Scaling Bayesian Network Discovery Through Incremental Recovery

R. Castelo and A. Siebes

Information Systems (INS)

INS-R9901 March 1999

Report INS-R9901
ISSN 1386-3681

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Scaling Bayesian Network Discovery Through Incremental Recovery

Robert Castelo Arno Siebes

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{robert,arno}@cwi.nl

ABSTRACT

Bayesian networks are a type of graphical models that, e.g., allow one to analyze the interaction among the variables in a database. A well-known problem with the discovery of such models from a database is the "problem of high-dimensionality". That is, the discovery of a network from a database with a moderate to large number of variables quickly becomes intractable.

Most solutions towards this problem have relied on prior knowledge on the structure of the network, e.g., through the definition of an order on the variables. With a growing number of variables, however, this becomes a considerable burden on the data miner. Moreover, mistakes in such prior knowledge have large effects on the final network.

Another approach is rather than asking the expert insight in the structure of the final network, asking the database. Our work fits in this approach. More in particular, before we start recovering the network, we first cluster the variables based on a chi-squared measure of association. Then we use an incremental algorithm to discover the network. This algorithm uses the small networks discovered for the individual clusters of variables as its starting point.

We illustrate the feasibility of our approach with some experiments. More in particular, we show that in the case where one knows the network, and thus the order, our algorithm yields almost the same network which is, moreover, still an I-map.

1991 Mathematics Subject Classification: 62-07, 62-09, 62A15, 62H20, 62H30, 68P10, 68T05, 90B40
1991 Computing Reviews Classification System: G.3, I.2.8
Keywords and Phrases: bayesian networks, incremental algorithm, hierarchical clustering of variables, heuristic search.
Note: This work has been carried out under the Esprit-IV project Keso.

1. Introduction

The analysis of interactions among variables is an important aspect of data analysis. Many models for such an analysis have a graphical representation which simplifies the interpretation of the results. Depending on the sort of interactions one wants to discover, a certain graphical representation is used. Our work is focused on Bayesian Networks [14], a type of recursive graphical models [12] . They represent conditional and marginal (in)dependencies by means of acyclic digraphs (also known as DAGs).

Algorithms for the discovery of Bayesian Networks from databases have been developed, based on Bayesian techniques and search techniques from machine learning and optimization; c.f., [4, 13, 6]. As an aside, note that from the search techniques in machine learning, hill-climbing or greedy search is probably the most popular.

The problem with the standard algorithms is that they do not scale well in the number of variables (or attributes) in the database. This problem is known as the problem of high-dimensionality. From

the perspective of data mining, however, scaling is of utmost importance, both in the number of records and in the number of variables. In this paper we present an algorithm that performs well for moderate sized networks of, say, around 50 nodes.

We are not the first to tackle the problem of high dimensionality. One set of approaches stems from a common idea: introduce topological constraints in the discovery process. These constraints may take form of a restriction in the sort of graphical models considered [7], or an order among variables that prohibits certain connections in the graph [4]. It is also possible to constrain the degree of connectivity of the graph, by setting a maximum number of parent nodes for every node[1].

In all of these cases, one specifies prior knowledge through these constraints. In the case of the order constraint, a complete order on the variables has to be defined. This order constrains the possible arcs: a node can only have arcs to nodes that are later in the order. Such prior knowledge is often not available or, at best, only partial. A mistake may lead to a model that reflects very little from the true model that underlies the data.

A different type of approach is to ask the database rather than the expert. That is, induce the prior information from the database. One approach of this kind was proposed by Singh and Valtorta [18]. They use χ^2 conditional independence tests to generate an order on the variables. Once the order is constructed they propose that the algorithm from Cooper and Herskovits [4] is used to discover the network from the database.

Our approach also relies on the database for prior information. However, it does not rely on an order at all. Rather, we use a χ^2-statistic on pairs of variables to compute a similarity matrix over the variables. Next we use a hierarchical cluster algorithm to group similar variables together. Then we use an incremental search strategy to discover the network.

The advantage of this approach vs. that of Singh and Valtorta is twofold. Firstly, we do not use the χ^2-values for testing and, hence, we are less sensitive for unreliable χ^2-values and we do not need α-tuning. Secondly, we do not constrain the search space at all; this advantage is not only with regard to Singh and Valtorta, but to all other proposed algorithms.

The downside is then, of course, that our approach does not work well with almost fully connected networks. From a exploratory analysis point of view, however, such tightly connected networks are not very informative anyway. It is the I-map (independency map) information ([14],p. 92) that interests the data miner during exploratory analysis. Note, that if one wants to use the Bayesian Network for prediction [15], rather than explanatory analysis, or, in general, as a probabilistic expert system [9], constraining the search space may be a more reasonable approach. In that case, one is more interested in *goodness of prediction* rather than *goodness of fit*.

In the next section we discuss the problem of high dimensionality in more depth. The third section describes our algorithm, both the clustering phase and the incremental algorithm are discussed in detail. Section 4 deals with experiments and results. It is shown for three given datasets that our algorithm performs as quick as using an order. In particular, we show that our algorithm recovers almost the original model from which the datasets are sampled, without requiring any order nor any other sort of prior knowledge from the expert. Moreover, the resulting network is still an I-map, i.e., the results the data miner is most interested in are preserved. In the fifth and final section we formulate our conclusions.

2. High Dimensionality in Bayesian Networks

The problem of high dimensionality in Bayesian Networks stems from the total number of possible models. This corresponds to the number of acyclic digraphs given n nodes [16] which is exponential in the number of nodes.

So, it is probably no surprise that finding the network that fits the database for a degree of connectivity greater than one is known to be NP-hard [6]. Therefore, greedy search techniques that at each step consider the addition of a new arc and the deletion and reversal of existing arcs are often used.

[1] every node in the graph represents a variable

If we consider this search process, given n variables, we realize that without an order we will generate at each step of the search a neighbourhood of n^2 networks (slightly less to avoid directed cycles) when generating all possible additions of arcs. If we use a complete order on the variables, then the size of the neighbourhood we generate at each step of the search reduces to the arithmetic sum of the first n natural numbers, i.e. $n(1+n)/2$. It may reduce even more if we consider also a maximum number of incident arcs per variable (a degree of connectivity).

This reduction of the search space leads the discovery process towards certain class of models, and our local maxima will reflect this bias. When the order is *good* then we are not only improving efficiency but also we are going to obtain a better model than without using any order. Because in the latter case the search process may get stuck in some *bad* local maxima.

If the order is *wrong* the bias will act negatively in addition to the already existing problem of finding a bad local maxima, if a greedy search strategy is used (e.g. hill-climbing).

In other words, a greedy search for the network, as outlined above, quickly becomes infeasible if the number of variables in the database grows. Constraining the search space using an order on the variables allows for higher dimensional databases. However, for a moderate sized database with, say, 50 variables, defining such an order is a non-trivial task. Moreover, mistakes in the order may yield rather bad networks. Therefore, we introduce in the next section a methodology to explore the search space in a way so that it is feasible to recover moderate sized networks without the need of a pre-defined order on the variables.

3. Smart Hill-Climbing

3.1 Hierarchical Clustering of Variables

Cluster analysis comprises a broad set of techniques that allow to find groups (clusters) of data units or variables such that, the entities within one group are more similar to each other than to those in other groups.

Hierarchical clustering organizes these groups as a tree where the root node contains the whole set of entities and the leaves contain one single entity each. The structure of the tree from the leaves to the root node aggregates at each level entities into clusters. Those procedures that generate the tree in this direction are known as *agglomerative*. For more details about cluster analysis in general, and this section in particular, the reader may consult [1].

To speed-up the discovery of Bayesian Networks, we are interested in groups of variables that are likely to interact (strongly) with each other. Such a kind of cluster analysis is, thus, based on a pairwise measure of association between variables. The measure used is discussed in detail below. Using this measure, we compute a *lower triangular similarity matrix* (or simply *similarity matrix*) over the variables. Using this similarity matrix, we cluster the variables agglomeratively using *complete linkage* [1]. Complete linkage ensures that all pairs of members in a cluster satisfy a minimum degree of similarity. This procedure is also discussed in detail below.

Cramér's V for similarity The similarity measure for variables we use should capture the existence of the sort of interaction we try to discover using the Bayesian Network, viz., marginal and conditional (in)dependencies. Independence of categorical variables is traditionally tested using a χ^2 statistic. It is well-known that when enough data is available, χ^2 statistics are rather accurate in characterizing the significance of a relationship.

Since we need to compare the relative *strengths* of the relationships between the variables in the clustering process, the χ^2 values themselves are not that useful given their dependence on the degrees of freedom. Therefore, we use the transformation of the χ^2 value introduced by Cramér [5]. This is a normalization of the χ^2 value to the interval $[0, 1]$, in which 0 means independence and 1 means a perfect association.

Let N be the sample size and i, j the cardinalities of the variables, then Cramér's V is given by:

$$\sqrt{\frac{\chi^2}{N \min(i-1, j-1)}}$$

As an aside, note that a known weak point of transformations of χ^2 statistics is that they lack an operational interpretation ([10],p. 740). Such as an interpretation in terms of power of prediction, or in terms of entropy. However, since we are interested in marginal and conditional (in)dependencies this poses no problem for us.

The complete linkage agglomerative procedure Using Cramér's V, we compute the *similarity matrix* of the variables in the database. Next we cluster the variables hierarchically using complete linkage [1] as illustrated in figure 1.

1. Begin with n clusters each consisting of exactly one variable. Let the clusters be labeled with the numbers 1 through n.
2. Search the similarity matrix for the most similar pair of clusters. Let the chosen clusters be labeled p and q and let their associated similarity be $s_{pq}, p > q$.
3. Reduce the number of clusters by 1 through merger of clusters p and q. Let the new cluster t be placed in row and column of cluster q.
4. Update the similarity matrix entries of former column q (now t) in order to reflect the revised similarities between the new cluster t and all other existing clusters in the following way:

$$s_{tr} = min(s_{pr}, s_{qr}) \ \ \forall r$$

5. Delete row and column of the similarity matrix pertaining to cluster p.
6. Perform steps 2 to 5 a total of $n-1$ times (at which point all variables will be in one cluster). At each stage, record the identity of the clusters which are merged and the value of similarity between them in order to retrieve the hierarchy.

Figure 1: Complete linkage agglomerative algorithm

Anderberg [1] proves that this procedure has an algorithmic cost of $\mathcal{O}(2n^2 - 9n/2)$. To build the similarity matrix we need to compute the measure of association over $n(n-1)/2$ pairs of variables. In other words, the whole clustering procedure is $\mathcal{O}(n^2)$ in the number of variables. Since this is a pre-processing phase of the Bayesian Network discovery algorithm and is, thus, only performed once, this puts no effective limit on the number of variables.

3.2 Building an incremental strategy from the hierarchy

Each search strategy has a unique way of searching through a vast, and often entangled, search space. Sophisticated search procedures, like simulated annealing, may perform very well, but require fine-tuning of their parameters. Hill-climbing is a very simple search strategy that has no parameters. Winston [19] describes improvements such as beam-search, best-first search and branch-and-bound. Moreover, he lists characteristics of the search space to which the hill-climber is sensitive. These are

foothills (large amounts of local maxima), *plateaus* (a practically flat search space with few and narrow peaks), and *ridges* (narrow paths to the top).

Empirically it is easy to see that search spaces for Bayesian Networks often have such characteristics. Moreover, if the search space is constrained, such as with an order on the variables, the problem of ridges may become aggravated. Our incremental algorithm is intended to avoid parts of the search space that contain such problems as far as possible.

In the previous subsection we have discussed how the variables are clustered. Now we have to extract the necessary clusters from the hierarchical tree of clusters; we have to *flatten* the tree. Deciding a priori which clustering yields the best results is a non-trivial task. Our experiments have shown us that in general many small clusters give good results. Therefore, the user may specify a cluster-size (cs) and the algorithm in figure 2 extracts clusters of more or less that size; they may contain one variable more, or one less.

```
algorithm incr_strategy(tree t, node top_root, int cs, list small) return list
    node r, s, w
    list inc_strat, children, sub_inc_strat
    r := root_node(t)
    children := t.children(r)
    children.gotop()
    while ¬children.end() do
        s := children.next()
        if |s| ≥ cs − 1 and |s| ≤ cs + 1 then inc_strat.add(s)
        else if |s| < cs − 1 then small.add(s)
        else
            t_s := t.sub_tree(s)
            sub_inc_strat := incr_strategy(t_s, top_root, cs, small)
            inc_strat.concatenate(sub_inc_strat)
    if r = top_root then
        small.gotop()
        while ¬small.end() do
            s := small.next()
            w.add_content(s)
            if |w| ≥ cs − 1 then
                inc_strat.add(w)
                w.empty_content()
        if ¬w.isempty() then inc_strat.add(w)
    return inc_strat
```

Figure 2: Algorithm to flatten a hierarchy of variables.

The algorithm explores recursively a tree t of a hierarchical clustering of variables in depth-first. At the moment it finds a node with a number of variables of, plus/minus one cluster size (cs), it does not go further in that branch. It stores that set of variables in an ordered list (inc_strat), and continues the exploration.

If a node contains less variables than the given cluster size minus one, then the algorithm stores these variables in a temporal ordered[2] list ($small$) This list containing the small sets of variables is used globally by the successive calls of the algorithm by passing it by reference. When the top call of the algorithm has explored all its branches, it creates sets of the proper size out of the $small$ list, by joining its elements. Every set created in this way is then added to the inc_strat list, which it

[2]note, this order is simply from left to right in the tree

will contain finally the whole set of variables in clusters of a certain size. The ordered list *inc_strat* returned by the algorithm, indicates which subnetworks must be recovered first, and in which order the incremental algorithm, that discovers the whole Bayesian Network, should operate.

```
algorithm incremental_bayesnet(list vars, int cs) return bayesian_network
    tree t
    list inc_strat, small, nets
    node r, s
    bayesian_network b, w
    t := hierarchical_clustering(vars)
    r := root_node(t)
    inc_strat := incr_strat(t, r, cs, small)
    inc_strat.gotop()
    while ¬inc_strat.end() do
        s := inc_strat.next()
        b := mine_bayesian_network(s, EMPTY_NET)
        nets.add(b)
    inc_strat.gotop()
    s := inc_strat.next()
    nets.gotop()
    w := nets.next()
    while ¬nets.end() do
        s := s ∪ inc_strat.next()
        w := w ∪ nets.next()
        w := mine_bayesian_network(s, w)
    return w
```

Figure 3: Algorithm to discover incrementally a Bayesian Network.

Figure 3 shows the incremental algorithm for Bayesian Networks discovery. In the specification of this algorithm, the function *mine_bayesian_network* is a call to the data mining server to create and run a mining task that recovers a Bayesian Network from the database. The function *mine_bayesian_network* returns the recovered network and has as input parameters the set of variables and the initial Bayesian Network. Note that during the incremental process, the initial network is formed by the current discovered network plus the next small network in *inc_strat*.

4. Experiments and Results

4.1 Experimental Design

The experiments have been carried out in a research prototype from an industrial data mining system [8] which, among other models[3], supports recovering Bayesian Networks from data. The system incorporates the BIC (or Schwarz [17]) model selection criterion, the Cooper and Herskovits bayesian measure [4] and the bayesian dirichlet equivalent (BDeu) [6] where uninformative priors for the dirichlet parameters are used [2]. We have used this latter model selection criterion in the experiments for this paper.

The search engine of the system harbours among others a beam search where we can tune the width of the beam. Thus we can use a hill-climber by setting this parameter to one. The neighbourhood is generated at each step of the search by an operator that creates all feasible[4] networks with one arc more, one arc less and one arc reversed. The operator takes into account whether we provide some

[3] association rules, decision rules and decision trees

[4] without directed cycles

complete/partial order among the set of variables and/or a maximum number of parents per node.

The system has a client-server architecture and the algorithm described in the previous section resides, in this prototype, in the client. Because we have ran the experiments in a shared server and using a shared network, we will report the time of our experiments normalized to the time the system takes to perform the analysis using the standard *order* approach. Our aim in this section is to show that our approach performs similarly[5] to the *order* approach when a perfect order is known; both in time and fit of the discovered models.

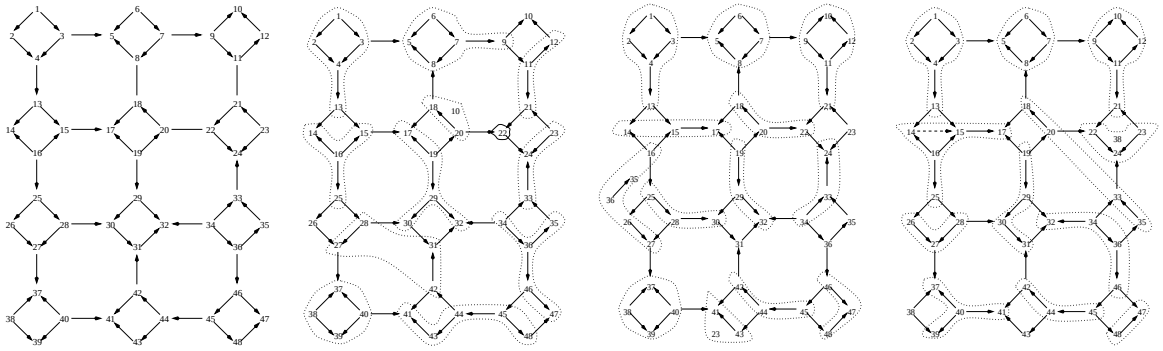
We use synthetic datasets created using a Monte-Carlo method [11]. This method was also used for the experimentation by [4], an important difference between their experiments and ours is, however, as follows. In [4], the authors set up the parameters of the network by hand, whereas we generated them from a dirichlet distribution[6]. The advantage of doing it this way is that our synthetic dataset is closer to a real world dataset. Because it means that the evidence supporting some dependencies may actually be too weak if the sample size is not large enough.

If one introduces sharp probabilities in the model from which one samples, then of course the learning process is going to be quicker because the shape of the search space becomes sharper.

4.2 Exploiting Loose Connections

First we are going to show that a hierarchical clustering of variables, which uses the association measure described in section 3.1.1, is able to capture clusters containing variables which are more likely to interact among each other, than with others.

For this purpose we design a Bayesian Network with a honeycombed structure that will help to see clearly how the groups of interacting variables are observed, see figure 4.a.



(a) (b) (c) (d)

Figure 4: Honeycombed structures: (a) original network, (b) (c) (d) clusters of networks obtained from the three samples

From this honeycombed Bayesian Network we sample three datasets, each of ten thousand records. For each sample the set of probabilities of the network is newly generated. Thus the three datasets reflect different amounts of evidence in the model. By doing this we can check empirically that the approach works as we expected.

Using the algorithm described in section 3.2 to flatten the hierarchy of clusters, with an input cluster size of four, we obtain the clusters of variables. Every cluster is used as the set of variables from which we recover an initial Bayesian Network. Every initial Bayesian Network resembles a connected part of the original network. Figures 4.b, 4.c and 4.d outline the clusters and networks found for each dataset.

[5] it performs better when the order is not known

[6] creating its parametric vector at random

4.3 Using an Incremental Strategy

Once the clusters have been obtained, the system recovers the Bayesian Networks corresponding to each cluster of variables. Since the amount of variables per network is very small we use a beam search strategy with a beam width of 3, to recover the small initial networks. When all these networks are recovered then the incremental strategy starts.

To make our point clear, we count at each step of the search the current number of models in the neighbourhood that improve the model currently selected by the hill-climber. From those models the hill-climber will pick up the best one. The number of models that improve the fit at every moment of the search gives an idea of the shape of the search space.

The time axis is normalized to the time necessary using a perfect order in every different dataset. The time measured comprises the whole process from doing the hierarchical clustering of variables first, until the last step of the incremental strategy is finished. It also includes the overhead of the client-server connection through the network and the time the system uses to set up and manage the mining tasks.

The first step in the incremental strategy is to recover the set of small networks. Once these networks are available, the incremental process uses them one by one to build the final network. The hierarchical clustering of variables took, on average over the three samples, around 4% of the time. The recovery of the small networks took less than 1% of the time[7].

Figure 5 shows the first four steps of the incremental strategy for the first sample of the honeycombed Bayesian Network. This picture illustrates how the network is discovered incrementally.

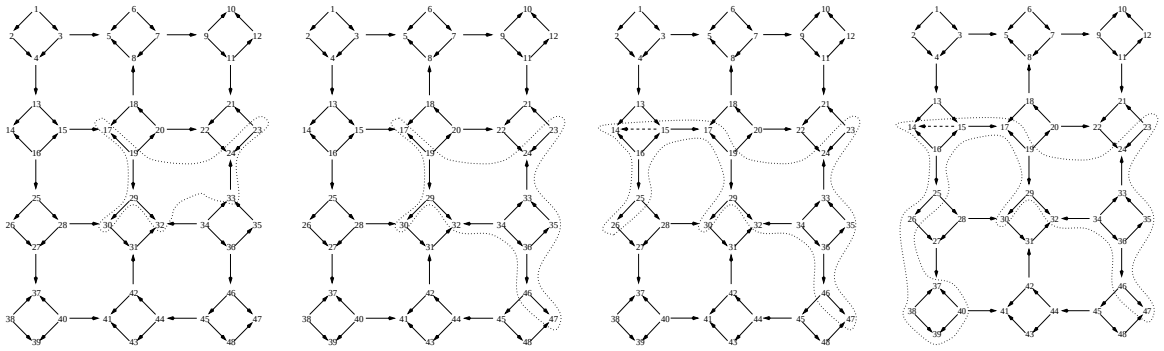


Figure 5: First four steps, from left to right, of the incremental strategy for the first sample of the honeycombed Bayesian Network. A dot line encloses the part of the network that at each step is being discovered.

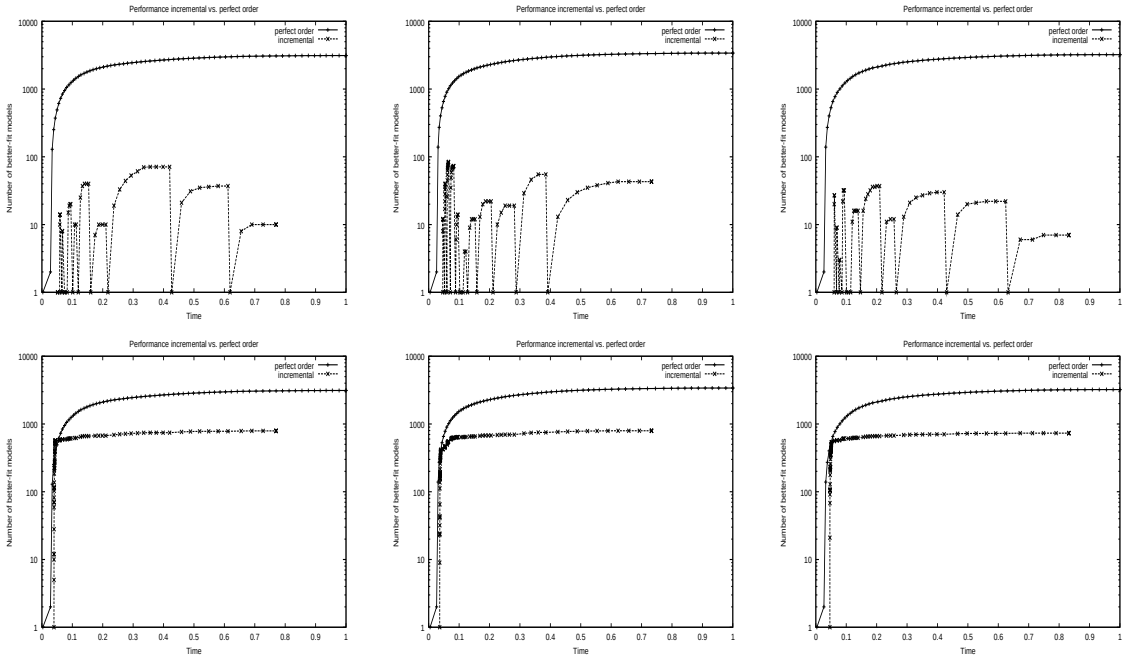
Figure 6, shows how the performance evolves at every step of the incremental strategy, by plotting in the amount of *better-fit* models w.r.t. time. In this picture there are two plots per sample. The upper one compares the perfect order performance[8] with the performance of the successive mining tasks where the Bayesian Network is recover incrementally. The incremental approach starts considering an analysis of much lower dimension than the original one, therefore less models improve the fit at each step, and the process takes short time. At the following step the dimension of the analysis is extended, but the initial model has already some reasonable fit, so that the amount of models that improve the fit is not as large as we started from an initial empty model.

In the lower plots, the accumulated number of models that improve the current fit are given for our incremental algorithm. It is immediate that the incremental algorithm searches through a smaller set of feasible models.

A fundamental issue we should discuss now is, how good is the fit of the models discovered using

[7] further these networks could be recovered in parallel

[8] which also uses a maximum number of parent nodes set to three



(sample 1) (sample 2) (sample 3)

Figure 6: Comparison between using a perfect order among variables, and using an incremental strategy without any order

Sample	Mis.	Ext.	Rev.	BDeu ord.	BDeu inc.	BDeu inc.(bs3)
1	1	3	4	-249864.9211	-249872.6124	-249866.6809
2	2	4	4	-256545.6734	-256566.0208	-256547.7731
3	4	3	5	-249231.1803	-249300.3495	-249231.2086

Table 1: Fit of the Bayesian Networks discovered using the incremental approach for the three different samples.

the incremental strategy. In general, for any synthetic dataset, if the sample size is large enough, one recovers the original[9] network, provide we use a perfect order among the variables. For the honeycombed structure from which we sampled, the sample size we used does not seem to be large enough. Using a perfect order, we recovered, in all three different samples, the original network with the exception of a few missing arcs. The evidence supporting those connections is not large enough, thus they are not discovered with the incremental strategy either.

On the table 1 we may see how good is the fit of the models discovered using the incremental approach. The table contains the structural difference on the second, third and fourth column, and the log-likelihood (BDeu measure [6]) of the models recovered using a perfect order and the incremental approach, in the sixth and seventh column respectively.

First note that the *missing* column is true for both the perfect order search and our incremental search. In other words, there was not enough evidence in the database for these arcs.

Secondly, real structural differences are given by the *extra* and *reversed*[10] columns. The important

[9] also known as gold standard

[10] arcs that were originally compelled and appear reversed in the discovered model

remark is that all reversed arcs are actually *covered*[11] [6] in the discovered model because of the effect of some missing or extra arc. This means that I-mapness holds given the evidence in the sample.

In other words, the differences between the true model and the recovered model are only in the extra arcs. Experiments show that these are caused by the greedy search. Using a beam search of width three, better networks were discovered, see table 1 (column 7). For instance, in the second sample the structural differences reduced from (4,4) to (2,1). The increase of time with respect to the incremental hill-climbing is a factor of 3.74 on average over the three samples.

5. Conclusions and Discussion

Bayesian Networks are a valuable tool in exploring the interactions among the variables in a database. Most discovery algorithms, however, suffer from the problem of high dimensionality. That is, with a growing number of variables the discovery of a network quickly becomes infeasible. Most solutions towards this problem require extensive a priori knowledge.

The approach we presented in this paper asks the data rather than the expert. More precisely, we cluster the variables using Cramér's V as a similarity measure and then we discover the network incrementally. With experiments we show that the resulting models are comparable with those recovered using a prior information if that is available. The big advantage is, of course, that we need not this a priori information. As an aside, note that our algorithm works in a rather general setting and is, thus, perhaps also applicable for other types of graphical models.

Improvements of the current algorithm can probably be found using less greedy search techniques and by using more sophisticated search operators, [3]. Experiments in this vein are underway. Combining our approach with a, partial, order provided by the expert is also a possibility. Preliminary experiments show that the networks are retrieved faster, but fit not as good as those retrieved using only our approach.

The size of the clusters to be used is the only parameter in our algorithm. We cannot provide a general policy on what the best cluster size is. In our experiments, we have seen that the size should be between four and eight. However, this may very well depend on the data. In other words, if it all possible, the domain-expert should pick out the good clusters!

[11] the parent set of the sink node is formed by the source node and its parent set

References

1. Michael R. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1973.
2. W.L. Buntine. Theory refinement on bayesian networks. In P. Smets B.D. D'Ambrosio and P.P. Bonissone, editors, *Proceedings of Uncertainty in Artificial Intelligence*, volume 7, pages 52–60, Los Angeles, 1991. Morgan Kaufmann.
3. D.M. Chickering. Learning equivalence classes of bayesian network structures. In *Proceedings of the 12th. Conference on Uncertainty in Artificial Intelligence*, pages 150–157. Morgan Kaufmann, 1996.
4. Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.
5. Harald Cramér. *Mathematical Methods of Statistics.* Princeton University Press, Princeton, 1946.
6. D. Geiger D. Heckerman and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
7. Azaria Paz Dan Geiger and Judea Pearl. Learning simple causal structures. *International Journal of Intelligent Systems*, 8:231–247, 1993.
8. Data Distilleries. *http://www.ddi.nl.*
9. J.M. Gutiérrez E. Castillo and A.S. Hadi. *Expert Systems and Probabilistic Network Models.* Springer, New York, 1997.
10. Leo A. Goodman and William H. Kruskal. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49:733–764, 1954.
11. Max Henrion. Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *Proceedings of the 2nd. Conference on Uncertainty in Artificial Intelligence*, pages 149–163. Elsevier Science, 1988.
12. Steffen L. Lauritzen. *Graphical Models.* Oxford Science, 1996.
13. David Madigan and Adrian E. Raftery. Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546, 1994.
14. J. Pearl. *Probabilistic Reasoning in intelligent systems.* Morgan Kaufmann, 1988.
15. Gregory M. Provan and Moninder Singh. Learning bayesian networks using feature selection. In

Fisher D. and Lenz H.-J., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 291–300, New York, 1996. Springer Verlag.

16. R.W. Robinson. Counting labeled acyclic digraphs. In Frank Harary, editor, *New Directions in the Theory of Graphs*, pages 239–273. Academic Press, New York, 1973.
17. Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.
18. Moninder Singh and Marco Valtorta. An algorithm for the construction of bayesian network structures from data. In *Proceedings of the 9th. Conference on Uncertainty in Artificial Intelligence*, pages 259–265. Morgan Kaufmann, 1993.
19. Patrick Henry Winston. *Artificial Intelligence*. Addison-Wesley, 1977.