Construction of Bayesian Network Structures from Data: a Brief Survey and an Efficient Algorithm *

Moninder Singh[†] **and Marco Valtorta**

Dept. of Computer Science, University of South Carolina
Columbia, SC 29208, U.S.A[‡]

ABSTRACT

Previous algorithms for the recovery of Bayesian belief network structures from data have been either highly dependent on conditional independence (CI) tests, or have required an ordering on the nodes to be supplied by the user. We present an algorithm that integrates these two approaches - CI tests are used to generate an ordering on the nodes from the database which is then used to recover the underlying Bayesian network structure using a non CI test based method. Results of the evaluation of the algorithm on a number of databases (e.g., ALARM, LED and SOYBEAN) are presented. We also discuss some algorithm performance issues and open problems.

KEYWORDS: *Bayesian Networks, Probabilistic Networks, Probabilistic Model Construction, Conditional Independence.*

1. INTRODUCTION

In very general terms, different methods of learning probabilistic network structures from data can be classified into three groups. Some of these

*A preliminary version of this paper was presented in [1].

†Current address is the Department of Computer and Information Science, University of Pennsylvania, 200 S 33rd St., Philadelphia, PA 19104

‡E-mail: msingh@gradient.cis.upenn.edu or mgv@usceast.cs.scarolina.edu

International Journal of Approximate Reasoning 1994 11:1–158

655 Avenue of the Americas, New York, NY 10010 0888-613X/94/$7.00

methods are based on linearity and normality assumptions [2, 3]; others are more general but require extensive testing of independence relations [4, 5, 6, 7, 8]; others yet take a Bayesian approach [9, 10, 11, 12].

In this paper, we do not consider methods of the first kind, namely, those that make linearity and normality assumptions. Our work concentrates on CI test based methods and Bayesian methods. A number of algorithms have been designed which are based on CI tests. However, there are two major drawbacks of such CI test based algorithms. Firstly, the CI test requires determining independence relations of order $n-2$, in the worst case. "Such tests may be unreliable, unless the volume of data is enormous" [10, page 332]. Also, as Verma and Pearl [5, pages 326–327] have noted, "in general, the set of all independence statements which hold for a given domain will grow exponentially as the number of variables grow". As such, CI test based approaches become rapidly computationally infeasible as the number of vertices increases. Spirtes and Glymour [6, page 62] have presented "an asymptotically correct algorithm whose complexity for fixed graph connectivity increases polynomially in the number of vertices, and may in practice recover sparse graphs with several hundred variables"; but for dense graphs with limited data, the algorithm might be unreliable [10].

On the other hand, Cooper and Herskovits [10] have given a Bayesian non-CI test based method, which they call the BLN (Bayesian learning of belief networks) method. Given that a set of four assumptions hold [10, page 338], namely, (i) The database variables are discrete, (ii) Cases occur independently, given a belief network model, (iii) All variables are instantiated to some value in every case, and finally (iv) Before observing the database, we are indifferent regarding the numerical probabilities to place on the belief network structure, Cooper and Herskovits [10] have shown the following result:

THEOREM 1. *(due to Cooper and Herskovits [10]). Consider a set Z of n discrete variables. Each variable $x_i \in Z$ has r_i possible value assignments: $(v_{i1}, \ldots, v_{ir_i})$. D is a database of m complete cases i.e. each case contains a value assignment for each variable in Z. Let B_S denote a belief network structure containing just the variables in Z. Each variable x_i in B_S has a set of parents π_i. w_{ij} denotes the jth unique instantiation of π_i relative to D and there are q_i such unique instantiations of π_i. N_{ijk} is the number of cases in D in which x_i is instantiated to v_{ik} while π_i is instantiated to w_{ij}. Let $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Then,*

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} g(i, \pi_i) \qquad (1)$$

where $g(i, \pi_i)$ is given by

$$g(i, \pi_i) \;=\; \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2}$$

This result can be used to find the most probable network structure given a database. However, since the number of possible structures grow exponentially as a function of the number of variables, it is computationally infeasible to find the most probable belief network structure, given the data, by exhaustively enumerating all possible belief network structures.

Herskovits and Cooper [10, 9] proposed a greedy algorithm, called the **K2** algorithm, to maximize $P(B_S, D)$ by finding the parent set of each variable that maximizes the function $g(i, \pi_i)$. In addition to the four assumptions stated above, K2 uses two more assumptions, namely, that there is an ordering available on the variables and that, a priori, all structures are equally likely. The K2 algorithm is described below:

for each node $i, 1 \leq i \leq n$, find π_i as follows:
  $\pi_i \leftarrow \phi$
  $P_{old} \leftarrow g(i, \pi_i)$;
  NotDone $\leftarrow$ True
  while NotDone do
    $\forall l \prec i \; (l \notin \pi_i),\; g_l \leftarrow g(i, \pi_i \cup \{l\})$
    $P_{new} \leftarrow \max_l \; g_l(i, \pi_i \cup \{l\})$;
    Let z be the node which maximizes g_l above
    if $P_{new} > P_{old}$ then
      $P_{old} \leftarrow P_{new}$;
      $\pi_i \leftarrow \pi_i \cup \{z\}$
    else NotDone $\leftarrow$ false;
  end {while};

The algorithm takes as input the n nodes, an ordering on the nodes and the database D of m cases. In order to find the parent set of a node, it first assumes that the node has no parents, and then adds incrementally that node (from among the predecessors in the ordering) to the parent set which increases the probability of the resultant structure by the largest amount. It stops adding parents to the node when no additional single parent can increase the probability of the resultant structure.

2. MOTIVATION

As stated at the end of the previous section, the K2 algorithm requires an ordering on the nodes to be given to it as an input along with the database of cases. The main thrust of this research is to combine both CI as well as non CI test based methods described above to come up with a computationally tractable algorithm which is not overdependent on the CI tests, nor does it require a node ordering[1].

In order to achieve this, we use CI tests to generate an ordering on the nodes, and then use the K2 algorithm to generate the underlying belief network from the database of cases using this ordering of nodes. Also, since we are interested in recovering the most probable Bayesian network structure given the data, we would like to generate an ordering on the nodes that is consistent with the partial order specified by the nodes of the underlying network. In a domain where very little expertise is available, or the number of vertices is fairly large, finding such an ordering may not be feasible. As such, we would like to avoid such a requirement. The remainder of this section elaborates on this point.

It is possible to find a Bayesian network for any given ordering of the nodes, since any joint probability distribution $P(x_1, x_2, \ldots, x_n)$ can be rewritten, by successive applications of the chain rule, as

$$P(x_{i1}, x_{i2}, \ldots, x_{in}) = \\ P(x_{i1} \mid x_{i2}, \ldots, x_{in}) \times P(x_{i2} \mid x_{i3}, \ldots, x_{in}) \times \ldots \times P(x_{in})$$

where $< i_1, i_2, \ldots, i_n >$ is an arbitrary permutation of $< 1, 2, \ldots, n >$. However, the sparseness of the Bayesian network structure representing the joint probability distribution $P(x_1, x_2, \ldots, x_n)$ will vary, sometimes dramatically, with respect to the choice of the ordering of the nodes[2]. It is desirable to use an ordering of the nodes that allows as many of the conditional independences true in the probability distribution describing the domain of interest as possible to be represented graphically[3].

It would be too expensive to search blindly among all orderings of nodes, looking for one that leads to a network that both fits the data and is sparse enough to be useful. In a small setting, grouping variables into generic classes, such as symptoms and diseases may be sufficient to limit

[1] Herskovits [9] suggested the use of the metric (on which K2 is based) with a CI test based method to do away with the requirement for an order of nodes.

[2] In this paper, no distinction is made between the nodes of a Bayesian network and the variables they represent.

[3] Whereas different types of graphical structures have different expressive powers, this paper is only concerned with dags, as used in Bayesian nets. We ignore Markov nets [13, chapter 3], chain graphs [14, 15], and other graphical representations (e.g., [16, 17]).

the number of orderings to be searched without having to use dramatically greedy heuristics. This was shown to be adequate for a medical application with 10 nodes in [11], where variables were divided in "blocks." In some applications, however, it may be impossible to divide variables into classes, or the classes may be too large to impose sufficient structure on the space of candidate orderings. We have implemented an algorithm, called **CB**[4], that uses a CI test based algorithm to propose a total order of the nodes that is then used by a Bayesian algorithm. We have tested the algorithm on some distributions generated from known Bayesian networks. (The results will be shown after the algorithm is presented.)

The Bayesian method used in the CB algorithm is a slightly modified version of Cooper and Herskovits's K2, implemented in C on a DECstation 5000. Herskovits proved an important result concerning the correctness of the metric that K2 uses to guide its search. He showed that the metric on which K2 is based is minimized, as the number of cases increases, without limit, on "those [Bayesian] network structures that, *for a given node order*, most parsimoniously capture all the independencies manifested in the data" [9, chapter 6]. More precisely, he showed that the K2 metric will always favor, as the number of cases in the database increase without limit, a minimal I-map consistent with the given ordering (see [13, chapter 3] for the definition of minimal I-map). Despite the convergence result, it is still important to provide K2 with a good node order, since there are too many orderings ($n!$ for n nodes) to search blindly among them, unless drastically greedy (myopic) search regimens are used. Moreover, for different orderings, we will get different I-maps of differing density. Note that an I-map only means that all independencies implied by it (through d-separation) are also in the underlying model. So more sparse networks will give us more information as compared to relatively denser networks. In this sense, the ordering given to K2 becomes very important. Given a random ordering, we might land up with a very dense dag which is an I-map (possibly minimal) but conveys very little information. So, we would like to use as informed an ordering as possible. For example, assuming that the data was generated using a Bayesian network whose structure is an I-map for the underlying distribution, it would be very desirable to provide K2 with an ordering of the nodes that allows the network to be recovered exactly, even though K2 may recover a different I-map when given a different ordering, because the generating structure is normally the sparsest one among all I-maps for a given distribution, or at least one of the sparsest ones. Our algorithm finds good node orderings by using a CI-based test. Since CB still uses K2 to compute the Bayesian network structure from an ordering, it is correct in the same sense that K2 is.

[4] The name reflects the initials of the two phases of the algorithm.

3. DISCUSSION OF THE ALGORITHM

3.1. Overview

The algorithm basically consists of two phases: Phase I uses CI tests to generate an undirected graph, and then orients the edges to get an ordering on the nodes. Phase II takes as input a total ordering consistent with the DAG generated by phase I, and applies the K2 algorithm to construct the network structure using the database of cases. The two phases are executed iteratively - first for 0th order CI relations, then for 1st order CI relations, and so on until the termination criteria is met.

Steps 1 to 4 of the algorithm are based on the algorithms given by Verma and Pearl [5] and Spirtes and Glymour [6]. We have allowed edges to be oriented in both directions because at any given stage, since CI tests of all orders have not been performed, all CI relations have not been discovered and there will be a number of extra edges. In such a case, it is quite possible for edges to be oriented in both directions by step 3. Although the bound used in step 2 is not necessary, it may be useful in decreasing the run time of the algorithm by not trying to generate the belief network structure if the undirected graph recovered from very low order CI relations (in step 2) is dense.

Once the edges have been oriented by steps 3 and 4, the algorithm finds the set of potential parents of each node by considering only the directed edges (step 5), and then uses a heuristic to choose an orientation for the edges which are still undirected, or are bidirected. Although, theoretically, equation 1 can be used to find the probability $P(\ i \ \rightarrow \ j \mid D)$ (and $P(\ i \ \leftarrow \ j \mid D))$ from the data [10, page 318] which can then be used to orient an edge $i \ - \ j$ (on the basis of which orientation is more probable), it is computationally infeasible do so because of the sheer number of network structures which have that edge. Hence the use of a heuristic. From equation 1, it should be clear that the orientation of an edge between vertices i and j affects only $g(i, \pi_i)$ and $g(j, \pi_j)$, and so to maximize $P(B_S, D)$, we would like to maximize the product $g(i, \pi_i) \times g(j, \pi_j)$ where π_i and π_j are the sets of parents of nodes i and j respectively. Accordingly, we compute the products $i_{val} \ = \ g(i, \pi_i) \ \times \ g(j, \pi_j \cup \{i\})$ and $j_{val} \ = \ g(j, \pi_j) \ \times \ g(i, \pi_i \cup \{j\})$ where π_i and π_j are the sets of potential parents recovered by step 5 of the algorithm. These products give us a way of selecting an orientation for the edge. If i_{val} is larger, we prefer the edge $i \ \rightarrow \ j$ (unless it causes a directed cycle in which case we choose the other orientation). Similarly, we choose $j \ \rightarrow \ i$ if j_{val} is larger (or the reverse in case of a directed cycle).

At this stage, the algorithm has constructed a DAG. It then finds a total ordering on the nodes consistent with the DAG and applies the K2 algorithm to find the set of parents of each node such that the K2 metric (i.e. $g(i, \pi_i)$) is maximized for each node i, allowing edges to be directed from a node only to nodes that are its successors in the ordering.

3.2. The Algorithm

Let $A_G ab$ be the set of vertices adjacent to a or b in graph G not including a and b. Also, let u be a bound on the degree of the undirected graph generated by step 2. ord is the order of CI relations being tested. Let π_i be the set of parents of node $i, 1 \leq i \leq n$.

1. Start with the complete graph G_1 on the set of vertices Z.
$ord \leftarrow 0$.
$old\_\pi_i \leftarrow \{\ \}\ \forall i, 1 \leq i \leq n$, and $old\_Prob \leftarrow 0$.

2. (based on [6])

Modify G_1 as follows :
For each pair of vertices a, b that are adjacent in G_1, if $A_{G_1}ab$ has a cardinality greater than or equal to ord, and $I(a, S_{ab}, b)$[5] where $S_{ab} \subseteq A_{G_1}ab$ of cardinality ord, remove the edge $a - b$, and store S_{ab}.
If for all pairs of adjacent vertices a, b in G_1, $A_{G_1}ab$ has cardinality $<$ ord, goto step 10.
If degree of $G_1 > u$, then

$ord \leftarrow ord + 1$

Goto beginning of step 2

3. Let G be a copy of G_1.
For each pair of non adjacent variables a, b in G, if there is a node c that is not in S_{ab} and is adjacent to both a and b, then orient the edges from $a \rightarrow c$ and $b \rightarrow c$ (see [5, 6]) unless such an orientation leads to the introduction of a directed cycle in the graph.
If an edge has already been oriented in the reverse direction, make that edge bidirected.

4. Try to assign directions to the yet undirected edges in G by applying the following four rules [5, 18] if this can be done without introducing directed cycles in the graph:

Rule 1: If $a \rightarrow b$ and $b - c$ and a and c are not adjacent, then direct $b \rightarrow c$.

Rule 2: If $a \rightarrow b$, $b \rightarrow c$ and $a - c$, then direct $a \rightarrow c$.

---

[5] We use the notation $I(S_1, S_2, S_3)$ to represent the fact that S_1 and S_3 are independent conditional on S_2

Rule 3: If $a - b$, $b - c$, $b - d$, $a \rightarrow d$, and $c \rightarrow d$, then direct $b \rightarrow d$.

Rule 4: If $a - b$, $b - c$, $a - c$, $c - d$, and $d \rightarrow a$, then direct $a \rightarrow b$ and $c \rightarrow b$.

Moreover, if $a \rightarrow b, b \rightarrow c$ and $a \leftrightarrow c$, then direct $a \rightarrow c$.

5. Let $\pi_i \leftarrow \{\ \} \ \forall i, 1 \leq i \leq n$.
For each node i, add to π_i the set of vertices j such that for each such j, there is an edge $j \rightarrow i$ in the pdag G.

6. For each undirected or bidirected edge in the pdag G choose an orientation as described below
If $i - j$ in an undirected edge, and π_i and π_j are the corresponding parent sets in G, then calculate the following products

$$i_{val} = g(i, \pi_i) \times g(j, \pi_j \cup \{i\})$$
$$j_{val} = g(j, \pi_j) \times g(i, \pi_i \cup \{j\})$$

If $i_{val} > j_{val}$, then $\pi_j \leftarrow \pi_j \cup \{i\}$ unless the addition of this edge, i.e. $i \rightarrow j$ leads to a cycle in the pdag. In that case, choose the reverse orientation, and change π_i (instead of π_j). Do a similar thing in case $j_{val} > i_{val}$

7. The sets π_i, $1 \leq i \leq n$ obtained by step 6 define a DAG since for each node i, π_i consists of those nodes that have a directed edge to node i.
Generate a total order on the nodes from this DAG by performing a topological sort on it.

8. Apply the K2 algorithm to find the set of parents of each node using the order in step 7. Let π_i be the set of parents, found by K2, of node i, $\forall\ 1 \leq i \leq n$.
Let $new\_Prob = \prod_{i=1}^{n} g(i, \pi_i)$.

9. If $new\_Prob > old\_Prob$, then

$old\_Prob \leftarrow new\_Prob$

$ord \leftarrow ord + 1$

$old\_\pi_i \leftarrow \pi_i \ \forall i, 1 \leq i \leq n$

Discard G

Goto Step 2

Else goto Step 10

10. Output $old\_\pi_i \ \forall i,\ 1 \leq i \leq n$
Output Old_Prob

3.3. Variations of the algorithm

3.3.1. Priors The **CB** algorithm, as does the **K2** algorithm, assumes that, a priori, all network structures are equally likely. In the absence of any information, this would seem to be the most logical strategy. However, there are situations in which some networks are clearly preferred to others—for example, in some diagnostic applications, networks that have the disease node as the root will be preferred over networks that do not have it as the root[6].

As such, we have tried out another strategy - forcing the class variable to be the root of the network structure learned by the **CB** algorithm. This restriction simply means that all networks that do not have the class variable as the root have a prior probability of zero, and all networks which have the class node as its root are, a priori, equally likely.

3.3.2. Search Strategy The **CB** algorithm uses a greedy search mechanism (**K2**) to search for the set of parents of each node. Although this has the advantage of being computationally efficient, it does not ensure optimality even though the metric used by **K2** is exact. Hence, we also explored the use of another search strategy, namely Simulated Annealing [19] though this has the disadvantage of being very computationally expensive.

Simulated annealing [19] is a stochastic optimization technique used to find the maximum probability (minimum cost) configuration of some cost function corresponding to combinatorial problems with cost functions having many local minima. The basic advantage of this method over a strictly descent algorithm like greedy search is that it provides a way of escaping local minima. The evolution of the state of the search process is controlled by varying a parameter called the 'temperature.' At the beginning of the process, the temperature is set to some large value which then allows large shifts in the state, thereby allowing the process to get out of local minima. This temperature parameter is successively decreased until the search stabilizes and converges to a global minima.

In the present situation, we are given a total ordering on the nodes with the constraint that the parents of any given node must belong to the set of nodes that precede this node in the total ordering. A suitable method for constructing the Bayesian network would be to find the parents of each node by simulated annealing.

Suppose that we want to find the set of parents of node i. Since any node can have parents only from the set of nodes occurring before it in the ordering, the solution space for this problem is the set of all possible subsets (including the empty set) of the set of nodes preceding the node i in the ordering. Each configuration, say π_{i_k}, is a subset of such nodes. Since we are interested in maximizing the value of the metric $g(\ i,\ \pi_i\ )$, we can define the cost function to be the negative of this i.e. $C(\pi_{i_k}) = -g(\ i\ , \pi_{i_k})$. Minimizing the cost function for the parent set of each node will give us the minimum cost of the entire network (equation 1).

The initial configuration (k = 0) at which the annealing algorithm starts consists of the parent sets, π_{i_k}, $1 \le i \le n$, obtained at the end of Step

[6] See [11] for a discussion of other similar situations.

6 of the **CB** algorithm. The algorithm randomly selects two nodes, and determines which of the two occurs earlier in the total ordering. Let this node be v, and the other be i. π_{i_k} is the current parent set configuration of node i. Since v is a possible parent of i, the algorithm computes the cost $C(\pi_{i_p})$ of the potential new configuration π_{i_p} that is obtained as follows: if $v \in \pi_{i_k}$, then $\pi_{i_p} = \pi_{i_k} - \{v\}$, otherwise $\pi_{i_p} = \pi_{i_k} \cup \{v\}$. The new configuration π_{i_p} is accepted if the following condition holds:

$$C(\pi_{i_p}) \leq C(\pi_{i_k}) - t_k \; log(u)$$

where u is a uniformly distributed random number on the interval [0,1], and t_k is the *temperature*, a control parameter at time k. This means that an uphill step (one that allows the cost to increase) of λt_k will be allowed with probability $e^{-\lambda}$. This means that the new configuration $\pi_{i_{k+1}}$ is set to π_{i_p} with probability

$$p = min\{1, e^{-\Delta | t_k}\}$$

and to π_{i_k} (i.e. unchanged) with probability $1 - p$, where

$$\Delta = C(\pi_{i_p}) - C(\pi_{i_k}).$$

The initial temperature is set to a large value, this process is repeated a large number of times, the temperature is then reduced and the process repeated and so on. To ensure that the process does converge to an almost optimal solution, a control schedule has to be defined which determines the initial temperature, the rate at which the temperature is decreased, the number of iterations at each temperature and the stopping criteria. For our experiments, we used a polynomial-time cooling schedule (which however does not give any guarantee for the deviation in cost between the final solution obtained by the algorithm and the optimal cost) described in [20].

4. RESULTS

The **CB** algorithm was implemented in C on a DEC Station 5000, and was used to reconstruct the underlying Bayesian network structure from four databases.

For two of the databases, namely the ALARM [21, 9, 10] and the LED [4] databases, the Bayesian network structure which had been used to generate the database was known. So it was possible to compare the network structure recovered by the **CB** algorithm against the actual structure to test the performance of the algorithm.

The **CB** algorithm was further tested on datasets for which the true underlying network structure is unknown. For this purpose, we selected two databases from the University of California (Irvine) Repository of Machine Learning Databases [22], namely the LETTER RECOGNITION Database [23] and the SOYBEAN Database [24]. In these cases, since the true underlying Bayesian network is unknown, we used the shell HUGIN [25] to test the predictive accuracy of the constructed networks. Once the network structure had been constructed, the required conditional probabilities were calculated from the database using the following relation [9]:

$$P(x_i = v_{ik} \mid \pi_i = w_{ij}) = \frac{N_{ijk} + 1}{N_{ij} + r_i}$$

where the various symbols have the same meanings as in Theorem 1.

For all tests, we used the χ^2 test with a fixed α level (set to 0.1) for testing Conditional Independence.

The results of the tests are described in the following sections:

4.1. The ALARM Database

The ALARM network [21] was constructed as a prototype to model potential anesthesia problems that could arise in the operating room. The network contains a total of 37 nodes and 46 arcs representing 8 diagnostic problems, 16 findings and 13 intermediate variables that relate diagnostic problems to findings. We used the **CB** algorithm to reconstruct the ALARM network (Figure 1(a)) by using 10,000 cases of a (20,000 case) database generated by Herskovits [9, 10] from the original network by using a Monte Carlo technique called probabilistic logic sampling [26]. We used a bound of 15 on the maximum degree of the undirected graph generated in step 2. The algorithm recovered the network shown in Figure 1(b) using CI tests up to *only order 2*. Due to the bound, it did not generate a network for CI relations of order 0. Out of 46 edges, it recovered 45 edges (Figure 1(b)).

The only missing edge was the edge $12 \rightarrow 32$ (an edge which is not strongly supported by the data [10]. Two of the edges recovered were incorrectly oriented. However, the algorithm also recovered 14 extra edges. This is probably due to the incorrectly oriented edges, and to some extent, due to the greedy nature of K2. One of the incorrectly oriented edge was between the variables 34 and 33. As can be observed from Figure 1(b), 7 of the extra edges were between 33 and some other node. Moreover, an analysis of the order in which K2 selected the parents of node 37 showed that the 3 other extra edges incident on node 37 were recovered due to the greedy nature of K2 which, after picking node 16 as a parent of 37, picked up 33 because of the incorrect orientation, and then recovered the 3 edges of node 37 with 24, 23 and 22 once again due to its greedy search regimen. Similarly, the three extra edges involving node 2, 17 and 18 were

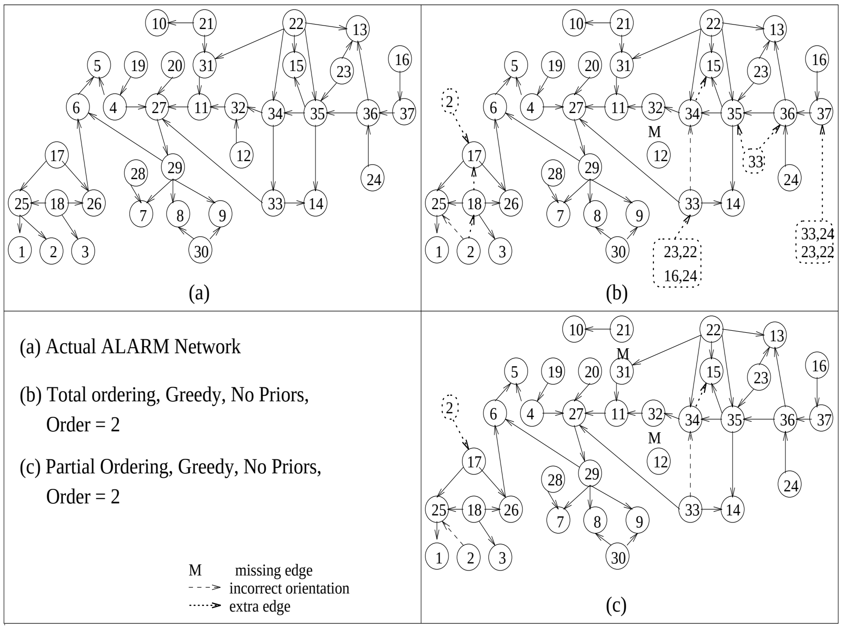

Figure 1. Results of testing the CB algorithm on the ALARM Database

recovered due to the fact that the edge between 2 and 25 was incorrectly oriented. The remaining extra edge was between nodes 15 and 34 which is recovered, once again, due to the greedy nature of K2. The total time taken was under 13 minutes.

Cooper and Herskovits [10] reported that K2, when given a total ordering consistent with the partial order of the nodes as specified by ALARM, recovered the complete network with the exception of one missing edge (between nodes 12 and 32) and one extra arc (from node 15 to 34). [27] reported similar results with the PC algorithm. They applied the PC algorithm [6] to the ALARM database split into two parts of 10000 cases each. The algorithm did not make any linearity assumption. In one case, the recovered network had no extra edge but had two missing edges while in the other case, the network had one extra edge and two missing edges.

To reduce the computational time, and to try to prevent the recovery of extra edges, we modified the algorithm by deleting step 7 of the algorithm. Instead of using a total order, K2 used a partial order defined on the nodes by the DAG constructed by step 6. The sets $\pi_i, 1 \leq i \leq n$, constructed

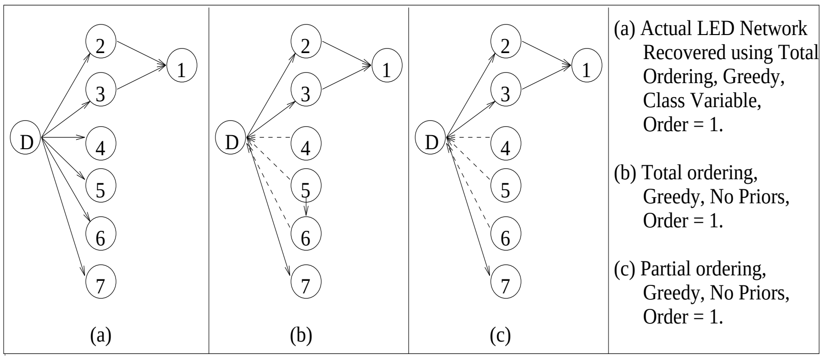

Figure 2. Results of testing the CB algorithm on the LED Database

by step 6 were given as input to K2 with the constraint that each node i could have parents only from the set π_i. The network recovered by the algorithm after having used CI relations of *up to only order 2* is shown in Figure 1(c). It recovered 44 edges (the extra missing edge being $21 \rightarrow 31$); there were 2 extra edges (between 2 and 17, and between 34 and 15) while 2 edges were incorrectly oriented. However, the metric used by K2 ranked the earlier network structure (Figure 1(b)) to be more probable. The time taken was reduced to under 7 minutes.

4.2. The LED Database

The LED database was used by Fung and Crawford [4] for the evaluation of their Markov network generating program called *Constructor*. The network (Figure 2(a)) represents a faulty LED display because the LED segment 1 is conditionally independent of the digit key given the state of the LED segments 2 and 3, whereas in a normal display knowledge about the depressed key is sufficient to indicate which LED segments are on. There are 8 variables, one representing the digit key and the remaining 7 corresponding to the seven segments of the display.

We used the algorithm to reconstruct the faulty LED network (Figure 2(a)) using a database of 199 cases. The **CB** algorithm reconstructed the network (Figure 2(b)) with 3 edges incorrectly oriented and one extra edge in less than 1 second using CI tests up to order 1. A subsequent analysis of the independence statements computed by CB found that the three incorrectly oriented edges were due to perceived independence of the pairs

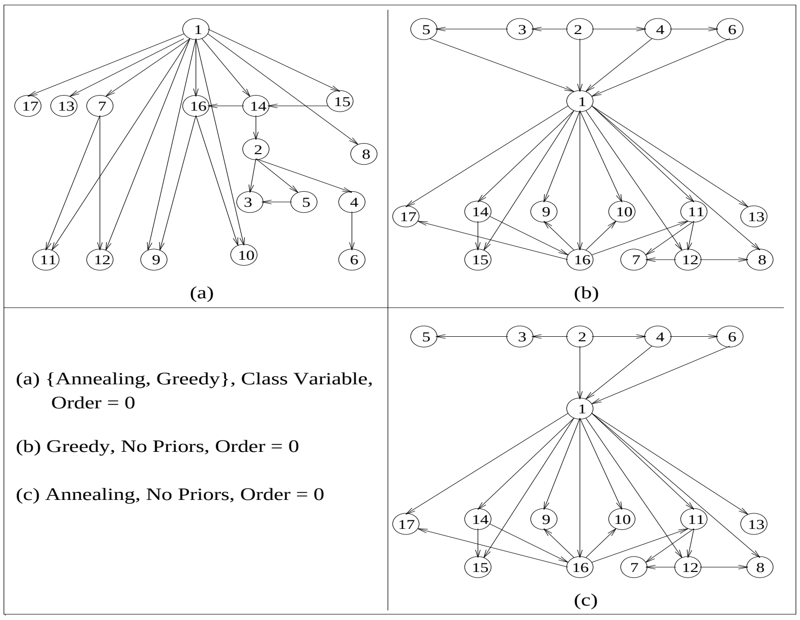

Figure 3. Results of testing the CB algorithm on the Letter Recognition Database

$(3,5)$, $(3,6)$, and $(4,5)$. While the underlying model did not support these independence statements, the data did. Or perhaps, the data was too little for the χ^2 test to be accurate, even for low order CI tests. As such, Step 3 oriented the edges according to the perceived independence. When we ran the modified version of CB using the partial order, the same network was recovered, except for the fact that there was no extra edge (Figure 2(c)).

We also used the **CB** algorithm to reconstruct the LED network by requiring the class variable (corresponding to the digit) to be the root of the resultant network structure. The algorithm recovered the original LED network (Figure 2(a)) in this case using, once again, CI tests up to order 1.

4.3. The LETTER RECOGNITION Database

The LETTER RECOGNITION database [23] was used to investigate the ability of several variations of Holland-style adaptive classifier systems to learn to correctly identify each of a large number of black-and-white rectangular pixel displays (presented as 16-attribute vectors) as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (representing the primitive statistical features of the pixel distribution).

We used the first 10000 cases for learning and the remaining 10000 cases for testing the constructed network for predictive accuracy. The algorithm was tested using both search strategies - greedy search as well as simulated annealing. Moreover, with each search strategy, we either used no priors at all, or forced the class variable to be the root node in the resultant network structure. The results obtained are shown in Figure 3. When no prior information was used, the network structure constructed using annealing (Figure 3(c)) was ranked as more probable than the one constructed using the greedy search method (Figure 3(b)). However, while the network formed using greedy search was too large for inference, the network formed using annealing had a predictive accuracy of about 82.5%. The best predictive accuracy achieved by [23] (using 16000 cases for learning and 4000 cases for testing) was similar.

On forcing the class variable to be the root of the resultant Bayesian network, both the probability and the predictive accuracy dropped (though they were the same for the two search strategies)(Figure 3(a)). The predictive accuracy obtained was around 80.5%.

4.4. The SOYBEAN Database

The SOYBEAN database is one the most widely used databases in the domain of machine learning. Created by Michalski and Chilausky [24], the 36 variable database consists of a 307 case training set and a 376 case evaluation set in the domain of soybean-plant-disease diagnosis.

There are 19 classes, only the first 15 of which have been used in prior work. The accompanying documentation states that the last four classes are unjustified by the data since they have so few examples. As such, the number of instances used for constructing the network is 290, while 340 samples are used from the test data for checking the accuracy of the network recovered. Missing values were just treated like another possible value of the variable concerned[7].

Once again, we constructed networks using both search strategies. As in the case of the Letter Recognition database, for each search strategy, we constructed the network without any priors as well as by forcing the class node to be the root of the network. The results are shown in Figure 4.

[7]Other methods of treating missing values are described in [28, 29]

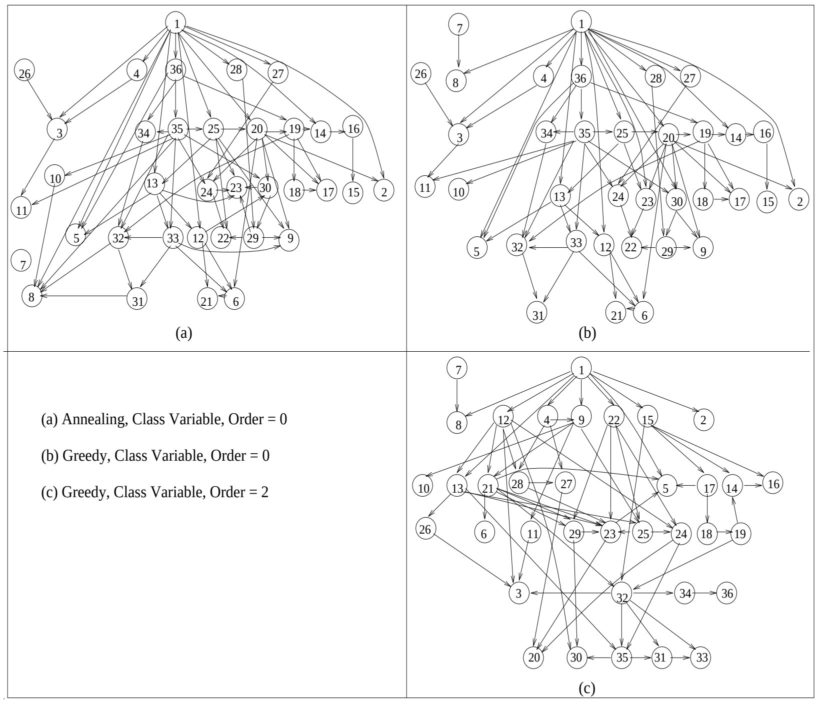

Figure 4. Results of testing the CB algorithm on the Soybean Database

When no prior information was used, the networks generated (with greedy search as well as with simulated annealing) were ranked as much more probable than the networks generated with the corresponding search strategy by forcing the class variable to be the root of the resultant network structure. However, these networks (generated without any priors) had low predictive accuracies (about 68%).

On the other hand, the networks generated by forcing the class variable to be the root of the recovered Bayesian network structure were ranked as less probable by the K2-metric, but had much better predictive accuracies. The network generated using annealing and CI tests of order 0 (Figure 4(a)) had a predictive accuracy of about 83%. The networks generated using the greedy search method produced even more interesting results. While the algorithm constructed the network shown in Figure 4(c) as the most probable network using CI tests of upto order 2, this network had a 79% accuracy as compared to the network generated using just CI tests of order 0 (Figure 4(b)), which was ranked as less probable but had a predictive accuracy of more than 86%.

Herskovits [9] had achieved a maximum predictive accuracy of about 84% with the K2 algorithm, and a predictive accuracy of 86% with K2-multiscore (an extension of the K2 algorithm in which the 'n' most probable networks corresponding to a particular ordering are generated, and are collectively used for prediction with the network probabilities being used as weighting factors).

Herskovits [9] also discusses in detail the possible reasons why the network generated by the K2 algorithm had a predictive accuracy much lower than the 98% accuracy achieved by [24]. Similar arguments would explain why the **CB** algorithm generated networks with much lower predictive accuracy, since both algorithms are based on the same metric.

5. SUMMARY AND OPEN PROBLEMS

In this paper, we have presented a method of recovering the structure of Bayesian belief networks from a database of cases by integrating CI test based methods and Bayesian methods.

These results are quite encouraging because they show that the **CB** algorithm can recover a reasonably complex Bayesian network structure from data using **substantially low order CI relations**. Moreover, since it generates an ordering on the nodes from the database of cases only, without any outside information, it **eliminates the requirement for the user to provide an ordering on the variables**.

In the worst case, the CB algorithm is exponential in the number of variables, as explained below. Steps 1 (initialization) and 10 (output) of the algorithm are executed only once. The number of times that steps 2 through 9 of the CB algorithm are executed is bound by the sum of the largest two degrees in the undirected graph constructed at the end of step 2, by an argument almost identical to that of [6, page 68]. Each of steps 3 through 9 have only polynomial complexity in the number of variables, by arguments that are either simple or described in [5, 10]. In step 2, the number of independence tests carried out is exponential in the size of the order of the independence relations to be tested, which is bounded by the maximum of $|A_G ab|$. Note that the CB algorithm is polynomial for graphs for which $|A_G ab|$ is constant as the number of vertices increases, i.e. sparse graphs. *Our results indicate that the CB algorithm recovers Bayesian network structures in polynomial time in the number of domain variables, because the highest order of independence relations to be tested is very low.*

Although CB works well on the tested databases and appears to be quite promising, a number of issues that could improve the performance of the algorithm need to be considered further.

Firstly, note that the method outlined in the paper constructs the single "best" model from the database of cases. This model is then used for future applications, like inference, *as if this particular model is the true model.* This would be reasonable only if the most probable network given the database of cases has a much higher probability as compared to the next most probable network given the data. However, this assumption may not be valid in many practical situations. Many authors (e.g., [30, 31]) have stressed the importance of (and have offered possible solutions) accounting for model uncertainty. As Madigan and Raftery [31] state, "a panacea is provided by the standard Bayesian formalism which averages the posterior distribution of the quantity of interest under each of the models, weighted by their posterior model probabilities". However, this problem is highly complicated by the extremely large number of possible network structures for any given set of variables. The **CB** algorithm can be easily extended on the lines of Herskovits' **K2-multiscore** algorithm [9] to construct the 'n' most probable models corresponding to a particular ordering of the nodes. However, since we cannot possibly find the set of all possible models and average over them, additional work is needed to develop methods of finding a tractable number of models which would give a reasonable approximation to the actual solution. Solutions proposed by Madigan and Raftery [31] and Draper [30] though feasible for small networks might be computationally very expensive for networks having a large number of variables.

Secondly, we have used a fixed α level for the χ^2 test. This will almost certainly introduce dependencies that are purely the result of chance. It is possible to use the technique of Cross Validation for tuning this parameter. Fung and Crawford [4] discuss the tuning of the alpha level in performing belief-network learning.

Also, since the quality of the recovered network structure is very sensitive to the ordering determined by phase I of the CB algorithm, efforts need to be made to find better and more efficient heuristics than the one presented in this paper that enable the selection of one orientation of an undirected edge over the other, since in general there will be a number of such undirected edges after steps 3 and 4 of the algorithm.

Moreover, most of the steps of the CB algorithm are inherently parallel. Hence, a huge reduction in the time required to recover the network structure can be possibly obtained by parallelizing the CB algorithm.

Finally, the CB algorithm uses a greedy strategy as a stopping criteria. It uses the probability of the entire network, as measured by the K2 metric, to decide when to stop; the algorithm stops when the value of the metric for the entire network is less than the value which had been computed for the network structure recovered in the previous iteration (i.e., for a lower order of the CI tests). There is a need to design and evaluate alternative methods of terminating the algorithm.

Acknowledgments

We are thankful to Prof. G. Cooper for providing the ALARM network database and to Dr. R. Fung for providing the LED network database. We are also grateful to the anonymous referees for their helpful comments and suggestions for improving the paper.

References

1. Singh, M., and Valtorta, M., An algorithm for the construction of Bayesian network structures from data, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, Washington D.C., Morgan Kaufmann, 259–265, 1993.
2. Glymour, C., Scheines, R., Spirtes, P., and Kelly, K., *Discovering Causal Structure*, Academic Press, San Diego, CA, 1987.
3. Pearl, J., and Wermuth, N., When can association graphs admit a causal interpretation? (first report), *Preliminary Papers of the 4th International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 141–150, January 3–6 1993.
4. Fung, R. M., and Crawford, S. L., Constructor: A system for the induction of probabilistic models, *Proceedings of AAAI*, Boston, MA, MIT Press, 762–769, 1990.
5. Verma, T., and Pearl, J., An algorithm for deciding if a set of observed independencies has a causal explanation, *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 323–330, 1992.
6. Spirtes, P., and Glymour, C., An algorithm for fast recovery of sparse causal graphs, *Social Science Computing Review*, 9(1), 62–72, 1991.
7. Pearl, J., and Verma, T., A theory of inferred causation, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 441–452, 1991.
8. Spirtes, P., Glymour, C., and Scheines, R., *Causality from Probability*, Pitman, London, 1990, 181-199.
9. Herskovits, E. H., Computer-based probabilistic-network construction, PhD thesis, Medical Information Sciences, Stanford University, Stanford, CA, 1991.
10. Cooper, G. F., and Herskovits, E. H., A Bayesian method for the induction of probabilistic networks from data, *Machine Learning*, 9, 309–347, 1992.
11. Lauritzen, S. L., Thiesson, B., and Spiegelhalter, D., Diagnostic systems created by model selection methods–a case study, *Preliminary Papers of the 4th International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 93–105, January 3-6 1993.

12. Lam, W., and Bacchus, F., Using causal information and local measures to learn Bayesian networks, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, Washington D.C., Morgan Kaufmann, 243–250, 1993.
13. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufman, San Mateo, CA, 1988.
14. Lauritzen, S. L., and Wermuth, N., Graphical models for associations between variables, some of which are qualitative and some quantitative, *Annals of Statistics*, 17, 31–57, 1989.
15. Lauritzen, S. L., and Wermuth, N., Graphical models for associations between variables, some of which are qualitative and some quantitative: Correction note, *Annals of Statistics*, 17, 1916, 1989.
16. Shachter, R. D., A graph-based inference method for conditional independence, *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, Los Angeles, Morgan Kaufmann, 353–360, 1991.
17. Geiger, D., and Heckerman, D., Advances in probabilistic reasoning, *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, Los Angeles, Morgan Kaufmann, 118–126, 1991.
18. Dor, D., and Tarsi, M., A simple algorithm to construct a consistent expression of a partially oriented graph, Technical Report R-185, Cognitive Systems Laboratory, Department of Computer Science, Univ. of California at Los Angeles, 1992.
19. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., Optimization by simulated annealing, *Science*, 220, 671–680, 1983.
20. Aarts, E., and Korst, J., *Simulated Annealing and Boltzmann Machines: a stochastic approach to combinatorial optimization and neural computing*, John Wiley and Sons, 1989.
21. Beinlich, I. A., Suermondt, H. J., Chavez, R. M., and Cooper, G. F., The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, London, England, 247–256, 1989.
22. Murphy, P. M., and Aha, D. W., UCI repository of machine learning databases, machine-readable data repository - Department of Information and Computer Science, University of California, Irvine.
23. Frey, P. W., and Slate, D. J., Letter recognition using holland-style adaptive classifiers, *Machine Learning*, 6(2), 1991.
24. Michalski, R. S., and Chilausky, R. L., Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis, *International Journal of Policy Analysis and Information Systems*, 4(2), 1980.
25. Anderson, S. K., Olesen, K. G., Jensen, F. V., and Jensen, F., HUGIN - A shell for building Bayesian belief universes for expert systems, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1080–1085, 1989.
26. Henrion, M., Propagating Uncertainty in Bayesian networks by probabilistic logic sampling, in *Uncertainty in Artificial Intelligence 2* (J. F. Lemmer,

L. N. Kanal, Eds.), Elsevier Science, North-Holland, 149-163, 1988.
27. Spirtes, P., Personal communication.
28. Buntine, W., Theory refinement on Bayesian networks, *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, Los Angeles, Morgan Kaufmann, 52–60, 1991.
29. Dempster, A., Laird, N., and Rubin, D., Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society (Series B)*, 39, 1–38, 1977.
30. Draper, D., Assessment and propagation of model uncertainty, *Preliminary Papers of the 4th International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 497–509, January 3-6 1993.
31. Madigan, D., and Raftery, A. E., Model selection and accounting for model uncertainty in graphical models using occam's window, Technical Report 213 (revised), Dept. of Statistics, Univ. of Washington, 1993.