An Algorithm for Bayesian Belief Network Construction from Data

Jie Cheng, David A. Bell, Weiru Liu

School of Information and Software Engineering
University of Ulster at Jordanstown
Northern Ireland, UK, BT37 0QB
e-mail: {j.cheng, da.bell, w.liu}@ulst.ac.uk

Abstract

This paper presents an efficient algorithm for constructing Bayesian belief networks from databases. The algorithm takes a database and an attributes ordering (i.e., the causal attributes of an attribute should appear earlier in the order) as input and constructs a belief network structure as output. The construction process is based on the computation of mutual information and cross entropy of attribute pairs. This algorithm guarantees that the *minimal Independent map* [1] of the underlying dependency model is generated, and at the same time, enjoys the time complexity of $O(N^2)$ on conditional independence (CI) tests. To evaluate this algorithm, we present the experimental results on three versions of the well-known ALARM network database, which has 37 attributes and 10,000 records. The correctness proof and the analysis of computational complexity are also presented. We also discuss the features of our work and relate it to previous works.

1 Introduction

The Bayesian belief network is a powerful knowledge representation and reasoning tool under conditions of uncertainty. A Bayesian belief-network is a directed acyclic graph (DAG) with a conditional probability distribution along each arc [1,2,3]. The DAG structure of such networks contains nodes representing domain variables, and arcs between nodes representing probabilistic dependencies. On constructing Bayesian networks from databases, we use nodes to represent database attributes.

In the last ten years, significant progress has been made in the area of probabilistic inference on belief networks, but the construction of belief networks remains a time consuming task, especially when the number of variables is large. Recently many belief network construction algorithms have be developed. Generally, these algorithms can be grouped into two categories: one category of algorithms uses heuristic search method to construct a model and use a scoring method to evaluate the model. This process continues until the score of the new model is not significantly better than the old one. Different scoring criteria have been applied in these algorithms, such as, Bayesian scoring method [5,6], entropy based method [21], and minimum description length method [20]. The other category of algorithms constructs Bayesian networks by analyzing dependency relationships among nodes. The dependency relationships are measured by using some kind of conditional independence (CI) test. The algorithms described in [4,8,9,10] and the proposed algorithm in this paper belong to this category. Both of these two categories of algorithms have their advantage and disadvantage: Generally, the first category of algorithms has less time complexity in the worst case (when the underlying DAG is densely connected), but it may not find the best solution due to its heuristic nature; The second category of algorithms is usually asymptotically correct when the probability distribution of data is *DAG-Isomorphic* (The definition is in [1]), but as Cooper *et al.* pointed out in [5], CI tests with large condition-sets may be unreliable unless the volume of data is enormous.

On developing this algorithm, we take the following two facts into consideration. First of all, real world situations usually yield sparse networks, and densely connected networks reveal very few independence relationships and thus contain little valuable information. Therefore, the algorithm should be particularly efficient when the database has a sparse underlying network. Secondly, since CI tests with large condition-sets are computational expensive and may be unreliable, we try to avoid CI tests with large condition-sets and use as few CI tests as possible. Considering the above discussion, we developed a three phases algorithm that constructs a draft of

the network structure in the first phase using mutual information and 'thickens' and 'thins' it using CI tests in the second and third phases. When the underlying network is sparse, the draft can be very similar to the underlying model. Therefore, our algorithm can avoid many unnecessary CI tests with large condition-sets and also reduce the number of CI tests.

The remainder of this paper is organized as follows. In Section 2, we give the background information and introduce our information theory based algorithm. In Section 3, we present our algorithm in detail and give the correctness proof. Section 4 contains the experimental results on 3 data set of alarm network. Finally, in Section 5, we discuss the related works and the important features of our work.

2 An Approach Based on Information Theory

The Bayesian belief network is a kind of probabilistic models. It uses DAG to represent dependency relationships between variables. Since every independence statement in belief networks satisfies a group of axioms (See [1] for details), we can construct belief networks from data by analyzing conditional independence relationships. This CI test based method is used by all the algorithms of the second category described in Section 1.

To introduce our approach, we first review the concept of *d-separation* [1], which plays an important role in our algorithm. For any three disjoint node sets X, Y, and Z in a belief network, X is said to be *d-separated* from Y by Z if there is no active undirected path between X and Y. A path between X and Y is active if: (1) every node in the path having head-to-head arrows is in Z or has a descendant in Z; (2) every other node in the path is outside Z. To understand *d-Separation*, we can use an analogy, which is similar to the one suggested in [3]. We view a belief network as a network system of information channels, where each node is a valve that is either active or inactive and the valves are connected by noisy information channels. The information flow can pass through an active valve but not an inactive one. When all the valves (nodes) on one undirected path between two nodes are active, we say this path is **open**. If any one valve in the path is inactive, we say the path is **closed**. When all paths between two nodes are **closed** given the status of a set of valves (nodes), we say the two nodes are *d-separated* by the set of nodes. The status of valves can be changed through the instantiation of a set of nodes. The amount of information flow between two nodes can be measured by using mutual information, when no nodes are instantiated, or Kullback-Leibler cross entropy, when some other nodes are instantiated.

In information theory, the mutual information of two nodes X_i, X_j is defined as

$$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}; \quad (1)$$

and Kullback-Leibler cross entropy is defined as

$$D(X_i, X_j | C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j | c)}{P(x_i | c), P(x_j | c)}, \quad (2)$$

where X_i, X_j are two nodes and C is a set of nodes. In our algorithm, we use cross entropy as CI tests to measure the average information between two nodes when the statuses of some valves are changed by the condition-set C. When $D(X_i, X_j | C)$ is smaller than a certain threshold value ε, we say that X_i, X_j are *d-separated* by the condition-set C, and they are conditionally independent.

This algorithm also makes the following assumptions: (1) The database attributes have discrete values and there are no missing values in all the records. (2) The volume of data is large enough for reliable CI tests. (3) The ordering of the attributes is available before the network construction, i.e., a node's parents nodes should appear earlier in the order.

3 The Algorithm for Belief Network Construction

This algorithm has three phases: drafting, thickening and thinning. In the first phase, this algorithm computes *mutual information* of each pair of nodes as a measure of closeness, and creates a draft based on this information. In the second phase, the algorithm adds arcs when the pairs of nodes cannot be *d-separated*. The result of Phase II is an *independence map* (*I-map*) [1] of the underlying dependency model. In the third phase, each arc of the *I-map* is examined using CI tests and will be removed if the two nodes of the arc can be *d-separated*. The result of Phase III is the *minimal I-map* [1].

3.1 The Algorithm

Phase I: (Drafting)

1. Initiate a graph $G(V, E)$ where V={all the nodes of a data set}, E={ }. Initiate two empty ordered set S, R.
2. For each pair of nodes (v_i, v_j) where $v_i, v_j \in V$, compute mutual information $I(v_i, v_j)$ using equation (1). For the pairs of nodes that have mutual information greater than a certain small value ε, sort them by their mutual information from large to small and put them into an ordered set S.
3. Get the first two pairs of nodes in S and remove them from S. Add the corresponding arcs to E. (the direction of the arcs in this algorithm is determined by the previously available nodes ordering.)
4. Get the first pair of nodes remained in S and remove it from S. If there is no **open** path between the two nodes (these two nodes are *d-separated* given empty set), add the corresponding arc to E; Otherwise, add the pair of nodes to the end of an ordered set R.
5. Repeat step 4 until S is empty.

In order to illustrate this algorithm's working mechanism, we use a simple multi-connected network example borrowed from [3]. Suppose we have a database that has underlying Bayesian network as Figure 1.a; and we also have a nodes' order as *A, B, C, D, E*. Our task is to find out the exact network structure. After step 2, we can get the mutual information of all 10 pair of nodes. Suppose we have $I(B,D) \geq I(C,E) \geq I(B,E) \geq I(A,B) \geq I(B,C) \geq I(C,D) \geq I(D,E) \geq I(A,D) \geq I(A,E) \geq I(A,C)$, and all the mutual information is greater than ε, we can construct a draft graph shown in Figure 1.b after step 5. Please note that the order of mutual information between nodes can not be arbitrary. For example, from information theory, we have $I(A,C) < Min(I(A,B),I(B,C))$. This is also the reason why Phase I can construct a graph close to the original graph to some extent. When the underlying graph is sparse, Phase I can construct a graph very close to the original one. In fact, if the underlying graph is a singly connected graph (a graph without undirected cycle), Phase I of this algorithm is essentially the algorithm of Chow and Liu[6], and it guarantees the constructed network is the same as the original one. Our algorithm can be viewed as an extension of Chow and Liu's algorithm to multi-connect networks. In this example, *(B,E)* is wrongly added and *(D,E)* is missing because of the existing **open** path *(D-B-E)* and *(D-B-C-E)*. The draft graph created in this phase is the base for next phase.

Phase II: (Thickening)

6. Get the first pair of nodes in R and remove it from R.
7. Find a block set that blocks each **open** path between these two nodes by a set of minimum number of nodes. (This procedure **find_block_set (current graph, node1, node2)** is given at the end of this subsection.) Conduct a CI test. If these two nodes are still dependent on each other given the block set, connect them by an arc.
8. go to step 6 until R is empty.

In our example, the graph after Phase II is shown in Figure 1.c. When this algorithm examines the pair of nodes *(D,E)* in step 7, it finds that $\{B\}$ is the minimum set which blocks all the **open** paths between D and E. Since the CI test can reveal that D and E are still dependent given $\{B\}$, arc *(D,E)* is added. Arc *(A,C)* is not added because the CI test can reveal that A and C are independent given block set $\{B\}$. Arc *(A,D)*, *(C,D)* and *(A,E)* are not added for the same reason. In this phase, the algorithm examines all pairs of nodes that have mutual information greater than ε, an arc is not added only when the two nodes are independent given some block set. It is possible that some arcs are wrongly added in this phase.

Phase III: (Thinning)

9. For each arc in E, if there are **open** paths between the two nodes besides this arc, remove this arc from E temporarily and call procedure **find_block_set (current graph, node1, node2)**. Conduct a CI test on the condition of the block set. If the two nodes are dependent, add this arc back to E; otherwise remove the arc permanently.

The 'thinned' graph of our example is shown in Figure 1.d, which is the same as the original graph. Arc *(B,E)* is removed because *B* and *E* are independent given $\{C,D\}$. This procedure generates the *minimal I-map* of the underlying dependency model.

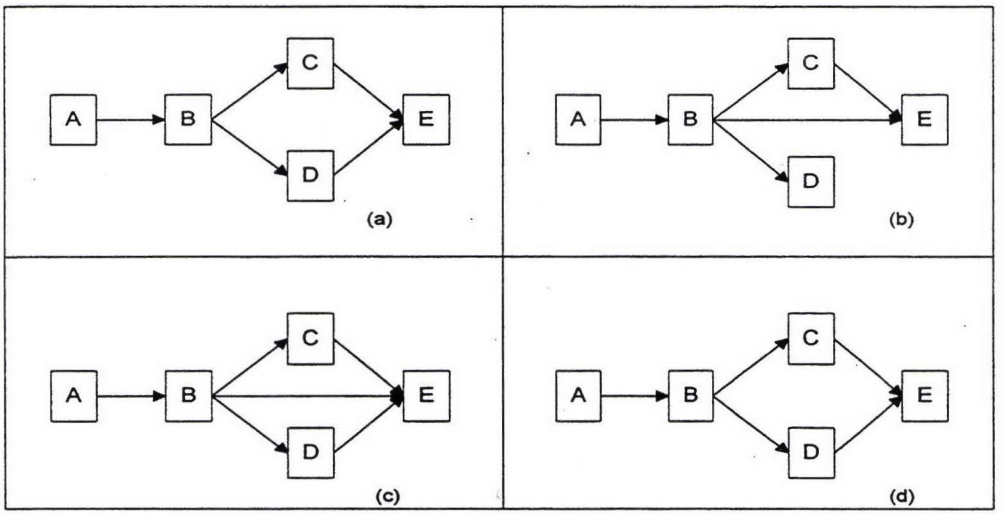

Figure 1. A simple multi-connected network.

Finding Minimum Block Set

As suggested by Acid *et al.* in [19], knowing the minimum block set of two nodes in belief networks can be very useful in several ways. In our algorithm, we try to avoid CI tests with large condition-sets by finding minimum block sets. The following simple procedure uses a heuristic-search method to find the block set. An algorithm for finding minimum *d-Separation* sets can be found in [19].

Procedure find_block_set (current graph, node1, node2)
begin
find all the undirected paths between node1 and node2;
store the **open** paths in open_path_set, store the **closed** paths in closed_path_set;
do
  while there are **open** paths which have only one node **do**
    store the nodes of each such path in the block set;
    remove all the blocked paths by these nodes from the open_path_set and closed_path_set;
    from the closed_path_set, find paths **opened** by the nodes in block set and move them to
    the open_path_set, shorten such paths by removing the nodes that are also in the block set;
  end while
  if there are **open** paths **do**
    find a node which can block maximum number of the rest paths and put it in the block set;
    remove all the blocked paths by the node from the open_path_set and the closed_path_set;
    from the closed_path_set, find paths **opened** by this node and move them to
    the open_path_set, shorten such paths by removing the nodes that are also in the block set;
  end if
until there are no **open** path
end procedure.

Because this procedure uses a greedy search method, it does not guarantee that a minimum block set is found. However, in the case of ALARM network, which will be discussed in Section 4, all the block sets, over 200, found

by this procedure are minimum. This procedure can also be easily extended to find the block set of two sets of nodes.

3.2 Correctness

Suppose a data set is large enough for reliable CI tests and has underlying dependency model M. We give the proofs of the following propositions.

Proposition 1 *Graph G2 generated after Phase II is an I-map of M.*
Proof: Phase I and Phase II of our algorithm examined all the arcs between any two nodes that are not independent. An arc is not added only if these two nodes are *d-separated* by a set of other nodes. Hence, any pair of not connected nodes of *G2* are conditional independent in *M*. Q.E.D.

Proposition 2 *Graph G3 generated after Phase III is a minimal I-map of M.*
Proof: Since an arc is removed in Phase III only if the pair of nodes are *d-separated*, *G3* is an *I-map* of *M*. Next, we shall prove that this *I-map* is a *minimal I-map*. Suppose *G3* is not a *minimal I-map*, then there must exist an arc (*a*, *b*) which can be removed from *G3* and the remained graph *G3'* is still *an I-map* of *M*. By the definition of *I-map*, node *a* and *b* are independent in *M* and can be *d-separated* by blocking a set of all the real **open** paths *Pr* in *M*. From our algorithm, we can get that *a* and *b* are connected in *G3* only if *a* and *b* cannot be *d-separated* by blocking a set of all the **open** paths *P* in *G3*. Since *G3* is an *I-map* of *M*, *P* includes *Pr* and some pseudo-paths. Because pseudo-paths cannot pass information, information must be passed through the paths in *Pr*. Therefore *Pr* cannot *d-separate a* and *b*. This contradicts our assumption that *a* and *b* are independent in *M*. Hence, *G3* is a *minimal I-map* of *M*. Q.E.D.

The above propositions ensure that our algorithm can construct a *minimal I-map* of the underlying dependency model. When the underlying model is *DAG-isomorphic* [1], our algorithm can construct the exact DAG given the assumptions of Section 2.

3.3 Complexity Analysis

Suppose a data set has N attributes, the maximum number of possible values of any attribute is r, and an attribute may have k parents at most. We give the complexity as follows.

Phase I: Since Phase I computes mutual information between any two nodes, it needs N^2 mutual information computations. By **equation (1)** of Section 2, each computation of mutual information requires $O(r^2)$ times of basic operations such as logarithm, multiplication and division. Sorting the nodes pairs can be finished in $O(N \log N)$ steps by using *quicksort* algorithm. The time complexity of this phase on basic operations is $O(N^2 r^2)$.

Phase II: This phase tries to add each arc to the graph and requires CI tests at most N^2 times. By equation (2) of Section 2, each CI test requires at most $O(r^{k+2})$ basic operations. The complexity of this phase on basic operations is $O(N^2 r^{k+2})$. In the worst case, it requires basic operations $O(N^2 r^N)$ times. Because Procedure **find_block_set** is just an optimization and can be replaced by a simple $O(N)$ procedure that may return larger block sets. We do not include the complexity of this procedure here.

Phase III: This phase tries to remove each arc from the graph and has the same time complexity as Phase II.

Overall, this algorithm requires CI tests of $O(N^2)$ complexity. (Mutual information computations can be regarded as CI tests with empty condition-sets.) The time complexity on basic operations is $O(N^2 r^{k+2})$. In the worst case, when all the CI tests require condition-sets on all the other nodes, the time complexity on basic operation is $O(N^2 r^N)$.

4 Results on ALARM Network

ALARM network[15] is a medical diagnostic alarm message system for patient monitoring, it contains 37 nodes and 46 arcs (see Figure 2). This belief network has become the *de facto* benchmark for evaluating belief network construction algorithms. Researchers in this field use data sets generated from three versions of this belief network. The three versions have the same structure but different probability distributions. To evaluate our algorithm, we use three data sets generated from each of the three versions of ALARM network. We call them dataset1, dataset2 and dataset3[i]. Each of them has 10,000 cases.

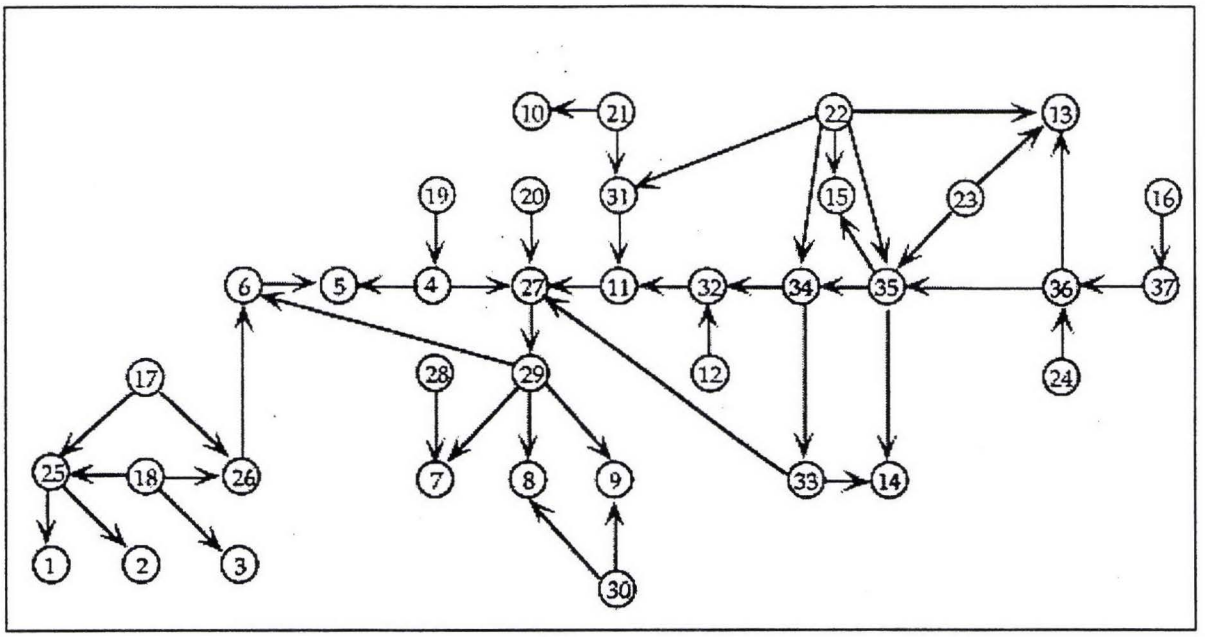

Figure 2. The ALARM belief network

Since this algorithm requires nodes ordering, we use the ordering described in the web page of Norsys Software Corp. for dataset1 and dataset2 and use the ordering described in [5] for dataset3. Please note that the actual orderings make no difference to the algorithm as long as they preserve the cause and effect relationships. To make CI tests more reliable when the volume of data is not large enough, we also modified equation (2) by taking the variable's *degree of freedom* into consideration. We use 0.003 as the value of ε.

We summarize our experimental results into the following two tables. Table 1 shows the detail test results of dataset1, and table 2 compares the results of the three data sets. All these experiments were conducted on a Pentium 90MHz PC and the data sets are stored in a Microsoft Access database.

Phase	Arcs	Missing	Wrongly Added	No. of CI Tests[ii]						Time (Minute)
				0	1	2	3	4	Total	
I	43	5 {22-15,22-13,33-27,23-13,35-14}	2 {34-13,34-14}	666	0	0	0	0	666	20.00
II	50	0	4 {34-13,34-14} + {35-13,32-27}	0	166	35	4	0	205	8.34
III	46	0	0	0	9	3	1	2	15	0.81

Table 1. Networks constructed at each phase for dataset1 of ALARM database.

Data sets	Results	No. of CI Tests						Time (Minute)			
		0	1	2	3	4	Total	Phase I	Phase II	Phase III	Total
Dataset1	Correct	666	175	38	5	2	886	20.00	8.34	0.83	29.17
Dataset2	Missing {11-27}	666	184	18	1	0	869	20.00	6.50	0.50	27.00
Dataset3	Missing {12-32,21-31}	666	86	26	1	0	779	20.00	3.67	0.50	24.17

Table 2. Results on dataset1, dataset2 and dataset3.

From Table 1, we can analyze the result of each phase. In Phase I, the mutual information of all 666 ($=37\times(37-1)/2$) pairs of nodes were computed; and our algorithm uses this information to construct a draft. The draft is quite similar to the underlying graph and has only 5 missing arcs and 2 wrongly added arcs. In Phase II, this algorithm conducted 205 CI tests and added 7 arcs; among which, all the 5 real arcs missed in the first phase were added. The result of this phase is an *I-map*. In Phase III, only 15 CI tests were conducted and the condition-sets have at most 4 nodes. This phase removed all the 4 wrongly added arcs correctly and constructed the exact belief network. In table 2, we compare the results on dataset1, dataset2 and dataset3. The difference in the running time and the number of CI tests among the three data sets is due to the difference in the obviousness of dependency relationships of the underlying probability distribution. For example, the dependency relationships are more obvious in dataset3 than those in dataset1. Our result on dataset3 missed two arcs of ALARM network; this is due to the fact that 12-32 and 21-31 are actually independent in dataset3: the mutual information between 12 and 32 is 7×10^{-5}, between 21 and 31 is 3×10^{-5}, while all the rest real arcs have mutual information greater than 0.01. χ^2 tests all show that these two pairs of nodes are independent. Therefore, we consider our result is correct. The reason for the missing arc 11-27 in our result on dataset2 is that this pair of nodes has very weak relationship when node 32 or 34 is instantiated. This may suggest that the dependency relationship between node 11 and 27 can be expressed through other dependency relationships.

Because this algorithm only requires CI tests less than $3N^2$ times and does not require high order CI tests (in the case of the ALARM network, it only requires condition-sets with 4 nodes at most.), it is very efficient and reliable. In our experiments, most running time is consumed by database engine on query preparing and data retrieving. The algorithm also constructed the correct network on 3,000 cases of dataset1 in less than 10 minutes. Moreover, although Phase II and Phase III have exponential time complexity on basic operations in the worst case, in the case of sparse networks like ALARM, they consume less time than Phase I, which has time complexity $O(N^2r^2)$ in the worst case.

5 Discussion

Some of the belief network construction algorithms require nodes ordering, such as the algorithms presented in [5,10,18,20,21] and the proposed algorithm in this paper; Others do not require nodes ordering and can orient the edges automatically, such as the algorithms presented in [4,6,12]. The former group of algorithms can be viewed as a special case of the latter group of algorithms where the nodes ordering is known. Based on the same ideas presented in this paper, we also developed a correct algorithm that does not require nodes ordering [22]. Since dropping the requirement of nodes ordering means many more possibilities to be considered, the new algorithm requires CI tests $O(N^4)$ times. Of course, the complexity on basic operations is exponential. Comparing to the algorithm presented in this paper, the new algorithm uses 20% more CI tests and is 30% slower on the data sets of ALARM network. The actual results are as follows. On dataset1, the result has one missing arc, one wrongly added arc and four not oriented arcs; On dataset2, it has one missing arc, four not oriented arcs and one wrongly oriented arc. On dataset3, it has one missing arc (excluding 12-32, 21-31 for the reason given in Section 4) and five not oriented arcs.

Both of our algorithms can be viewed as extension of the algorithm of [7] to multi-connected networks. One merit of the proposed algorithm is that it preserves the $O(N^2)$ complexity on CI tests. Algorithms described in [4,8,10] can also construct a belief network whose structure is a *minimal I-map* of the underlying dependency model. However, these algorithms require CI tests in exponential complexity. Comparing to the algorithms of [4,8,10], our algorithms have the following two features. (1) By generating a draft in Phase I, our algorithms prevent many pseudo-arcs from being added in Phase II, and therefore avoid high order CI tests in Phase II and III, and also reduce the number of CI test needed in Phase III. (2) By using cross entropy as a measure of dependency relationship, our algorithms can compare two relationships quantitatively, and therefore avoid exponential complexity in the case of no nodes ordering (the detail is in [22].)

Our subsequent research will focus on handling continuous variable nodes and missing values. We also plan to develop a commercial software based on our algorithms.

Acknowledgments

We wish to thank G. Cooper and E. Herskovits for providing their ALARM network data. We also thank D. Heckerman and Norsys Software Corp. for providing their ALARM network probability distributions.

References

[1] Pearl, J. , *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, 1988.

[2] Neapolitan, R.E. , *Probabilistic reasoning in expert systems: theory and algorithms*, John Wiley & Sons, 1990.

[3] Spirtes, P., Glymour, C. and Scheines, R., *Causation, Prediction, and Search* (Book), http://hss.cmu.edu/html/departments/philosophy/TETRAD.BOOK/book.html,1996.

[4] Spirtes, P., Glymour, C. and Scheines, R., An algorithm for fast recovery of sparse causal graphs, *Social Science Computer Review*, 9, 62-72, 1991.

[5] Cooper, G.F., Herskovits, E., A Bayesian Method for the induction of probabilistic networks from data, *Machine Learning*, 9, 309-347, 1992.

[6] Heckerman, D., Geiger, D. and Chickering, D.M., Learning Bayesian networks: the combination of knowledge and statistical data, *Technical Report MSR-TR-94-09*, Microsoft Research, 1994.

[7] Chow, C.K. and Liu, C.N., Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14, 462-467, 1968.

[8] Wermuth, N. and Lauritzen, S., Graphical and recursive models for contingency tables. *Biometrika*, 72, 537-552, 1983.

[9] Fung, R.M. and Crawford, S.L., Constructor: a system for the induction of probabilistic models. *Proceedings of AAAI (pp. 762-769)*, Boston, MA: MIT Press.

[10] Srinivas, S. Russell, S. and Agogino, A., Automated construction of sparse Bayesian networks from unstructured probabilistic models and domain information, In Henrion, M., Shachter, R.D., Kanal, L.N. and Lemmer, J.F. (Eds.), *Uncertainty in artificial intelligence 5*, Amsterdam: North-Holland, 1990.

[11] Buntine, W., A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2), 195-210, 1996.

[12] Singh, M. and Valtorta, M. Construction of Bayesian network structures from data: a brief survey and an efficient algorithm, *International Journal of Approximate Reasoning*, 12, 111-131, 1995.

[13] Wong, S.K.M. and Xiang, Y. Construction of a Markov network from data for probabilistic inference, Proc. *Third International Workshop on Rough Sets and Soft Computing*, San Jose, CA, 562-569, 1994.

[14] Sarkar, S. and Murthy, I. Constructing efficient belief network structures with expert provided information, *IEEE Transactions on knowledge and data engineering*, 8-1, 1996.

[15] 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 Second European Conference on Artificial Intelligence in Medicine* (pp.247-256), London, England, 1989.

[16] Bouckaert, R.R., Properties of Bayesian belief network learning algorithms. In *Proc. of tenth conference on uncertainty in artificial intelligence*, Mantaras, R. and Poole, D. (Ed.), Morgan Kaufmann, 1994.

[17] Madigan, D., Mosurski, K. and Almond R.G., Explanation in belief networks. *Technical Report*, Department of Statistics, University of Washington, 1994.

[18] Acid, S., and Campos, L.M., BENEDICT: An Algorithm for Learning Probabilistic Belief Networks, *Sixth International Conference IPMU'96*, 1996.

[19] Acid, S., and Campos, L.M., An Algorithm for Finding Minimum d-Separating Sets in Belief Networks. *Proceedings of UAI'96*, 1996.

[20] Suzuki, J., Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique, *Proceedings of the international conference on machine learning*, Bally, Italy, 1996.

[21] Herskovits, E.H., Computer-based probabilistic network construction, *Doctoral dissertation*, Medical information sciences, Stanford University, Stanford, CA, 1991.

[22] Cheng, J., Bell, DA and Liu W., Learning belief networks based on information theory, forth coming.

---

[i] Dataset1 has the underlying belief network described in the web page of Norsys Software Corp. Dataset2 has the underlying belief network used by David Heckerman. Dataset3 is generated by Gregory F. Cooper and Edward Herskovits. We generated dataset1 and dataset2 using a Monte Carlo technique.

[ii] The CI tests are grouped by the cardinalities of condition-sets.