Machine Learning, 59, 213–235, 2005
2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Learning Bayesian Network Classifiers: Searching in a Space of Partially Directed Acyclic Graphs

SILVIA ACID acid@decsai.ugr.es
LUIS M. DE CAMPOS lci@decsai.ugr.es
JAVIER G. CASTELLANO fjgc@decsai.ugr.es
Departamento de Ciencias de la Computación e I.A., E.T.S.I. Informática, Universidad de Granada, 18071, Granada, Spain

Editors: Pedro Larrañaga, Jose A. Lozano, Jose M. Peña and Iñaki Inza

Abstract. There is a commonly held opinion that the algorithms for learning unrestricted types of Bayesian networks, especially those based on the score+search paradigm, are not suitable for building competitive Bayesian network-based classifiers. Several specialized algorithms that carry out the search into different types of directed acyclic graph (DAG) topologies have since been developed, most of these being extensions (using augmenting arcs) or modifications of the Naive Bayes basic topology. In this paper, we present a new algorithm to induce classifiers based on Bayesian networks which obtains excellent results even when standard scoring functions are used. The method performs a simple local search in a space unlike unrestricted or augmented DAGs. Our search space consists of a type of partially directed acyclic graph (PDAG) which combines two concepts of DAG equivalence: classification equivalence and independence equivalence. The results of exhaustive experimentation indicate that the proposed method can compete with state-of-the-art algorithms for classification.

Keywords: classification, Bayesian networks, learning algorithms, scoring functions, directed acyclic graphs, partially directed acyclic graphs

1. Introduction

Supervised classification is an important task in data mining, assigning predefined class labels to data items described by means of a set of features or attributes. A *classifier* is a function that maps an instance into a class label. The problem of the automatic induction of classifiers from data sets of preclassified instances has long received considerable attention within the machine learning community. Traditional approaches to this problem include decision trees, neural networks, and classical statistical methods (Hand, 1981). More recently, Bayesian networks have also been successfully applied to classification problems in many ways by inducing classifiers using different types of Bayesian network learning algorithms.

One way to do so is to perform unsupervised or unrestricted learning of Bayesian networks (in the sense that the learning method does not distinguish the class variable from the attribute variables) and then use the model as the classifier. Within this approach, if the search method underlying the learning algorithm is guided by a scoring function, the networks with the best scores do not necessarily give rise to the best classifiers: optimizing the scoring function in the learning process does not necessarily maximize the accuracy in predicting the class label given the evidence over the attributes. Consequently, neither the scoring functions nor the search procedures used by these unrestricted learning algorithms take into account the specificity of the classification problem, where there is a distinguished class variable, and therefore the objective is not to get a good representation of a joint probability distribution for the class and the attributes, but one for the posterior probability distribution of the class given the attributes. As a result, several specialized algorithms that carry out the search into different types of directed acyclic graph (DAG) topologies have been developed, most of these being extensions (using augmenting arcs) or modifications of the Naive Bayes basic topology. This approach generally obtains more satisfactory results in terms of classification accuracy.

In this paper, we present a new algorithm to induce classifiers based on Bayesian networks which uses standard scoring functions and achieves excellent results. The method performs a simple local search, focused on the classification task, in a different space to that of unrestricted or augmented DAGs. Our search space consists of a type of partially directed acyclic graph (PDAG), which we call C-RPDAG and which combines two concepts of DAG equivalence: classification equivalence and independence equivalence. This enables us, on one hand, to focus the search with a classification perspective in mind and, on the other hand, to explore a more reduced and robust search space.

The rest of the paper is organized as follows. We begin in Section 2 with the preliminaries. Section 3 describes the kind of graphical structure which comprises the search space of our algorithm and Section 4 gives details of the method used to search for good solutions within this space, including the operators selected and the way of evaluating configurations. Section 5 contains the experimental results of the evaluation of the proposed algorithm on a wide range of databases and an empirical comparison with other state-of-the-art Bayesian network classifiers. Section 6 contains the concluding remarks. Finally, the Appendix includes proof of all the propositions set out in the paper.

2. Preliminaries

We shall begin by giving some background about Bayesian network learning algorithms and their application to the induction of classifiers.

2.1. *Learning Bayesian networks*

Let us consider a finite set $\mathbf{U} = \{X_1, X_2, \dots, X_n\}$ of discrete random variables, each variable taking on values from a finite set. We shall use capital letters for variable names, and lower-case letters to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters and assignments of values to the variables in these sets are denoted by boldface lower-case letters.

A *Bayesian Network* (BN) is a graphical representation of a joint probability distribution (Pearl, 1988). It comprises two components: firstly, a directed acyclic graph (DAG) $G = (\mathbf{U}, E_G)$, where $\mathbf{U}$, the set of nodes, represents the system variables, and E_G, the set of arcs, represents direct dependence relationships between the variables; and secondly, a set of parameters is also stored for each variable in $\mathbf{U}$, usually conditional probability distributions. For each variable $X \in \mathbf{U}$ we have a family of conditional distributions $p(X \mid pa_G(X))$, one for each *configuration*, $pa_G(X)$, from the *parent set* of X in the graph, $Pa_G(X) = \{Y \in \mathbf{U} \mid Y \rightarrow X \in E_G\}$. From these conditional distributions we can recover the joint distribution over $\mathbf{U}$:

$$p(x_1, x_2, \ldots, x_n) = \prod_{X_i \in \mathbf{U}} p(x_i \mid pa_G(X_i)) \tag{1}$$

The independence relationships which make this decomposition possible are graphically encoded (through the d-separation criterion) by means of the presence or absence of direct connections between pairs of variables in the DAG G. For example, each variable X is independent of its nondescendants given its parents in G. Another interesting pattern of independence is the one generated by the *Markov blanket*: the Markov blanket of a variable X in G, $MB_G(X)$, is the set of variables comprising X's parents, X's children and the parents of X's children in G; X is then independent of the remaining variables (those in $\mathbf{U}\backslash(MB(X) \cup \{X\})$) given its Markov blanket.

The problem of learning a BN is that given a *training set* $D = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ of instances of $\mathbf{U}$, find the network that, in some sense, best matches D. There are a great many learning algorithms which may be subdivided into two general approaches: methods based on *conditional independence tests* (also called *constraint-based* methods), and methods based on a *scoring function* and a *search* procedure.[1]

Constraint-based algorithms (Cheng et al., 2002; de Campos & Huete, 2000; Spirtes, Glymour, & Scheines, 1993) perform a qualitative study of the dependence and independence relationships between the variables in the domain (obtained from the data by means of conditional independence tests), and attempt to find a network that represents these relationships as far as possible.

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. The scoring functions are based on different principles, such as entropy (Chow & Liu, 1968), Bayesian approaches (Cooper & Herskovits, 1992; Heckerman, Geiger, & Chickering, 1995), or the Minimum Description Length (Lam & Bacchus, 1994). With respect to the search methods, the algorithms in this category have commonly used local search methods (Cooper & Herskovits, 1992; Heckerman, Geiger, & Chickering, 1995), due to the exponentially large size of the search space. However, there is a growing interest in other heuristic search methods (for references, see Acid & de Campos, 2003).

2.2. Learning Bayesian network classifiers

In this section, we shall review previous work on the application of Bayesian networks in the field of classification. In this context, the set of variables considered will be $\mathbf{V} = \mathbf{U} \cup \{C\}$. C is a distinguished variable, called the *class* variable, and the variables in $\mathbf{U}$ will play

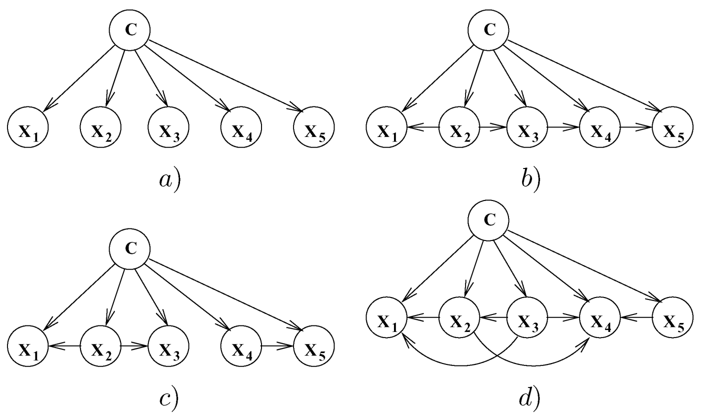

Figure 1. Different models for representing dependencies among the attributes: (a) Naive Bayesian network classifier (NB). (b) Tree-augmented Naive Bayesian network classifier (TAN). (c) Forest-augmented Naive Bayesian network classifier (FAN). (d) Bayesian network augmented Naive Bayesian classifier (BAN).

the role of the *attributes* used to predict the value of C. The objective is to compute the posterior probability of the class given any configuration of the attributes, $p(C \mid \mathbf{u})$.

Any algorithm for learning BNs can be used to induce a classifier. In this case, the resulting graph structure G may be a general, unrestricted DAG. If the sample to be classified has no missing data, the classification process does not require the use of complicated BN inference algorithms: as C is independent of all the variables given its Markov blanket, $p(c \mid \mathbf{u}) \propto p(c \mid pa_G(C)) \prod_{X \in \text{Children}(C)} p(x \mid pa_G(X))$, and the computation of the posterior probability is quite simple. The results obtained using this approach are not encouraging when the learning algorithms considered are score+search-based (Friedman, Geiger, & Goldszmidt, 1997). Using constraint-based algorithms instead seems to produce better results (Cheng & Greiner, 1999).

At the other extreme, the simplest classifier which can be considered as a Bayesian network classifier, although coming from the pattern recognition field, is the famous *Naive Bayesian* (NB) classifier (Langley, Iba, & Thompson, 1992) (Figure 1(a) depicts its DAG structure). Its success is mainly due to its simplicity: firstly, no structure learning is required (the network structure is fixed), and the parameters need only be estimated from the data set; secondly, the required parameters may be reliably computed using only the bi-dimensional statistics for the class and each attribute; and thirdly, the classification process is very efficient: $p(c \mid \mathbf{u}) \propto p(c) \prod_{X \in \mathbf{U}} p(x \mid c)$.

The NB model exhibits a surprisingly competitive predictive accuracy, considering its clearly unrealistic assumptions (each attribute is conditionally independent from the other attributes given the class and all the attributes have a bearing on the class), and outperforms many more sophisticated classifiers over a large number of datasets, especially when the attributes are not strongly correlated (Langley, Iba, & Thompson, 1992; Domingos & Pazzani, 1997; Friedman, Geiger, & Goldszmidt, 1997).

The relative success of the NB classifier has motivated the development of many methods which attempt to improve it by relaxing some of its basic assumptions.

One way to relax the independence assumption of the NB classifier is to use restricted DAG topologies which starting from the NB structure, complete it by adding *augmenting arcs* between attributes. The resulting structures are *augmented naive Bayesian networks* (ABNs) (Friedman, Geiger, & Goldszmidt, 1997). Several ways to include augmenting arcs have been proposed. Borrowing the terminology introduced by Sahami (1996), a *k-dependence Bayesian classifier* is a Bayesian network whose associated DAG contains the structure of the naive Bayes classifier and allows each attribute to have a maximum of k attribute nodes as parents. Within this approach, *tree-augmented naive Bayesian network* (TAN) (Friedman, Geiger, & Goldszmidt, 1997) is a 1-dependence Bayesian classifier (see Figure 1(b)). In TAN the attributes are linked together as a directed tree. This algorithm generates an optimal 1-dependence structure by extending Chow and Liu's method for learning trees (1968), using the conditional mutual information measure between attributes given the class variable. A variant of TAN is *forest-augmented naive Bayesian network* (FAN) (Lucas, 2002), the difference being that in FAN the dependencies between attributes are restricted to form a forest (see Figure 1(c)). Another type of TAN classifier (Keogh & Pazzani, 2002) performs hill-climbing searches of TAN structures, using leaving-one-out cross validation to estimate the accuracy of the classifier.

At the other extreme of the range of k-dependence ABNs we have the so-called *Bayesian network augmented naive Bayesian classifier* (BAN), an n-dependence ABN (see Figure 1(d)). In this case, the NB structure is fixed and the search of the augmenting arcs is carried out using an algorithm for learning Bayesian networks. Examples of this approach which use a scoring-based search algorithm can be found in the work by Friedman, Geiger, and Goldszmidt (1997), which use an MDL scoring metric and local search, or Ezawa, Singh, and Norton (1996), based on the K2 learning algorithm. On the other hand, constraint-based learning algorithms instead of scoring-based ones are used by Cheng and Greiner (1999).

In between TAN and BAN there is the KDB algorithm (Sahami, 1996), which builds k-dependence Bayesian classifiers, for fixed *k*, using greedy search and mutual information.

There are also other methods to relax the independence assumption of the NB model that are not strictly based on ABNs. This is the case of the *semi-naive Bayesian classifier* (Kononenko, 1991) and the model proposed by Pazzani (1995). They share the idea of merging several correlated attributes into new compound attributes. These compound attributes are considered independent among each other given the class, but no assumption of independence is made about the attributes within a group. Thus, the resulting model is an NB model where the new attributes can be either groups or individual attributes. Another proposal within this category (Zhang, Nielsen, & Jensen, 2004), is to introduce latent variables to model violations of the independence between attributes given the class. This model uses hierarchical naive Bayes (HNB) structures, which are tree-shaped BNs with latent variables between the class node (the root) and the attributes (the leaves).

Finally, another method, which is closer to our proposal, is based on the concept of the Markov blanket (Sierra & Larrañaga, 1998). This method, called MB-GA, searches in a space of DAGs where all the attributes must belong to the Markov blanket of the class variable. The search is carried out by means of genetic algorithms and each candidate network is evaluated using its accuracy as the classifier. Another variant of this method, RMB-GA, uses a modified search space, where not all the attributes must belong to the Markov blanket and some structural restrictions are imposed in order to obtain simpler models (if an attribute is a parent of the class, it cannot be a parent of a class' child; moreover, any attribute may be a parent of at most one class' child).

The other basic assumption of the NB classifier (i.e. that all the attributes have some influence on the class variable) can also create some problems: firstly, if two attributes are strongly correlated, their influence on the class variable may be overamplified by the NB classifier; and secondly, if an attribute is truly irrelevant, its use only contributes to adding noise to the classification. In either case, the solution may be to remove some of the attributes by using *feature subset selection* (FSS). The *selective naive Bayesian classifier* (Langley & Sage, 1994; John & Kohavi, 1994) searches for a good subset of attributes, starting from an empty set and iteratively introducing the attribute that most improves the predictive accuracy, using cross-validation. Another selective Bayesian classifier was proposed by Singh and Provan (1996), and this combines several FSS strategies with an unrestricted BN learning algorithm over the previously selected attribute subset. FSS is also used by Ezawa, Singh, & Norton (1996) and Pazzani (1995). Other algorithms for FSS are based on the idea of identifying Markov blankets (Aliferis, Tsamardinos, & Statnikov, 2003; Frey et al., 2003; Koller & Sahami, 1996).

3. The C-RPDAG search space

In this section, we shall describe the elements that comprise the search space of our algorithm for learning Bayesian network classifiers.

3.1. Class-focused DAGs

The first important fact to be taken into consideration when designing our search space is that from a classification point of view, many different DAGs are equivalent, in the sense that they will produce the same posterior probabilities for the class variable.[2] Let us formalize this idea:

Definition 1 *(Classification equivalence).* Let $G = (\mathbf{V}, E_G)$ and $G' = (\mathbf{V}, E_{G'})$ be two DAGs. Let p be any joint probability distribution on $\mathbf{V}$, and p_G and $p_{G'}$ be the probability distributions that factorize according to G and G', respectively, defined as $p_G(X \mid pa_G(X)) = p(X \mid pa_G(X))$ and $p_{G'}(X \mid pa_{G'}(X)) = p(X \mid pa_{G'}(X))$, $\forall X \in \mathbf{V}$. If $p_G(C \mid \mathbf{u}) = p_{G'}(C \mid \mathbf{u})\ \forall \mathbf{u}$, we say that G and G' are *classification-equivalent*.

Proposition 1. *Given any DAG $G = (\mathbf{V}, E_G)$, let $G_c = (\mathbf{V}, E_c)$ be the subgraph of G defined as follows:*
1. $Pa_{G_c}(C) = Pa_G(C)$,
2. $\forall X \in \mathbf{V}, X \neq C$, if $C \in Pa_G(X)$ then $Pa_{G_c}(X) = Pa_G(X)$,
3. $\forall X \in \mathbf{V}, X \neq C$, if $C \notin Pa_G(X)$ then $Pa_{G_c}(X) = \emptyset$.

Then G and G_c are classification-equivalent. Moreover, for any other subgraph $H = (\mathbf{V}, E_H)$ of G such that G and H are classification-equivalent, G_c is also a subgraph of H.

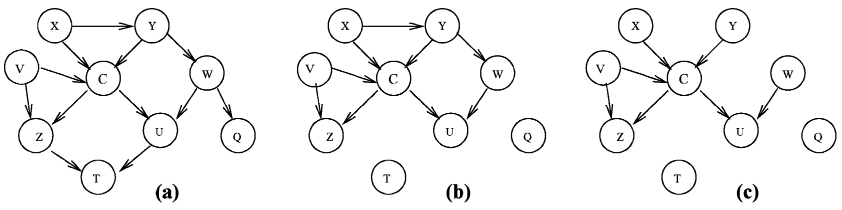

Figure 2. (a) A DAG G; (b) subgraph of G induced by the Markov blanket of C; (c) subgraph G_c.

It should be noted that G_c is not equal to the subgraph of G induced by C and the Markov blanket of C, as the first contains fewer arcs than the second. Figure 2 illustrates the differences between these two subgraphs.

Proposition 1 identifies the minimum subgraph of any DAG G that performs exactly as G as a classifier for the variable C. This suggests that we could try to find good BN classifiers by focusing the search in the set of DAGs of this kind, instead of searching in the whole space of DAGs. First, we shall formally define this type of structure.

Definition 2 *(Class-focused DAG).* A DAG $G = (\mathbf{V}, E_G)$ is a *class-focused* DAG (C-DAG) with respect to the variable C if and only if it satisfies the following condition:

$$\forall X, Y \in \mathbf{V},\ \text{if } X \rightarrow Y \in E_G \text{ then either } Y = C \text{ or } X = C \text{ or } C \rightarrow Y \in E_G\,. \quad (2)$$

C-DAGs may therefore contain only arcs linking the class with any attribute (in either direction) and arcs between attributes only in the case that the class is a parent of some of them.

A class-focused DAG can be seen as the canonical representative of a class of DAGs, all of which are equivalent as classifiers for C. The following result formalizes this idea:

Proposition 2. *Given any C-DAG $H = (\mathbf{V}, E_H)$, let $\mathcal{C}_H$ be the set*

$$\mathcal{C}_H = \{G \mid G \text{ is a DAG on } \mathbf{V} \text{ and } G_c = H\}\,.$$

Then

1. *$G \in \mathcal{C}_H$ if and only if $Pa_G(X) = Pa_H(X)$ $\forall X \in \mathbf{V}$ such that $C \in Pa_G(X)$, and $Pa_G(C) = Pa_H(C)$.*
2. *The family of sets $\{\mathcal{C}_H \mid H$ is a C-DAG on $\mathbf{V}\}$ is a partition of the set of DAGs on $\mathbf{V}$.*

3.2. Class-focused RPDAGs

Although the use of C-DAGs instead of general DAGs helps to reduce the search space, further reductions are possible if we also consider the concept of *independent-equivalence* classes of DAGs (Pearl & Verma, 1990), i.e. classes of DAGs where each represents a different set of conditional independence assertions.[3]

The graphical objects used to represent independent-equivalence classes of DAGs are *PDAGs* (*partially directed acyclic graphs*). These graphs may contain both directed (arcs) and undirected (links) edges, but no directed cycles. In the context of learning general, unrestricted Bayesian networks, a number of algorithms which carry out the search in a space of independent-equivalence classes of DAGs have been developed. This feature reduces the size of the search space, which has a smoother landscape, and avoids some early decisions on arc directions. The price that these algorithms must pay for this reduction, namely that the evaluation of candidate structures does not take advantage of the property of decomposability of many scoring functions (thus rendering these algorithms less efficient), has recently been overcome (Acid & de Campos, 2003; Chickering, 2002).

Our proposal is to apply these ideas about independence equivalence of DAGs to the case of class-focused DAGs. Although there is a kind of PDAG that can be used to canonically represent independent-equivalence classes of DAGs (called *completed PDAGs* by Chickering, 1996), the non-canonical scheme of representation that we shall use is that of *restricted PDAGs* (*RPDAGs*)(Acid & de Campos, 2003), which are considerably simpler than completed PDAGs. Let us introduce some additional notation and then the concept of RPDAG. The *skeleton* of a DAG is the undirected graph that results from ignoring the directionality of every arc. A *h-h pattern* (*head-to-head pattern*) in a DAG G is an ordered triplet of nodes, (X, Y, Z), such that G contains the arcs $X \rightarrow Y$ and $Y \leftarrow Z$. Given a PDAG $G = (\mathbf{V}, E_G)$, the following subsets of nodes are defined for each node $X \in \mathbf{V}$:

- $Ch_G(X) = \{Y \in \mathbf{V} \mid X \rightarrow Y \in E_G\}$, the set of *children* of X.
- $Sib_G(X) = \{Y \in \mathbf{V} \mid X — Y \in E_G\}$, the set of *siblings* of X.
- $Ad_G(X) = Pa_G(X) \cup Ch_G(X) \cup Sib_G(X)$, the set of *adjacents* to X.

***Definition 3** (Restricted PDAG).* (Acid & de Campos, 2003). A PDAG $G = (\mathbf{V}, E_G)$ is a restricted PDAG (RPDAG) if and only if it satisfies the following conditions:

1. $\forall X \in \mathbf{V}$, if $Pa_G(X) \neq \emptyset$ then $Sib_G(X) = \emptyset$.
2. G does not contain any directed cycle.
3. G does not contain any completely undirected cycle, i.e. a cycle containing only links.
4. If $X \rightarrow Y \in E_G$ then either $|Pa_G(Y)| \geq 2$ or $Pa_G(X) \neq \emptyset$.

As RPDAGs are intended as representations of sets of independent-equivalent DAGs, the set of DAGs which is represented by a given RPDAG G must be defined:

***Definition 4** (Extension of an RPDAG) (Acid & de Campos, 2003).* Given an RPDAG $G = (\mathbf{V}, E_G)$, we say that a DAG $H = (\mathbf{V}, E_H)$ is an extension of G if and only if:

1. G and H have the same skeleton.
2. If $X \rightarrow Y \in E_G$ then $X \rightarrow Y \in E_H$ (no arc is redirected).
3. G and H have the same h-h patterns.

We shall use $Ext(G)$ to denote the set of DAGs that are extensions of a given RPDAG G. It can be proved (Acid & de Campos, 2003) that (1) $Ext(G) \neq \emptyset$, (2) $\forall H, H' \in Ext(G)$

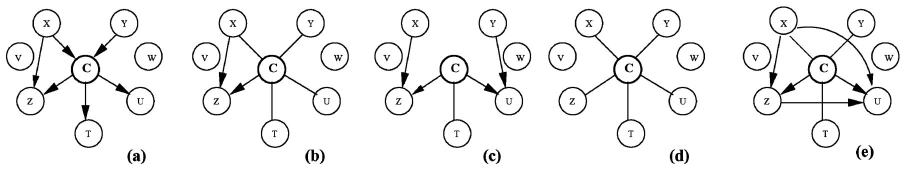

Figure 3. Examples of C-RPDAGs.

H and H' are independence equivalent, and (3) $H, H' \in Ext(G)$ if and only if H and H' have the same skeleton and the same h-h patterns.

Returning to our problem of searching for BN classifiers, our proposal is to define the search space as the set of RPDAGs which are different from a classification point of view. More formally, we define:

Definition 5 *(Class-focused RPDAG).* A class-focused RPDAG or C-RPDAG is an RPDAG whose extensions are class-focused DAGs.

The following proposition gives a useful characterization of the concept of C-RPDAG.

Proposition 3. *A PDAG G is a class-focused RPDAG if and only if the following conditions hold:*

1. *G does not contain any directed cycle.*
2. *If $Pa_G(C) \neq \emptyset$ then $|Pa_G(C)| \geq 2$ and $Sib_G(C) = \emptyset$.*
3. *$\forall X \in \mathbf{V}$, $X \neq C$, if $Pa_G(X) \neq \emptyset$ then $C \in Pa_G(X)$ and either $|Pa_G(X)| \geq 2$ or $|Pa_G(C)| \geq 2$.*
4. *$\forall X \in \mathbf{V}$, $X \neq C$, if $Sib_G(X) \neq \emptyset$ then $Sib_G(X) = \{C\}$ and $Pa_G(X) = \emptyset$.*

Some examples of C-RPDAGs are displayed in Figure 3. Another interesting result concerning C-RPDAGs is:

Proposition 4. *Let $G = (\mathbf{V}, E_G)$ be a C-RPDAG. Then, $\forall H, H' \in Ext(G)$, H and H′ are classification-equivalent.*

In this way, the elements in our search space of C-RPDAGs represent sets of classification-equivalent C-DAGs, each in turn representing a set of classification-equivalent DAGs. It should be noted that C-RPDAGs are not always canonical representations of the classes of classification-equivalent DAGs: two classification-equivalent DAGs may be associated to different C-RPDAGs. Anyway, a useful reduction in the search space with respect to that of DAGs is obtained.

We believe that this is not the only advantage of using C-RPDAGs. On one hand, a search process in the space of DAGs may lose its way, in the sense that the inclusion of some arcs, irrelevant for the classification task, may prevent or hinder the inclusion of other relevant arcs. We do not expect this problem to occur in a classification-oriented search process, such as the one based on C-RPDAGs. On the other hand, with respect to search methods based on augmented DAGs, those where $C \in Pa_G(X)$ $\forall X \in \mathbf{V}\backslash\{C\}$ (such as TAN, FAN and BAN), our search space does not impose the naive Bayes substructure, although this can be obtained if it is really appropriate (it should be observed that augmented DAGs are always C-DAGs). We can also obtain more general structures, such as those where a node is a parent of a class' child, but this node is not connected to the class. It should be noted that feature selection is also an integral part of our approach, since all the variables that remain isolated at the end of the search process will not be considered by the classifier. Moreover, the structures obtained by other classification methods can also be reproduced using C-RPDAGs. This is the case, for example, of the semi-naive Bayesian classifier and the selective naive Bayesian classifier in their different formulations.

4. The search method

We shall use a local method to explore the search space of C-RPDAGs. The starting point of the search process will be an empty C-RPDAG (corresponding to an empty DAG). We must then define the operators to move from one configuration to another neighboring configuration.

Our basic operators essentially are the same as those used by Acid and de Campos (2003) for RPDAGs, i.e. the inclusion of an edge between a pair of non-adjacent nodes and the removal of an existing edge between a pair of adjacent nodes. These edges may be either directed or undirected.

4.1. The operators

The five operators used by Acid and de Campos to move in the space of RPDAGs were: A_arc(X, Y), addition of an arc $X \rightarrow Y$; A_link(X, Y), addition of a link X–Y; D_arc(X, Y), deletion of an arc $X \rightarrow Y$; D_link(X, Y), deletion of a link X–Y; A_hh(X, Y, Z), addition of an arc $X \rightarrow Y$ and creation of the h-h pattern $X \rightarrow Y \leftarrow Z$ by transforming the link Y–Z into the arc $Y \leftarrow Z$. However, taking into account the special character of the class variable C in our case, these operators will be reformulated in terms of C's parents, C's children and the parents of C's children, in order to make their meaning clearer from the perspective of the class variable:

– A_ParentOfC(X), addition of the arc $X \rightarrow C$.
– A_ChildOfC(X), addition of the arc $C \rightarrow X$.
– A_SiblingOfC(X), addition of the link X–C.
– A_HHOfC(X, Y), creation of the h-h pattern $X \rightarrow C \leftarrow Y$ by adding the arc $X \rightarrow C$ and transforming the link Y–C into the arc $Y \rightarrow C$.
– A_ParentOfChild(X, Y), addition of the arc $X \rightarrow Y$.
– A_HHOfChild(X, Y), creation of the h-h pattern $X \rightarrow Y \leftarrow C$ by adding the arc $X \rightarrow Y$ and transforming the link C–Y into the arc $C \rightarrow Y$.
– D_ParentOfC(X), deletion of the arc $X \rightarrow C$.
– D_ChildOfC(X), deletion of the arc $C \rightarrow X$, together with all the other arcs pointing to X.
– D_SiblingOfC(X), deletion of the link X–C.

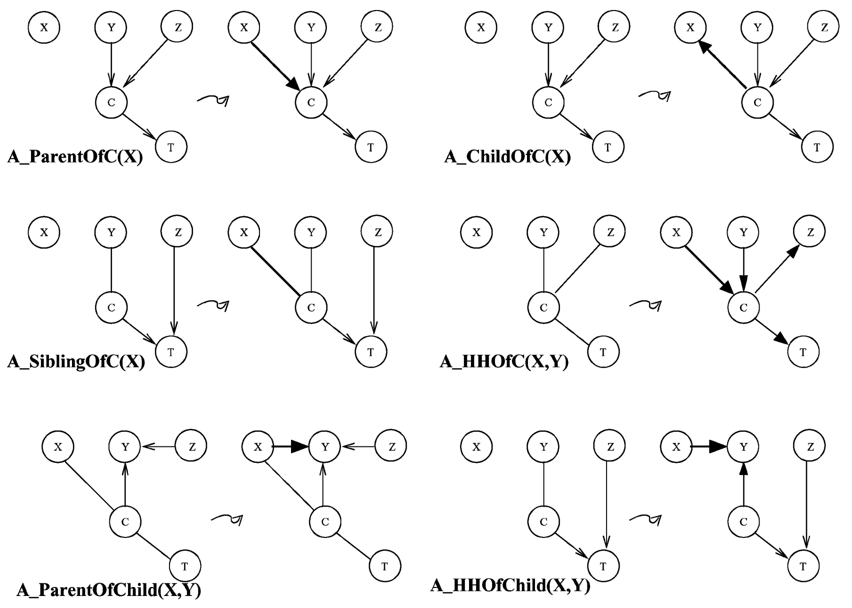

Figure 4. Examples of the six operators of addition.

- D_HHOfC(X, Y), destruction of the h-h pattern $X \rightarrow C \leftarrow Y$ by deleting the arc $X \rightarrow C$ and transforming the arc $Y \rightarrow C$ into the link Y–C.
- D_ParentOfChild(X, Y), deletion of the arc $X \rightarrow Y$.
- D_HHOfChild(X, Y), destruction of the h-h pattern $X \rightarrow Y \leftarrow C$ by deleting the arc $X \rightarrow Y$ and transforming the arc $C \rightarrow Y$ into the link Y–C.

The conditions that the current C-RPDAG G must verify so that each of these operators can be applied in order to obtain a valid neighboring C-RPDAG, G', are shown in Table 1. These conditions can be easily derived from the result in Proposition 3. Table 1 also shows the actions required to transform G into the corresponding neighbor G', including those necessary to ensure that G' is a C-RPDAG. Figures 4 and 5 show an example for each one of the operators. It should be observed that as each operator for insertion has a corresponding operator for deletion, we can obtain any C-RPDAG starting from any other C-RPDAG.

4.2. *Evaluation of candidate structures*

The search method we have described may be applied in combination with any function which is capable of measuring the fitness between each explored C-RPDAG and the data available to induce the classifier (the training set). Several options could be considered. One is to use the degree of accuracy of the resulting classifier (i.e. the percentage of well-classified samples) for the assessment of each candidate network. This is a *wrapper* approach, commonly used in many algorithms for inducing classifiers or performing feature

Table 1. The operators, their conditions of applicability and the required actions.

Operator	Conditions	Actions
A_ParentOfC(X)	$X \notin Ad_G(C)$; $Pa_G(C) \neq \emptyset$	insert($X \rightarrow C$)
A_ChildOfC(X)	$X \notin Ad_G(C)$; $Pa_G(C) \neq \emptyset$	insert($C \rightarrow X$)
A_SiblingOfC(X)	$X \notin Ad_G(C)$; $Pa_G(C) = \emptyset$	insert(X—C)

A_HHOfC(X,Y)	$X \notin Ad_G(C)$; $Pa_G(C) = \emptyset$
	$Y \in Sib_G(C)$
		insert($X \rightarrow C$)
		delete(Y—C); insert($Y \rightarrow C$)
		$\forall Z \in Sib_G(C) \setminus \{Y\}$ {
		delete(C—Z); insert($C \rightarrow Z$)}

A_ParentOfChild(X,Y)	$X \notin Ad_G(Y)$; $Y \in Ch_G(C)$ There is no directed path from Y to X in G	insert($X \rightarrow Y$)

A_HHOfChild(X, Y)	$X \notin Ad_G(Y)$; $Y \in Sib_G(C)$
	There is no directed path
	from Y to X in G
		insert($X \rightarrow Y$)
		delete(C—Y)
		insert($C \rightarrow Y$)

D_ParentOfC(X)	$X \in Pa_G(C)$; $\mid Pa_G(C) \mid \geq 3$	delete($X \rightarrow C$)
D_ChildOfC(X)	$X \in Ch_G(C)$	$\forall Z \in Pa_G(X)$ delete($Z \rightarrow X$)
D_SiblingOfC(X)	$X \in Sib_G(C)$	delete(X—C)

D_HHOfC(X,Y)	$X \in Pa_G(C)$; $Y \in Pa_G(C)$
	$\mid Pa_G(C) \mid = 2$
		delete($X \rightarrow C$)
		delete($Y \rightarrow C$); insert(Y—C)
		$\forall Z \in Ch_G(C)$
		if $Pa_G(Z) = \{C\}$ {
		delete($Z \rightarrow C$)}; insert(Z—C)

D_ParentOfChild(X, Y)	$X \in Pa_G(Y)$; $Y \in Ch_G(C)$ $\mid Pa_G(Y) \mid \geq 3$ or $Pa_G(C) \neq \emptyset$	delete($X \rightarrow Y$)

D_HHOfChild(X, Y)	$X \in Pa_G(Y)$; $Y \in Ch_G(C)$
	$\mid Pa_G(Y) \mid < 3$; $Pa_G(C) = \emptyset$
		delete($X \rightarrow Y$)
		delete($C \rightarrow Y$); insert(C—Y)

selection, as we mentioned earlier (John & Kohavi, 1994; Keogh & Pazzani, 2002; Langley, Iba & Thompson, 1992; Langley & Sage, 1994; Pazzani, 1995; Sierra & Larrañaga, 1998). As this kind of measure tends to cause problems of overfitting, the usual solution is to use an estimation scheme, such as cross-validation, to repeatedly evaluate the predictive accuracy of the candidate classifier on various data subsets. This approach is computationally expensive, unless the selected model is simple, such as the naive Bayesian classifier.

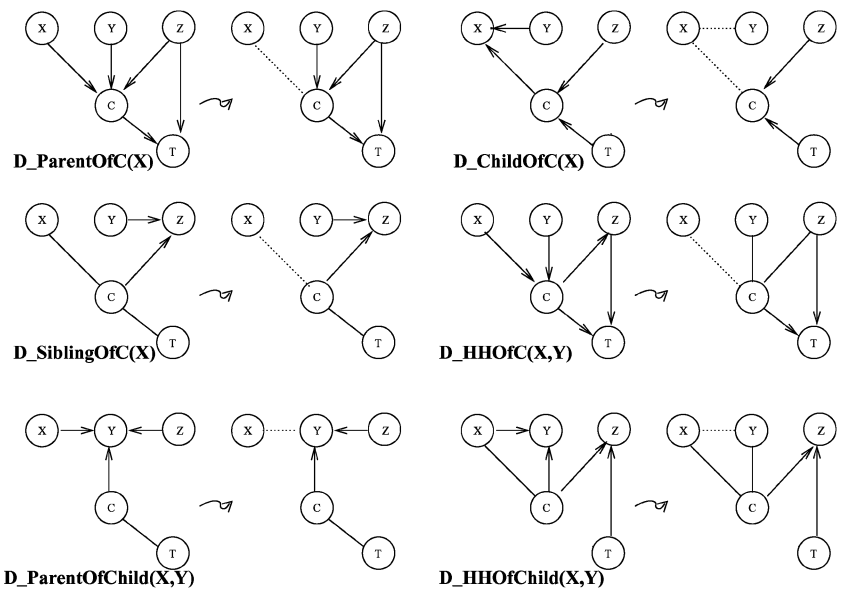

Figure 5. Examples of the six operators of deletion.

A second option would be to develop a specialized scoring metric that could account how well a network describes the probability distribution of the class variable given the attributes. Unfortunately, attempts in this direction have so far resulted computationally intractable (Friedman, Geiger, & Goldszmidt, 1997).

Although we do not discard the possibility of pursuing these approaches in the future, in this paper we shall consider a simpler and more efficient approach: a non-specialized scoring function, such as those commonly used by the algorithms for learning unrestricted Bayesian networks. Examples of these functions are BIC, BDe (Heckerman, Geiger, & Chickering, 1995) K2 (Cooper & Herskovits, 1992) and MDL (Lam & Bacchus, 1994). It has been argued that these scoring functions measure the fitness between the joint distribution associated to the network and the data, but do not measure the fitness between the conditional distribution of the class given the attributes and the data. Consequently, their use may result in poor classifiers. While the first of these assertions is undoubtedly true, the second is more controversial. The results of our experiments in Section 5 indicate that some non-specialized scoring functions, such as BDe, in combination with our C-RPDAG search space, also perform very well for the classification task.

In principle, a scoring function, g, evaluates DAGs. However, the elements in our search space are not DAGs but classes of equivalent DAGs. A simple way to use our method would be to select an extension H of the candidate C-RPDAG G to be evaluated, and compute $g(H : D)$. In order to do so, it is reasonable to use a *score-equivalent function*[4] in order to ensure that the same result is obtained regardless of the selected DAG H. Moreover, for the sake of efficiency in the evaluation process, the scoring function should be *decomposable*[5]; in this way the DAG obtained by adding or removing an arc from the current DAG H can be evaluated by modifying only one local score. For the operators being considered in our C-RPDAG space, a similar property can be obtained: by using a score equivalent and decomposable function, firstly, it is not necessary to transform the C-RPDAG into a DAG—it can be evaluated directly; and secondly, the score of any neighboring C-RPDAG can be obtained by computing at most two local scores. In this way, all the advantages of the search methods on the space of DAGs are preserved, but a more reduced and robust search space is used. The following proposition proves these assertions.

Proposition 5. *Let G be a C-RPDAG and G' be any C-RPDAG obtained by applying one of the operators described in Table 1 to G. Let g be a score equivalent and decomposable function.*

(a) If the operator is A_ParentOfC(X) then

$$g(G' : D) = g(G : D) - g_D(C, Pa_G(C)) + g_D(C, Pa_G(C) \cup \{X\})$$

(b) If the operator is A_ChildOfC(X) then

$$g(G' : D) = g(G : D) - g_D(X, \emptyset) + g_D(X, \{C\})$$

(c) If the operator is A_SiblingOfC(X) then

$$g(G' : D) = g(G : D) - g_D(C, \emptyset) + g_D(C, \{X\})$$

(d) If the operator is A_HHOfC(X, Y) then

$$g(G' : D) = g(G : D) - g_D(C, \{Y\}) + g_D(C, \{X, Y\})$$

(e) If the operator is A_ParentOfChild(X, Y) then

$$g(G' : D) = g(G : D) - g_D(Y, Pa_G(Y)) + g_D(Y, Pa_G(Y) \cup \{X\})$$

(f) If the operator is A_HHOfChild(X, Y) then

$$g(G' : D) = g(G : D) - g_D(Y, \{C\}) + g_D(Y, \{C, X\})$$

(g) If the operator is D_ParentOfC(X) then

$$g(G' : D) = g(G : D) - g_D(C, Pa_G(C)) + g_D(C, Pa_G(C) \backslash \{X\})$$

(h) If the operator is D_ChildOfC(X) then

$$g(G' : D) = g(G : D) - g_D(X, Pa_G(X)) + g_D(X, \emptyset)$$

(i) If the operator is D_SiblingOfC(X) then

$$g(G' : D) = g(G : D) - g_D(C, \{X\}) + g_D(C, \emptyset)$$

(j) If the operator is D_HHOfC(X, Y) then

$$g(G' : D) = g(G : D) - g_D(C, Pa_G(C)) + g_D(C, Pa_G(C) \backslash \{X\})$$

(k) If the operator is D_ParentOfChild(X, Y) then

$$g(G' : D) = g(G : D) - g_D(Y, Pa_G(Y)) + g_D(Y, Pa_G(Y) \backslash \{X\})$$

(l) If the operator is D_HHOfChild(X, Y) then

$$g(G' : D) = g(G : D) - g_D(Y, \{C, X\}) + g_D(Y, \{C\})$$

5. Experimental results

In this section we shall describe the experiments carried out with the proposed C-RPDAG-based classifier, the obtained results, and a comparative study with other Bayesian network classifiers. We have selected 31 well-known data sets, obtained from the *UCI repository of machine learning databases* (Blake & Merz, 1998), except “mofn-3-7-10” and “corral”, that were designed by Kohavi and John (1997). All these data sets have been widely used in specialist literature for comparative purposes in classification.

Table 2 gives a brief description of the characteristics of each data-base, including the number of instances, attributes and states for the class variable. These data sets have been preprocessed in the following way: the continuous variables have been discretized using the procedure proposed by Fayyad and Irani (1993), and the instances with undefined/missing values were eliminated. For this preprocessing stage, we have used the MLC++ System (Kohavi et al., 1994).

The other classifiers being considered are Naive Bayes (NB), Tree augmented naive Bayes (TAN), Bayesian network augmented naive Bayes (BAN), and two unrestricted Bayesian network learning algorithms (UBN and RPDAG). All the algorithms are implemented in the Elvira System (Elvira Consortium, 2002) a Java tool to construct probabilistic decision support systems, which works with Bayesian networks.

Both BAN and UBN algorithms search locally in the space of augmented DAGs and unrestricted DAGs, respectively (with the classical operators of arc addition, arc deletion, *Table 2*. Description of the data sets used in our experiments.

#	Data set	Instances	Attributes	Classes
1	Adult	45222	14	2
2	Australian	690	14	2
3	Breast	682	10	2
4	Car	1728	6	4
5	Chess	3196	36	2
6	Cleve	296	13	2
7	Corral	128	6	2
8	crx	653	15	2
9	Diabetes	768	8	2
10	DNA-nominal	3186	60	3
11	Flare	1066	10	2
12	German	1000	20	2
13	Glass	214	9	7
14	Glass2	163	9	2
15	Heart	270	13	2
16	Hepatitis	80	19	2
17	Iris	150	4	3
18	Letter	20000	16	26
19	Lymphography	148	18	4
20	mofn-3-7-10	1324	10	2
21	Mushroom	8124	22	2
22	Nursery	12960	8	5
23	Pima	768	8	2
24	Satimage	6435	36	6
25	Segment	2310	19	7
26	Shuttle-small	5800	9	7
27	Soybean-large	562	35	19
28	Splice	3190	60	3
29	Vehicle	846	18	4
30	Vote	435	16	2
31	Waveform-21	5000	21	3

and arc reversal), and are scoring-based. RPDAG, which is also scoring-based, searches locally in the RPDAG space using its specific operators. In these three methods, and also in C-RPDAG, the selected scoring function is the same: BDeu (Heckerman, Geiger, & Chickering, 1995), with the parameter representing the equivalent sample size set to 1 and a uniform structure prior. The starting point of the local search in UBN, RPDAG and C-RPDAG was the empty graph.

Table 3. Experimental results. Predictive accuracy.

#	NB	TAN	BAN	UBN	RPDAG	CRPDAG
1	*83.13* (–)	*85.23*	**85.54**	85.43	85.47	85.29
2	85.22	*84.98*	85.94	85.75	**86.09**	86.04
3	97.56	*95.60*	97.56	**97.65**	97.56	97.56
4	*85.40* (–)	93.86	93.09	92.98	**93.87**	93.07
5	*87.84* (–)	92.34 (–)	96.88	97.02	**97.53**	96.53
6	83.24	*80.89*	82.57	83.57	**84.03**	81.78
7	*86.82* (–)	99.49	**100.00**	**100.00**	**100.00**	**100.00**
8	86.97	*86.36*	86.56	86.71	86.66	**87.02**
9	*78.17*	79.00	78.65	79.09	**79.39**	78.78
10	95.40 (–)	*94.82* (–)	95.31 (–)	95.95	95.96	**96.15**
11	*80.40*	82.90	82.93	82.27	82.36	**83.27**
12	**75.27**	*72.80*	74.67	73.80	74.57	74.30
13	72.78	*69.05*	71.83	69.06	**73.84**	73.10
14	*83.38*	85.04	85.04	84.82	84.61	**85.86**
15	**83.46**	82.84	*82.10*	83.33	**83.46**	82.22
16	*85.00*	90.42	88.75	89.58	**94.58**	88.75
17	94.22	*93.56*	94.44	**94.89**	94.00	94.67
18	*74.01* (–)	**85.83** (+)	84.35 (–)	84.43 (–)	84.69	85.08
19	84.41	81.94	**84.43**	*79.08*	82.43	81.54
20	*85.40* (–)	90.71 (–)	87.29 (–)	99.17	**100.00**	99.50
21	*95.38* (–)	99.98	**100.00**	**100.00**	**100.00**	**100.00**
22	*90.26* (–)	92.27 (–)	91.82 (–)	93.12	93.51	**93.52**
23	*77.91*	78.78	78.03	78.78	**79.29**	78.03
24	*82.43* (–)	**88.47**	88.29	83.49 (–)	83.15 (–)	88.19
25	*92.16* (–)	**95.09**	94.34	94.56	94.46	94.34
26	*99.01* (–)	99.66	**99.72**	99.69	99.69	99.60
27	91.70	*86.05*	**92.53**	88.55	89.85	90.33
28	95.45 (–)	*94.89* (–)	95.28	95.73	95.78	**96.27**
29	*62.85* (–)	71.05	71.56	65.13 (–)	64.97 (–)	**71.91**
30	*90.04*	94.42	94.34	**94.57**	94.49	93.49
31	81.82	**82.89**	82.36	*79.81* (–)	79.96 (–)	82.45
Average	*85.39*	87.46	87.94	87.68	88.19	**88.34**
B	2	4	6	5	12	9
W	18	10	1	2	0	0

Once the graphical structure of each algorithm has been determined, the required marginal and conditional probabilities must be estimated from the data set. In all cases, we have used the same smoothed estimator, based on the Laplace law of succession (Good, 1965) (we

Table 4. Number of times that the algorithm in row *i* is better/**significantly better** than the algorithm in column *j*.

	NB	TAN	BAN	UBN	RPDAG	CRPDAG	Tot. better
NB	–	12/**1**	7/**0**	7/**2**	6/**1**	5/**0**	37/ **4**
TAN	19/**12**	–	12/**2**	10/**4**	7/**4**	12/**1**	60/**23**
BAN	23/**11**	18/**3**	–	13/**4**	11/**4**	10/**0**	75/**22**
UBN	24/**10**	20/**2**	16/**1**	–	8/**0**	12/**0**	80/**13**
RPDAG	23/**11**	24/**3**	17/**2**	20/**0**	–	16/**0**	100/**16**
CRPDAG	25/**14**	19/**5**	15/**4**	17/**4**	12/**3**	–	88/**30**
Tot. worse	114/**58**	93/**14**	67/**9**	67/**14**	44/**12**	55/**1**	

try to avoid problems of unreliability and overfitting of the maximum likelihood estimator in small data sets).

The results of this comparative study are displayed in Table 3. For each classifier and data set we show the predictive accuracy (the percentage of successful predictions on a test set different from the training set). This accuracy was measured as the average of three runs, the accuracy of each run being estimated using 10-fold cross-validation. Within each run, the cross-validation folds were the same for all the classifiers on each data set. We used repeated runs and 10-fold cross-validation to get a good balance between bias and variance of the estimation (Kohavi, 1995). The best and worst results obtained for each problem are emphasized using bold and italic fonts, respectively. Also, at the bottom of the table we show the average accuracy and how many times each algorithm has been the best (B) and the worst (W).

We can observe that no algorithm is always better than another for all the datasets, and the different approximations can obtain the best result depending on the problem. However, in general C-RPDAG and RPDAG obtain a greater number of best results and there is no case where they were the worst classifier. In order to confirm the good behavior of our algorithm, we have used a non-parametric statistical significance test: the Wilcoxon paired signed Rank Test. In Table 3 we also indicate whether an algorithm is significantly worse (–) or better (+) than C-RPDAG, on each data set. In Table 4 we compare each classifier with the others. The entry in row *i* column *j* represents the number of times that classifier *i* is better/significantly better (using the Wilcoxon test) than classifier *j*. Each row displays the times that the corresponding classifier is better, whereas each column says how many times the classifier is worse. These numbers clearly indicate that our algorithm does indeed behave generally better than the others.

Another important aspect to be taken into account, in addition to effectiveness, is that of efficiency. Although running time is only a rough measure of efficiency, we have therefore collected information about timing results of all the algorithms, excluding NB, which is obviously much faster than the others. In Table 5 we show the ratio of the time spent by each algorithm to the time spent by C-RPDAG, averaged across all the data sets and across the larger data sets (those including at least 20 attributes). As expected, C-RPDAG is faster than BAN and UBN (the difference increases when we consider only the largest data sets, *Table 5*. Average ratios of running times spent by each algorithm to the time spent by C-RPDAG.

	TAN	BAN	UBN	RPDAG
Across all the data sets	0.539	1.081	1.192	0.546
Across selected data sets	1.155	1.306	1.383	0.567

and in this case C-RPDAG runs even faster than TAN). Surprisingly, however, RPDAG is about twice as fast as C-RPDAG (although we believe that this is mainly due to a finer implementation of RPDAG).

6. Concluding remarks

Until recently, it was thought that a score-based learning algorithm could build Bayesian networks which maximize the scoring function but which may behave poorly as classifiers. This is only partially true, however, as the use of search algorithms specialized for classifications tasks permits us to obtain better results than search algorithms which are not oriented to classification, even within a score-based approach. In fact, what we have developed in this paper is a new score+search-based algorithm for learning Bayesian network classifiers from databases. It is based on the use of C-RPDAGs, a kind of partially directed acyclic graph, as the elements of the search space. These graphs combine the ideas of classification equivalence and independence equivalence to produce a more reduced, focused and robust search space. The experimental results show that our method can efficiently induce classifiers that compete favorably with several state-of-the-art Bayesian network classifiers. For future work we plan to use other search algorithms, more powerful than local search algorithms, in combination with our C-RPDAG space, in order to test whether the possible increase in predictive accuracy may compensate the extra cost of inducing the classifier. Candidate algorithms may be any of the search methods already used for learning unrestricted Bayesian networks.

Acknowledgments

This work has been supported by the Junta de Comunidades de Castilla-La Mancha and Fondo de Investigación Sanitaria, under Projects PBC-02-002 and FIS-PI021147, respectively.

Appendix.

Proof of Proposition 1: The conditional probability $p_G(C \mid x_1, \ldots, x_n)$ can be expressed in terms of the probabilities $p(C \mid pa_G(C))$ and $p(X \mid pa_G(X)), \forall X$ such that $C \in Pa_G(X)$. As the parent sets $Pa_G(C)$ and $Pa_G(X)$ are the same in the subgraph G_c, the first assertion follows immediately. Moreover, these parent sets must also be the same for any other classification-equivalent subgraph of G. As $Pa_{G_c}(Y) = \emptyset$ for any other node Y, it follows that G_c is a subgraph of H. □

Proof of Proposition 2: The first assertion is obvious. For the second, it is clear that given any DAG, G, we can find a C-DAG H such that $G \in \mathcal{C}_H$: $H = G_c$. Moreover, if $G \in \mathcal{C}_H \cap \mathcal{C}_{H'}$, then $H = G_c = H'$, and $\{\mathcal{C}_H\}$ is a partition. □

Proof of Proposition 3: *Necessary condition*: we assume that G is a C-RPDAG and we shall prove the four conditions 1–4 in Proposition 3. □

Condition 1: this is obvious from Condition 2 in Definition 3.

Condition 2: if $Pa_G(C) \neq \emptyset$, then $Sib_G(C) = \emptyset$ by using Condition 1 in Definition 3. Moreover, there is at least one arc $X \to C$ in G. Using Condition 4 in Definition 3 we obtain $|Pa_G(C)| \geq 2$ or $Pa_G(X) \neq \emptyset$. If $Pa_G(X) \neq \emptyset$, then we have an arc $Y \to X$ such that $Y \neq C$, $X \neq C$ and the arc $C \to X$ does not exist, which contradicts the fact that the extensions of G are C-DAGs. Hence, $|Pa_G(C)| \geq 2$.

Condition 3: if $Pa_G(X) \neq \emptyset$, $X \neq C$, then there is an arc $Y \to X$ in G and in any extension of G. As these extensions are C-DAGs, then either $Y = C$ or $\exists C \to X$. In either case $C \in Pa_G(X)$. Using Condition 4 in Definition 3 $|Pa_G(X)| \geq 2$ or $Pa_G(C) \neq \emptyset$. In this last case, the previous Condition 2 ensures that $|Pa_G(C)| \geq 2$.

Condition 4: if $Sib_G(X) \neq \emptyset$, $X \neq C$, then we obtain $Pa_G(X) = \emptyset$, using Condition 1 in Definition 3. Moreover, a link $Y—X$ exists in G. This link will be an arc in the extensions of G. As these extensions are C-DAGs, the only possible options are that either $Y = C$ or the h-h pattern $C \to Y \leftarrow X$ exists in some extension of G. The last option would imply that the h-h pattern also exists in G, and this contradicts the fact that $Y—X$ is a link and not an arc in G. Therefore, $C \in Sib_G(X)$ and C is the single sibling of X.

Sufficient condition: we assume the four conditions in Proposition 3, and we shall prove the four conditions in Definition 3 and that the extensions of G are C-DAGs.

Condition 1: assume that $Pa_G(X) \neq \emptyset$. If $X = C$ then Condition 2 in Proposition 3 guarantees that $Sib_G(X) = \emptyset$; if $X \neq C$, then Condition 3 in Proposition 3 asserts that $C \in Pa_G(X)$, hence $C \notin Sib_G(X)$. From Condition 4 in Proposition 3 we can now obtain $Sib_G(X) = \emptyset$.

Condition 2: It follows immediately from Condition 1 in Proposition 3.

Condition 3: if a completely undirected cycle exists in G, then at least three nodes have more than one sibling node. However, from Condition 4 in Proposition 3 we know that all the nodes except the class have at most one sibling.

Condition 4: assume that the arc $X \to Y$ exists in G. If $Y = C$ then $Pa_G(C) \neq \emptyset$, and Condition 2 in Proposition 3 ensures that $|Pa_G(Y)| = |Pa_G(C)| \geq 2$. If $X = C$ then $Pa_G(Y) \neq \emptyset$ and Condition 3 in Proposition 3 ensures that $|Pa_G(Y)| \geq 2$ or $|Pa_G(X)| = |Pa_G(C)| \geq 2$. If $X \neq C$ and $Y \neq C$ then from Condition 3 in Proposition 3 we know that $C \in Pa_G(Y)$, hence $|Pa_G(Y)| \geq 2$.

Therefore, G is an RPDAG. In order to prove that it is also a C-RPDAG, let us consider that its extensions are C-DAGs. Let H be an extension of G and assume that $X \to Y$ is an arc in H and $X \neq C$. This arc may correspond with either the arc $X \to Y$ or the link $X—Y$ in G. In the first case $Pa_G(Y) \neq \emptyset$, and from Condition 3 in Proposition 3 we obtain $C \in Pa_G(Y)$, hence the arc $C \to Y$ exists in G. In the second case, as $Sib_G(X) \neq \emptyset$, from Condition 4 in Proposition 3 we obtain $Sib_G(X) = \{C\}$, hence $Y = C$. Therefore, G is a C-DAG.

Proof of Proposition 4: As H and H' are extensions of an RPDAG, they are independence equivalent, so that $p_H(c, x_1, \ldots, x_n) = p_{H'}(c, x_1, \ldots, x_n)$, $\forall c, x_1, \ldots, x_n$, and therefore $p_H(c|x_1, \ldots, x_n) = p_{H'}(c|x_1, \ldots, x_n)$. □

Proof of Proposition 5: The twelve operators defined in the space of C-RPDAGs, except D_ChildOfC(X), are merely particularizations of the five operators defined by Acid and de Campos for RPDAGs. In effect, A_ParentOfC(X) =A_arc*(X,C)*, A_ChildOfC*(X)* = A_arc*(C, X)*, A_ParentOfChild*(X, Y)* = A_arc(*X, Y*), A_SiblingOfC(*X*) = A_link(*C, X*), A_HHOfC(*X, Y*) = A_hh(*X, C, Y*), A_HHOfChild(*X, Y*) = A_hh(*X, Y, C*), D_ParentOfC(*X*) = D_arc(*X, C*), D_ParentOfChild(*X, Y*) = D_arc(*X,Y*), D_HHOfC(*X, Y*) = D_arc(*X, C*), D_HHOfChild(*X, Y*) = D_arc(*X, Y)* and D_SiblingOfC(*X*) = D_link(*C, X*). Therefore, we can apply directly the corresponding results stated in Acid and de Campos (2003).

With respect to the operator D_ChildOfC(X), this involves removing not only the arc $C \to X$ but all the arcs in G pointing to X. Only the local score corresponding to the variable X must therefore be modified in order to reflect that the current parent set of X, $Pa_G(X)$ is transformed into $Pa_{G'}(X) = \emptyset$. This transformation gives the result stated in (h). □

Notes

1. There are also hybrid algorithms that use a combination of constraint-based and scoring-based methods (Acid & de Campos, 2001).
2. Although they are not equivalent in the usual sense, i.e. independence or distribution equivalence. See the next section.
3. If all the variables are discrete, as in our case, *independence equivalence* coincides with the concept of *distribution equivalence*, in the sense that each class represents a different set of (unrestricted multinomial) probability distributions.
4. A function that gives the same score to independent-equivalent DAGs.
5. A scoring function g is decomposable if the score of any Bayesian network structure may be expressed as a product (or a sum in the log-space) of local scores involving only one node and its parents, i.e. $g(G : D) = \sum_{X \in \mathbf{V}} g_D(X, Pa_G(X))$.

References

Acid, S., & de Campos, L. M. (2001). A hybrid methodology for learning belief networks: Benedict. *International Journal of Approximate Reasoning, 27*, 235–262.

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.

Aliferis, C. F., Tsamardinos, I., & Statnikov, A. (2003). HITON: A novel Markov blanket algorithm for optimal variable selection. In *Proceedings of the 2003 American Medical Informatics Association* (pp. 21–25).

Blake, C. L., & Merz, C. J. (1998). UCI Repository of machine learning databases. *http://www.ics.uci.edu/~mlearn/MLRepository.html*, University of California, Irvine, Dept. of Information and Computer Sciences.

Cheng, J., & Greiner, R. (1999). Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 101–108).

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

Chickering, D. M. (1996). Learning equivalence classes of Bayesian network structures. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (pp. 150–157).

Chickering, D. M. (2002). Learning equivalence classes of Bayesian network structures. *Journal of Machine Learning Research, 2*, 445–498.

Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory, 14*, 462–467.

Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–348.

de Campos, L. M., & Huete, J.F. (2000). A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning, 24*, 11–37.

Domingos, P., & Pazzani, M. J. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*, 103–130.

Elvira Consortium. (2002). Elvira: An environment for probabilistic graphical models. In *Proceedings of the First European Workshop on Probabilistic Graphical Models* (pp. 222–230).

Ezawa, K., Singh, M., & Norton, S. (1996). Learning goal oriented Bayesian networks for telecommunications risk management. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 139–147).

Fayyad, U. M., & Irani, K. B. (1993). Multi-valued interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Inteligence* (pp. 1022–1027).

Frey, L., Fisher, D., Tsamardinos, I., Aliferis, C. F., & Statnikov, A. (2003). Identifying Markov blankets with decision tree induction. In *Proceedings of the Third IEEE International Conference on Data Mining* (pp. 59–66).

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*, 131–163.

Good, I. J. (1965). *The estimation of probabilities*. Cambridge: The MIT Press.

Hand, D. J. (1981). *Discrimination and classification*. Wiley.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*, 197–243.

John, G. H., & Kohavi, R. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 121–129).

Keogh, E., & Pazzani, M. (2002). Learning augmented Bayesian classifiers. *International Journal on Artificial Intelligence Tools, 11*, 587–601.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1137–1143).

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*, 273–324.

Kohavi, R., John, G. H., Long, R., Manley, D., & Pfleger, K. (1994). MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence* (pp. 740–743).

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 284–292).

Kononenko, I. (1991). Semi-naive Bayesian classifier. In *Proceedings of the Second International Conference on Knowledge Discovery in Databases* (pp. 206–219).

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence, 10*, 269–293.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 223–228).

Langley, P., & Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 399–406).

Lucas, P. (2002). Restricted Bayesian network structure learning. In *Proceedings of the First European Workshop on Probabilistic Graphical Models* (pp. 117–126).

Pazzani, M. J. (1995). Searching for dependencies in Bayesian classifiers. *Lecture Notes in Statistics, 112*, 239–248.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. San Francisco, CA: Morgan Kaufmann.

Pearl, J., & Verma, T.S. (1990). Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence* (pp. 220–227).

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 335–338).

Sierra, B. & Larrañaga, P. (1998). Predicting survival in malignant skin melanoma using Bayesian networks automatically induced by genetic algorithms. An empirical comparison between different approaches. *Artificial Intelligence in Medicine, 14*, 215–230.

Singh, M., & Provan, G. M. (1996). Efficient learning of selective Bayesian network classifiers. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 453–461).

Spirtes, P., Glymour, C., & Scheines, R. (1993). *Causation, Prediction and Search*. Lecture Notes in Statistics 81. New York: Springer Verlag.

Zhang, N. L., Nielsen, T. D., & Jensen, F. V. (2004). Latent variable discovery in classification models. *Artificial Intelligence in Medicine, 30*, 283–299.

Accepted November 26, 2004