Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
45 (2007) 233–254

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Bayesian network learning algorithms using structural restrictions

Luis M. de Campos *, Javier G. Castellano

Departamento de Ciencias de la Computación e I.A., E.T.S.I. Informática y Telecomunicaciones, Universidad de Granada, 18071 Granada, Spain

Received 30 December 2005; received in revised form 16 June 2006; accepted 30 June 2006
Available online 11 August 2006

Abstract

The use of several types of structural restrictions within algorithms for learning Bayesian networks is considered. These restrictions may codify expert knowledge in a given domain, in such a way that a Bayesian network representing this domain should satisfy them. The main goal of this paper is to study whether the algorithms for automatically learning the structure of a Bayesian network from data can obtain better results by using this prior knowledge. Three types of restrictions are formally defined: existence of arcs and/or edges, absence of arcs and/or edges, and ordering restrictions. We analyze the possible interactions between these types of restrictions and also how the restrictions can be managed within Bayesian network learning algorithms based on both the score + search and conditional independence paradigms. Then we particularize our study to two classical learning algorithms: a local search algorithm guided by a scoring function, with the operators of arc addition, arc removal and arc reversal, and the PC algorithm. We also carry out experiments using these two algorithms on several data sets.

Keywords: Bayesian networks; Learning algorithms; Structural restrictions

* Corresponding author. Tel.: +34 958 244019; fax: +34 958 243317.
E-mail addresses: lci@decsai.ugr.es (L.M. de Campos), fjgc@decsai.ugr.es (J.G. Castellano).

0888-613X/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2006.06.009

1. Introduction

Nowadays, Bayesian networks [26] constitute a widely accepted formalism for representing uncertain knowledge and for efficiently reasoning with it. A Bayesian network (BN) is a graphical representation of a joint probability distribution, which consists of a qualitative part, a directed acyclic graph (DAG) representing conditional (in)dependence relationships, and a quantitative one, a collection of numerical parameters representing conditional probability distributions. There has been a lot of work in the last ten years on the automatic learning of Bayesian networks from data and, consequently, many learning algorithms have been developed, based on different methodologies. However, little attention has been paid to the use of additional expert knowledge, not present in the data, in combination with a given learning algorithm. This knowledge could help in the learning process and contribute to get more accurate results, and even reduce the search effort of the BN representing a given domain of knowledge.

In this paper, we address this problem by considering some kinds of expert knowledge that will be codified by defining several types of restrictions, which will be used in conjunction with algorithms for learning Bayesian networks. More precisely, we shall consider three types of restrictions: (1) existence of arcs and edges, (2) absence of arcs and edges, and (3) ordering restrictions. All of them will be considered "hard" restrictions (as opposed to "soft" restrictions [19]), in the sense that they are assumed to be true for the BN representing the domain of knowledge, and therefore all the candidate BNs must necessarily satisfy them. We can consider our method as a mixture of automatic learning from data and manual construction of Bayesian networks using expert knowledge. Therefore, we can move from one extreme to the other using more or less structural restrictions.

The paper is structured as follows: in Section 2 we briefly give some preliminary basic concepts about learning the structure of Bayesian networks. Section 3 formally introduces the three types of restrictions that we are going to study. In Section 4 we describe how to represent the restrictions and how to manage them, including their self-consistency and the consistency of the restrictions with a given graph. Section 5 studies how to combine the restrictions with learning algorithms based on the score + search paradigm, and particularizes this study to the case of algorithms based on local search. Section 6 carries out a similar study for learning algorithms based on independence tests, focusing on the PC algorithm.[1] Section 7 discusses the experimental results and Section 8 contains the concluding remarks. Finally, although the proofs of the propositions set out in the paper are relatively simple, for the sake of completeness we have included them in the appendix.

2. Notation and preliminaries

Let us consider a finite set $\mathscr{V} = \{x_1, x_2, \ldots, x_n\}$ of discrete random variables, each variable taking on values from a finite set. We shall use lower-case letters for variable names, and capital letters to denote sets of variables. The structure of a Bayesian network on this domain is a directed acyclic graph (DAG) $G = (\mathscr{V}, E_G)$, where $\mathscr{V}$ is the set of nodes[2] and E_G represents the set of arcs in the graph.

[1] Examples of existing systems that can manage some kinds of structural restrictions within PC are TETRAD [27] and Hugin [25].

[2] We do not distinguish between a node and the random variable it represents.

The problem of learning the structure of a BN from data is that given a training set $\mathscr{D}$ of instances of the variables in $\mathscr{V}$, find the network that, in some sense, best matches $\mathscr{D}$. The learning algorithms may be subdivided into two general approaches: methods based on conditional independence tests, and methods based on a scoring function and a search procedure.

The algorithms based on the score + search paradigm attempt to find a graph that maximizes the selected score. All use a scoring function [12,19,21], usually defined as a measure of fit between the graph and the data, in combination with a search method in order to measure the goodness of each explored structure from the space of feasible solutions. Most of these algorithms use different search methods [5,6,13,14,16,19,20,23] but the same search space: the space of DAGs.[3]

The algorithms based on independence tests generate a list of conditional independence relationships among the variables in the domain (obtained from $\mathscr{D}$ by means of conditional independence tests), and attempt to find a network that represents these relationships as far as possible. The number, complexity and reliability of the required independence tests are the main concerns regarding this type of algorithms [9,15,27]. The class of graphs where these methods implicitly search for the best solution is that of partially directed acyclic graphs (PDAGs), which may contain both undirected links[4] (edges) and directed links (arcs) but no directed cycle.

In both cases our objective is to narrow the corresponding search space (which is hyper-exponential) by introducing several types of restrictions that the elements in this space must satisfy.

3. Types of restrictions

We are going to study three types of restrictions on the graph structures defined for the domain $\mathscr{V}$, namely existence, absence and ordering restrictions.

3.1. Existence restrictions

We shall consider two kinds of existence restrictions, existence of arcs and existence of edges. Let $\mathscr{E}_a, \mathscr{E}_e \subseteq \mathscr{V} \times \mathscr{V}$ be two subsets of pairs of variables, with $\mathscr{E}_a \cap \mathscr{E}_e = \emptyset$. They will be interpreted as follows:

- $(x, y) \in \mathscr{E}_a$: the arc $x \to y$ must belong to any graph in the search space.
- $(x, y) \in \mathscr{E}_e$: there is either a directed or a undirected link between the nodes x and y in any graph in the search space. In the case of the DAG space this means that either the arc $x \to y$ or the arc $y \to x$ must appear in any DAG.

An example of the use of existence restrictions may be any BAN algorithm [8], a BN learning algorithm for classification, which fixes the naive Bayes structure (i.e. arcs from

[3] Although other alternatives are possible, as searching in a space of equivalence classes of DAGs [2,11] or in a space of orderings [17,24], in this paper we shall focus only on the space of DAGs.

[4] Because the directionality of some links cannot be determined by using only information about conditional independence relationships between sets of variables.

the class variable to all the attribute variables) and searches for the appropriate additional arcs connecting pairs of attribute variables.

3.2. Absence restrictions

We shall also consider two kinds of absence restrictions, absence of arcs and absence of edges. Let $\mathscr{A}_a, \mathscr{A}_e \subseteq \mathscr{V} \times \mathscr{V}$ be two subsets of pairs of variables, with $\mathscr{A}_a \cap \mathscr{A}_e = \emptyset$. Their meaning is the following:

- $(x, y) \in \mathscr{A}_a$: the arc $x \to y$ cannot be present in any graph in the search space.
- $(x, y) \in \mathscr{A}_e$: there is neither a directed nor a undirected link connecting nodes x and y in any graph in the search space. For the DAG space this means that neither the arc $x \to y$ nor the arc $y \to x$ can appear in any DAG.

An example of the use of absence restrictions is a selective naive Bayesian classifier [22], which forbids arcs between attribute variables and also arcs from the attributes to the class variable.

3.3. Partial ordering restrictions

We need some additional concepts to better understand the meaning of this kind of restriction. We shall say that a total ordering, σ, of the set of variables $\mathscr{V}$ is *compatible* with a partial ordering, μ, of the same set of variables if

$$\forall x, y \in \mathscr{V}, \quad \text{if } x <_\mu y \text{ then } x <_\sigma y, \tag{1}$$

i.e. if x precedes y in the partial ordering μ then also x precedes y in the total ordering σ. Notice that a DAG (and also a PDAG) G determines a partial ordering on its variables: if there is a directed path from x to y in G, then x precedes y. Therefore, we can also say that a total ordering σ on the set $\mathscr{V}$ is *compatible* with a graph $G = (\mathscr{V}, E_G)$ if

$$\forall x, y \in \mathscr{V}, \quad \text{if } x \to y \in E_G \text{ then } x <_\sigma y. \tag{2}$$

Now, let us consider a subset $\mathscr{R}_o \subseteq \mathscr{V} \times \mathscr{V}$. In this case the interpretation is:

- $(x, y) \in \mathscr{R}_o$: every graph in the search space has to satisfy that x precedes y in some total ordering of the variables compatible with the graph.

Notice that the restriction $(x, y) \in \mathscr{R}_o$ is equivalent to assert that there is not a directed path from y to x in any of the graphs in the search space. The ordering restrictions may represent, for example, temporal or functional precedence between variables. Examples of use of ordering restrictions are all the BN learning algorithms that require a fixed total ordering of the variables (as the well-known K2 algorithm [12] or the algorithm in [15]).

4. Representation and management of the restrictions

In order to manage the restrictions it is useful to represent them also graphically. So, the existence restrictions can be represented by means of a partially directed graph $G_e = (\mathscr{V}, E_e)$, where each element (x, y) in $\mathscr{E}_a$ is associated with the corresponding arc

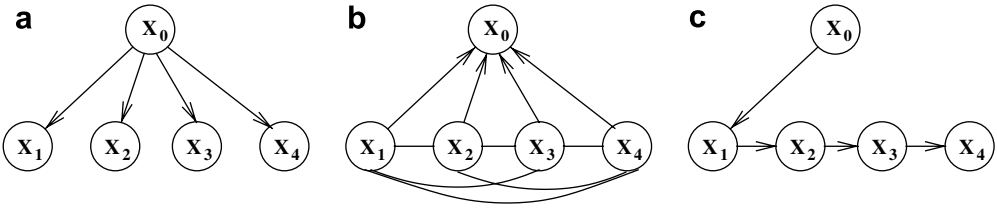

Fig. 1. (a) Existence graph G_e used by a BAN algorithm; (b) absence graph G_a corresponding to a selective naive Bayesian classifier; (c) ordering graph G_o used by the K2 algorithm. In cases (a) and (b) x_0 represents the class variable.

$x \to y \in E_e$, and each element (x, y) in $\mathscr{E}_e$ is associated with the edge $x—y \in E_e$. Fig. 1(a) displays the existence graph used by a BAN algorithm. The absence restrictions may be represented by means of another partially directed graph $G_a = (\mathscr{V}, E_a)$, where the elements (x, y) in $\mathscr{A}_a$ correspond with arcs $x \to y \in E_a$ and the elements (x, y) in $\mathscr{A}_e$ are associated with edges $x—y \in E_a$. Fig. 1(b) represents the absence graph corresponding to a selective naive Bayesian classifier. Finally, the ordering restrictions will be represented by using a directed graph $G_o = (\mathscr{V}, E_o)$, with each (x, y) in $\mathscr{R}_o$ being associated with the arc $x \to y \in E_o$. Notice that, as we are assuming that the ordering restrictions form a partial ordering (i.e. the relation is transitive), we are not forced to include in G_o an arc for each element in $\mathscr{R}_o$. G_o may be any graph such that its transitive closure contains an arc for each element in $\mathscr{R}_o$. For example, to represent a total ordering restriction $x_1 < x_2 < \cdots < x_n$ it suffices to include in G_o the $n - 1$ arcs $x_i \to x_{i+1}$, $i = 1, \ldots, n - 1$, instead of having a complete graph with all the arcs $x_i \to x_j$, $\forall i < j$. Fig. 1(c) displays the ordering graph used by the K2 algorithm.

Now, let us formally define when a given graph is *consistent* with a set of restrictions (i.e. the graph satisfies the restrictions):

Definition 1. Let $G_e = (\mathscr{V}, E_e)$, $G_a = (\mathscr{V}, E_a)$ and $G_o = (\mathscr{V}, E_o)$ be the graphs representing the existence, absence and ordering restrictions, respectively. Let $G = (\mathscr{V}, E_G)$ be a DAG and $H = (\mathscr{V}, E_H)$ be a PDAG. We say that

(1) G is consistent with the existence restrictions if and only if
   - $\forall x, y \in \mathscr{V}$, if $x \to y \in E_e$ then $x \to y \in E_G$,
   - $\forall x, y \in \mathscr{V}$, if $x—y \in E_e$ then $x \to y \in E_G$ or $y \to x \in E_G$.

(2) G is consistent with the absence restrictions if and only if
   - $\forall x, y \in \mathscr{V}$, if $x \to y \in E_a$ then $x \to y \notin E_G$,
   - $\forall x, y \in \mathscr{V}$, if $x—y \in E_a$ then $x \to y \notin E_G$ and $y \to x \notin E_G$.

(3) G is consistent with the ordering restrictions if and only if
   - there exists a total ordering σ of the variables in $\mathscr{V}$ compatible with both G and G_o.

(4) H is consistent with the existence restrictions if and only if
   - $\forall x, y \in \mathscr{V}$, if $x \to y \in E_e$ then $x \to y \in E_H$,
   - $\forall x, y \in \mathscr{V}$, if $x—y \in E_e$ then $x \to y \in E_H$ or $y \to x \in E_H$ or $x—y \in E_H$,
   - H can be transformed into a DAG, which is consistent with the restrictions, by directing its edges.

(5) H is consistent with the absence restrictions if and only if
   - $\forall x, y \in \mathscr{V}$, if $x \to y \in E_a$ then $x \to y \notin E_H$,
   - $\forall x, y \in \mathscr{V}$, if $x—y \in E_a$ then $x \to y \notin E_H$, $y \to x \notin E_H$ and $x—y \notin E_H$,

- H can be transformed into a DAG, which is consistent with the restrictions, by directing its edges.

(6) H is consistent with the ordering restrictions if and only if
- there exists a total ordering σ of the variables in $\mathscr{V}$ compatible with both H and G_o,
- H can be transformed into a DAG, which is consistent with the restrictions, by directing its edges.

When we are specifying the set of restrictions to be used within a given domain, it is necessary to make sure that these restrictions can indeed be satisfied. In this sense, we shall say that a set of restrictions is *self-consistent* if there is some DAG that is consistent with them. Testing the self-consistency of each of the three types of restrictions separately is very simple:

Proposition 2. *Let* $G_e = (\mathscr{V}, E_e)$, $G_a = (\mathscr{V}, E_a)$ *and* $G_o = (\mathscr{V}, E_o)$ *be the graphs representing existence, absence and ordering restrictions, respectively. Then*

(a) *The set of existence restrictions is self-consistent if and only if the graph* G_e *has no directed cycle.*
(b) *The set of absence restrictions is always self-consistent.*
(c) *The set of ordering restrictions is self-consistent if and only if* G_o *is a DAG.*

When several types of self-consistent restrictions are considered simultaneously, some interactions can occur among each other. These interactions may give rise to inconsistencies. For example, the existence and absence of the same arcs; the ordering restrictions may also contradict with the existence of some arcs (as they implicitly also represent partial ordering restrictions). For instance, $x \to v$, $v \to y \in E_e$ contradicts with $y \to z$, $z \to t$, $t \to x \in E_o$.

It is also possible that some absence or ordering restrictions force an existence restriction. For instance, if an arc must exist in either direction (i.e. $x—y \in E_e$) but either an absence or an ordering restriction indicates that some direction is forbidden (e.g. $x \to y \in E_a$ or $y \to x \in E_o$), then the other direction is forced ($x—y$ should be replaced by $y \to x$ in E_e). This can also produce interactions among the three types of restrictions, giving rise to inconsistencies. For example, if $y \to t$, $t \to x$, $x—z$, $z—y \in E_e$, $y \to z \in E_a$ and $x \to z \in E_o$, the absence and ordering restrictions force the orientation of the edges $z—y$ and $x—z$ which, together with the other existence restrictions, generate a directed cycle. The following result characterizes *global* self-consistency of the restrictions, in terms of simple operations on graphs.

Proposition 3. *Let* $G_e = (\mathscr{V}, E_e)$, $G_a = (\mathscr{V}, E_a)$ *and* $G_o = (\mathscr{V}, E_o)$ *be the graphs representing existence, absence and ordering restrictions, respectively. Let* $G_{re} = (\mathscr{V}, E_{re})$ *be the refined graph of existence restrictions*[5] *defined as*

$$\begin{aligned} E_{re} = {} & \{x \to y | x \to y \in E_e\} \cup \{y \to x | x—y \in E_e, x \to y \in E_a\} \\ & \cup \{x—y | x—y \in E_e, x \to y \notin E_a, y \to x \notin E_a\} \end{aligned} \tag{3}$$

Then the three sets of restrictions are self-consistent if and only if $G_{re} \cap G_a = G_\emptyset$ *and* $G_{re} \cup G_o$ *has no directed cycle.* $G_\emptyset$ *is the empty graph*[6] *and both the union and the intersection*

[5] G_{re} is the graph G_e with some edges being replaced by arcs (those ones whose direction is forced because of an absence restriction).

[6] A graph having neither arcs nor edges.

of two partially directed graphs use the convention that $\{x \to y\} \cup \{x—y\} = \{x \to y\}$ *and* $\{x \to y\} \cap \{x—y\} = \{x \to y\}$.

The following result shows that testing the consistency of a DAG with a set of restrictions can also be reduced to simple graph operations.

Proposition 4. *Let* $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ *and* $G_o = (\mathcal{V}, E_o)$ *be graphs representing self-consistent existence, absence and ordering restrictions, respectively, and let* $G = (\mathcal{V}, E_G)$ *be a DAG. Then G is consistent with the restrictions if and only if* $G \cup G_e = G$, $G \cap G_a = G_\emptyset$ *and* $G \cup G_o$ *is a DAG.*

Testing the consistency of a PDAG with a set of restrictions is only a bit more complicated, because of the possible interaction between the edges in the PDAG and the arcs in the absence restrictions:

Proposition 5. *Let* $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ *and* $G_o = (\mathcal{V}, E_o)$ *be graphs representing self-consistent existence, absence and ordering restrictions, respectively, and let* $H = (\mathcal{V}, E_H)$ *be a PDAG and* $H_r = (\mathcal{V}, E_{H_r})$ *be the refined PDAG*[7] *defined as*

$$\begin{aligned} E_{H_r} = {} & \{x \to y | x \to y \in E_H\} \cup \{y \to x | x—y \in E_H, x \to y \in E_a\} \\ & \cup \{x—y | x—y \in E_H, x \to y \notin E_a, y \to x \notin E_a\} \end{aligned} \tag{4}$$

Then H is consistent with the restrictions if and only if $H \cup G_e = H$, $H_r \cap G_a = G_\emptyset$ *and* $H_r \cup G_o$ *has no directed cycle.*

5. Using the restrictions within score + search learning algorithms

If we have a set of self-consistent restrictions and we want to build a Bayesian network from data using a score + search learning algorithm, it seems natural to use them to reduce the search space and force the algorithm to return a DAG consistent with the restrictions. A general mechanism to do it, which is valid for any algorithm of this type, is very simple: each time the search process selects a candidate DAG G to be evaluated by the scoring function, we can use the result in Proposition 4 to test whether G is consistent with the restrictions, and reject it otherwise.

However, this general procedure may be somewhat inefficient. It would be convenient to adapt it to the specific characteristics of the learning algorithm being used. We are going to do that for the case of the classical score + search learning algorithm based on local search [19], which uses the operators of arc insertion, arc deletion and arc reversal.

5.1. Conditions to apply the search operators

We start from the current DAG G, which is consistent with the restrictions, and let G' be the DAG obtained from G by applying one of the aforementioned operators. Let us see which are the conditions necessary and sufficient to assure that G' is also consistent with the restrictions.

[7] As before, H_r is the graph H with the edges whose direction is forced because of an absence restriction being replaced by arcs.

Proposition 6. *Let* $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ *and* $G_o = (\mathcal{V}, E_o)$ *be graphs representing self-consistent existence, absence and ordering restrictions, respectively, and let* $G = (\mathcal{V}, E_G)$ *be a DAG consistent with the restrictions.*

(a) *Arc insertion*: *Let* $G' = (\mathcal{V}, E'_G)$, $E'_G = E_G \cup \{x \to y\}$, *with* $x \to y \notin E_G$. *Then* G' *is a DAG consistent with the restrictions if and only if*
- $x \to y \notin E_a$ *and* $x—y \notin E_a$,
- *there is not any directed path from* y *to* x *in* $G \cup G_o$.

(b) *Arc deletion*: *Let* $G' = (\mathcal{V}, E'_G)$, $E'_G = E_G \setminus \{x \to y\}$, *with* $x \to y \in E_G$. *Then* G' *is a DAG consistent with the restrictions if and only if*
- $x \to y \notin E_e$ *and* $x—y \notin E_e$.

(c) *Arc reversal*: *Let* $G' = (\mathcal{V}, E'_G)$, $E'_G = (E_G \setminus \{x \to y\}) \cup \{y \to x\}$, *with* $x \to y \in E_G$. *Then* G' *is a DAG consistent with the restrictions if and only if*
- $x \to y \notin E_e$, $y \to x \notin E_a$ *and* $x \to y \notin E_o$,
- *excluding the arc* $x \to y$, *there is not any other directed path from* x *to* y *in* $G \cup G_o$.

Notice that the conditions about the absence of directed paths between x and y in the previous proposition have also to be checked by the algorithm that does not consider the restrictions (using in this case the DAG G instead of $G \cup G_o$), so that the extra cost of managing the restrictions is quite reduced: only two or three tests about the absence of either an arc or an edge from a graph.

It is also interesting to notice that other score + search learning algorithms, more sophisticated that a simple local search, can also be easily extended to efficiently deal with the restrictions. There are many BN learning algorithms that perform a search more powerful than local search but use the same three basic operators (as variable neighborhood search [16], tabu search [2] or GRASP[8] [14]), or even a subset of them (as ant colony optimization [13] which only uses arc insertion[9]). These algorithms can be used together with the restrictions with almost no additional modification.

5.2. Search initialization

Another question to be considered is the initialization of the search process. In general, the learning algorithms start from one or several initial DAGs that, in our case, must be consistent with the restrictions. A very common starting point is the empty DAG $G_\emptyset$. In our case $G_\emptyset$ should be replaced by the graph G_e or, even better, by the graph G_{re}. However, as G_{re} is not necessarily a DAG, it must be transformed into a DAG. An easy way to do it is to iteratively select an edge $x—y \in E_{re}$, randomly choose an orientation and test whether the restrictions are still self-consistent, choosing the opposite orientation if the test is negative. This process is based on the following result:

Proposition 7. *Let* $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ *and* $G_o = (\mathcal{V}, E_o)$ *be graphs representing self-consistent existence, absence and ordering restrictions, respectively, and let* $G_{re} = (\mathcal{V}, E_{re})$ *be the refined graph of existence restrictions. Let* $x—y \in E_{re}$ *and define the graph* $G_{e(x \to y)} = (\mathcal{V}, E_{e(x \to y)})$, *where* $E_{e(x \to y)} = (E_e \setminus \{x—y\}) \cup \{x \to y\}$. *Then* $G_{e(x \to y)}$, G_a *and* G_o *are still*

[8] Greedy randomized adaptive search procedure.
[9] The B algorithm [7] also uses arc insertion only, together with local search.

self-consistent if and only if there is not a directed path from y to x in $G_{re} \cup G_o$. Moreover, either $G_{e(x \to y)}$ or $G_{e(y \to x)}$, together with G_a and G_o, are self-consistent.

In other cases the search algorithm is initialized with one (or several) random DAGs. The process of selecting a random DAG, checking the restrictions and iterating until the generated DAG satisfies the restrictions may be time-consuming, specially when there are many restrictions. In these cases it would be quite useful to have a repair operator, i.e. a method to transform any DAG G into one verifying the restrictions. This method can also be useful for learning algorithms using population-based search processes (as genetic algorithms [23] and EDAs [5]). There are many ways to define this repair operator. Here we propose a quite simple method: we start from a DAG G_{red} containing only the arcs[10] (not the edges) in G_{re}; then, given a random ordering of the arcs in G we iteratively try to insert each of these arcs into G_{red}, using the conditions in Proposition 6(a); finally, for the edges in G_{re}, we include them in G_{red} with the appropriate orientation, using the test in Proposition 7 (replacing the graph $G_{re} \cup G_o$ by $G_{red} \cup G_o$). The result is a DAG consistent with the restrictions and containing as many arcs from G as possible.

6. Using the restrictions within independence-based learning algorithms

The learning algorithms based on independence tests typically proceed by eliminating edges connecting pairs of nodes which are conditionally independent given some subset of nodes, and by directing edges to form head-to-head patterns (triplets of nodes x, y, z such that x and y are not adjacent and the arcs $x \to z$ and $y \to z$ exist). Both activities are guided by the results of some statistical tests of conditional independence applied to the available data. For example, the SGS and PC algorithms [27] first eliminate as many edges as they can, and after they give direction to some of the non-removed edges by forming head-to-head patterns. Finally, several additional edges may be directed by using some coherence rules.

In this general context, a simple method to use a set of restrictions, in order to reduce the number of necessary tests, is the following: before applying an independence test $I(x, y|Z)$, to either eliminate an edge x—y or to create a head-to-head pattern $x \to z \leftarrow y$, we could test whether the graph obtained by applying this operation is consistent with the restrictions (using the result in Proposition 5); if the consistency test fails, the independence test will not be carried out. However, in order to improve the efficiency of the procedure, it is convenient to adapt the general consistency test in Proposition 5 to the specific characteristics of the operators being used.

Proposition 8. *Let $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ and $G_o = (\mathcal{V}, E_o)$ be graphs representing self-consistent existence, absence and ordering restrictions, respectively, and let $H = (\mathcal{V}, E_H)$ be a PDAG consistent with the restrictions.*

(a) *Arc deletion: Let $H' = (\mathcal{V}, E'_H)$, $E'_H = E_H \setminus \{x \to y\}$, with $x \to y \in E_H$. Then H' is a PDAG consistent with the restrictions if and only if*
- *$x \to y \notin E_e$ and x—$y \notin E_e$.*

[10] $G_{red} = (\mathcal{V}, E_{red})$, with $E_{red} = \{x \to y | x \to y \in E_{re}\}$.

(b) *Edge deletion*: *Let* $H' = (\mathcal{V}, E'_H)$, $E'_H = E_H \setminus \{x—y\}$, *with* $x—y \in E_H$. *Then* H' *is a PDAG consistent with the restrictions if and only if*
- $x \to y \notin E_e$, $y \to x \notin E_e$ *and* $x—y \notin E_e$.

(c) *Head-to-head insertion*: *Let* $x, y, z \in \mathcal{V}$ *and define a subset of links* S *as either* $S = \{x \to z, z—y\}$, $S = \{x—z, y \to z\}$ *or* $S = \{x—z, z—y\}$. *If* x *and* y *are not adjacent in* H *and* $S \subseteq E_H$, *let* $H' = (\mathcal{V}, E'_H)$, *with* $E'_H = (E_H \setminus S) \cup \{x \to z, y \to z\}$. *Then* H' *is a PDAG consistent with the restrictions if and only if*
- $x \to z \notin E_a$ *and* $y \to z \notin E_a$,
- *there is a directed path neither from* z *to* x *nor from* z *to* y *in* $H \cup G_o$.

(d) *Edge orientation*: *Let* $H' = (\mathcal{V}, E'_H)$, $E'_H = (E_H \setminus \{x—y\}) \cup \{x \to y\}$, *with* $x—y \in E_H$. *Then* H' *is a PDAG consistent with the restrictions if and only if*
- $x \to y \notin E_a$,
- *there is not a directed path from* y *to* x *in* $H \cup G_o$.

Moreover, another way to reduce the number and complexity of the required independence tests, is to use the restrictions to reduce the size of the sets of nodes which are candidate to form the separating sets employed by the tests. Focusing on the PC algorithm, it tries to determine whether two nodes x and y are not adjacent by testing whether y is independent on x conditional on some subset of the current adjacencies of x (and after testing whether x is independent on y conditional on some subset of the current adjacencies of y). The reason is that, if x and y are not adjacent, then they will be conditionally independent given either the parents of x or the parents of y in the true graph, and these sets will always be subsets of the current adjacencies of either x or y, respectively. In this context, we can use the restrictions to remove, from any subset of the current adjacencies of node x, all the nodes that cannot be parents of x: for each node z (excluding y) adjacent to x in the current graph, if either there is a directed path from x to z in $G_{\mathrm{re}} \cup G_{\mathrm{o}}$, or $z \to x \in E_{\mathrm{a}}$ or $x—z \in E_{\mathrm{a}}$ then, according to Proposition 5, $z \to x$ cannot be an arc in any graph consistent with the restrictions and therefore it can be safely removed from any candidate separating set.

As in the case of the score + search based methods, we have also to consider the initialization step of the algorithm. A common starting point for independence-based algorithms is the complete undirected graph $G_{\mathrm{c}} = (\mathcal{V}, E_{\mathrm{c}})$, with $E_{\mathrm{c}} = \{x—y | x, y \in \mathcal{V}\}$. In our case this initial graph should be transformed by removing the edges in the absence restrictions and giving direction to some edges taking into account the arcs in the existence, ordering and absence restrictions. More precisely, let us define the following graphs:

- The graph of undirected absence restrictions:

$$G_{\mathrm{au}} = (\mathcal{V}, E_{\mathrm{au}}), \quad E_{\mathrm{au}} = \{x—y | x—y \in E_{\mathrm{a}}\}$$

- The graph of inverted absence restrictions:

$$G_{\mathrm{ai}} = (\mathcal{V}, E_{\mathrm{ai}}), \quad E_{\mathrm{ai}} = \{x \to y | y \to x \in E_{\mathrm{a}}\}$$

- The transitive closure of the graph of ordering and existence restrictions:

$$G_{\mathrm{eot}} = (\mathcal{V}, E_{\mathrm{eot}}), \quad E_{\mathrm{eot}} = \{x \to y | \text{ there is a directed path from } x \text{ to } y \text{ in } G_{\mathrm{re}} \cup G_{\mathrm{o}}\}$$

The complete undirected graph G_{c} should then be replaced by the graph $(G_{\mathrm{c}} \backslash G_{\mathrm{au}}) \cup G_{\mathrm{eot}} \cup G_{\mathrm{ai}}$. Fig. 2 illustrates this transformation.

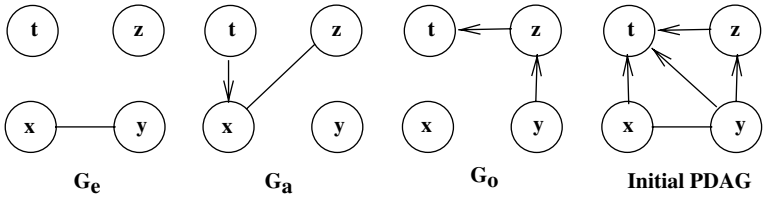

Fig. 2. Transforming the complete undirected graph into an initial PDAG consistent with the restrictions G_e, G_a and G_o.

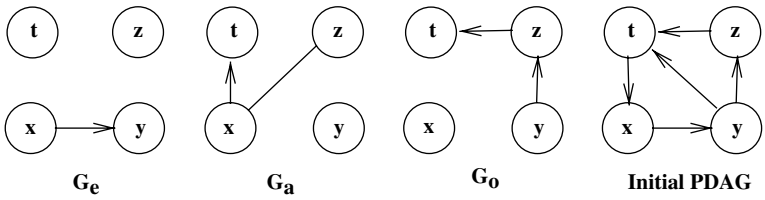

Fig. 3. Transforming the complete undirected graph into an initial graph which is not consistent with the restrictions G_e, G_a and G_o.

However, a problem may appear, namely that this initial PDAG may be non-consistent with the restrictions. For example, the set of self-consistent restrictions in Fig. 3 gives rise to a graph $(G_c \backslash G_{au}) \cup G_{eot} \cup G_{ai}$ which is not consistent. It is quite simple to show that this situation will occur if and only if the graph $G_e \cup G_o \cup G_{ai}$ has a directed cycle.

The reason for this situation is that in these cases the set of restrictions, *together*, implies another absence restrictions which have not been explicitly stated. For the example in Fig. 3, the explicit restrictions imply that the arc $t \to x$ cannot exist (so that the restriction $x \to t \in E_a$ in fact is $x—t \in E_a$). A possible solution is to detect this situation and then remove the corresponding arc (the arc $t \to x$ in the example). However, there are cases where more than one arc could be removed (i.e. the set of implicit absence restrictions that can be deduced from the explicit restrictions does not form a conjunction). For example, the set of restrictions $x \to y \in E_e$, $y \to z \in E_o$ and $x \to t$, $t \to z \in E_a$ implies that either $x—t \in E_a$ or $t—z \in E_a$. In these cases we would have to choose eliminating one of these arcs without using additional information. For that reason we believe that a better solution is to postpone the removal of these arcs, temporarily allowing the intermediate graphs obtained to be non-consistent with the restrictions. Once the phase of elimination of edges/arcs guided by independence tests has finished, we would remove the additional arcs (if any) which are necessary to restore the consistency. This can be easily done by testing for the presence of directed cycles in the graph $H \cup G_o$, being H the current graph, and then removing one of the arcs in the cycle that comes from an absence restriction. The advantage of this strategy is that some of the arcs that could generate the lack of consistency may have already been eliminated by the independence tests, and we avoid making an arbitrary selection. In the last example, if the arc $z \to t$ is eliminated by an independence test, we avoid the risk of having arbitrarily removed the arc $t \to x$ at the beginning (to after eliminate also the arc $z \to t$).

The proposed adaptation of the PC algorithm to use restrictions is therefore quite simple: we start with the PDAG $(G_c \backslash G_{au}) \cup G_{eot} \cup G_{ai}$. Next, the phase of edge elimination of

PC is carried out, considering the conditions (a) and (b) in Proposition 8 for arc and edge deletion, respectively, using also the reduced candidate separating sets. Then we test for the consistency of the resulting PDAG and eliminate some arcs if necessary, as we explained previously. The following step is the PC phase of detection of head-to-head patterns, using in this case the conditions (c) in Proposition 8. Finally, the orientation of additional edges using the coherence rules is carried out using the conditions (d) in Proposition 8. If the final graph H is not a DAG, we can direct the remaining edges in either direction that avoids the creation of new head-to-head patterns, as long as we do not create directed cycles in $H \cup G_o$.

7. Experimental results

In this section we shall describe the experiments carried out to test the effect of using restrictions on BN learning algorithms, and the obtained results. The score + search learning algorithm considered is the previously mentioned classical local search (with addition, removal and reversal of arcs), using the BDeu scoring function [19], with the parameter representing the equivalent sample size set to 1 and a uniform structure prior. The independence-based learning algorithm used is PC. We have selected four different problems. The Alarm network (left-hand side of Fig. 4) displays the relevant variables and relationships for the Alarm Monitoring System [3], a diagnostic application for patient monitoring. This network contains 37 variables and 46 arcs. Insurance [4] is a network for evaluating car insurance risks. The Insurance network (Fig. 5) contains 27 variables and 52 arcs. Hailfinder [1] is a normative system that forecasts severe summer hail in northeastern Colorado. The Hailfinder network contains 56 variables and 66 arcs. Asia (right-hand side of Fig. 4) is a small Bayesian network that calculates the probability of a patient having tuberculosis, lung cancer or bronchitis respectively based on different factors. All these networks have been widely used in specialist literature for comparative purposes.

The collected performance measures are: (1) The scoring value (BDeu) of the obtained network; this measure is interesting because it is just the criterion which guides the local search algorithm. (2) Three measures of the structural difference between the learned network and the true one, which measure the capacity to reconstruct the graphical structure: the number of added arcs (A), the number of deleted arcs (D) and the number of inverted arcs (I) in the learned network with respect to the original. To eliminate fictitious differences or similarities between the two networks, caused by different but equivalent sub-DAG structures, before comparing the two networks we have converted them to their corresponding completed PDAG (also called essential graph) representation,[11] using the algorithm proposed in [10]. (3) A measure of the ability to reconstruct the joint probability distribution: we use the Kullback–Leibler divergence (KL) between the distributions associated to the original and the learned networks.

For each problem we have randomly selected fixed percentages of restrictions of each type, extracted from the whole set of restrictions corresponding to the true network. More precisely, if $G = (\mathcal{V}, E_G)$ is the true network, then each arc $x \to y \in E_G$ is a possible existence restriction (we may select the restriction $x \to y \in E_e$ if this arc is also present in the

[11] A completed PDAG is a partially directed acyclic graph which is a canonical representation of all the DAGs belonging to the same equivalence class of DAGs.

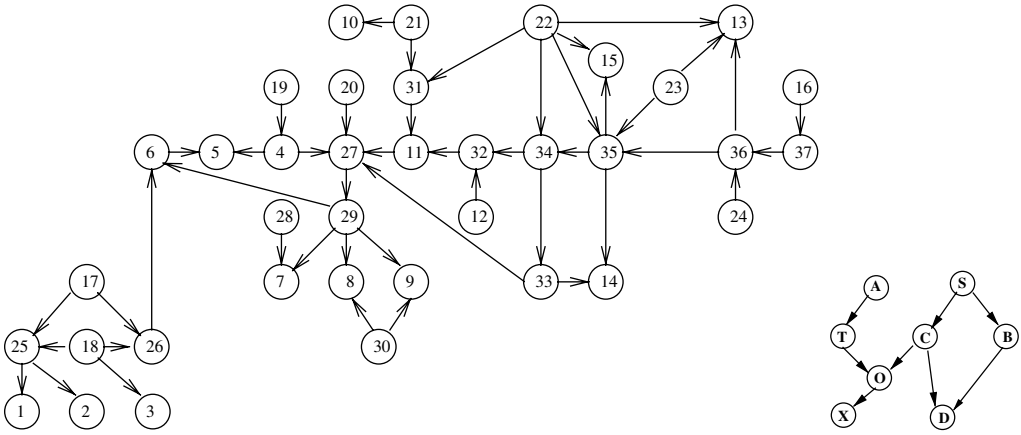

Fig. 4. The Alarm (left) and the Asia (right) networks.

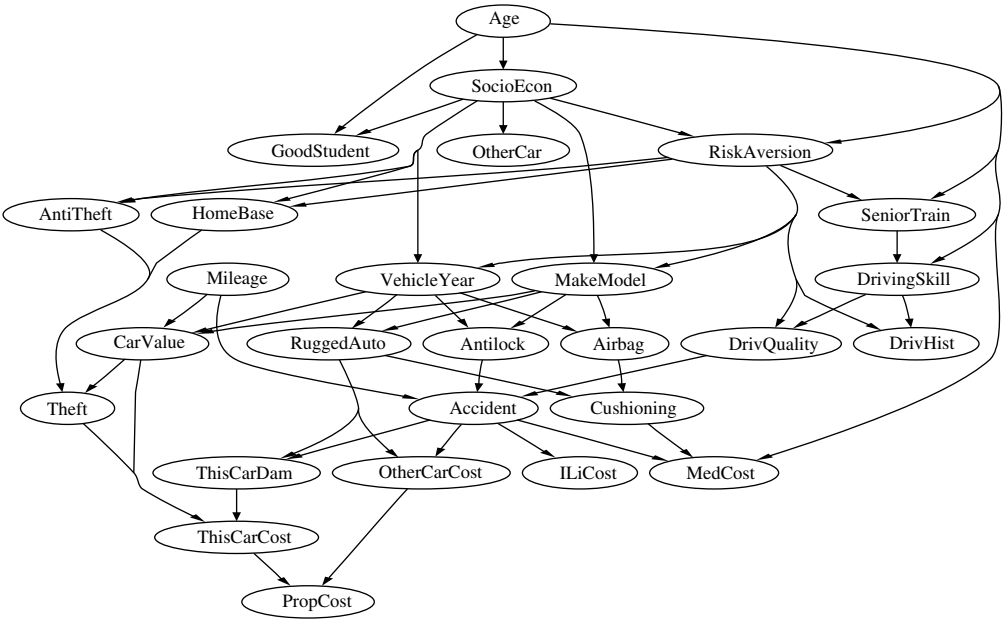

Fig. 5. The Insurance network.

completed PDAG representation of G; otherwise we would use the restriction $x—y \in E_e$); each arc $x \to y \notin E_G$ is a possible absence restriction (in case that also $y \to x \notin E_G$ we randomly select whether to use the restriction $x \to y \in E_a$ or $x—y \in E_a$); finally, if there is a directed path from x to y in the completed PDAG representation of G then $x \to y \in E_o$ is a possible ordering restriction. The selected percentages have been 10%, 20%, 30% and 40%.

We have run the learning algorithms for each percentage of restrictions of each type alone, and also using the three types of restrictions together.

Each network has been used to generate 10 databases, each of which contains 1000 instances, except for Asia, where the sample size is 100. The results displayed in the following sections represent the average values of the performance measures across 50 iterations (i.e. 5 random subsets of restrictions for each of the 10 datasets). All the implementations have been carried out within the Elvira System [18], a Java tool to construct probabilistic decision support systems, which works with Bayesian networks and influence diagrams.

7.1. Results for the local search algorithm

Tables 1–4 display the results obtained using the local search algorithm, including the results obtained by the learning algorithm without using restrictions (0%), and the KL values of the true networks, with parameters retrained from the corresponding databases, which may serve as a kind of scale. Tables 5 and 6 display the corresponding BDeu values, as well as the scoring values of the true networks.

First, let us analyze the results from the perspective of the structural differences. What it was expected is that the number of deleted arcs, added arcs and inverted arcs decreases as the number of existence, absence and ordering restrictions, respectively, increases. This behaviour is indeed clearly observed in the results. Moreover, another less obvious effect, frequently observed in the experiments, is that the use of any of the three types of restrictions also tends to decrease the other measures of structural difference. For example, the existence restrictions decrease the number of deleted arcs, but also the number of added and inverted arcs.

With respect to the analysis of the results from the perspective of the KL divergence, we have to distinguish Hailfinder from the other three datasets. For these datasets the use of each type of restriction leads to better network structures, and the improvement almost systematically increases with the number of restrictions being used. Nevertheless, there are a few cases (with Alarm) where using the absence restrictions gives worse results than those of the unrestricted local search. We believe that the explanation of this behaviour lies in the following fact: when a local search-based learning algorithm mistakes the direction of some arc connecting two nodes,[12] then the algorithm tends to 'cross' the parents of these nodes to compensate the wrong orientation; if some of these 'crossed' arcs are used as absence restrictions, then the algorithm cannot compensate the mistake and has to stop in a worse configuration. These results suggest another way of using the absence restrictions: once the algorithm, using only existence and ordering restrictions, has found a local maximum, we could delete all the forbidden arcs and run another local search. However, the case of Hailfinder is completely different, all the types of restrictions give rise to worse networks, the more restrictions we use the greater the KL divergence (except in the case of the absence restrictions). For the present we do not have an explanation for this unexpected behaviour of the KL divergence for the Hailfinder datasets.

Finally, concerning the BDeu values we can observe that they are always greater (better) than the BDeu scores of the true networks, which indicates some kind of overfitting

[12] This situation may be quite frequent at early stages of the search process.

Table 1
Average results obtained for Asia using local search

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	0.1385	2.3	1.8	0.8	0.1448	2.8	2.0	0.8	0.1418	2.4	2.1	0.5	0.1491	2.9	2.3	0.6
20	0.1319	1.7	1.4	0.5	0.1438	2.6	1.6	1.0	0.1409	2.1	2.1	0.6	0.1481	2.9	2.3	0.6
30	0.1225	1.5	1.1	0.4	0.1378	2.5	1.4	0.9	0.1408	1.9	2.0	0.6	0.1481	2.9	2.3	0.6
40	0.1169	1.1	0.8	0.3	0.1308	2.4	1.1	0.9	0.1376	1.4	1.8	0.5	0.1485	2.9	2.3	0.6
0	0.1491	2.9	2.3	0.6	KL true Asia network (retrained): 0.0955											

Table 2
Average results obtained for Alarm using local search

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	0.2875	6.7	3.0	7.9	0.2896	8.0	3.0	12.5	0.3419	9.9	3.9	15.7	0.3163	9.5	3.6	13.7
20	0.2620	4.2	2.4	4.3	0.2595	6.5	2.4	9.5	0.3608	8.7	4.2	12.8	0.3079	8.9	3.6	11.5
30	0.2338	2.7	2.0	2.7	0.2435	5.7	2.0	7.1	0.3401	6.5	4.1	9.6	0.2833	7.6	3.4	8.5
40	0.2240	2.1	1.7	1.6	0.2309	4.8	1.6	5.4	0.3183	4.9	3.6	7.0	0.2714	7.0	3.3	6.7
0	0.3199	10.5	3.7	18.4	KL true Alarm network (retrained): 0.2112											

Table 3
Average results obtained for Insurance using local search

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	0.5592	3.3	15.3	11.5	0.5871	4.6	15.9	13.6	0.6053	4.7	17.5	14.3	0.5984	5.4	17.9	14.2
20	0.5403	2.1	13.4	8.9	0.5571	3.5	13.9	12.3	0.5819	3.6	17.0	13.0	0.5729	4.9	17.8	13.0
30	0.5144	1.0	11.3	6.3	0.5293	2.5	11.8	9.8	0.5685	2.7	16.6	11.1	0.5614	4.5	17.6	11.8
40	0.5092	0.5	9.3	4.8	0.5263	1.9	9.8	7.3	0.5587	2.0	16.2	8.1	0.5513	4.1	17.4	10.4
0	0.6295	6.1	18.3	16.1	KL true Insurance network (retrained): 0.5531											

Table 4
Average results obtained for Hailfinder using local search

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	1.4409	11.7	16.2	13.2	1.4322	14.1	16.6	16.4	1.4329	13.4	17.7	15.6	1.4382	15.0	18.3	14.8
20	1.4974	9.2	14.3	12.4	1.4946	13.3	14.9	16.7	1.4249	11.3	17.3	13.1	1.4536	15.1	18.7	13.6
30	1.5209	6.7	12.3	10.4	1.5330	11.9	13.0	15.7	1.4049	8.9	16.2	11.5	1.4465	14.7	18.6	12.0
40	1.5453	4.8	10.7	8.6	1.5415	11.1	11.2	14.4	1.3886	7.2	15.6	9.8	1.4517	14.5	18.7	10.6
0	1.4216	15.5	18.1	17.3	KL true Hailfinder network (retrained): 1.1609											

to the data. Moreover, the BDeu values, as we increase the number of restrictions, tend to decrease for Asia and Hailfinder and tend to increase for Alarm and Insurance. We believe

Table 5
Average BDeu values for Asia and Alarm

%	Asia				Alarm			
	G_e, G_a, G_o	Only G_e	Only G_a	Only G_o	G_e, G_a, G_o	Only G_e	Only G_a	Only G_o
10	−218.97	−218.68	−218.66	−218.35	−11184.80	−11192.55	−11253.17	−11233.99
20	−219.50	−219.15	−218.73	−218.38	−11151.20	−11157.59	−11264.57	−11226.59
30	−219.67	−219.22	−219.01	−218.38	−11123.20	−11134.51	−11237.61	−11190.76
40	−220.00	−219.41	−219.25	−218.38	−11113.99	−11119.93	−11209.80	−11171.16
0	−218.35	BDeu true network: −258.91			−11233.37	BDeu true network: −11256.62		

Table 6
Average BDeu values for Insurance and Hailfinder

%	Insurance				Hailfinder			
	G_e, G_a, G_o	Only G_e	Only G_a	Only G_o	G_e, G_a, G_o	Only G_e	Only G_a	Only G_o
10	−14105.75	−14137.72	−14115.24	−14113.08	−52638.00	−52612.30	−52599.14	−52595.80
20	−14100.71	−14129.56	−14082.38	−14080.21	−52785.04	−52720.74	−52616.60	−52607.93
30	−14117.11	−14138.98	−14060.97	−14068.16	−52977.60	−52868.29	−52620.29	−52608.47
40	−14158.52	−14178.02	−14047.71	−14052.52	−53207.97	−53020.67	−52641.89	−52622.05
0	−14152.92	BDeu true network: −14439.10			−52580.32	BDeu true network: −55268.49		

that this is due to a greater degree of overfitting of the BDeu score in the first two cases, caused by a smaller relative sample size.

7.2. Results for the PC algorithm

In some preliminary experiments we observed poor results when using the absence restrictions, especially concerning the KL divergence and the number of inverted arcs. We think that the reason for this behaviour is the following: the algorithm does not perform independence tests for any two nodes x and y involved in undirected absence restrictions. Therefore, if these nodes become related through another node z, x—z—y, we have not the information required to determine whether they form a head-to-head pattern (namely that z is not in the subset which separated x and y). For that reason we have modified the algorithm in such a way that we do not eliminate from the initial graph the edges x—$y \in G_a$ but we remove them after the step where the algorithm eliminates edges using independence tests. The results obtained after this modification are displayed in Tables 7–10.

Table 7
Average results obtained for Asia using PC

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	0.2520	0.3	4.0	2.7	0.2639	0.2	4.1	3.0	0.2839	0.2	4.8	2.8	0.2847	0.2	4.8	2.9
20	0.2224	0.4	3.3	2.0	0.2388	0.2	3.4	2.8	0.2796	0.2	4.8	2.7	0.2802	0.3	4.7	2.8
30	0.2023	0.5	2.6	1.6	0.2242	0.3	3.0	2.6	0.2837	0.4	4.7	2.5	0.2740	0.4	4.7	2.6
40	0.1818	0.5	2.1	1.1	0.2145	0.4	2.4	2.4	0.2774	0.4	4.6	2.2	0.2735	0.6	4.7	2.3
0	0.2895	0.2	4.8	2.8	KL true Asia network (retrained): 0.0955											

Table 8
Average results obtained for Alarm using PC

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	2.2590	1.5	13.5	7.6	2.3850	1.7	15.0	8.5	2.7508	1.5	16.9	9.2	2.7265	1.6	17.1	8.8
20	1.5953	1.2	9.5	6.4	1.9995	1.8	12.6	7.2	2.5969	1.5	15.8	8.6	2.6471	1.7	16.5	8.3
30	1.1720	1.1	6.8	4.5	1.7976	1.9	10.5	6.2	2.3655	1.2	14.0	7.7	2.4914	1.6	15.2	8.3
40	0.8204	1.0	4.9	2.9	1.5681	1.9	8.5	5.6	2.0665	1.1	12.1	6.9	2.3195	1.7	14.2	8.2
0	2.7482	1.7	17.8	9.6	KL true Alarm network (retrained): 0.2112											

Table 9
Average results obtained for Insurance using PC

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	2.2192	1.1	26.9	11.4	2.4062	1.4	27.8	12.0	2.3718	1.1	31.2	7.9	2.3385	1.4	31.3	7.5
20	1.8538	1.0	22.7	13.0	2.2920	1.4	24.7	14.4	2.2248	1.1	30.4	8.4	2.2134	1.4	30.7	7.9
30	1.6081	0.8	18.1	13.5	2.1726	1.3	20.8	15.9	2.0282	1.1	29.4	9.1	2.1007	1.3	30.2	7.8
40	1.4259	0.5	14.8	13.4	2.0900	1.3	17.7	17.2	1.8690	0.6	28.3	9.6	1.9995	1.3	29.7	7.9
0	2.4314	1.4	31.6	7.4	KL true Insurance network (retrained): 0.5531											

Table 10
Average results obtained for Hailfinder using PC

%	G_e, G_a, G_o				Only G_e				Only G_a				Only G_o			
	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*	KL	*A*	*D*	*I*
10	8.2310	12.2	32.3	9.7	8.4367	11.1	32.8	10.4	8.9021	11.8	36.3	8.8	9.0700	10.9	36.6	9.0
20	7.7158	13.0	27.9	9.9	8.2728	11.3	28.8	11.4	8.6166	12.3	35.9	8.3	8.9622	11.2	36.3	8.8
30	7.4199	14.0	23.6	10.1	8.2550	11.5	24.9	12.6	8.2566	13.6	35.1	7.7	8.8277	11.6	35.9	8.3
40	7.4649	15.7	19.4	10.3	8.3642	11.7	20.9	13.5	8.0320	15.4	34.3	7.2	8.7295	11.7	35.5	7.8
0	9.1548	10.7	36.8	9.4	KL true Hailfinder network (retrained): 1.1609											

In this case, the use of the restrictions always gives rise to better network structures than the unrestricted PC, from the point of view of the KL divergence. With respect to the structural differences, all the types of restrictions decrease the number of deleted arcs (possibly this is due to the smaller number of independence tests carried out). However, for the same reason the number of added arcs tends to increase (except in the case of Insurance). The number of inverted arcs tends to decrease (once again except in the case of Insurance).

8. Concluding remarks

We have formally defined three types of structural restrictions for Bayesian networks, namely existence, absence and ordering restrictions, and studied their use in combination with BN learning algorithms that use score + search methods and independence tests. We have illustrated it for the specific cases of a local search learning algorithm and the PC

algorithm. The experimental results show that the use of additional knowledge in the form of restrictions may lead to improved network structures (usually in less time). For future work we plan to study the use of restrictions within score + search based learning algorithms that do not search directly in the DAG space [2,17]. We would also like to study another type of restrictions, namely conditional independence relationships between pairs of variables that should be true.

Acknowledgments

This work has been supported by the Spanish Junta de Comunidades de Castilla-La Mancha and Ministerio Educación y Ciencia, under Projects PBC-02-002 and TIN2004-06204-C03-02, respectively.

Appendix

We include here the proofs of all the propositions stated in the paper.

Proof of Proposition 2. (a) *Necessary condition*: We know that a DAG G exists which is consistent with the restrictions. If the graph G_e has some directed cycle, as all the arcs in G_e must also be arcs in G, then G will contain a directed cycle too, which contradicts the fact that G is a DAG.

Sufficient condition: We know that G_e has no directed cycle. Starting from G_e we shall build a graph $G = (\mathcal{V}, E_G)$ as follows: $\forall x, y \in \mathcal{V}$, if $x \to y \in E_e$ then $x \to y \in E_G$; if $x \to y \notin E_e$, $y \to x \notin E_e$ and $x—y \notin E_e$ then $x \to y \notin E_G$ and $y \to x \notin E_G$; if $x—y \in E_e$ then we include in E_G either the arc $x \to y$ or the arc $y \to x$. This graph G is obviously consistent with the restrictions. We can always select at least one orientation of the edges $x—y$ that does not generate a directed cycle and hence G will be a DAG: if the arc $x \to y$ generates a directed cycle this is because there was a directed path from y to x before including this arc; if the arc $y \to x$ also generates a directed cycle, then there was also a directed path from x to y. As we have a directed path from x to y and another directed path from y to x, we would have a directed cycle before introducing any arc.

(b) The empty graph $G_\emptyset$ always verifies the absence restrictions.

(c) *Necessary condition*: We know that there is a DAG G and a total ordering compatible with both G and G_o. If G_o is not a DAG, then it has a directed cycle, and then no total ordering can be compatible with G_o.

Sufficient condition: We know that G_o is a DAG. There is at least one total ordering compatible with G_o. Therefore, for the graph $G = G_o$ this ordering is obviously compatible with G and G_o.

Proof of Proposition 3. *Necessary condition*: We know that there exists a DAG G consistent with the restrictions. First, let us see that $G_{re} \cap G_a = G_\emptyset$:

$x \to y \in E_{re}$ if and only if either $x \to y \in E_e$ or $x—y \in E_e$ and $y \to x \in E_a$. In the first case, from $x \to y \in E_e$ we get $x \to y \in E_G$ and from this $x \to y \notin E_a$ and $x—y \notin E_a$. In the second case, from $x—y \in E_e$ we get either $x \to y \in E_G$ or $y \to x \in E_G$; if $x \to y \in E_G$ then we also obtain $x \to y \notin E_a$ and $x—y \notin E_a$; the second case, $y \to x \in E_G$, cannot happen because we also have that $y \to x \in E_a$. Therefore, we have that if $x \to y \in E_{re}$ then $x \to y \notin E_a$ and $x—$

$y \notin E_a$. On the other hand, $x—y \in E_{re}$ if and only if $x—y \in E_e$, $x \to y \notin E_a$ and $y \to x \notin E_a$. From $x—y \in E_e$ we obtain that either $x \to y \in E_G$ or $y \to x \in E_G$, and in any case this implies that $x—y \notin E_a$. So, we have that if $x—y \in E_{re}$ then $x \to y \notin E_a$, $y \to x \notin E_a$ and $x—y \notin E_a$. Therefore, $G_{re} \cap G_a = G_\emptyset$.

Now, let us see that $G_{re} \cup G_o$ has not directed cycles:

If $G_{re} \cup G_o$ has some directed cycle $x_1 \to x_2 \to \cdots \to x_k \to x_1$, then each one of these arcs belongs to either E_{re} or E_o. If $x_i \to x_{i+1} \in E_o$ then every total ordering σ compatible with G_o has to verify that $x_i <_\sigma x_{i+1}$. If $x_i \to x_{i+1} \in E_{re}$ then $x_i \to x_{i+1} \in E_G$ and again every total ordering σ compatible with G has to verify that $x_i <_\sigma x_{i+1}$. Consequently, every total ordering compatible with both G and G_o would have to verify that $x_1 <_\sigma x_2 <_\sigma \cdots <_\sigma x_k <_\sigma x_1$, which is obviously not possible. Therefore, $G_{re} \cup G_o$ has not directed cycles.

Sufficient condition: We know that $G_{re} \cap G_a = G_\emptyset$ and $G_{re} \cup G_o$ has not directed cycles. As $G_{re} \cup G_o$ is a graph without directed cycles we can give direction to the edges in this graph to get a DAG. Let us select only the arcs from this completed DAG that come from G_{re}, and we obtain a new subDAG $G = (\mathcal{V}, E_G)$. Obviously G satisfies the existence restrictions. Let us see that G also verifies the absence and ordering restrictions: If $x \to y \in E_a$ then $x \to y \notin E_{re}$ and $x—y \notin E_{re}$, hence $x \to y \notin E_G$. If $x—y \in E_a$ then $x—y \notin E_{re}$, $x \to y \notin E_{re}$ and $y \to x \notin E_{re}$, hence $x \to y \notin E_G$ and $y \to x \notin E_G$, and G satisfies the absence restrictions. As $G \cup G_o$ is a DAG then there exists an ordering compatible with $G \cup G_o$. This ordering is clearly compatible with both G and G_o, and then G satisfies the ordering restrictions.

Proof of Proposition 4. *Necessary condition*: We know that G is consistent with the restrictions. First let us prove that $G \cup G_e = G$. If $x \to y \in E_G \cup E_e$ then either $x \to y \in E_G$ or $x \to y \in E_e$. In the second case we also deduce that $x \to y \in E_G$. If $x—y \in E_G \cup E_e$ then $x—y \in E_e$, $x \to y \notin E_G$ and $y \to x \notin E_G$, but this situation is not possible because G is consistent with G_e. Therefore, we obtain that $G \cup G_e \subseteq G$, and obviously $G \subseteq G \cup G_e$.

Now, let us prove that $G \cap G_a = G_\emptyset$. If $x \to y \in E_a$ then $x \to y \notin E_G$. If $x—y \in E_a$ then $x \to y \notin E_G$ and $y \to x \notin E_G$. Hence $G \cap G_a = G_\emptyset$.

Finally, let us prove that $G \cup G_o$ is a DAG. In case that $G \cup G_o$ is not a DAG, there exists a directed cycle $x_1 \to x_2 \to \cdots \to x_k \to x_1$, each one of these arcs belonging to $E_G \cup E_o$. Then, as in Proof of Proposition 3 to prove that $G_{re} \cup G_o$ had not directed cycles, we can deduce that every total ordering σ compatible with both G and G_o would have to verify that $x_1 <_\sigma x_2 <_\sigma \cdots <_\sigma x_k <_\sigma x_1$.

Sufficient condition: We know that $G \cup G_e = G$, $G \cap G_a = G_\emptyset$ and $G \cup G_o$ is a DAG. As $E_G \cup E_e = E_G$, if $x \to y \in E_e$ then $x \to y \in E_G$; if $x—y \in E_e$ then either $x \to y \in E_G$ or $y \to x \in E_G$. Therefore, G is consistent with the existence restrictions.

As $E_a \cap E_G = \emptyset$, if $x \to y \in E_a$ then $x \to y \notin E_G$; if $x—y \in E_a$ then $x \to y \notin E_G$ and $y \to x \notin E_G$, and G is consistent with the absence restrictions.

As $G \cup G_o$ is a DAG, there exists a total ordering σ compatible with $G \cup G_o$. Then this ordering is compatible with both G and G_o.

Proof of Proposition 5. The proof is quite similar to that of Propositions 3 and 4, so that we shall omit the details. The justification to use H_r instead of H is that, as it happened with the existence restrictions G_e in Proposition 3, the edges in H may interact with the arcs in G_a (for example, if $E_a = \{x_1 \to x_2\}$ and $E_H = \{x_1—x_2\}$, then $G_a \cap H \neq G_\emptyset$ because $\{x_1 \to x_2\} \cap \{x_1—x_2\} = \{x_1 \to x_2\}$). The edges of H whose orientation becomes forced

because of the arcs in G_a can also interact with the arcs in G_o (for example, if $E_a = \{x_1 \to x_2, x_2 \to x_3\}$ and $E_o = \{x_1 \to x_3\}$, then the graph $H = (\mathscr{V}, E_H)$ with $E_H = \{x_1 — x_2, x_2 — x_3\}$, verifies that $H \cup G_o$ has not directed cycles although H is not consistent with the restrictions, whereas $H_r \cup G_o$ has a directed cycle).

Proof of Proposition 6. (a) *Necessary condition*: We know that G' is consistent with the restrictions. As $E'_G \cap E_a = \emptyset$, then $(E_G \cup \{x \to y\}) \cap E_a = \emptyset$. Therefore, $\{x \to y\} \cap E_a = \emptyset$ and this means that $x \to y \notin E_a$ and $x — y \notin E_a$. On the other hand, if $G \cup G_o$ had a directed path from y to x, then we would have a cycle in $G' \cup G_o$, and then $G' \cup G_o$ would not be a DAG.

Sufficient condition: We know that G is consistent with the restrictions.

As $E_G \cup E_e = E_G$, then $E'_G \cup E_e = (E_G \cup \{x \to y\}) \cup E_e = E_G \cup \{x \to y\} = E'_G$ and therefore $G' \cup G_e = G'$.

As $E_G \cap E_a = \emptyset$, then $E'_G \cap E_a = (E_G \cup \{x \to y\}) \cap E_a = \{x \to y\} \cap E_a$, and this last intersection is empty because $x \to y \notin E_a$ and $x — y \notin E_a$. Therefore, $G' \cap G_a = G_\emptyset$.

Finally, as $G \cup G_o$ is a DAG and there is not a directed path from y to x in this graph, then the graph $G' \cup G_o$ obtained from $G \cup G_o$ by including the arc $x \to y$ is a DAG.

(b) *Necessary condition*: We know that G and G' are consistent with the restrictions. As $G' \cup G_e = G'$, then $E'_G \cup E_e = E'_G$, i.e. $(E_G \backslash \{x \to y\}) \cup E_e = E_G \backslash \{x \to y\}$. In case that $x \to y \in E_e$ then we would have $(E_G \backslash \{x \to y\}) \cup E_e = E_G \cup E_e = E_G$. In case that $x — y \in E_e$ then $(E_G \backslash \{x \to y\}) \cup E_e = ((E_G \cup E_e) \backslash \{x \to y\}) \cup \{x — y\} = (E_G \backslash \{x \to y\}) \cup \{x — y\}$, and both cases contradict the hypothesis.

Sufficient condition: As $E_G \cap E_a = \emptyset$ then $E'_G \cap E_a = (E_G \setminus \{x \to y\}) \cap E_a = \emptyset$, and $G' \cap G_a = G_\emptyset$.

As $E_G \cup E_e = E_G$, $x \to y \notin E_e$ and $x — y \notin E_e$, then $E'_G \cup E_e = (E_G \setminus \{x \to y\}) \cup E_e = (E_G \cup E_e) \setminus \{x \to y\} = E_G \setminus \{x \to y\} = E'_G$. Thus, $G' \cup G_e = G'$.

Finally, as $G \cup G_o$ is a DAG and $G' \cup G_o$ is a subgraph, it is also a DAG.

(c) *Necessary and sufficient conditions*: Reversing an arc $x \to y$ can be seen as first deleting it and after inserting the arc $y \to x$. Then, by applying the conditions for deleting and inserting we would have $x \to y \notin E_e$, $x — y \notin E_e$, $y \to x \notin E_a$, $x — y \notin E_a$ and there is not any directed path from x to y in $(G \backslash \{x \to y\}) \cup G_o$. However, the condition $x — y \notin E_e$ is not necessary, because we are not eliminating the arc $x \to y$ but replacing it by $y \to x$. The condition $x — y \notin E_a$ will be always true, because $x \to y$ was in G and G is consistent. Finally, the condition stating that there is not any directed path from x to y in $(G \backslash \{x \to y\}) \cup G_o$ is equivalent to say that there is not any directed path from x to y in $(G \cup G_o) \backslash \{x \to y\}$ and that $x \to y \notin E_o$.

Proof of Proposition 7. Let $G_{re(x \to y)} = (\mathscr{V}, E_{re(x \to y)})$ be the refined graph of existence restrictions using $G_{e(x \to y)}$ instead of G_e. It is clear that $E_{re(x \to y)} = (E_{re} \backslash \{x — y\}) \cup \{x \to y\}$. As $x — y \in E_{re}$ we also know that $x \to y \notin E_a$ and $y \to x \notin E_a$. Then, according to Proposition 3, $G_{e(x \to y)}$, G_a and G_o will be self-consistent if and only if $G_{re(x \to y)} \cap G_a = G_\emptyset$ and $G_{re(x \to y)} \cup G_o$ has no directed cycle.

Let us assume first that $G_{e(x \to y)}$, G_a and G_o are self-consistent. If there is a directed path from y to x in $G_{re} \cup G_o$, all the arcs in this path are also in $G_{re(x \to y)} \cup G_o$ and, together with the arc $x \to y$, they form a directed cycle in $G_{re(x \to y)} \cup G_o$, which contradicts the hypothesis.

Now, let us assume that there is not any directed path from y to x in $G_{re} \cup G_o$. As G_e, G_a and G_o are self-consistent, we also know that $G_{re} \cap G_a = G_\emptyset$ and $G_{re} \cup G_o$ has no directed cycle.

$E_{re(x\to y)} \cap E_a = ((E_{re} \setminus \{x—y\}) \cup \{x \to y\}) \cap E_a = ((E_{re} \setminus \{x—y\}) \cap E_a) \cup (\{x \to y\} \cap E_a) = \emptyset$. Therefore, $G_{re(x\to y)} \cap G_a = G_\emptyset$.

As $G_{re} \cup G_o$ has no directed cycle, if after directing the edge $x—y$ as $x \to y$, in order to obtain $G_{re(x\to y)} \cup G_o$, we get a cycle, then there was a directed path from y to x in $G_{re} \cup G_o$, which contradicts the hypothesis. Therefore, we have that $G_{re(x\to y)} \cap G_a = G_\emptyset$ and $G_{re(x\to y)} \cup G_o$ has not a directed cycle, hence $G_{e(x\to y)}$, G_a and G_o are self-consistent.

Finally, let us prove that either $G_{e(x\to y)}$ or $G_{e(y\to x)}$, together with G_a and G_o are self-consistent. If we assume that this is not true, then there is a directed path from y to x and another from x to y in $G_{re} \cup G_o$, and this means that we have a directed cycle in $G_{re} \cup G_o$, which contradicts the fact that G_e, G_a and G_o are self-consistent.

Proof of Proposition 8. (a) and (b) The proof is completely similar to that of Proposition 6(b). The difference is that we use H_r instead of H, but we only have to take into account that H'_r is always a subgraph of H_r.

(d) Directing an edge $x—y$ is similar to inserting an arc $x \to y$, so that the conditions that assure the consistency are those in Proposition 6(a); the only difference is that the condition $x—y \notin E_a$ in Proposition 6(a) is not necessary (it is always true), since the edge $x—y$ is already in H and H is consistent with the restrictions.

(c) As in the previous case, creating a head-to-head pattern is similar to insert two arcs, so that the conditions for consistency are again those in Proposition 6(a) applied to the arcs $x \to z$ and $y \to z$; as before, as we know that there are links joining nodes x and z and nodes y and z in H, we do not need to check that $x—z \notin E_a$ and $y—z \notin E_a$, because, as H is consistent with the restrictions, these conditions will be true.

References

[1] B. Abramson, J. Brown, A. Murphy, R.L. Winkler, Hailfinder: A Bayesian system for forecasting severe weather, International Journal of Forecasting 12 (1996) 57–71.

[2] S. Acid, L.M. de Campos, Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs, Journal of Artificial Intelligence Research 18 (2003) 445–490.

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

[4] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning 29 (1997) 213–244.

[5] R. Blanco, I. Inza, P. Larrañaga, Learning Bayesian networks in the space of structures by estimation of distribution algorithms, International Journal of Intelligent Systems 18 (2003) 205–220.

[6] R.R. Bouckaert, Bayesian belief networks: from construction to inference, Ph.D. thesis, University of Utrecht, 1995.

[7] W. Buntine, Theory refinement of Bayesian networks, in: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, 1991, pp. 52–60.

[8] J. Cheng, R. Greiner, Comparing Bayesian network classifiers, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, 1999, pp. 101–108.

[9] J. Cheng, R. Greiner, J. Kelly, D.A. Bell, W. Liu, Learning Bayesian networks from data: an information-theory based approach, Artificial Intelligence 137 (2002) 43–90.

[10] D.M. Chickering, A transformational characterization of equivalent Bayesian network structures, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 1995, pp. 87–98.
[11] D.M. Chickering, Learning equivalence classes of Bayesian network structures, Journal of Machine Learning Research 2 (2002) 445–498.
[12] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (1992) 309–348.
[13] L.M. de Campos, J.M. Fernández-Luna, J.A. Gámez, J.M. Puerta, Ant colony optimization for learning Bayesian networks, International Journal of Approximate Reasoning 31 (2002) 291–311.
[14] L.M. de Campos, J.M. Fernández-Luna, J.M. Puerta, Local search methods for learning Bayesian networks using a modified neighborhood in the space of dags, Lecture Notes in Computer Science 2527 (2002) 182–192.
[15] L.M. de Campos, J.F. Huete, A new approach for learning belief networks using independence criteria, International Journal of Approximate Reasoning 24 (2000) 11–37.
[16] L.M. de Campos, J.M. Puerta, Stochastic local and distributed search algorithms for learning belief networks, in: Proceedings of the III International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Model, 2001, pp. 109–115.
[17] L.M. de Campos, J.M. Puerta, Stochastic local search algorithms for learning belief networks: Searching in the space of orderings, Lecture Notes in Artificial Intelligence 2143 (2001) 228–239.
[18] Elvira Consortium, Elvira: an environment for probabilistic graphical models, in: J.A. Gámez, A. Salmerón (Eds.), Proceedings of the First European Workshop on Probabilistic Graphical Models, 2002, pp. 222–230.
[19] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, Machine Learning 20 (1995) 197–243.
[20] T. Kocka, R. Castelo, Improved learning of Bayesian networks, in: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, 2001, pp. 269–276.
[21] W. Lam, F. Bacchus, Learning Bayesian belief networks. An approach based on the MDL principle, Computational Intelligence 10 (1994) 269–293.
[22] P. Langley, S. Sage, Induction of selective Bayesian classifiers, in: Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, 1994, pp. 399–406.
[23] P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, C. Kuijpers, Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (1996) 912–926.
[24] P. Larrañaga, C. Kuijpers, R. Murga, Learning Bayesian network structures by searching for the best ordering with genetic algorithms, IEEE Transactions on System, Man and Cybernetics 26 (1996) 487–493.
[25] A.L. Madsen, M. Lang, U.B. Kjaerulff, F. Jensen, The Hugin tool for learning Bayesian networks, Lecture Notes in Artificial Intelligence 2711 (2003) 594–605.
[26] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufman, San Mateo, 1988.
[27] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction and SearchLecture Notes in Statistics, 81, Springer-Verlag, New York, 1993.