The Search of Causal Orderings: A Short Cut for Learning Belief Networks

Silvia Acid, Luis M. de Campos, and Juan F. Huete

Departamento de Ciencias de la Computación e I.A.
E.T.S.I. Informática, Universidad de Granada
18071 - Granada, SPAIN
{acid, lci, jhg}@decsai.ug.es

Abstract. Although we can build a belief network starting from any ordering of its variables, its structure depends heavily on the ordering being selected: the topology of the network, and therefore the number of conditional independence relationships that may be explicitly represented can vary greatly from one ordering to another. We develop an algorithm for learning belief networks composed of two main subprocesses: (a) an algorithm that estimates a causal ordering and (b) an algorithm for learning a belief network given the previous ordering, each one working over different search spaces, the ordering and dag space respectively.

1 Introduction

Belief Networks (also called Bayesian Networks or causal networks) are Knowledge-Based Systems that represent uncertain knowledge by means of both graphical structures and numerical parameters. In a belief network, the qualitative component is a directed acyclic graph (dag), where the nodes represent the variables in the domain, and the arrows represent dependence or causality relationships among the variables. The quantitative component is a collection of conditional probability measures, which measure our uncertainty [15]. The reasons for the success of belief networks are that they allow: (i) to represent the available information in a intelligible way (using causal relationships), (ii) to decompose and store the information efficiently (by means of independence relationships) and (iii) to perform inference tasks.

One of the most interesting problems when dealing with belief networks is that of developing methods capable of learning the network directly from data. As learning belief networks is NP-hard [12], then any kind of previous information about the model to be recovered may be quite useful, in order to facilitate the learning process. This information may be an ordering of the variables in the network [2,10,13,17] or knowledge about the (possible) presence of some causal or (in)dependence relationships [16]. Perhaps an expert may provide this kind of information, but the development of tools capable of obtaining this information as a first step to the learning process is clearly an interesting task.

In this work we focus on the problem of learning belief networks by first obtaining a good ordering on the set of variables. In general, if we look for an optimal ordering then obtaining it may require as much information as the learning of the complete structure itself, and the calculus may be quite complex as well [6,14]. So, we propose to use only partial (and easily available) information about the problem in order to get a 'good' approximation of the ordering. The type of partial information we use will be a subset of the set of dependence/independence relationships that could be represented in the network (more precisely, marginal and conditional (in)dependence relationships of order one), and the method to perform the search of the ordering will be simulated annealing. Once we have obtained an ordering, it will be supplied to an algorithm for learning belief networks that will use the ordering to reduce the search space. This algorithm, called BENEDICT-*step* [3], is based on a (hybrid) methodology which is a combination of the methods based on independence criteria and the ones based on scoring metrics.

The rest of the paper is organized as follows: in section 2 we briefly recall some general ideas about belief networks, and some basic concepts about simulated annealing. In the next two Sections we describe the components of our method: in Section 3 we present our algorithm to estimate an ordering and Section 4 describes the algorithm BENEDICT-step. Section 5 shows some experiments with the proposed method. Finally, Section 6 contains the concluding remarks.

2 Preliminaries

Given a belief network G, we can extract an ordering θ for its variables in the following way: if there is an arrow $x_j \rightarrow x_i$ in the graph then x_j precedes x_i in the ordering θ, i.e., $\theta(x_j) < \theta(x_i)$. Such an ordering θ is a *causal ordering* [6]. It is interesting to note that, given a dag, the causal ordering is not unique; for example $\theta_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and $\theta_2 = \{x_1, x_4, x_2, x_3, x_5, x_6\}$ are two valid causal orderings for the first network in Figure 1.

Given any ordering θ, the Markov condition provides a systematic (but impractical) method to build a belief network [15]: for each node x_i, assign, as the parents of x_i in the dag, the minimal subset of predecessors of x_i in the ordering θ which makes x_i conditionally independent of the rest of its predecessors.

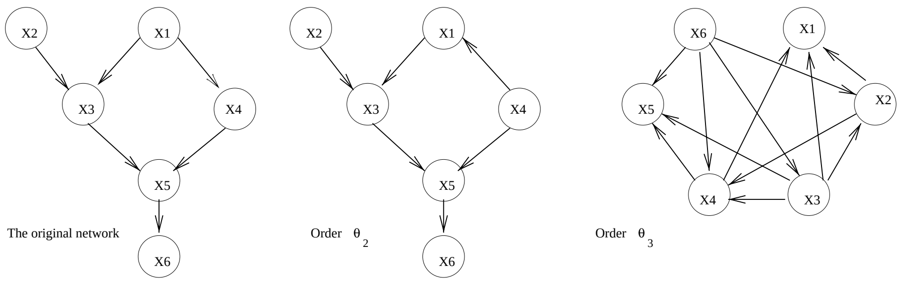

Fig. 1. Original dag and those obtained by using orderings θ_2 and θ_3

However, different orderings may give rise to different networks. For example, let us start from the network in the left hand side of Figure 1. Let $\theta_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $\theta_2 = \{x_4, x_2, x_1, x_3, x_5, x_6\}$ and $\theta_3 = \{x_6, x_3, x_2, x_4, x_1, x_5\}$ be three different orderings. If we apply the previous process, for θ_1 we recover the original graph, for θ_2 we obtain the second graph, and for the ordering θ_3 we get the much more dense graph on the right hand side of the same figure.

After assigning the corresponding conditional probabilities to the nodes, the three models represent the same joint probability distribution. However, the set of independence relationships represented in these dags is not the same. In the graph associated to θ_3 only a few independences are preserved, whereas using θ_2 we get the same set of dependence/independence relationships as in the original model (the dags corresponding to θ_1 and θ_2 are equivalent according to [18]).

2.1 Learning belief networks

There are a big number of algorithms for learning belief networks from data. 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. The number and complexity of the tests are critical for the efficiency and reliability of these methods. Some of the algorithms based on this approach can be found in [10,11,16].

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 an heuristic search method to explore the space of possible solutions. The algorithms that use this approach when the search is in the space of general dags almost invariably use greedy searches. The scoring metrics are based on different principles, such as entropy, Bayesian approaches or Minimum Description Length [7].

2.2 Simulated Annealing

In this section we briefly recall some basic ideas about simulated annealing, the search method we shall use to find a good ordering for the variables in a belief network.

The idea behind simulated annealing [4] is to model numerically the physical annealing process of solids in order to solve optimization problems:

Consider a system composed of N variables and a function E to optimize (called the energy function). Our purpose is to find a configuration c of the N variables that minimizes (or maximizes) the function E. Starting from a random configuration (c_i), representing the current state, we can compute the energy $E(c_i)$ which measures the ‘quality’ of this configuration. A new configuration c_j can be obtained by applying a perturbation mechanism on c_i. Let $E(c_j)$ be the energy of this state, and ΔE be the difference of energy, i.e., $\Delta E =$

$E(c_j) - E(c_i)$. If the energy decreases, $\Delta E \leq 0$, we accept c_j as the new current state, otherwise c_j is only accepted with a probability given by $\exp\left(-\frac{\Delta E}{T}\right)$, being T the temperature, a control parameter that decreases with time. This criterion allows a 'uphill climb' from a configuration with a lower energy to another with higher energy, thus preventing the process from being trapped at local minima. The procedure continues the search until a stopping criterion is satisfied. This criterion may be based on considering the final temperature (close to zero), the value of the energy function or using a fixed number of iterations.

3 Approximating a Causal Ordering

We seek to find a good causal ordering for the variables in a (unknown) belief network. Given any ordering, it is possible to build a belief network representing the joint probability distribution, this network being an Independence map [15] of the underlying probabilistic model. However, the density of the resultant dag may change drastically depending on the selected ordering. Our goal is to find an ordering able to represent as much true independence relationships as possible. Given this ordering, the search space to find an optimal belief network reduces considerably.

Taking into account that for a network with n nodes, the size of the set of candidate orderings is $n!$, the task of finding an optimal ordering may be quite complex. Several approaches to deal with this problem can be found in the literature:

- *Singh and Valtorta* [17] use conditional independence tests to learn a draft of the network, which is then used to get an ordering. Next, they utilize the K2 algorithm [13] to learn the network.
- *Bouckaert* [6] proposes an algorithm which takes as the input a complete dependence model and an initial ordering, and gives as the output an optimal causal ordering.
- *Larrañaga et al.* [14] use a genetic algorithm to search for the best ordering. Each element of the population is a possible ordering, and their fitness function is the K2 metric.

Our approach is situated between the works of Singh and Valtorta and those of Larrañaga et al. The basic idea is to use only a subset of the (in)dependence relationships of the model to learn a draft of the network and next apply a combinatorial optimization tool to search for the ordering which preserves as much of these dependences and independences as possible.

When dealing with conditional independence relationships whose true values have to be estimated from a database by means of conditional independence tests, two problems appear: the number of tests and their order (i.e., the number of variables involved in the conditioning set). On one hand, the number of conditional independence tests that may be necessary to perform can increase exponentially with the number of variables; on the other hand, computing the truth value of a conditional independence test requires a number of calculations which grows exponentially with the order of the test. Moreover, another problem is not related with efficiency but reliability: conditional independence tests of high order are not reliable except if the size of the database is enormous. So, it may be interesting to restrict the kind of conditional independence tests that we are going to carry out to tests of low order.

We propose using only conditional independence tests of order zero and one (i.e. $I(x_i, x_j|\emptyset)$ and $I(x_i, x_j|x_k)$, respectively) for several reasons: i) these tests are quite reliable even for moderate datasets, ii) the number of tests is polynomial $O(n^3)$, and iii) this set of independences is quite expressive for sparse structures, as those we usually find in real applications. These independences are sufficient even for characterizing and learning some specific kinds of belief networks [8,9]. We shall call *0-1 Independences* to the set of conditional independence relationships of order zero and one which are true for a given model, also denoted as I^M_{0-1}.

Our algorithm will take the set I^M_{0-1} obtained from the data set as the input. In an initialization step, we build a undirected graph (denoted G_{0-1}) as a basic skeleton of the network: starting from the complete undirected graph, we remove those links $x_i - x_j$ such that there is a 0-1 independence between x_i and x_j in I^M_{0-1}. For example, let us suppose that the underlying model is isomorphic to the graph I) in Figure 2. In this case the set of 0-1 Independences is $I^M_{0-1} = \{I(x_2, x_3|x_1)\}$. The initialization step produces the undirected graph II) in Figure 2. In a second step we shall execute the search process, which tries to find an optimal ordering. For any ordering θ being considered, we direct the skeleton G_{0-1} as follows: if $x_i - x_j \in G_{0-1}$, and $\theta(x_j) < \theta(x_i)$ then we direct the link as $x_j \rightarrow x_i$. For the example in Figure 2, let us consider the following orderings: $\theta_1 = \{x_1, x_2, x_3, x_4\}$; $\theta_2 = \{x_2, x_3, x_4, x_1\}$; $\theta_3 = \{x_1, x_2, x_4, x_3\}$ and $\theta_4 = \{x_3, x_1, x_2, x_4\}$. Using these orderings we obtain the dags III), IV), V) and VI) respectively in Figure 2.

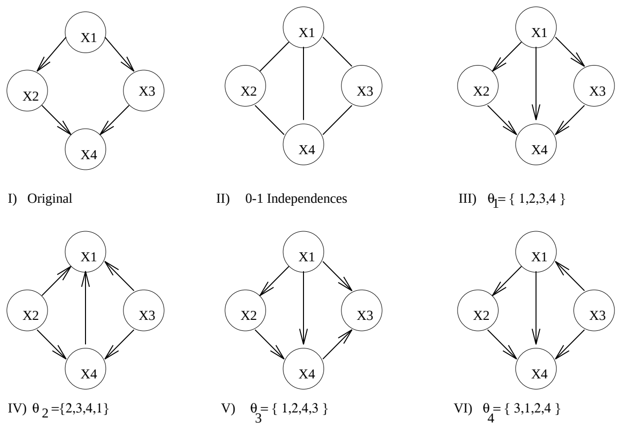

Fig. 2. Different orderings of G_{0-1}

Now, let us describe the different components of the search process:

• *Energy function*: For each configuration (ordering) θ we try to measure the degree $g(\theta)$ in which, after directing G_{0-1} according to the ordering θ (thus obtaining a dag G^{θ}_{0-1}), the dependence and independence relationships in I^{M}_{0-1} are preserved in the dag G^{θ}_{0-1}. Let us denote $I^{G^{\theta}}_{0-1}$ to the set of independence relationships of order zero and one that are valid in G^{θ}_{0-1} (using d-separation) and $\langle .,.|.\rangle$ to any d-separation statement in a dag. So, we count the number of dependence and independence relationships that are true in I^{M}_{0-1} but are not in $I^{G^{\theta}}_{0-1}$. Therefore, our energy function is:

$$g(\theta) = \sum_{x_i,x_j \wedge x_i \neq x_j} (I(x_i,x_j|\emptyset) \otimes \langle x_i,x_j|\emptyset\rangle) + \sum_{x_i,x_j,x_k \wedge x_i \neq x_j \neq x_k} (I(x_i,x_j|x_k) \otimes \langle x_i,x_j|x_k\rangle) \quad (1)$$

where we assume that an independence relationship takes on a binary value (1 for dependence, 0 for independence) and $\otimes$ corresponds to the exclusive-or operator[1]. A value $g(\theta)=0$ represents that I^{M}_{0-1} and $I^{G^{\theta}}_{0-1}$ are equivalent, and the greater the value of $g(\theta)$ is, the greater number of dependence and independence relationships are not preserved. Obviously, we shall prefer those orderings giving a value of g as low as possible. For the example in Figure 2, we have $g(\theta_1)=0$, $g(\theta_2)=2$, $g(\theta_3)=1$ and $g(\theta_4)=0$, thus θ_1 and θ_4 are the preferred orderings.

• *Perturbation mechanism:* Each configuration representing an ordering θ is codified as a chain of variables, when x_j appears before x_i then x_j precedes x_i in θ. Given a configuration, the new configuration is obtained by modifying a randomly selected segment s in the current configuration. Two mechanisms (randomly selected with 0.5 probability) have been implemented. The first one, a *transportation* function that moves the segment toward a new random position p (interchanging the elements); the second one is the *inverse* function that inverts the ordering of the variables within the segment.

• *Temperature function:* A proportional decreasing function has been implemented, i.e., $T_k = \alpha T_{k-1}$, where $\alpha \in (0,1)$ and T_0 is a fixed initial temperature.

• *Stopping criterion:* The algorithm stops when: i) all the 0-1 independences have been captured by the current configuration θ, ii) the fitness is not modified after two consecutive iterations or iii) the process has been iterated 10 times.

4 Learning Belief Networks with a Given Ordering

The algorithm we are going to describe, BENEDICT-*step*, utilizes a hybrid methodology: it uses a specific metric and a search procedure (so, it belongs to the group of methods based on scoring metrics), although it also explicitly makes use of the conditional independences embodied in the topology of the network to elaborate the scoring metric and carries out independence tests to limit the search effort

[1] We also tried more quantitative ways of evaluating the goodness of the ordering. The idea is that a link may actually represent a very weak correlation, so its absence may not be so important as the absence of other links representing strong correlations. However, the best results were obtained by using the qualitative measure of eq.(1).

(hence it has also strong similarities with the algorithms based on independence tests). It is part of a family of algorithms [2,3] that share a common methodology for learning belief networks, which we have called BENEDICT.

Let us briefly describe the BENEDICT methodology. The basic idea is to measure the discrepancies between the conditional independences (d-separation statements) 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.

To measure the discrepancy of each one of the independences in the graphical model and the numerical model (the database), BENEDICT uses the Kullback-Leibler cross entropy:

$$Dep(X, Y|Z) = \sum_{\mathtt{x},\mathtt{y},\mathtt{z}} P(\mathtt{x}, \mathtt{y}, \mathtt{z}) \log \frac{P(\mathtt{x}, \mathtt{y}|\mathtt{z})}{P(\mathtt{x}|\mathtt{z})P(\mathtt{y}|\mathtt{z})},$$

where $\mathtt{x}$, $\mathtt{y}$, $\mathtt{z}$ denote instantiations of the sets of variables X, Y and Z respectively, and P is a probability estimated from the database.

As the number and complexity of the d-separation statements in a dag G may grow exponentially with the number of nodes, we cannot use all the d-separations displayed by G, but some selected subset of 'representative' d-separation statements. Given any candidate network G, BENEDICT will take into account the conditional independencies for every two non-adjacent single variables, x_i and x_j given the set of minimum size, $S_G(x_i, x_j)$, that d-separates x_i and x_j [1]. Finding this set takes some additional effort, but it is compensated by a decreasing computing time of the corresponding dependence degree. Moreover, it also increases the reliability of the results, because less data is needed to reliably compute a conditional dependence measure of lower order. The method BENEDICT uses for efficiently finding the sets $S_G(x_i, x_j)$ is described in [1].

In order to give a score to a specific network structure G given a database D, BENEDICT uses the aggregation (the sum) of the local discrepancies, as the measure of global discrepancy $g(G, D)$ (which has to be minimized). Finally, the type of search method used by BENEDICT is a simple greedy search that allows to insert into the structure the candidate arc that produces a greater improvement of the score (removal of arcs is not permitted).

Let us describe more specifically the algorithm BENEDICT-*step*. It works under the assumption that the total ordering of the variables is known (this ordering θ is just the one obtained by the simulated annealing algorithm). BENEDICT-*step* consists in a process composed of n steps, where each step i represents the inclusion of a new node x_i (the i-th node in the ordering θ) in the (initially empty) structure and the inclusion of the necessary arcs to construct the best graph with i nodes, G_i.

At each step i only the d-separation relationships between x_i, the node just introduced, and the previous ones are considered, hence the metric used by BENEDICT-*step* is

$$g(G_i, D) = \sum_{x_j,\, x_j <_\theta x_i \wedge x_j \notin \pi_{G_i}(x_i)} Dep(x_i, x_j | S_{G_i}(x_i, x_j)) \qquad (2)$$

Every step i is composed of a series of substeps. Each substep looks for the arc whose addition to the current graph results in a greater decrease of the discrepancy measure between the new graph (that one with the added arc) and the data. The process continues in this way, adding at each substep the single arc $x_j \rightarrow x_i$, $x_j <_\theta x_i$, which most decreases the discrepancy, until the stopping condition holds.

This condition is related with the fact that the algorithm also uses independence tests to remove candidate arcs; in this way, the process stops naturally when there is no more candidate arcs to consider (either because they are already inserted into the structure or because their extreme nodes are found to be independent). At the end of the algorithm a pruning process (also based on independence tests) is triggered (see [3] for details). This pruning partially overcomes some of the problems due to the use of an irrevocable search strategy.

5 Experimental Results

We will consider the performance of the proposed methodology to recover the so-called Alarm belief network (see Figure 3), which has been considered as a benchmark for evaluating learning algorithms. This network contains 37 variables and 46 arcs. The input data commonly used are subsets of the Alarm database which contains 20,000 cases, specifically we used the first 3,000 cases in our experiments.

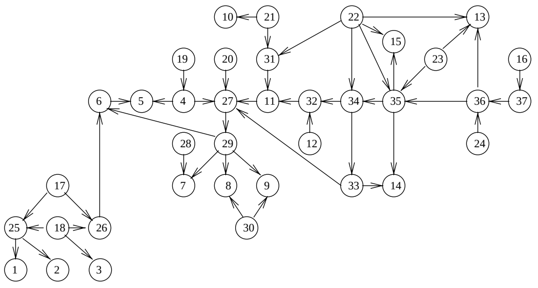

Fig. 3. The Alarm network

As we explained, the learning process is divided in two main processes. Let us first analyze the results of each one separately, and then the final results of the whole process.

The first subprocess consists on searching the 'best' ordering, θ, of the variables using a simulated annealing algorithm. The fitness value used was the number (in percentage) of 0-1 (in)dependences preserved by the current ordering. In order to measure the quality of the ordering we will compare with the Alarm 'correct' ordering. Due to the stochastic nature of the simulated annealing algorithm, we run several times the algorithm with the same training set. In every case the final fitness was 97.0%, resulting different indistinguishable orderings (one of these orderings is the 'correct' ordering). After analyzing the orderings obtained in our experiments we can extract some conclusions:

1. The degree in which the database reflects the set of 0-1 (in)dependences in the true model is important for getting a good output ordering.
2. For those subsets W of variables in the model with no 0-1 independence relationships, we do not have enough information to discriminate which partial orderings, involving the variables in W, are the correct ones. We found different orderings, involving changes in the relative ordering of variables in the set $\{35, 15, 34\}$, with the same fitness value. As we will see, the study of these orderings will be relevant in order to get a better network.
3. There are several orderings with the same fitness value that are indistinguishable even with more information. For example, considering the variables 4 and 19, regardless of the relative ordering used, we get equivalent structures.

The second subprocess consists on, using an ordering θ, let the algorithm BENEDICT-*step* to learn the structure of the network. In order to analyze its behaviour, we supply the BENEDICT-*step* algorithm with the 'correct' ordering and the same training data. From the topology of the learned network we observe: a) There are three missing edges, $11 \rightarrow 27$, $12 \rightarrow 32$ and $21 \rightarrow 31$. The last two arcs are not strongly supported by the data, as it was reported by several authors. b) There is one extra arc between variables 31 and 27 which has not been determined as independent by the independence tests; this arc is set while trying to compensate the loss of the arc $11 \rightarrow 27$ (a total of 4 different arcs from the original model). We have also computed several other measures to evaluate the quality of the learned network from different points of view. These measures are: 1.- the Kullback[2] distance between the probability distribution associated to the database and the probability distribution associated to the learned network. 2.- The K2 metric [13] (log version) and 3.- the BIC metric (Bayesian Information Criterion) which includes a penalty term. Finally we compare all these collected measures with those obtained by the K2 algorithm [13], running both algorithms on the same conditions. The results shown in the first row of Table 1 allow us to conclude that our algorithm is competitive and recovers a good model.

Now we are going to analyze the results obtained in the whole process. Usually when no ordering is known, the learning algorithm has to cope with the entire dag space to learn the network. We can make a comparison[3] between the two steps method proposed and the single searching process. For that purpose we have used a constraint based algorithm, BN Power Constructor (BNPC) [11] (we use the software package available at `http://www.cs.ualberta.ca/~jcheng /bnsoft.htm`). In Table 2 we show the results obtained by the BNPC algorithm which are worse than those obtained by any of the different orderings θ, used as entry to the algorithms BENEDICT-*step* and K2. All these orderings were score-equivalent for the simulated annealing process.

[2] Actually, we have calculated a decreasing monotonic transformation of the Kullback distance computed in a very efficient form [8]. The interpretation is: the higher this parameter the better is the network.

[3] We do not compare the running times because the three algorithms considered, run on different platforms and are implemented using different programming languages.

In order to focus on how some local misplacements in the ordering can modify the resulting learned network, we have studied the possible cases found by simulated annealing involving the relative orderings between $34, 35$ and 15. As we said, simulated annealing is not capable to discriminate among these orderings, thus any configuration would be a possible input to the BENEDICT-*step* algorithm. As we could expect, they give rise to different networks. Table 1 presents the results obtained when we consider three orderings that differ only in the relative ordering of the variables $34, 35$ and 15. As we can observe, some of the orderings are worse than others, and the output belief network structure depends on how lucky we were, in the first subprocess. As the ordering is obtained by using only partial information (0-1 Independences), its quality might be questioned. Hopefully, we do not have to reconsider the global ordering. In the general case, the set of 0-1 Independences is quite significant, so the learned structure is a 'good' representation of the model. Anyway, we thought about that BENEDICT-*step* with more information could detect some misplacements between variables in θ and could discriminate among indistinguishable orderings. Note that a 'bad' ordering (as for example the order θ_3 in Figure 1) tends to create cliques (we consider cliques having at least three nodes) among the variables involved.

Table 1. Comparison between the algorithms BENEDICT-*step* and K2

	BENEDICT-*step*				K2				Relative
Ordering	Kullback	Hamming	BIC	K2	Kullback	Hamming	BIC	K2	ordering
'correct'	9.20	4	-33919	-14425	9.23	2	-34351	-14424	
θ_1	9.11	14	-34830	-14624	9.21	12	-35697	-14520	$34 \prec 15 \prec 35$
θ_2	9.18	12	-34606	-14533	9.23	8	-36205	-14494	$34 \prec 35 \prec 15$
θ_3	9.21	8	-34358	-14479	9.23	5	-35203	-14450	$35 \prec 34 \prec 15$

Table 2. Performance measures for the network learned by BNPC

Kullback	Hamming	BIC	K2
9.12	7	-35197	-14541

We have developed a heuristic rule that, using the information stored in the output network, allows us to obtain a sparser representation of the same model. This refinement is carried out by determining local rearrangements in θ, giving rise to also local changes in the structure, but improving the quality of the output network. Basically the heuristics consists on selecting a variable x_i in a clique and generating a new ordering θ^*, where this variable changes its relative position with respect to some variables in the clique.

In Table 1, from the ordering θ_1 to θ_3, we can follow the steps of our heuristic focused on variables $34, 35$ and 15. We make the comparisons taking as reference the structure obtained by BENEDICT-*step* when it uses the Alarm 'correct' ordering as the input (the initial erroneous arcs remain). Thus, taking the worst ordering, $34 \prec 15 \prec 35$, as the input, our first change involves the variables 15 and 35 (the last two variables in the clique), giving rise to a sparser network and also with a better fitness, which is accepted as the current one. Then, using the same reasoning, the change between variables 35 and 34 is performed. Note that the resulting ordering is the correct relative ordering. The algorithm stops at this point.

6 Concluding Remarks

We have addressed the problem of learning belief networks from data by means of a two steps process: estimating a good ordering of the variables (thus reducing the search space for the belief network learning algorithm) and then using a learning algorithm that exploits this ordering. The search for a good ordering is carried out by means of a simulated annealing method, which uses a function based on independence tests of order zero and one to measure the fitness. The algorithm that uses this ordering to learn the network is a member of the BENEDICT family, whose main characteristics are its hybrid nature and the use of d-separating sets of minimum size.

In addition to the specific algorithms that we use to develop our method, the main methodological difference with respect to other approaches [14,17] is that the two subprocesses are run independently on each other (this means that we carry out two 'simple' different search processes (over different spaces) instead of a single search process which intermingles the orderings and the graph structures).

In general, to obtain an optimal solution to the problem of finding a causal ordering, it would be necessary to learn the network (i.e., to have information about the complete set of valid conditional independence statements). Our experiments show that our approximate method (based on conditional independence tests of low order, for reasons of reliability, expressiveness and efficiency) is quite successful. Its combination with BENEDICT-*step* gives a very general and competitive algorithm for learning belief networks. However, a thorough experimental work, using networks of different complexity, is necessary in order to obtain definitive conclusions.

In future works we plan to continue the development of heuristic rules allowing the algorithm BENEDICT-*step* (or any other learning algorithm that requires an ordering) to make local rearrangements in the ordering to improve the quality of the learned network. We will also study methods that refine a given ordering (e.g., the output ordering provided by our algorithms) to obtain an optimal solution. These methods could be based on the idea of the reliability about the particular position of any given variable x_i in the ordering, which in turn is directly related to the number of 0-1 independences where this variable x_i is involved.

Acknowledgments. This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología under Project n. TIC2000-1351. We would like to thank the anonymous referees for their useful comments.

References

1. S. Acid and L.M. de Campos. An algorithm for finding minimum d-separating sets in belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, E. Horvitz, F. Jensen (Eds.), pages 3-10 Morgan and Kaufmann, 1996.
2. S. Acid and L.M. de Campos. Benedict: An algorithm for learning probabilistic belief netwoks. In *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 979-984, 1996.
3. S. Acid and L.M. de Campos. An hybrid methodology for learning belief networks: Benedict. To appear in International Journal of Approximate Reasoning.
4. N. Ansari and E. Hou. *Computational Intelligence for Optimization.* Kluwer Academic Publishers, 1997.
5. I. Beinlich, H. Seurmondt, R. Chavez, and G. Cooper. The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pages 247-256, 1989.
6. R. Bouckaert. Optimizing causal orderings for generating dag's from data. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, pages 9-16. Morgan-Kaufmann, 1992.
7. W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transaction on Knowledge and Data Engineering,* 8:195-210, 1996.
8. L.M. de Campos. Independency relationships and learning algorithms for singly connected networks. *Journal of Experimental and Theoretical Artificial Intelligence* 10:511-549, 1998.
9. L.M. de Campos and J.F. Huete. On the use of independence relationships for learning simplified belief networks. *Int. J. of Intelligent Systems*, 12:495-522, 1997.
10. L.M. de Campos and J.F. Huete. A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, 24(1)11-37, 2000.
11. J. Cheng, D.A. Bell and W. Liu. Learning belief networks form data: An information theory based approach In *Proc. of ACM CIKM'97*, pages 325-331, 1997.
12. D. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian Networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research, 1994.
13. G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309-347, 1992.
14. P. Larrañaga, C.M. Kuijpers, R.H. Murga, and Y. Yurramendi. Learning Bayesian network structure by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics- Part A: Systems and Humans*, 26(4):487-493, 1996.
15. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan and Kaufmann, 1988.
16. P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search.* Lecture Notes in Statistics 81. Springer Verlag, New York, 1993.
17. M. Singh and M. Valtorta. Construction of Bayesian networks structures from data: A survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12:111-131, 1995.
18. T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 220-227, Mass, 1990.