# On the Use of Restrictions for Learning Bayesian Networks

Luis M. de Campos and Javier G. Castellano

Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática, Universidad de Granada, 18071 – Granada, Spain
{lci, fjgc}@decsai.ugr.es

**Abstract.** In this paper we explore the use of several types of structural restrictions within algorithms for learning Bayesian networks. These restrictions may codify expert knowledge in a given domain, in such a way that a Bayesian network representing this domain should satisfy them. Our objective is to study whether the algorithms for automatically learning Bayesian networks from data can benefit from this prior knowledge to get better results. We formally define three types of restrictions: existence of arcs and/or edges, absence of arcs and/or edges, and ordering restrictions, and also study their interactions and how they can be managed within Bayesian network learning algorithms based on the score+search paradigm. Then we particularize our study to the classical local search algorithm with the operators of arc addition, arc removal and arc reversal, and carry out experiments using this algorithm on several data sets.

## 1 Introduction

Nowadays, Bayesian networks [15] 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), and a quantitative one, a collection of numerical parameters, usually conditional probability tables. There has been a lot of work in recent years on the automatic learning of Bayesian networks from data and, consequently, there are a great many learning algorithms, 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 defining several types of restrictions, that codify some kinds of expert knowledge, to 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 [13]), 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. 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 DAG. 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 discusses the experimental results. Finally, Section 7 contains the concluding remarks.

## 2 Notation and Preliminaries

Let us consider a finite set $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$ 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) $G = (\mathcal{V}, E_G)$, where $E_G$ represents the set of arcs.

The problem of learning the structure of a BN from data is that given a training set $\mathcal{D}$ of instances of the variables in $\mathcal{V}$, find the network that, in some sense, best matches $\mathcal{D}$. 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 (for references, see [2]). In this paper we are more interested in the algorithms based on the score+search paradigm, which attempt to find a graph that maximizes the selected score. All use a scoring function, 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 but the same search space: the space of DAGs[1]. Our objective is to narrow this (hyper-exponential) search space 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 DAG structures defined for the domain $\mathcal{V}$, namely existence, absence and ordering restrictions.

### 3.1 Existence or Arcs and/or Edges

Consider two subsets of pairs of variables $\mathcal{E}_a, \mathcal{E}_e \subseteq \mathcal{V} \times \mathcal{V}$, with $\mathcal{E}_a \cap \mathcal{E}_e = \emptyset$. They will be interpreted as follows:

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

– $(x, y) \in \mathcal{E}_a$: the arc $x \rightarrow y$ must belong to any DAG in the search space.
– $(x, y) \in \mathcal{E}_e$: the edge (i.e. the arc without direction) $x$—$y$ must belong to any DAG in the search space. In other words, either the arc $x \rightarrow y$ or the arc $y \rightarrow x$ must appear in any DAG.

An example of the use of existence restrictions may be any BAN algorithm [5], a BN learning algorithm for classification, which fixes the naive Bayes structure (i.e. arcs from the class variable to all the attribute variables) and searches for the appropriate additional arcs, linking pairs of attribute variables.

### 3.2 Absence of Arcs and/or Edges

Now, consider the subsets $\mathcal{A}_a, \mathcal{A}_e \subseteq \mathcal{V} \times \mathcal{V}$, with $\mathcal{A}_a \cap \mathcal{A}_e = \emptyset$. Their meaning is the following:

– $(x, y) \in \mathcal{A}_a$: the arc $x \rightarrow y$ cannot be present in any DAG in the search space.
– $(x, y) \in \mathcal{A}_e$: the edge $x$—$y$ cannot appear in any DAG in the search space (i.e. neither the arc $x \rightarrow y$ nor the arc $y \rightarrow x$ can appear).

An example of the use of absence restrictions is a selective naive Bayesian classifier [14], which forbids arcs between attribute variables and also arcs from the attributes to the class variable.

### 3.3 Partial Ordering

Consider the subset $\mathcal{R}_o \subseteq \mathcal{V} \times \mathcal{V}$. In this case the interpretation is:

– $(x, y) \in \mathcal{R}_o$: all the DAGs in the search space have to satisfy that $x$ precedes $y$ in some total ordering of the variables compatible with the DAG structure.

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

$$\forall x, y \in \mathcal{V}, \text{ if } x <_\mu y \text{ then } x <_\sigma y\,,$$

i.e. if $x$ precedes $y$ in the ordering $\mu$ then also $x$ precedes $y$ in the ordering $\sigma$. Notice that a DAG determines a partial ordering on its variables: if there is a directed path from $x$ to $y$ in a DAG $G$, then $x$ precedes $y$. Therefore, we can also say that a total ordering $\sigma$ on the set $\mathcal{V}$ is compatible with a DAG $G = (\mathcal{V}, E)$ if

$$\forall x, y \in \mathcal{V}, \text{ if } x \rightarrow y \in E \text{ then } x <_\sigma y\,.$$

The ordering restrictions may represent, for example, temporal or functional precedence between variables. Notice that the restriction $(x, y) \in \mathcal{R}_o$ also means that there is not a directed path from $y$ to $x$ in any of the DAGs in the search space. Examples of use of ordering restrictions are the BN learning algorithms that require a fixed total ordering of the variables (as the K2 algorithm [7]).

## 4 Representing and Managing the Restrictions

In order to manage the restrictions it is useful to represent them graphically. So, the existence restrictions can be represented by means of a partially directed graph $G_e = (\mathcal{V}, E_e)$, where each element $(x, y)$ in $\mathcal{E}_a$ is associated with the corresponding arc $x \rightarrow y \in E_e$, and each element $(x, y)$ in $\mathcal{E}_e$ is associated with the edge $x$—$y \in E_e$. The absence restrictions are represented by means of another partially directed graph $G_a = (\mathcal{V}, E_a)$, where the elements $(x, y)$ in $\mathcal{A}_a$ correspond with arcs $x \rightarrow y \in E_a$ and the elements $(x, y)$ in $\mathcal{A}_e$ are associated with edges $x$—$y \in E_a$. Finally, the ordering restrictions are represented by using a directed graph $G_o = (\mathcal{V}, E_o)$, with $(x, y)$ in $\mathcal{R}_o$ being associated with the arc $x \rightarrow y \in E_o$. 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 $\mathcal{R}_o$. $G_o$ may be any graph such that its transitive closure contains an arc for each element in $\mathcal{R}_o$. For example, to represent a total ordering restriction $x_1 < x_2 < \ldots < x_n$ it suffices to include in $G_o$ the $n-1$ arcs $x_i \rightarrow x_{i+1}$, $i = 1, \ldots, n-1$, instead of a having a complete graph with all the arcs $x_i \rightarrow x_j$, $\forall i < j$.

Now, let us formally define when a given DAG $G$ is consistent with a set of restrictions (i.e. $G$ verifies them):

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

- *$G$ is consistent with the existence restrictions if and only if*
  - *$\forall x, y \in \mathcal{V}$, if $x \rightarrow y \in E_e$ then $x \rightarrow y \in E$, and*
  - *$\forall x, y \in \mathcal{V}$, if $x$—$y \in E_e$ then $x \rightarrow y \in E$ or $y \rightarrow x \in E$.*
- *$G$ is consistent with the absence restrictions if and only if*
  - *$\forall x, y \in \mathcal{V}$, if $x \rightarrow y \in E_a$ then $x \rightarrow y \notin E$, and*
  - *$\forall x, y \in \mathcal{V}$, if $x$—$y \in E_a$ then $x \rightarrow y \notin E$ and $y \rightarrow x \notin E$.*
- *$G$ is consistent with the ordering restrictions if and only if*
  - *there exists a total ordering $\sigma$ of the variables in $\mathcal{V}$ compatible with both $G$ and $G_o$.*

Before using a set of restrictions we must be sure that we are not demanding conditions impossible to satisfy. 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 type of restriction separately is very simple[2]:

**Proposition 1.** *Let $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ and $G_o = (\mathcal{V}, E_o)$ be the graphs representing existence, absence and ordering restrictions, respectively. Then*

- *The set of existence restrictions is self-consistent if and only if the graph $G_e$ has no directed cycle.*

[2] The proofs of this and all the other propositions stated in the paper are not given, because of their relative simplicity and space limitations.

– *The set of absence restrictions is always self-consistent.*
– *The set of ordering restrictions is self-consistent if and only if $G_o$ is a DAG.*

When several types of 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; or the existence of some arcs that (as they implicitly also represent partial ordering restrictions) may contradict with 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 also may happen 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 an absence or 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 $y \to t$, $t \to x$, $x—z$, $z—y \in E_e$, $x \to z \in E_o$ and $y \to z \in E_a$, the absence and ordering restrictions force the orientation of the edges $x—z$ and $z—y$ 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 2.** *Let $G_e = (\mathcal{V}, E_e)$, $G_a = (\mathcal{V}, E_a)$ and $G_o = (\mathcal{V}, E_o)$ be the graphs representing existence, absence and ordering restrictions, respectively. Let $G_{re} = (\mathcal{V}, E_{re})$ be the refined graph of existence restrictions*[3] *defined as*

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

*Then the three sets of restrictions are self-consistent if and only if*

$$G_{re} \cap G_a = G_\emptyset \text{ and } G_{re} \cup G_o \text{ has no directed cycle,}$$

*where $G_\emptyset$ is the empty graph (a graph having neither arcs nor edges), and both the union and the intersection of two partially directed graphs use the convention that $\{x \to y\} \cup \{x—y\} = \{x \to y\}$ and $\{x \to y\} \cap \{x—y\} = \{x \to y\}$.*

Testing the consistency of a DAG with a set of restrictions can also be reduced to simple graph operations, as the following result shows:

**Proposition 3.** *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 $G = (\mathcal{V}, E)$ a DAG. Then $G$ is consistent with the restrictions if and only if*

$$G \cup G_e = G,\ G \cap G_a = G_\emptyset \text{ and } G \cup G_o \text{ is a DAG.}$$

[3] This is the same graph $G_e$ with the edges whose direction is forced by virtue of some absence restriction being replaced by the corresponding arcs.

## 5 Using the Restrictions for Learning

In case that we want to get a Bayesian network from data using a score+search learning algorithm and we have a set of (self-consistent) restrictions, 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, 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 the previous proposition 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 [13], which uses the operators of arc insertion, arc deletion and arc reversal. 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 operators. Let us see which are the conditions necessary and sufficient to assure that $G'$ is also consistent with the restrictions.

**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 $G = (\mathcal{V}, E)$ a DAG consistent with the restrictions.*

*(a) Arc insertion: Let $G' = (\mathcal{V}, E')$, $E' = E \cup \{x \rightarrow y\}$, with $x \rightarrow y \notin E$. Then $G'$ is consistent with the restrictions if and only if*
- *$x \rightarrow y \notin E_a$ 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 $G' = (\mathcal{V}, E')$, $E' = E \setminus \{x \rightarrow y\}$, with $x \rightarrow y \in E$. Then $G'$ is consistent with the restrictions if and only if*
- *$x \rightarrow y \notin E_e$ and $x — y \notin E_e$.*

*(c) Arc reversal: Let $G' = (\mathcal{V}, E')$, $E' = (E \setminus \{x \rightarrow y\}) \cup \{y \rightarrow x\}$, with $x \rightarrow y \in E$. Then $G'$ is consistent with the restrictions if and only if*
- *$x \rightarrow y \notin E_e$, $y \rightarrow x \notin E_a$ and $x \rightarrow y \notin E_o$,*
- *if we exclude the arc $x \rightarrow y$, 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: 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 basic operators, as variable neighborhood search [10], tabu search [2] or GRASP[4] [9],

[4] Greedy Randomized Adaptive Search Procedure.

or even a subset of them (arc insertion), as ant colony optimization [8]. These algorithms can be used together with the restrictions with almost no additional modification.

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 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 $G_{re} = (\mathcal{V}, E_{re})$ be the refined graph of existence restrictions. Let $x$—$y \in E_{re}$ and define the graph $G_{e(x \to y)} = (\mathcal{V}, (E_e \setminus \{x—y\}) \cup \{x \to y\})$. Then $G_{e(x\to y)}$, $G_a$ and $G_o$ are still 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 into one verifying the restrictions. This method can also be useful for learning algorithms using population-based search processes (as genetic algorithms and EDAs).

## 6 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. We have selected four different problems. The Alarm network (left hand side of Figure 1) 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 (Figure 2) 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 Figure 1) 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.

For Alarm, the input data commonly used are subsets of a standard database containing 20000 cases. In our experiments, we have used a subset containing

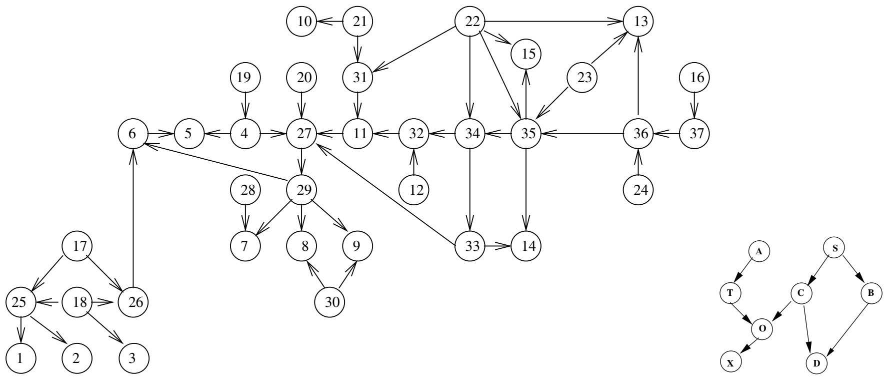

**Fig. 1.** The Alarm (left) and the Asia (right) networks

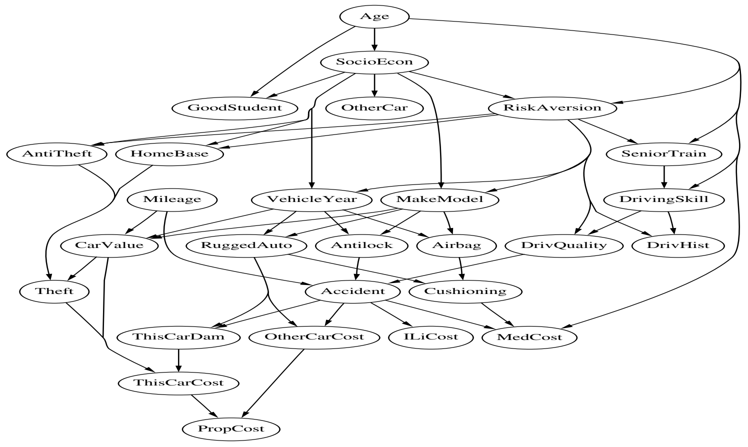

**Fig. 2.** The Insurance network

the first 10000 cases. In each of the other three problems, a database containing 10000 cases generated from the corresponding network has been used.

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 [13], with the parameter representing the equivalent sample size set to 1 and a uniform structure prior. The collected performance measures are the scoring value of the obtained network (BDeu) and three measures of the structural difference between the learned network and the true one: 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 true network. To eliminate fictitious differences or similarities between the two networks, caused by different but equivalent subDAG structures, before comparing the two networks we have converted them to their corresponding completed PDAG (also called essential graph) representation[5], using the algorithm proposed in [6]. The percentages of running time of the algorithm using restrictions (T) with respect to the running time of the algorithm without using them have also been computed. All the implementations have been carried out within the Elvira System [12], a Java tool to construct probabilistic decision support systems, which works with Bayesian networks and influence diagrams.

For each dataset 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 $G = (\mathcal{V}, E)$ is the true network, then each arc $x \rightarrow y \in E$ is a possible existence restriction (we may select the restriction $x \rightarrow y \in E_e$ if this arc is also present in the completed PDAG representation of $G$; otherwise we would use the restriction $x—y \in E_e$); each arc $x \rightarrow y \notin E$ is a possible absence restriction (in case that also $y \rightarrow x \notin E$ we randomly select whether to use the restriction $x \rightarrow y \in E_a$ or $x—y \in E_a$); finally, if there is a directed path from $x$ to $y$ in completed PDAG representation of $G$ then $x \rightarrow y \in E_o$ is a possible ordering restriction. The selected percentages have been 10%, 20%, 30% and 40%. We have run the learning algorithm for each percentage of restrictions of each type alone, and also using the three types of restrictions together.

The results in Tables 1–4 represent the average values of the performance measures across 50 iterations (i.e. 50 random subsets of restrictions for each percentage and each dataset). For comparative purposes, these tables display also the results obtained by the learning algorithm without using restrictions (0%), its running time and the scoring value of the true network.

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 observed in the results. Moreover, another less obvious effect, almost systematically observed in the experiments (except in Asia), 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 scoring function, we have to distinguish Hailfinder from the other three datasets, the reason being that in the first case the learning algorithm, without using restrictions, finds a network with a score much better than the true Hailfinder network. The true Insurance network is also worse in score than the learned one but at a much lesser extend, whereas the true Asia and Alarm networks are better than the learned ones. This is important because the use of restrictions tries to guide the search process towards the true network. On the one hand, in the last three cases, both the existence and the ordering restrictions lead to better network

[5] 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.

**Table 1.** Results obtained for Asia

| | $G_e$, $G_a$, $G_o$ | | | | | only $G_e$ | | | | | only $G_a$ | | | | | only $G_o$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T |
| 10% | -2258.48 | 1.8 | 0.9 | 1.3 | 76 | -2257.42 | 1.8 | 0.8 | 2.4 | 76 | -2260.59 | 1.8 | 1.0 | 2.2 | 76 | -2257.65 | 1.9 | 1.0 | 2.4 | 77 |
| 20% | -2256.95 | 1.5 | 0.8 | 0.2 | 56 | -2256.94 | 1.8 | 0.8 | 1.6 | 57 | -2260.34 | 1.6 | 1.1 | 1.1 | 56 | -2257.37 | 1.9 | 1.1 | 1.9 | 58 |
| 30% | -2256.71 | 1.1 | 0.5 | 0.0 | 43 | -2256.69 | 1.9 | 0.5 | 0.8 | 44 | -2258.96 | 1.4 | 1.1 | 0.7 | 43 | -2256.76 | 2.0 | 1.1 | 1.0 | 42 |
| 40% | -2256.87 | 0.7 | 0.5 | 0.0 | 28 | -2256.61 | 1.9 | 0.5 | 0.4 | 29 | -2260.00 | 0.9 | 1.0 | 0.4 | 28 | -2256.59 | 2.0 | 1.1 | 0.8 | 30 |
| 0% | -2257.90 | 2 | 1 | 3 | | running time: 0.51 sec. | | | | | | BDeu true network: -2257.55 | | | | | | | | |

**Table 2.** Results obtained for Alarm

| | $G_e$, $G_a$, $G_o$ | | | | | only $G_e$ | | | | | only $G_a$ | | | | | only $G_o$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T |
| 10% | -108551 | 1.6 | 1.2 | 1.4 | 77 | -108666 | 3.0 | 1.5 | 2.0 | 78 | -108773 | 4.1 | 1.7 | 2.4 | 77 | -108758 | 4.1 | 1.7 | 3.1 | 78 |
| 20% | -108550 | 0.9 | 1.0 | 1.0 | 60 | -108613 | 2.4 | 1.4 | 2.3 | 61 | -108806 | 3.4 | 1.7 | 2.5 | 60 | -108739 | 3.7 | 1.3 | 2.5 | 61 |
| 30% | -108513 | 0.3 | 0.8 | 0.4 | 46 | -108562 | 1.8 | 1.0 | 1.9 | 47 | -108788 | 2.6 | 1.5 | 2.3 | 46 | -108718 | 3.3 | 1.1 | 2.1 | 47 |
| 40% | -108504 | 0.2 | 0.7 | 0.3 | 33 | -108486 | 0.9 | 0.8 | 2.0 | 35 | -108782 | 1.8 | 1.3 | 1.8 | 34 | -108675 | 2.8 | 1.1 | 1.9 | 35 |
| 0% | -108828 | 5 | 2 | 3 | | running time: 2.53 min. | | | | | | BDeu true network: -108452 | | | | | | | | |

**Table 3.** Results obtained for Insurance

| | $G_e$, $G_a$, $G_o$ | | | | | only $G_e$ | | | | | only $G_a$ | | | | | only $G_o$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T |
| 10% | -132369 | 3.1 | 8.7 | 9.0 | 79 | -132406 | 4.2 | 8.8 | 10.7 | 79 | -132410 | 4.6 | 9.7 | 10.6 | 79 | -132417 | 5.3 | 10.0 | 10.5 | 79 |
| 20% | -132271 | 1.4 | 7.4 | 6.4 | 58 | -132429 | 3.4 | 8.0 | 8.5 | 59 | -132434 | 3.0 | 9.3 | 8.5 | 59 | -132362 | 4.6 | 9.8 | 9.4 | 59 |
| 30% | -132225 | 0.5 | 6.0 | 4.4 | 42 | -132313 | 1.8 | 6.4 | 6.8 | 43 | -132509 | 2.5 | 9.3 | 8.4 | 43 | -132309 | 3.8 | 9.8 | 8.4 | 44 |
| 40% | -132233 | 0.2 | 5.1 | 3.8 | 31 | -132409 | 1.6 | 5.7 | 5.2 | 32 | -132308 | 1.4 | 8.9 | 6.4 | 31 | -132257 | 3.3 | 9.5 | 7.5 | 33 |
| 0% | -132488 | 6 | 10 | 11 | | running time: 1.60 min. | | | | | | BDeu true network: -132512 | | | | | | | | |

structures. For Hailfinder, the convergence towards the true network results in worse networks. On the other hand, the use of absence restrictions seems to be self-defeating: the obtained networks frequently are worse in score than the one obtained without using restrictions. 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[6], 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 that perhaps it is not a good idea to limit the search space using absence restrictions. Instead, 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.

Finally, with respect to the efficiency of the learning algorithm, it can be observed that the running times decrease considerably when using the restrictions, these times being progressively lesser as the number of restrictions increases.

In order to test the behavior of the restrictions in more realistic situations, where the number of available cases is much smaller (and therefore the expert knowledge that the restrictions represent is less probable to be already embedded in the data), we have also carried out experiments (20 iterations) with data sets containing only 500 cases. The results are displayed in Table 5. We can observe,

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

**Table 4.** Results obtained for Hailfinder

| | $G_e$, $G_a$, $G_o$ | | | | | only $G_e$ | | | | | only $G_a$ | | | | | only $G_o$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T | BDeu | A | D | I | T |
| 10% | -498306 | 13.1 | 10.0 | 11.6 | 80 | -497977 | 14.8 | 10.4 | 15.2 | 80 | -498146 | 16.1 | 11.5 | 17.3 | 80 | -498245 | 17.4 | 12.7 | 15.8 | 81 |
| 20% | -498354 | 10.1 | 8.4 | 4.7 | 64 | -498098 | 13.2 | 9.1 | 8.7 | 64 | -498475 | 14.2 | 11.2 | 14.3 | 64 | -498444 | 17.1 | 12.7 | 12.5 | 65 |
| 30% | -498424 | 7.0 | 6.6 | 3.2 | 51 | -498219 | 12.1 | 8.0 | 5.5 | 52 | -498535 | 11.2 | 10.4 | 12.4 | 51 | -498726 | 16.8 | 12.6 | 9.5 | 53 |
| 40% | -498550 | 5.0 | 5.4 | 1.0 | 41 | -498347 | 11.0 | 6.9 | 3.7 | 42 | -498516 | 8.5 | 9.3 | 8.9 | 41 | -498722 | 15.9 | 12.5 | 7.5 | 43 |
| 0% | -497904 | 17 | 12 | 19 | | running time: 6.53 min. | | | | | | | BDeu true network: -503095 | | | | | | | |

**Table 5.** Results obtained using data sets with only 500 cases

| | $G_e$, $G_a$, $G_o$ | | | | only $G_e$ | | | | only $G_a$ | | | | only $G_o$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | BDeu | A | D | I | BDeu | A | D | I | BDeu | A | D | I | BDeu | A | D | I |
| | | | | | | | | Asia | | | | | | | | |
| 10% | -1075.94 | 0.0 | 0.8 | 0.6 | -1075.94 | 0.0 | 0.8 | 0.6 | -1075.36 | 0.0 | 1.0 | 0.0 | -1075.36 | 0.0 | 1.0 | 0.0 |
| 20% | -1075.74 | 0.0 | 0.6 | 0.3 | -1075.87 | 0.0 | 0.6 | 0.4 | -1075.36 | 0.0 | 1.0 | 0.0 | -1075.36 | 0.0 | 1.0 | 0.0 |
| 30% | -1075.75 | 0.0 | 0.6 | 0.3 | -1075.75 | 0.0 | 0.6 | 0.3 | -1075.36 | 0.0 | 1.0 | 0.0 | -1075.36 | 0.0 | 1.0 | 0.0 |
| 40% | -1075.51 | 0.0 | 0.6 | 0.0 | -1075.51 | 0.0 | 0.6 | 0.0 | -1075.36 | 0.0 | 1.0 | 0.0 | -1075.36 | 0.0 | 1.0 | 0.0 |
| 0% | -1075.36 | 0 | 1 | 0 | BDeu true network: -1075.69 | | | | | | | | | | | |
| | | | | | | | | Alarm | | | | | | | | |
| 10% | -5990 | 9.7 | 4.0 | 12.7 | -5972 | 10.4 | 3.8 | 15.6 | -5998 | 11.0 | 5.0 | 19.3 | -6002 | 13.2 | 4.7 | 14.6 |
| 20% | -5971 | 6.2 | 3.3 | 7.6 | -5959 | 9.1 | 3.2 | 12.1 | -5990 | 9.2 | 4.6 | 14.2 | -6005 | 12.4 | 5.0 | 12.2 |
| 30% | -5946 | 4.4 | 2.2 | 5.8 | -5949 | 7.7 | 2.6 | 10.2 | -5984 | 7.3 | 4.6 | 10.8 | -6003 | 11.3 | 4.9 | 10.2 |
| 40% | -5943 | 3.2 | 1.6 | 4.4 | -5946 | 7.1 | 2.0 | 8.4 | -5989 | 6.5 | 4.6 | 8.8 | -5986 | 9.8 | 4.6 | 8.5 |
| 0% | -5986 | 11 | 5 | 22 | BDeu true network: -5935 | | | | | | | | | | | |
| | | | | | | | | Insurance | | | | | | | | |
| 10% | -7262 | 4.8 | 17.6 | 8.8 | -7270 | 5.6 | 18.2 | 9.0 | -7274 | 7.2 | 20.6 | 7.1 | -7270 | 8.0 | 20.9 | 7.8 |
| 20% | -7280 | 3.4 | 15.2 | 7.8 | -7286 | 4.6 | 16.2 | 8.7 | -7270 | 5.9 | 19.6 | 7.4 | -7270 | 7.9 | 20.7 | 7.4 |
| 30% | -7310 | 2.0 | 12.2 | 6.0 | -7325 | 3.6 | 13.4 | 8.0 | -7264 | 4.6 | 18.7 | 6.2 | -7268 | 7.8 | 20.6 | 7.2 |
| 40% | -7353 | 1.4 | 9.9 | 6.2 | -7358 | 3.1 | 11.2 | 6.8 | -7273 | 3.9 | 18.0 | 5.8 | -7265 | 7.8 | 20.4 | 6.9 |
| 0% | -7270 | 8 | 21 | 8 | BDeu true network: -7592 | | | | | | | | | | | |
| | | | | | | | | Hailfinder | | | | | | | | |
| 10% | -27270 | 13.3 | 22.2 | 11.1 | -27231 | 15.4 | 22.5 | 13.0 | -27202 | 14.7 | 23.8 | 11.4 | -27183 | 16.8 | 25.2 | 11.3 |
| 20% | -27408 | 11.0 | 19.9 | 9.2 | -27330 | 14.2 | 19.9 | 12.1 | -27244 | 12.5 | 23.3 | 10.4 | -27190 | 16.4 | 25.3 | 10.7 |
| 30% | -27582 | 8.7 | 17.2 | 8.0 | -27461 | 13.3 | 17.4 | 12.2 | -27280 | 10.3 | 22.2 | 9.8 | -27198 | 16.3 | 25.6 | 10.2 |
| 40% | -27781 | 7.0 | 14.2 | 7.6 | -27649 | 12.4 | 14.6 | 11.4 | -27309 | 8.0 | 21.2 | 9.2 | -27204 | 16.2 | 26.0 | 9.6 |
| 0% | -27171 | 17 | 25 | 12 | BDeu true network: -29347 | | | | | | | | | | | |

in general, the same behavior as in the previous experiments, although in this case it must be taken into account that in all the databases (except Alarm) the true networks have a score worse than the learned ones without using restrictions.

## 7 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 scoring functions and search methods. We have illustrated it for the specific case of a learning algorithm using local search. The experimental results show that the use of additional knowledge in form of restrictions may lead to improved network structures 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, 11] or within algorithms based on independence tests [16]. Finally, we would like to study another type of restriction, namely conditional independence relationships between variables that must be true.

**Acknowledgments.** This work has been supported by the Spanish Ministerio de Ciencia y Tecnología and Junta de Comunidades de Castilla-La Mancha, under Projects TIC2001-2973-CO5-01 and PBC-02-002, respectively.

## References

1. Abramson, B., Brown, J., Murphy, A., & Winkler, R. L. (1996). Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting, 12*, 57–71.
2. Acid, S., & de Campos, L.M. (2003). Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18, 445–490.
3. Beinlich, I. A., Suermondt, H. J., Chavez, R. M., & Cooper, G. F. (1989). 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*, 247–256.
4. Binder, J., Koller, D., Russell, S., & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning, 29*, 213–244.
5. Cheng, J., & Greiner, R. (1999). Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth UAI Conference*, 101–108.
6. Chickering, D.M. (1995). A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Eleventh UAI Conference*, 87–98.
7. Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–348.
8. de Campos, L. M., Fernández-Luna, J. M., Gámez, J. A., & Puerta, J. M. (2002). Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning, 31*, 291–311.
9. de Campos, L. M., Fernández-Luna, J. M., & Puerta, J. M. (2002). Local search methods for learning Bayesian networks using a modified neighborhood in the space of dags. *Lecture Notes in Computer Science, 2527*, 182–192.
10. de Campos, L. M., & Puerta, J. M. (2001). 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*, 109–115.
11. de Campos, L. M., & Puerta, J. M. (2001). Stochastic local search algorithms for learning belief networks: Searching in the space of orderings. *Lecture Notes in Artificial Intelligence, 2143*, 228–239.
12. Elvira Consortium. (2002). Elvira: an environment for probabilistic graphical models. In J.A. Gámez, A. Salmerón (Eds.), *Proceedings of the 1st European Workshop on Probabilistic Graphical Models*, 222–230.
13. Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*, 197–243.
14. Langley, P., & Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth UAI Conference*, 399–406.
15. Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo: Morgan Kaufmann.
16. Spirtes, P., Glymour, C., & Scheines, R. (1993). *Causation, Prediction and Search.* Lecture Notes in Statistics 81, New York: Springer Verlag.