ELSEVIER International Journal of Approximate Reasoning 27 (2001) 235–262

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

A hybrid methodology for learning belief networks: BENEDICT

Silvia Acid, Luis M. de Campos *

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

Received 1 March 2000; accepted 1 March 2001

Abstract

Previous algorithms for the construction of belief networks structures from data are mainly based either on independence criteria or on scoring metrics. The aim of this paper is to present a hybrid methodology that is a combination of these two approaches, which benefits from characteristics of each one, and to develop two operative algorithms based on this methodology. Results of the evaluation of the algorithms on the well-known Alarm network are presented, as well as the algorithms performance issues and some open problems. © 2001 Elsevier Science Inc. All rights reserved.

Keywords: Belief networks; Learning; Independence; Scoring metrics; Minimum d-separating sets

1. Introduction

Graphical models such as belief networks [35] have become very attractive tools because of their ability to represent knowledge with uncertainty and to efficiently perform reasoning tasks. The knowledge that they manage is in the form of dependence and independence relationships, two basic notions in the human reasoning. Both relationships are coded by means of the qualitative component of the model, i.e., the directed acyclic graph (dag). The presence of links in the graph may represent the existence of direct dependency

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

PII: S0888-613X(01)00041-X

relationships between the linked variables (that sometimes may be interpreted as causal influence or temporal precedence), and the absence of some links means the existence of certain conditional independency relationships between the variables. The strength of the dependencies is measured by means of a collection of numerical parameters, usually conditional probabilities. This quantitative component allows to introduce uncertainty in the knowledge represented by the model.

The good performance of the inference processes when using belief networks is due to the existence of different algorithms that use local computations, which take advantage of the independences represented in the dag. In fact, independence modularizes knowledge in such a way that what is relevant is quickly identifiable, and easily accessible in the whole knowledge base. Moreover, independence allows a suitable factorization of the global numerical representation (a joint probability distribution) which gives rise to an important saving in the storage requirements. In addition to their methodological advantages, belief networks provide a very intuitive graphical representation of the available knowledge, reasons for the growing interest on these networks in the Artificial Intelligence community.

First of all, it is necessary to have a belief network in order to exploit it in reasoning tasks. For this purpose, in recent years a strong effort in automatic learning of belief networks from data has been invested. The reason is that partially automated learning methods may alleviate the knowledge acquisition bottleneck, common to most techniques for knowledge representation and reasoning. The result is a lot of learning algorithms from different approaches and different principles [11]. However, they can be grouped in two main approaches: methods based on conditional independence tests, and methods based on a scoring metric.

The algorithms based on *independence tests* (also called *constraint-based*) carry out a qualitative study on the dependence and independence properties among the variables in the domain, and then they try to find a network representing most of these properties. This quality isolates learning methods from the current formalism used to represent quantitative information. So, they take a list of conditional independence relationships as the input and the output is a graph displaying these relationships as far as possible. This list, in a learning process from a database, is obtained by means of conditional independence tests. They represent the major computational cost for these algorithms as for their number and their complexity. Unreliable results may be another consequence of the high complexity of the tests applied on the database (unless huge amounts of data were available). In spite of these drawbacks the methods based on independence tests, under certain conditions, have been proven to be accurate. Some of the algorithms based on this approach can be found in [12,23,29], which work on polytrees, [24,13] on simple graphs, and [15,40,43,44] for general dags.

The algorithms based on *scoring metrics* try to find a graph which has the minimum number of links that 'best' represents the data according to their own metric. They all use a function (the scoring metric) that measures the quality of each candidate structure and a heuristic search method to explore the space of possible solutions, trying to select the best one. During the search process, the scoring metric is applied to evaluate the fitness of each candidate structure to the data. So, each one of these algorithms is characterized by the specific scoring metric (there are very different expressions) and the search procedure used.

The algorithms based on a scoring metric are generally more efficient than the algorithms mentioned above, although given their greedy nature, they may not find the optimal solution, but a local optimal one. Some algorithms that use this approach may be found in [12,18,37], where the search space is restricted to trees or polytrees, and the scoring function is based on marginal and/or first-order conditional dependence measures (such as the Kullback–Leibler cross-entropy). Other algorithms, which work on general dags, almost invariably use greedy searches, and the scoring metrics are based on different principles, such as entropy [27], Bayesian approaches [10,19,22,25,26,33,36], or Minimum Description Length [8,21,32,42].

The algorithms we are going to describe in this paper do not fall clearly in any of these two categories but they utilize a hybrid methodology: they use a specific metric and a search procedure (so, they belong to the group of methods based on scoring metrics), although they also explicitly make use of the conditional independences embodied in the topology of the network to elaborate the scoring metric, and carry out independence tests to limit the search effort (hence they have also strong similarities with the algorithms based on independence tests). The only precedents we know about hybrid learning algorithms are the works in [38,41], where a scoring-based algorithm (K2 [19]) is, in some sense, inserted into a constraint-based one (PC [40]), with the objective of removing the need of an ordering of the variables.

The rest of the paper is organized as follows. In Section 2 we describe our methodology, which we have called BENEDICT. Section 3 presents two algorithms, BENEDICT-*dsep* and BENEDICT-*step*, for recovering the graph with the restriction that the ordering between the variables is given. Section 5 discusses the different rules used to stop the search and evaluation processes of our algorithms. In Section 6, we carry out a comparative study of the results obtained by both algorithms using the well-known Alarm network. Finally, Section 7 contains the concluding remarks and some proposals for future works.

2. The methodology

The basic idea of the methodology is to measure the discrepancies between the conditional independences represented in any given candidate network G

and the ones displayed by the database D. The lesser these discrepancies are, the better the network fits the data. The aggregation of all these (local) discrepancies will result in a measure $g(G,D)$ of global discrepancy between the network and the database. From this perspective, our proposal falls on the methods which use a scoring metric. So, this measure of global discrepancy will be used by a search procedure, which will explore the feasible solutions space searching for optimum (no discrepancy) or at least good (low discrepancy) solutions.

To construct our discrepancy-based scoring metric, first we have to study, on one hand, *what conditional independences* must be contrasted in both models, the graphical model and the numerical model (i.e., the database), and on the other hand, *how to measure* the discrepancy of these independences. Next, we will decide on how to aggregate all the local discrepancies. We also need to specify the kind of search process that our method will use.

In the graphical model, the independence assertions correspond to a graphical criterion, the d-separation criterion [35,43] for directed acyclic graphs:

- *d-separation*: Given a dag G, a chain C (a chain in a directed graph is a sequence of adjacent nodes, the direction of the arrows does not matter) from node α to node β is said to be blocked by the set of nodes Z, if there is a vertex $\gamma \in C$ such that, either
  - $\gamma \in Z$ and arrows of C do not meet head to head at γ, or
  - $\gamma \notin Z$, nor has γ any descendants in Z, and the arrows of C do meet head to head at γ.

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

In the underlying distribution of the numerical model, the concept of conditional independence, $I(X,Y\,|\,Z)$, corresponds to stochastic independence, i.e.,

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

for every instantiation x, y and z of the sets of variables X, Y and Z.

So, given a candidate network, in principle the conditional independence statements which have to be contrasted in the numerical model are all those obtained applying the d-separation criterion on this network. [1] Now, we need a metric to measure the discrepancy of each one of these independences. We are going to use the Kullback–Leibler cross-entropy [30], a function that measures the degree to which a specific independency is supported by the database.

[1] This set of independences will be considerably reduced later.

The Kullback–Leibler cross-entropy is defined as follows:

$$Dep(X, Y \mid Z) = \sum_{\mathrm{x},\mathrm{y},\mathrm{z}} P(\mathrm{x}, \mathrm{y}, \mathrm{z}) \log \frac{P(\mathrm{x}, \mathrm{y} \mid \mathrm{z})}{P(\mathrm{x} \mid \mathrm{z}) P(\mathrm{y} \mid \mathrm{z})}, \tag{1}$$

where x, y, z denote instantiations of the sets of variables X, Y and Z, respectively, and P is a probability estimated from the database. Observe that the value of $Dep(X, Y \mid Z)$ is zero if X and Y are indeed independent, given Z, and the more dependent X and Y are, given Z, the greater $Dep(X, Y \mid Z)$ is.

Given a d-separation statement $\langle X, Y \mid Z \rangle_G$, $Dep(X, Y \mid Z)$ measures the degree of dependence between X and Y, given that we know Z. The more strongly supported by the database is the independence, the smaller is the value of the measure, i.e., the discrepancy. We could also use other different dependency measures, such as those considered in [5] or in [1] which exhibit a similar behaviour. However, the Kullback–Leibler cross entropy has one important advantage from our point of view: it is possible to design an independence test based on this statistic, as we will see later.

To get an idea of the complexity of this initial proposal, let us consider the network in Fig. 1, defined over the set of variables $\mathscr{U} = \{A, B, C, E, P, T\}$. Some d-separation statements which are true in this model are:

$$\langle C, T \mid \emptyset \rangle, \langle C, E \mid \emptyset \rangle, \langle C, A \mid \emptyset \rangle, \langle C, P \mid \emptyset \rangle,$$
$$\langle E, B \mid T \rangle, \langle E, A \mid T \rangle, \langle E, P \mid T \rangle,$$
$$\langle B, A \mid T \rangle, \langle B, P \mid T \rangle, \langle A, P \mid T \rangle.$$

Other true d-separation statements, where the involved subsets of variables are not singletons are:

$$\langle E, B \mid \{T, A\} \rangle, \langle E, B \mid \{T, P\} \rangle, \langle E, B \mid \{T, C\} \rangle,$$
$$\langle E, B \mid \{T, A, P\} \rangle, \langle E, B \mid \{T, A, C\} \rangle, \ldots,$$
$$\langle E, \{A, P\} \mid T \rangle, \langle E, \{A, P, B\} \mid T \rangle, \ldots,$$
$$\langle E, \{A, P, B, C\} \mid T \rangle, \langle \{B, C\}, \{A, E, P\} \mid T \rangle, \ldots$$

As we can observe, the number of d-separation statements may grow exponentially with the number of nodes and also their complexity. Both are critical

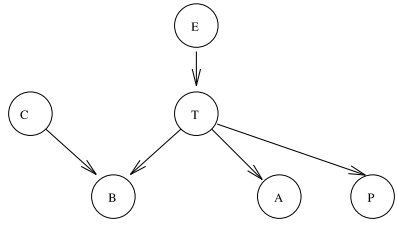

Fig. 1. An example of network structure.

factors to our methodology in order to configure the list of independences to be contrasted in the numerical model by means of $Dep(X, Y \mid Z)$. Therefore, we have to focus on some selected subset of 'representative' d-separation statements and ignore the remainders.

As a first approach to get the selected subset, we may use the concept of *causal input list* or *recursive basis* [43]: given a set of variables, $\mathscr{U} = \{x_1, x_2, \ldots, x_n\}$, and given an ordering l of the variables in $\mathscr{U}$, let $U_i = \mathrm{Pred}_l(x_i) = \{x_j \in \mathscr{U} \mid x_j <_l x_i\}$ be the set of predecessors of x_i in the ordering l, and let $B_i \subseteq U_i$ be a minimal subset satisfying the independence statement $I(x_i, U_i \setminus B_i \mid B_i)$ (i.e., given B_i, x_i is independent of all its predecessors that are not in B_i). The causal input list has the important property that the graph G formed by designing, for each i, the variables in the set B_i as the parents of x_i ($\pi_G(x_i) = B_i$), is a minimal I-map [35] of the underlying dependency model (i.e., all the d-separation statements found in the graph correspond to true independence relationships in the model). Moreover, all the d-separation statements displayed by this graph can be deduced from those in the causal input list, using the graphoid axioms [35], hence we have a completely representative set. So, our list of independences would be the set composed by n d-separation statements in the form: $\{\langle x_i, U_i \setminus B_i \mid B_i\rangle, i = 1, \ldots, n\}$. To sum up, we have reduced the initial set of independence statements to be considered (with exponential size) to n. The problem is that each one of these statements involves all the variables in the set $U_i \cup \{x_i\}$, and therefore the computation of $Dep(x_i, U_i \setminus \pi_G(x_i) \mid \pi_G(x_i))$ will take an exponential time.

A refinement of the selected set of independences is to reduce the complexity of the current statements (coming from the causal input list). For this purpose we can use the decomposition property (one of the graphoid axioms [34]), so that, supposing that $U_i \setminus \pi_G(x_i) = \{x_{i_1}, \ldots, x_{i_k}\}$, instead of considering the relationship $\langle x_i, U_i \setminus \pi_G(x_i) \mid \pi_G(x_i)\rangle_G$, we can decompose it as $\langle x_i, x_{i_j} \mid \pi_G(x_i)\rangle_G$, $\forall j = 1, \ldots, k$. In this way we considerably reduce the size of the sets involved in the calculus of the dependence measures, thus gaining in efficiency and reliability (although we have to compute $O(n^2)$ dependence measures instead of only $O(n)$). So, given any candidate network G, we have to calculate, from the database, the conditional dependence degrees of any two non-adjacent single variables, x_i and x_j (assuming that $x_j <_l x_i$ in the ordering l) given the parent set $\pi_G(x_i)$ of x_i in the network G, $Dep(x_i, x_j \mid \pi_G(x_i))$.

For example, for the network in Fig. 1, and assuming that the ordering l of the variables in $\mathscr{U}$ is (E, C, T, B, A, P), we would need to compute the following dependence degrees:

$$Dep(E, C \mid \emptyset), Dep(E, B \mid \{C, T\}), Dep(E, A \mid T), Dep(E, P \mid T),$$
$$Dep(C, T \mid E), Dep(C, A \mid T), Dep(C, P \mid T),$$
$$Dep(B, A \mid T), Dep(B, P \mid T), Dep(A, P \mid T).$$

However, we can try to further reduce the complexity of each independence statement, by reducing the size of the d-separating sets or cut-sets. Given two nodes x_i and x_j, such that $x_j <_l x_i$, instead of using the parent set of x_i, we could use the set of minimum size that still d-separates x_i and x_j. Of course, finding this set will take some additional effort, but it will be compensated by a decreasing computing time of the corresponding dependence degree. Moreover, it will also increase the reliability of the results, because less data are needed to reliably compute a conditional dependence measure of lower order. For the example in Fig. 1, the dependence degrees that would have to be calculated are the following:

$$Dep(E, C \mid \emptyset), Dep(E, B \mid T), Dep(E, A \mid T), Dep(E, P \mid T),$$
$$Dep(C, T \mid \emptyset), Dep(C, A \mid \emptyset), Dep(C, P \mid \emptyset),$$
$$Dep(B, A \mid T), Dep(B, P \mid T), Dep(A, P \mid T).$$

In many other cases, depending on the topology of the network, the savings may be quite remarkable, even cutting down the exponential complexity of the computations. For example, for the network displayed in Fig. 2, if we were to compute the conditional dependence measures of any pair of nodes given the parent set of the largest numbered node in the pair, this would represent 10 measures of order zero, 30 of order one and 5 of order five. However, if we use the minimum d-separating set instead of the parent set, we need 40 measures of order zero and 5 of order one.

So, in the general case, given a candidate network G and given an ordering l [2] on the set of nodes, for any pair x_i and x_j of non-adjacent nodes in G, such that $x_j <_l x_i$, we propose to replace $Dep(x_i, x_j \mid \pi_G(x_i))$ by $Dep(x_i, x_j \mid S_G(x_i, x_j))$, where $S_G(x_i, x_j)$ is a d-separating set for x_i and x_j in G with minimum size. The method we use for efficiently finding the sets $S_G(x_i, x_j)$ will be outlined in section 2.1 (see [2,3] for a detailed study of this matter).

In order to give a score g to a specific network structure G given a database D (that would measure the global discrepancy), we propose to define $g(G, D)$ as a weighted average of all the local discrepancies $Dep(x_i, x_j \mid S_G(x_i, x_j))$ being considered (the weighting factors could reflect some kind of confidence about the reliability of each local discrepancy). Finally, the type of search method we propose is a simple greedy search (which will become more definite in Section 3).

2.1. Finding d-separating sets of minimum size

The problem we are considering is the following: Find a d-separating set of minimum size for two non-adjacent nodes x_i and x_j in a dag G. To solve it we use a series of stepwise transformations of the original problem.

[2] Any ordering l coherent with the graph: if x_j is an ancestor of x_i then $x_j <_l x_i$.

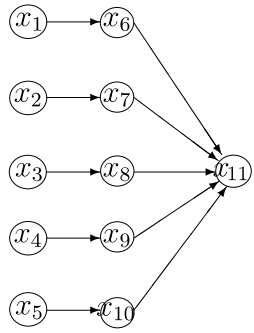

Fig. 2. Using minimum d-separating sets cuts down the complexity.

First, we reduce the problem to the simpler one [3] of finding a minimum separating set in an undirected graph. For this purpose the searching process has to be applied on the undirected graph $(G_{\mathrm{An}(\{x_i,x_j\})})^m$, where $(G_{\mathrm{An}(\{x_i,x_j\})})^m$ is the moral graph of the subgraph of G induced by $\mathrm{An}(\{x_i,x_j\})$ [2]; $\mathrm{An}(\{x_i,x_j\})$ is the smallest ancestral set of $\{x_i,x_j\}$, i.e., x_i, x_j and all their predecessors. Figs. 3 and 4 show an example of this transformation.

Second, we develop an algorithm for solving this new problem, `Minimum-cut-set`. To design the algorithm for solving the current problem we took into account the strong relationship that exists between problems of connectivity and problems of flow in graphs. The problem of finding the maximum flow between a pair of nodes in a graph (which is equivalent to finding a minimum cut-set) is efficiently solved by the well-known Ford–Fulkerson algorithm [20]. However, this result is not directly applicable, because the cited algorithm works on directed graphs while the state of the current graph $(G_{\mathrm{An}(\{x_i,x_j\})})^m$ is undirected, and also because it looks for cut-sets composed by arcs but not by nodes as we are interested in. Thus, we would have to transform separating arc-sets for directed graphs into separating node-sets for undirected graphs. This last transformation is illustrated in Fig. 5 for only one link, and in Fig. 6 for the graph $(G_{\mathrm{An}(\{x_3,x_{15}\})})^m$ (see [2,3] for details).

Our algorithm `Minimum-cut-set` does not really need to explicitly make all these transformations. It works on the original dag G where we want to test for d-separation and all the transformations are implicit, with the consequent savings and increment on performance.

[3] Simpler because the separation criterion is easier than the d-separation one as the separation criterion manages the independences in a more uniform way; on the other hand, the resultant graphs are simpler in complexity, number of nodes and links involved.

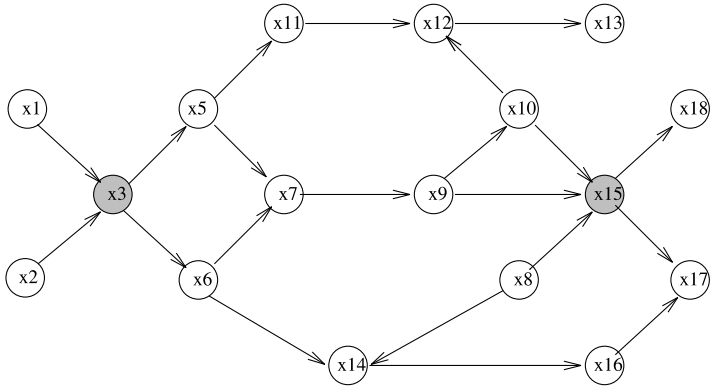

Fig. 3. Original graph G where to look for the minimum d-separating set of x_3 and x_{15}.

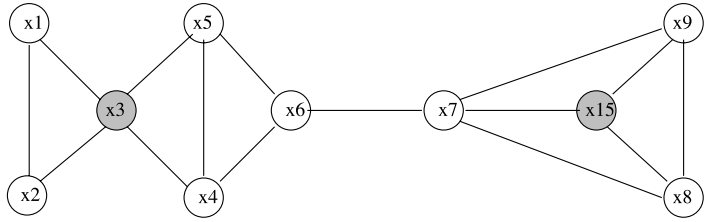

Fig. 4. Transformed undirected graph $(G_{\mathrm{An}(\{x_3,x_{15}\})})^m$ where to look for the minimum separating set of x_3 and x_{15}.

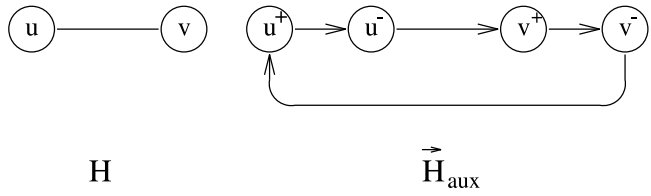

Fig. 5. Last transformation for one link.

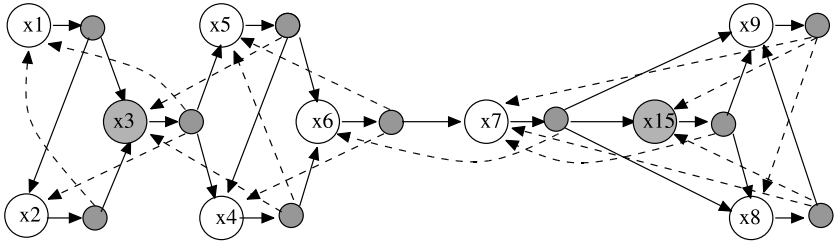

Fig. 6. Last transformation for G in Fig. 3, where the Ford–Fulkerson algorithm would have to be applied. The `Minimum-cut-set` algorithm does not use this graph but the original one.

2.2. The BENEDICT methodology

Although the main ideas of our learning methodology have already been presented, we will now try to summarize and systematize them. Thus, we are going to describe the different elements that compose the methodology for learning belief networks, which we have called BENEDICT. The name is an acronym of *BE*lief *NE*tworks *DI*scovery using *C*ut-set *T*echniques, and is motivated by the use of d-separating sets or cut-sets to define the metric.

The first two elements refer to the search process and the next three elements concern the scoring metric:

- A current candidate network G to be evaluated. The initial candidate network G_0 usually will be empty (containing no arcs), although we may start from any other network by fixing an initial set of arrows (thus imposing some constraints to the learning process: knowledge, supplied by an expert, about the existence of some arcs).
- A set L of candidate arcs to be introduced in the current network. This set will dynamically vary as the algorithm progresses. Initially, L may contain all the possible arcs or a more restricted set of arcs (in this case we impose another type of constraint to the learning: knowledge about the absence of some arcs or about the existence of some conditional independence relationships).
- A subset $\mathscr{M}_G$ of the set of valid graphical independence relationships in the current candidate network G, $\mathscr{M}_G \subseteq \mathscr{I}_G$, where

$$\mathscr{I}_G = \{\langle x_i, x_j \,|\, S_G(x_i, x_j)\rangle_G,\ i = 1, \ldots, n,\ x_j <_l x_i,\ x_j \notin \pi_G(x_i)\}, \tag{2}$$

  where $S_G(x_i, x_j)$ is a d-separating set for x_i and x_j in G with minimum size.
- A measure of (local) discrepancy between each $\langle x_i, x_j \,|\, S_G(x_i, x_j)\rangle_G$ and $I(x_i, x_j \,|\, S_G(x_i, x_j))$, i.e., between each graphical independence relationship in $\mathscr{M}_G$ and the database D. At present we are using the Kullback–Leibler cross-entropy $Dep(x_i, x_j \,|\, S_G(x_i, x_j))$ defined in Eq. (1).
- A measure of global discrepancy $g(G, D)$ (which has to be minimized) between the candidate network and the database, based on the aggregation of the local discrepancies. We define $g(G, D)$ as a weighted sum (the weighting factors could reflect some kind of confidence about the reliability of each local discrepancy or whatever):

$$g(G, D) = \sum_{\langle x_i, x_j \,|\, S_G(x_i, x_j)\rangle_G \in \mathscr{M}_G} \big(\alpha_{ij} * Dep(x_i, x_j \,|\, S_G(x_i, x_j))\big). \tag{3}$$

3. Algorithms based on the BENEDICT methodology

Now, our objective is to materialize the methodology proposed in the previous section into operative algorithms. We are going to develop two

algorithm, BENEDICT-*dsep* and BENEDICT-*step*. Both algorithms determine the network structure under the assumption that a total ordering l for the variables is known. This assumption, although somewhat restrictive, is quite frequent for learning algorithms [17,19,27,41,43,44]. We can understand this ordering as either causal (causes before effects) or temporal (determined by the occurrence time for each variable), although it may also be arbitrary. This ordering could be obtained, for example, from an expert; alternatively, it may be automatically constructed using a specific algorithm [7,14,16].

The two algorithms we are going to describe also share the following elements:

- The weights α_{ij} in Eq. (3) are all equal to 1 (the same confidence degrees for all the discrepancies).
- There is no prior knowledge about either the presence or the absence of some arcs in the network that we are looking for.
- The search process is greedy and allows to insert into the structure the candidate arc that produces a greater improvement of the score (removal of arcs is not permitted).

3.1. Algorithm BENEDICT-dsep

The first algorithm we are going to describe based on the BENEDICT methodology is called BENEDICT-*dsep*. As we have already commented, the search algorithm to explore the space of feasible solutions is greedy: the set of configurations explored at each step is obtained as the result of adding a new single arc to the previous better configuration. The selected candidate arc will be the one that produces the network with one extra arc having the best score. The initial set of candidate arcs in this case is the set of all the arcs compatible with the given ordering l, i.e.:

$$L = \{x_j \rightarrow x_i \mid x_j <_l x_i\}. \tag{4}$$

The subset $\mathscr{M}_G$ of valid graphical independence relationships considered for a given network G is $\mathscr{M}_G = \mathscr{I}_G$ (i.e., all the minimal d-separation relationships between non-adjacent nodes). Thus, the metric used by BENEDICT-*dsep* is

$$g(G,D) = \sum_{i=2}^{n} \sum_{\substack{x_j, x_j <_l x_i \\ x_j \notin \pi_G(x_i)}} Dep(x_i, x_j \mid S_G(x_i, x_j)). \tag{5}$$

Initially, BENEDICT-*dsep* starts out from a completely disconnected network, i.e., it is assuming marginal independence among all the variables. The scoring of this 'empty' graph G_0 (considered the current best network) is then calculated, from Eq. (5), as $g(G_0,D) = \sum_{i=2}^{n} \sum_{x_j <_l x_i} Dep(x_i, x_j \mid \emptyset)$. Next, the algorithm looks for the arc whose addition to the graph results in a greater decrease of $g(.,D)$, thus obtaining a new graph, G_1, containing only one arc. The process

continues in this way, adding at each step, k, the single arc, say $x_j \rightarrow x_i$, which verifies

$$g(G_{k-1} \cup (x_j \rightarrow x_i), D) = \min_{\substack{x_h \rightarrow x_l \in L \\ x_h \rightarrow x_l \notin G_{k-1}}} g(G_{k-1} \cup (x_h \rightarrow x_l), D).$$

Then, normally an arc $x_j \rightarrow x_i$ is added because the variables x_i and x_j are quite dependent on each other and because the current d-separation statements displayed by this (new) graph are more in agreement with the database. The addition of an arc always implies to eliminate an independence relationship from the previous configuration. If the dependence relationship were strong enough, then a high value $Dep(x_i, x_j \,|\, S_{G_{k-1}}(x_i, x_j))$ would be removed from the sum $g(G_{k-1}, D)$ after adding the arc $x_j \rightarrow x_i$ to G_{k-1}, thus favouring the selection of $x_j \rightarrow x_i$ as the best candidate arc.

In other cases the improvement of the score because of the addition of an arc may be less local to the eliminated independence. This arc changes the connectivity of the network, thus modifying the d-separation statements because of the change in the minimal d-separating sets. For example, let us consider the graph G_4 in Fig. 7, and suppose that we are going to add the arc $x_3 \rightarrow x_5$.

Table 1 shows which are the d-separation statements in the model before and after inserting the arc $x_3 \rightarrow x_5$. We can see how some d-separations remain unchanged but some others are modified, and one disappears. For this reason it is necessary to recompute completely the global discrepancy after the inclusion of a new arc (the effort to determine the terms of the sum which have changed is roughly equivalent to compute the global discrepancy). In other words, our metric is not decomposable [26].

In this way arcs are added in a stepwise process until a stopping rule is satisfied. The stopping rule we use, as well as some alternative rules, will be discussed later in Section 5. A pseudocode description of BENEDICT-*dsep* is given in Fig. 8.

The algorithm takes $\mathscr{U} = \{x_1, x_2, \ldots, x_n\}$, the ordering l of the variables and the database D as the inputs, and returns a network structure compatible with the ordering l as the output.

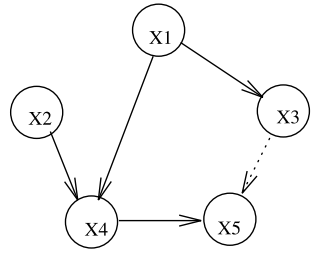

Fig. 7. Adding the arc $\{x_3 \rightarrow x_5\}$ to the graph G_4.

Table 1
List of d-separations before and after inserting an arc

Without $\{x_3 \rightarrow x_5\}$	With $\{x_3 \rightarrow x_5\}$
$\langle x_1, x_2 \mid \emptyset \rangle_{G_4}$	$\langle x_1, x_2 \mid \emptyset \rangle_{G_4 \cup \{x_3 \rightarrow x_5\}}$
$\langle x_1, x_5 \mid x_4 \rangle_{G_4}$	$\langle x_1, x_5 \mid x_3, x_4 \rangle_{G_4 \cup \{x_3 \rightarrow x_5\}}$
$\langle x_2, x_3 \mid \emptyset \rangle_{G_4}$	$\langle x_2, x_3 \mid \emptyset \rangle_{G_4 \cup \{x_3 \rightarrow x_5\}}$
$\langle x_2, x_5 \mid x_4 \rangle_{G_4}$	$\langle x_2, x_5 \mid x_3, x_4 \rangle_{G_4 \cup \{x_3 \rightarrow x_5\}}$
$\langle x_3, x_4 \mid x_1 \rangle_{G_4}$	$\langle x_3, x_4 \mid x_1 \rangle_{G_4 \cup \{x_3 \rightarrow x_5\}}$
$\langle x_3, x_5 \mid x_4 \rangle_{G_4}$	$\neg \langle x_3, x_5 \mid \mathscr{U}_5 \setminus \{x_3\} \rangle_{G_4 \cup \{x_3 \rightarrow x_5\}}$

1. *Let* $G_0 = (\mathcal{U}, \mathcal{E}_0)$, *where* $\mathcal{U} = \{x_1, x_2, \ldots, x_n\}, \mathcal{E}_0 := \emptyset$
2. *Let* $L = \{x_j \rightarrow x_i | x_j <_l x_i\}$
3. $g := 0$
4. *For* **each** *node* $x_t \in \mathcal{U}$ **do**
   (a) *For* **each** *node* $x_s \in Pred_l(x_t)$ **do**
      i. $g := g + Dep(x_t, x_s | \emptyset)$
5. $min := g$
6. $k := 1$
7. **while** *not stop* **do**
   (a) *For* **each** *arc* $x_j \rightarrow x_i \in L$ **do**
      i. $G'_k = (\mathcal{U}, \mathcal{E}_{k-1} \cup \{x_j \rightarrow x_i\})$
      ii. $g := 0$
      iii. *For* **each** *node* $x_t \in \mathcal{U}$ **do**
         For **each** *node* $x_s \in Pred_l(x_t) \setminus \pi_{G'_k}(x_t)$ **do**
            $S_{G'_k}(x_s, x_t) :=$*Minimum-cut-set*(x_s, x_t)
            $g := g + Dep(x_t, x_s | S_{G'_k}(x_s, x_t))$
      iv. **if** $g < min$ **then** {*select the best link*}
         $min := g$
         $X := x_i$
         $Y := x_j$
   (b) $\mathcal{E}_k := \mathcal{E}_{k-1} \cup \{Y \rightarrow X\}$
   (c) $L := L \setminus \{Y \rightarrow X\}$
   (d) $k := k + 1$

Fig. 8. Algorithm BENEDICT-dsep.

To study the algorithm complexity, consider a graph with n nodes. Since every pair of nodes may be adjacent, $O(n^2)$ arcs have to be considered before the best one could be added; when testing the addition of any arc, we have to

calculate the scoring $g(.,D)$ of the resultant graph, which amounts to the calculation of $O(n^2)$ values. Finally, if the algorithm does not stop before all the possible arcs have been added (for example, this would be the case if the underlying graph were complete), we have to perform $O(n^2)$ iterations, thus resulting in a worst case complexity of $O(n^6)$ (which does not include the dependence measure calculations; however, the dependence measures needed by BENEDICT are normally expected to be of low order. Note that if we use the dependence measures obtained directly from the causal input list, we would obtain a complexity of $O(n^5)$, but the cost of computing them would be exponential, and the result less reliable). As we will detail later, to save on computing time, the algorithm uses some additional heuristics; for example, if, at any stage, the algorithm detects an independence relationship between two variables (i.e., the dependence degree being nearly zero), then it removes the arc linking this pair of variables from the list L and it will never reconsider it as a candidate arc during the rest of the process.

4. Algorithm BENEDICT-step

The second algorithm also based on the BENEDICT methodology is called BENEDICT-*step*, which works under the same assumption as -*dsep* (the total ordering of the variables is known). The purpose of designing a new algorithm is to improve the efficiency of BENEDICT-*dsep* by a better use of the ordering. The BENEDICT-*dsep* algorithm exploits the ordering just to give an orientation to the links and therefore to obtain a saving in the process of exploration. However, the ordering may also guide the search process and save computing effort during the evaluation process.

The main difference of this algorithm with respect to the previous one is the different search processes, which we are going to describe now. Instead of starting out from the completely disconnected network with n nodes (containing all the variables), it will begin with a network composed of a single node (the first variable in the ordering). The first step consists in including a second node in the initial structure and then to evaluate the different graph configurations that may appear (in this first step we only have to evaluate two graphs: the empty graph with two nodes and the complete graph with two nodes). Once the best configuration with the current set of nodes is reached, another node is introduced, the best graph configuration with respect to the just introduced node and the previous ones, is looked for, and several arcs may be introduced. A new node is included and so on, until completing the set of all the n variables. So, the network is growing in nodes as well as in arcs. The searching process is very simple at the beginning and implies progressively more and more variables, therefore a saving in the total effort is obtained.

In this way we have a process composed of n steps, where each step i represents the inclusion of a new node x_i in the structure and the inclusion of the necessary arcs to construct the best graph with i nodes, G_i. Each step i has its own list of initial candidate arcs, $L_i = \{x_j \rightarrow x_i \,|\, x_j <_l x_i\}$ (only the arcs going to the just introduced node are candidate). Consequently, the set of d-separation relationships considered at each step i is also different: $\mathscr{M}_{G_i} = \{\langle x_i, x_j \,|\, S_{G_i}(x_i, x_j)\rangle_{G_i}, x_j <_l x_i, x_j \notin \pi_{G_i}(x_i)\}$ (only the relationships between the last node introduced and the previous ones).

This strategy of considering a node and all its predecessors before to include the next node in the structure (which resembles the search strategy of the K2 algorithm [19]) gives rise to a double saving with respect to BENEDICT-*dsep*. On one hand, the number of configurations (and their complexity) to explore and to evaluate has been reduced (every candidate arc is considered less times). On the other hand, once all the parents of a node x_i have been introduced at step i, the d-separation relationships (and the *Dep* values) of x_i with its other predecessors remain unchanged for subsequent steps. Thus, in order to measure the discrepancy of a network structure, these terms would be constant in the sum that defines the global discrepancy. Therefore, to discriminate between competing candidate networks, we only have to take into account the d-separation relationships which involve the last node introduced in the current set of nodes (i.e., the set $\mathscr{M}_{G_i}$ defined above). Hence the metric used by BENEDICT-step is

$$g(G_i, D) = \sum_{\substack{x_j, x_j <_l x_i \\ x_j \notin \pi_{G_i}(x_i)}} Dep(x_i, x_j \,|\, S_{G_i}(x_i, x_j)). \tag{6}$$

In this way the number of independences and the computational effort necessary to evaluate each candidate structure has decreased significantly. The pseudocode of BENEDICT-step is given in Fig. 9.

As we said, BENEDICT-*step* has a lower complexity order than BENEDICT-*dsep*. As it begins from a network with a single node, $n-1$ nodes remain to be included, and every time it is done, at step i, there are $i-1$ candidate arcs. So, in the worst case, if all the arcs have to be included, the while loop is repeated $i-1$ times. Each time it tries to include the $i-1-(|\mathscr{E}_i| - |\mathscr{E}_{i-1}|)$ remaining arcs in L_i. Finally, in order to compute the global discrepancy measure, the local discrepancies have to be computed for the $i-1-(|\mathscr{E}_i| - |\mathscr{E}_{i-1}|)-1$ pairs of non-adjacent nodes. Thus, we obtain a complexity of $O(n^4)$ for BENEDICT-*step*.

5. Stopping rules

In the description of the algorithms BENEDICT-*dsep* and BENEDICT-*step* we have not specified the way in which we stop the learning process,

1. *Let* $G_1^0 \equiv (\mathcal{U}_1, \mathcal{E}_1^h)$, *where* $\mathcal{U}_1 := \{x_1\}, \mathcal{E}_1^h := \emptyset$, *and* $h := 0$, *number of links*

2. *For* $i = 2$ *to* n *do*

   (a) *Let* $L_i = \{x_j \to x_i | x_j <_l x_i\}$

   (b) $\mathcal{U}_i := \mathcal{U}_{i-1} \cup \{x_i\}$

   (c) $g := 0$

   (d) *For* **each** *node* $x_j \in Pred_l(x_i)$ **do**

      i. $g := g + Dep(x_i, x_j | \emptyset)$

   (e) $min := g$

   (f) **While** *not stop* **do**

      i. *For* **each** *link* $x_j \to x_i \in L_i$ **do**

         A. $G_i^{h'} = (\mathcal{U}_i, \mathcal{E}_{i-1}^h \cup \{x_j \to x_i\})$

         B. $g := 0$

         C. *For* **each** *node* $x_k \in Pred_l(x_i) \setminus \pi_{G_i^{h'}}(x_i)$ *do*
            $S_{G_i^{h'}}(x_k, x_i) :=$ *Minimum-cut-set*(x_k, x_i)
            $g := g + Dep(x_i, x_k | S_{G_i^{h'}}(x_k, x_i))$

         D. **if** $g < min$ **then**
            $min := g$
            $X := x_i$
            $Y := x_k$

      ii. $h := h + 1$

      iii. $\mathcal{E}_i^h := \mathcal{E}_i^{h-1} \cup \{Y \to X\}$

      iv. $L_i := L_i \setminus \{Y \to X\}$

Fig. 9. Algorithm BENEDICT-step.

i.e., how to decide whether the size of the graph is adequate and then do not add more arcs to the current structure. The extent to which the resultant network is a good description of the domain strongly depends on this question.

For example, if we let the graph grow until no more arcs can be included (thus obtaining a complete graph), this network has a zero global discrepancy with every database, since it does not explicitly represent any independence relationship. However, this network is far from being descriptive (although it may represent any independence relationship, that remains hidden in the conditional probability tables). In general, if we stop the learning process too late the resultant networks are very complex to be understood, estimated and used for inference purposes (moreover, an overfitting phenomenon can appear). On the other hand, if we stop too early the growing process, in order to get a simpler structure, we may lose the chance

to improve it (i.e., to get a better score). So, the network displays more independences than those they are really supported by the data, thus resulting in a bad approximation of the distribution underlying the data. Therefore, we have to focus on finding appropriate stopping rules, or in other words, on finding a criterion for determining a good trade-off between network complexity and accuracy.

The first stopping rule we tried consisted in setting a threshold α and deciding not to include an arc if the global discrepancy with the data, $g(G_k, D)$, was beneath α. This means that if the discrepancy is quite low, close to zero, then all the d-separation statements considered in the current graph are almost completely supported by the database, hence this graph is already a good approximation, and we should stop adding arcs. A second alternative, also using another threshold β, was not to include an arc if the decreasing of discrepancy in two consecutive steps was below β, $g(G_{k-1,}, D) - g(G_k, D) < \beta$, i.e., no significant improvement is achieved by adding the best arc to the current graph, so that we should also stop the process.

After having studied the behaviour of our algorithms using the two aforementioned stopping rules [4] (and using also a combination of both rules), by means of several databases, we get clear that these rules produced in general unsatisfactory results. The problem has two aspects: First, the threshold values are critical to the density of the learned networks; if the threshold α is set too low, then the algorithms stop essentially when there are very few terms in the summatory representing the global discrepancy, which in turn entails that a lot of arcs are added to the structure (and this may hide real independences). However, if the value of the threshold α is high we may stop too early, thus including very few arcs in the graph (hence the network would represent more independences than the ones really supported by the database). Using β instead of α results in a similar behaviour. Second, it is quite difficult to determine automatically, from the data distribution, their appropriate values.

Thus, we decided to tackle the problem of finding the optimum size of the learned network from another perspective. Instead of looking for the right stopping rule (or the right threshold) we will use two different procedures that have a direct influence on the size of the network. These are:

- Using independence tests to remove arcs from the set of candidate arcs, and thus avoiding to include them in the structure regardless of the value of the discrepancy reached by the current network.
- Pruning instead of stopping. Grow a network in a not very restrictive manner and then prune it by revisiting each arc (dependency) established in the building process, in order to check whether it was introduced prematurely. This is inspired by the common practice of using methods for pruning decision trees, which often produce excellent results [9].

5.1. Independence tests

The basic idea for using independence tests is to consider as candidate, at every step of the learning process, only the arcs between pairs of (non-adjacent) variables which have not been found independent. In the case that the test establishes an independence relationship, then the arc linking the independent variables is deleted from the set of candidate arcs L. Therefore, the learning process will stop naturally when the set of candidate arcs becomes empty (the arcs are removed from this set either because they are inserted into the current graph or because their extreme nodes are found to be independent by the test).

In order to define the kind of test, we will take advantage of the fact that the statistic $2 * N * Dep(x_i, x_j | Z)$ is approximately χ^2-distributed with $\|Z\|(\|x_i\| - 1)(\|x_j\| - 1)$ degrees of freedom [31], where N is the number of samples in the database D, and $\|\cdot\|$ represents the number of different states in the corresponding set of variables.

Let us see how to use the independence tests in our algorithms: At each step, once the best candidate arc $x_j \rightarrow x_i$ has been selected, it is included in the current structure G (and removed from the set of candidate arcs); then the remaining candidate arcs in L are reexamined in order to see if some may be removed because their extreme nodes are found to be independent by the test (the newly introduced arc changes the connectivity of the network, and can turn into independent variables that were not yet determined as independent). If $G' = G \cup \{x_j \rightarrow x_i\}$ denotes the new current graph, then for any pair of nodes x_h and x_l such that $x_h \rightarrow x_l \in L$, we apply an independence test based on the value $Dep(x_l, x_h | S_{G'}(x_l, x_h))$. As all these values have already been calculated, the tests entail almost null additional computational cost.

5.2. The pruning process

With the same goal to get the optimal size of a network during the learning task, the new approach is to use a pruning process. The first step is to grow the network by letting the building process (any of the BENEDICT algorithms) to work with a non-very restrictive stopping criterion (for instance that the set of candidate arcs is empty). In the resultant structure G, some right arcs will be introduced and also some additional (incorrect) ones. Then a review of the arcs included in the structure may reveal that some of them are superfluous and thus can be removed from the network.

An arc $x_j \rightarrow x_i$ in G may be considered superfluous if the nodes it connects are conditionally independent (i.e., the dependence degree between x_i and x_j is very low). Perhaps this fact was not discovered before because, due to the greedy nature of the process, the d-separating set used previously to try to separate these nodes was not correct. However, the information obtained in subsequent steps (new arcs have been introduced) changes the connectivity of

the network and therefore may also change the relationships between the variables. So, to verify the possible independence relationship between the linked variables x_i and x_j, the arc $x_j \rightarrow x_i$ is removed momentarily and the *Dep* value is calculated to be used in an independence test. If the independency is true, the arc is removed definitively, otherwise it is put back in the network. The value *Dep* we compute is the following: let $G' = G \setminus \{x_j \rightarrow x_i\}$, then we determine the minimum d-separating set between x_i and x_j in G', $S_{G'}(x_i, x_j)$ and compute $Dep(x_i, x_j | S_{G'}(x_i, x_j))$.

In summary, the pruning process consists of reviewing the arcs, one by one, in the same ordering that they were introduced in the graph, and applying a test of conditional independence on the nodes connected by each arc. For the pruning process, every arc in G is a candidate to be removed. When a link is definitively removed, we get the new graph G', a subgraph of G. In this case G is an I-map of G', i.e., every d-separation assertion true in G is also true in G'. Therefore, the pruning process consists of a set of local verifications by means of tests, that in no case affect the rest of the structure. The ordering to applying these tests is precisely the ordering in which the arcs were introduced in the structure: in this way we can reexamine the established dependences in the light of the information obtained after any arc $x_j \rightarrow x_i$ is introduced (so, the d-separating set that, at the moment in which $x_j \rightarrow x_i$ was inserted, was not able to make independent x_i and x_j could change, thus being able to render x_i and x_j independent). Finally, in order to recover the right ordering to prune is necessary to remember the arcs introduced during the building process, by means of a list of arcs $\mathscr{P}$.

To illustrate how the independence tests and the pruning process [4] are integrated in the BENEDICT algorithms, Fig. 10 shows the pseudocode of BENEDICT-*dsep* using these elements to make up the stopping rule.

The pseudocode of the pruning process is shown in Fig. 11. The inputs to the algorithm are the graph G, and the ordered list of arcs $\mathscr{P}$, both elaborated by the learning process.

6. Experimental results

In this section, we are going to present the results obtained by our algorithms using different databases in the learning process. In order to do a comparative study of the proposed algorithms, all the experiments were carried out on a sun4m sparc station at 100 MHz, to reconstruct the so-called Alarm

[4] Both approaches can be used together to improve even more their performance. In fact, combining independence tests and pruning produces networks that, in our experiments, have been close to the optimal size, as we shall see in the next section.

1. *Let* $G_0 = (\mathcal{U}, \mathcal{E}_0)$, *where* $\mathcal{U} = \{x_1, x_2, \ldots, x_n\}, \mathcal{E}_0 := \emptyset$
2. *Let* $L = \{x_j \rightarrow x_i | x_j <_l x_i\}$
3. $g := 0$
4. *For* **each** *node* $x_t \in \mathcal{U}$ **do**
   (a) *For* **each** *node* $x_s \in Pred_l(x_t)$ **do**
      i. $g := g + Dep(x_t, x_s|\emptyset)$
      ii. **if** $I(x_t, x_s|\emptyset)$ **then** $L := L \setminus \{x_s \rightarrow x_t\}$
5. $min := g$
6. $k := 1$
7. **while** $L \neq \emptyset$ **do**
   (a) *For* **each** *arc* $x_j \rightarrow x_i \in L$ **do**
      i. $G'_k = (\mathcal{U}, \mathcal{E}_{k-1} \cup \{x_j \rightarrow x_i\})$
      ii. $g := 0$
      iii. *For* **each** *node* $x_t \in \mathcal{U}$ **do**
         For **each** *node* $x_s \in Pred_l(x_t) \setminus \pi_{G'_k}(x_t)$ **do**
         $S_{G'_k}(x_s, x_t) :=$ *Minimum-cut-set*(x_s, x_t)
         $g := g + Dep(x_t, x_s|S_{G'_k}(x_s, x_t))$
      iv. **if** $g < min$ **then**
         $min := g$
         $X := x_i$
         $Y := x_j$
   (b) $\mathcal{E}_k := \mathcal{E}_{k-1} \cup \{Y \rightarrow X\}$
   (c) $L := L \setminus \{Y \rightarrow X\}$
   (d) $\mathcal{P} := \mathcal{P} + \{Y \rightarrow X\}$
   (e) *For* **each** *arc* $x_s \rightarrow x_t \in L$
      $S_{G_k}(x_s, x_t) :=$ *Minimum-cut-set*(x_s, x_t)
      if $I(x_t, x_s|S_{G_k}(x_s, x_t))$ **then** $L := L \setminus \{x_s \rightarrow x_t\}$
   (f) $k := k + 1$
8. *Prune*$(G, \mathcal{P})$

Fig. 10. Algorithm BENEDICT-dsep using independence tests and pruning.

belief network (see Fig. 12). This network displays the relevant variables and relationships for the Alarm Monitoring System [6], a diagnostic application for patient monitoring. The Alarm network contains 37 variables and 46 arcs.

The input data commonly used are subsets of the Alarm database [28], which contains 20,000 cases that were stochastically generated using the Alarm network.

1. *For* **each** *arc* $\{Y \to X\}$ *in* $\mathcal{P}$ **do**
   (a) $\mathcal{E} := \mathcal{E} \setminus \{Y \to X\}$
   (b) $S_G(Y, X) :=$ *Minimum-cut-set*(Y, X)
   (c) **if** $\neg I(Y, X | S_G(Y, X))$ **then** $\mathcal{E} := \mathcal{E} \cup \{Y \to X\}$

Fig. 11. Pruning process.

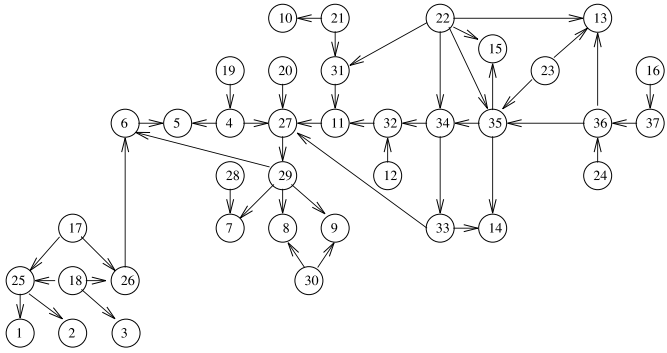

Fig. 12. The Alarm network.

The information we will show about each experiment with our algorithms is the following:

- *Time*. The time, measured in minutes and seconds, spent by the algorithm when learning the Alarm network.
- $g(G, D)$. The value of the measure of global discrepancy, $g(G, D)$, used by the algorithm, on each learned network G with respect to the database D it comes from.
- *Ham*. The Hamming distance, number of different arcs (missing or added) in the learned network with respect to the original model.
- *N.Arc.* Number of arcs in the learned network.
- *Kull.* The Kullback distance (cross-entropy) between the probability distribution, P, associated to the database (the empirical frequency distribution) and the probability distribution associated to the learned network, P_G. In this way we try to assess the performance of the algorithms, not only from the perspective of how much of the target network is reconstructed, but also of how closely the probability distribution learned approximates the empirical frequency distribution. Actually, we have calculated a decreasing monotonic linear transformation of the Kullback distance, because this one has exponential complexity and the transformation can be computed in a very efficient form: If P_G is the joint probability distribution associated to a network G whose set of nodes is $\mathcal{U} = \{x_1, \ldots, x_n\}$, then the Kullback distance can be written in the following way:

$$\mathrm{Kull}(P,P_G) = -H_P(\mathscr{U}) + \sum_{i=1}^{n} H_P(x_i) - \sum_{i=1,\pi_G(x_i)\neq\emptyset}^{n} Dep(x_i,\pi_G(x_i)\,|\,\emptyset), \qquad (7)$$

where H_P denotes the Shannon entropy with respect to the distribution P. As the first two terms of the expression above do not depend on the graph G, our transformation consists in calculating only the third term in Eq. (7). So, the interpretation of our transformation of the Kullback distance is: the higher this parameter is the better is the network.

We will use databases of different sizes for training, with the aim of evaluating the robustness of the algorithms. More precisely, we use, to reconstruct the Alarm network, four training sets containing the first 500, 1000, 2000 and 3000 cases in the Alarm database.

Our first experiment concentrates around the stopping rules, aiming to determine the quality of the different rules proposed: using thresholds (α,β), independence tests, and pruning. In the first case we use a conjunction of the two thresholds (low discrepancy and no significant improvement). Both thresholds were set 'ad hoc' as a fraction of the initial discrepancy calculated on the empty graph. We use the χ^2 test for the conditional independence tests and the pruning process with a fixed confidence level of 0.99. The results are displayed in Table 2 for BENEDICT-*dsep* and Table 3 for BENEDICT-*step*. For this experiment we used the database containing 3000 cases.

From these results it is clear that the use of independence tests improves remarkably the performance of the algorithms with respect to the use of thresholds, as much in the quality of the results as in the efficiency. They present lower discrepancy values and also higher values of the objective Kullback measure. On the other hand, we can observe that the times spent are

Table 2
Comparing stopping rules: results using BENEDICT-*dsep*

	Time	$g(G,D)$	Kullback	Ham.	N.Arc
(α,β)	20:58	1.3754	8.2329	17	31
Indep.	7:10	0.6995	9.2213	10	50
Indep. + Pruning	7:12	0.6818	9.2036	4	44

Table 3
Comparing stopping rules: results using BENEDICT-*step*

	Time	$g(G,D)$	Kullback	Ham.	N.Arc
(α,β)	14:58	4.6205	6.2994	27	31
Indep.	5:04	0.6954	9.2143	8	48
Indep. + Pruning	5:20	0.6923	9.2077	5	45

markedly lower, specially with BENEDICT-*step*, than using thresholds (α, β). This gives us an idea of the savings in evaluating candidate networks obtained by removing some arcs from the candidate set L. We can also see that final process of pruning, using almost null additional time, simplifies the final structures, although slightly decreasing the Kullback distances; anyway, the results are comparable. Furthermore, using pruning overcomes, at least partially, some of the problems due to the use of the irrevocable search strategy selected. In the light of these results, in the following experiments we will always use the combination of independence tests and pruning as the stopping rule.

In the second experiment we are interested in evaluating the impact of using d-separating sets of minimum size, $S_G(x_i, x_j)$, against directly using the parent set $\pi_G(x_i)$. Then we have also implemented versions of BENEDICT-*dsep* and BENEDICT-*step* using $\pi_G(x_i)$ instead of $S_G(x_i, x_j)$ to measure the dependence degree between the non-adjacent nodes x_i and x_j. The results are displayed in Table 4 for BENEDICT-*dsep* and Table 5 for BENEDICT-*step*. For this experiment we also used the database containing 3000 cases.

The obtained results are quite conclusive: the use of d-separating sets of minimum size is clearly preferable to the use of parent sets, not only with respect to the quality of the learned network (lower values of the Hamming distance and greater values of the transformed Kullback distance) but also in terms of efficiency (the use of d-separating sets of minimum size reduces around four times the execution time of the algorithm).

In the third experiment our interest is focused mainly in evaluating the performance of the algorithms using databases of different sizes. As we already mentioned, we have used four sizes (500, 1000, 2000 and 3000). The results of this experiment are displayed in Table 6 for BENEDICT-*dsep* and Table 7 for BENEDICT-*step*.

Several conclusions may be drawn from this experiment. First, it seems that the learned networks approach the original one as the sample size increases, as

Table 4
Comparing minimum d-separating set versus parent sets using BENEDICT-*dsep*

BENEDICT-*dsep*	Time	Kullback	Ham.	N.Arc
Using $\pi_G(x_i)$	27:16	8.7720	6	42
Using $S_G(x_i, x_j)$	7:12	9.2036	4	44

Table 5
Comparing minimum d-separating set versus parent sets using BENEDICT-*step*

BENEDICT-*step*	Time	Kullback	Ham.	N.Arc
Using $\pi_G(x_i)$	22:40	9.1961	7	43
Using $S_G(x_i, x_j)$	5:20	9.2077	5	45

Table 6
Using different sample sizes: results with BENEDICT-*dsep*

Sample Size	Time	$g(G,D)$	Kullback	Ham.	N.Arc
500	2:22	2.7586	8.9759	11	43
1000	3:49	2.4367	9.0927	9	45
2000	5:06	0.9605	9.1345	5	45
3000	7:12	0.6818	9.2036	4	44

Table 7
Using different sample sizes: results with BENEDICT-*step*

Sample size	Time	$g(G,D)$	Kullback	Ham.	N.Arc
500	0:36	3.2907	8.8019	13	43
1000	1:54	2.4367	9.0927	9	45
2000	3:28	0.9599	9.1351	7	45
3000	5:20	0.6923	9.2077	5	45

shown by the Hamming distances and the Kullback measures, in both tables. The time employed by the two algorithms is almost linear with respect to the sample size. Nevertheless, as we could expect, the running time of BENEDICT-*step* is significatively lower than the one of BENEDICT-*dsep*. However, the quality of the resultant networks, judging by the rest of the parameters, is very similar in both cases.

Specifically, for the training set of 3000 samples, the mistaken arcs are in common to both algorithms. There are three missing arcs, $11 \rightarrow 27$, $12 \rightarrow 32$ and $21 \rightarrow 31$. The last two arcs are not strongly supported by the data, as it was reported by several authors. There are one or two additional erroneous arcs, linking variables which have not been determined as independent by the tests. These arcs are $31 \rightarrow 27$ for BENEDICT-*dsep*, and $24 \rightarrow 9$ and $31 \rightarrow 27$ for BENEDICT-*step* (the arc $31 \rightarrow 27$ is set in both cases while trying to compensate the loss of the arc $11 \rightarrow 27$).

7. Concluding remarks

We have proposed a new hybrid methodology for learning Bayesian belief networks from databases, and two algorithms based on it. These algorithms belong to the class of learning algorithms that use a scoring metric and heuristic search to find a good graphical representation for the database. However, they also exhibit several characteristics related to the class of learning algorithms based on independency relationships. As we explained, our algorithms use conditional independence relationships of order as low as possible, by using the `Minimum-cut-set` algorithm. This fact benefits our algorithms, gaining in efficiency and also in reliability.

The algorithm BENEDICT-*dsep* as well as the algorithm BENEDICT-*step* recovers reasonably well complex belief network structures from data in polynomial time. Both algorithms work under the assumption that the ordering among the variables is known. However, BENEDICT-*step* is more efficient in exploiting the ordering and reaches a better performance than BENEDICT-*dsep*.

Several possible extensions of this work, that we plan to study in the future, include:

- Avoiding the requirement of an ordering on the nodes. One approach is to leave the search procedure to also cope with the directions of the arcs to be added, although this may considerably increase the search effort. For this task the algorithm BENEDICT-*dsep* would act as the starting point (because BENEDICT-*step* is so dependent on the ordering that it cannot be generalized to work without this requirement). Another option is to perform a two-step process: the first one is to run an algorithm for learning a good ordering [7,14,16], the second one would be carried out by one of our algorithms (in this case BENEDICT-*step* would probably be more appropriate).
- Modification of the pure greedy search used for arc addition. Sometimes the algorithm selects one arc instead of another one by a very small difference in the respective scorings. It might be interesting in these cases to design another criterion to 'break the tie', or to postpone the decision until it becomes clearer in subsequent steps. Anyway, the search process may be trapped in a local minimum. Other search techniques that may escape from local minima, such as branch and bound or simulated annealing, could be used in combination with our scoring metric. We could modify the search process in a different direction, by using not only arc addition but allowing more operations, as arc removal (or even arc reversal).
- More empirical work to investigate the practicality of our approach in real domains, particularly in the field of classification. For this kind of problems, our algorithms (and all other algorithms for learning belief networks) should probably be modified to give them a more classification-oriented perspective.

Acknowledgements

This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) under Project No. TIC96-0781.

References

[1] S. Acid, Métodos de aprendizaje de redes de creencia. Aplicación a la clasificación, Ph.D. thesis, Universidad de Granada, Spain, 1999 (in Spanish).

[2] S. Acid, L.M. de Campos, An algorithm for finding minimum d-separating sets in belief networks, in: E. Horvitz, F. Jensen (Eds.), Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, 1996, pp. 3–10.
[3] S. Acid, L.M. de Campos, An algorithm for finding minimum d-separating sets in belief networks, Decsai Technical Report 960214, Universidad de Granada, 1996, pp. 1–22.
[4] S. Acid, L.M. de Campos, Benedict: an algorithm for learning probabilistic belief networks, in: Proceedings of the Sixth IPMU Conference, 1996, pp. 979–984.
[5] S. Acid, L.M. de Campos, A. González, R. Molina, N. Pérez de la Blanca, Learning with castle, in: R. Kruse, P. Siegel (Eds.), Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science 548, Springer, Berlin, 1991, pp. 99–106.
[6] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, G.F. Cooper, The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks, in: European Conference on Artificial Intelligence in Medicine, 1989, pp. 247–256.
[7] R. Bouckaert, Optimizing causal orderings for generating dags from data, in: D. Dubois, M.P. Wellman, D.B. D'Ambrosio (Eds.), Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, Morgan-Kaufmann, San Mateo, 1992, pp. 9–16.
[8] R.R. Bouckaert, Belief networks construction using the minimum description length principle. in: M. Clarke, R. Kruse, S. Moral (Eds.), Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science 747, Springer, Berlin, 1993, pp. 41–48.
[9] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth Statistics Probability Series, Belmont, 1984.
[10] W. Buntine, Operations for learning with graphical models, Journal of Artificial Intelligence Research 2 (1994) 159–225.
[11] W. Buntine, A guide to the literature on learning probabilistic networks from data, IEEE Transactions on Knowledge and Data Engineering 8 (1996) 195–210.
[12] L.M. de Campos, Independency relationships and learning algorithms for singly connected networks, Journal of Experimental and Theoretical Artificial Intelligence 10 (4) (1998) 511–549.
[13] L.M. de Campos, J.F. Huete, On the use of independence relationships for learning simplified belief networks, International Journal of Intelligent Systems 12 (1997) 495–522.
[14] L.M. de Campos, J.F. Huete, Aproximación de una ordenación de variables en redes causales mediante algoritmos genéticos, Inteligencia Artificial 4 (1998) 30–39 (in Spanish).
[15] L.M. de Campos, J.F. Huete, A new approach for learning belief networks using independence criteria, International Journal of Approximate Reasoning 24 (2000) 11–37.
[16] L.M. de Campos, J.F. Huete, Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing, in: Proceedings of the Eighth IPMU Conference, vol. I, 2000, pp. 333–340.
[17] J. Cheng, D.A. Bell, W. Liu, An algorithm for Bayesian belief network construction from data, in: Proceedings of AI and STAT'97, 1997, pp. 83–90.
[18] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE transactions on Information Theory 14 (1968) 462–467.
[19] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (4) (1992) 309–348.
[20] L.R. Ford, D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ, 1962.
[21] N. Friedman, M. Goldszmidt, Learning Bayesian networks with local structure. in: E. Horvitz, F. Jensen (Eds.), Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, 1996, pp. 252–262.
[22] D. Geiger, D. Heckerman, A characterisation of the Dirichlet distribution with application to learning Bayesian networks, in: P. Besnard, S. Hanks (Eds.) Proceedings of the Eleventh

Conference on Uncertainty in Artificial Intelligence, Morgan and Kaufmann, San Francisco, 1995, pp. 196–207.
[23] D. Geiger, A. Paz, J. Pearl, Learning causal trees from dependence information. in: Eighth National Conference on Artificial Intelligence (AAAI 90), 1990, pp. 770–776.
[24] D. Geiger, A. Paz, J. Pearl, Learning simple causal structures, International Journal of Intelligent Systems 8 (1993) 231–247.
[25] D. Heckerman, Bayesian networks for knowledge discovery, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, MA, 1996, pp. 273–305.
[26] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Machine Learning 20 (1995) 197–243.
[27] E. Herskovits, G.F. Cooper, Kutató: An entropy-driven system for the construction of probabilistic expert systems from databases, in: P. Bonissone (Ed.), Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, Cambridge, 1990, pp. 54–62.
[28] E.H. Herskovits, Computer-based probabilistic networks construction, Ph.D. thesis, Medical Information Sciences, Stanford University, 1991.
[29] J.F. Huete, L.M. de Campos, Learning causal polytrees, in: M. Clarke, R. Kruse, S. Moral (Eds.), Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science 747, Springer, Berlin, 1993, pp. 180–185.
[30] S. Kullback, R.A. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1951) 79–86.
[31] S. Kullback, Information Theory and Statistics, Dover Publication, New York, 1968.
[32] W. Lam, F. Bacchus, Learning Bayesian belief networks. An approach based on the mdl principle, Computational Intelligence 10 (4) (1994) 269–293.
[33] D. Madigan, A. Rafterym, Model selection and accounting for model uncertainty in graphical models using Occam's window, Journal of the American Statistics Association 89 (1994) 1535–1546.
[34] J. Pearl, A. Paz, Graphoids: A graph-based logic for reasoning about relevance relations, Technical Report 850038 (R-53-L), Cognitive Systems Laboratory, UCLA, 1985.
[35] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Morgan Kaufmann, Morgan Kaufmann, San Mateo, 1988.
[36] M. Ramoni, P. Sebastiani, Learning Bayesian networks from incomplete databases, in: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Morgan and Kaufman, San Mateo, 1997, pp. 401–408.
[37] G. Rebane, J. Pearl, The recovery of causal poly-trees from statistical data, in: L.N. Kanal, T.S. Levitt, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, vol. 3, 1987, pp. 222–228.
[38] M. Singh, M. Valtorta, An algorithm for the construction of Bayesian network structures from data, in: D. Heckerman, A. Mamdani (Eds.), Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, 1993, pp. 259–265.
[39] M. Singh, M. Valtorta, Construction of Bayesian network structures from data: a brief survey and an efficient algorithm, International Journal of Approximate Reasoning 12 (1995) 111–131.
[40] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction and Search, Lecture Notes in Statistics 81, Springer, New York, 1993.
[41] S. Srinivas, S. Russell, A. Agogino, Automated construction of sparse Bayesian networks from unstructured probabilistic models and domain information, in: M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, vol. 5, North-Holland, Amsterdam, 1990, pp. 295–308.
[42] J. Suzuki, A construction of Bayesian networks from databases based on the MDL principle, in: D. Heckerman, A. Mamdani (Eds.) Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, 1993, pp. 266–273.

[43] T. Verma, J. Pearl, Causal networks: Semantics and expressiveness. in: R.D. Shachter, T.S. Lewitt, L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, vol. 4, North-Holland, Amsterdam, 1990, pp. 69–76.
[44] N. Wermuth, S. Lauritzen, Graphical and recursive models for contingence tables, Biometrika 72 (1983) 537–552.