Efficient Learning of Hierarchical Latent Class Models

Nevin L. Zhang
Department of Computer Science.
Hong Kong University of Science & Technology
Hong Kong, China

Tomáš Kočka
Laboratory of Intelligent Systems Prague
Prague University of Economics
Prague, Czech Republic

Abstract

Hierarchical latent class (HLC) models are tree-structured Bayesian networks where leaf nodes are observed while internal nodes are hidden. In earlier work, we have demonstrated in principle the possibility of reconstructing HLC models from data. In this paper, we address the scalability issue and develop a search-based algorithm that can efficiently learn high-quality HLC models for realistic domains. There are three technical contributions: (1) the identification of a set of search operators; (2) the use of improvement in BIC score per unit of increase in model complexity, rather than BIC score itself, for model selection; and (3) the adaptation of structural EM for situations where candidate models contain different variables than the current model. The algorithm was tested on the COIL Challenge 2000 data set and an interesting model was found.

1 Introduction

Hierarchical latent class (HLC) models [8] are tree-structured Bayesian networks (BNs) where leaf nodes are observed while internal nodes are hidden. They generalize latent class (LC) models [3] and were first identified as a potentially useful class of Bayesian networks by Pearl [5]. This paper is concerned with the problem of learning HLC models from data. The problem is interesting for three reasons. First, HLC models represent complex dependencies among observed variables and yet are computationally simple to work with. Second, the endeavor of learning HLC models can reveal latent causal structures. Researchers have already been inferring latent causal structures from observed data. One example is the reconstruction of phylogenetic trees, which can be viewed as special HLC models. Third, HLC models alleviate disadvantages of LC models as models for cluster analysis.

When learning BNs with latent variables, one needs to determine not only model structures, but also cardinalities of latent variables, i.e. the numbers of values they can take. Although not using the terminology of HLC models, Connolly [1] proposed the first, somewhat *ad hoc*, algorithm for learning HLC models and tested it on one small toy example. A more principled algorithm was proposed by Zhang [8]. This algorithm hill-climbs in the space of HLC models guided by a scoring function. It starts with an LC model. At each step of search, it first generates a number of candidate structures by modifying the structure of the current model. It then optimizes cardinalities of latent variables, resulting in candidate models. Finally, it evaluates the candidate models and picks the best one to seed the next step of search. Search terminates when the best candidate model is no better than the current model. To optimize the cardinalities of the latent variables in a model structure, the algorithm employs another hill-climbing routine. Hence we call it the *double hill-climbing (DHC) algorithm*.

Zhang [8] has empirical shown that the DHC algorithm performs well in terms of model quality when coupled with the BIC score [6]. However, it has a serious drawback, namely its high complexity. Let n be the number of observed variables. At each step of search, DHC generates $O(n^2)$ models structures. To optimize the cardinalities of the latent variables in a given model structure, $O(n^2k)$ models are examined, where k is the maximum cardinality for a latent variable. Hence there are totally $O(n^4k)$ models to evaluate. Before a model can be evaluated, its parameters must be optimized. Due to the presence of latent variables, parameter optimization requires the EM algorithm, which is known to be computationally expensive.

In this work, we reduce the complexity of DHC in two steps. The first step is to reduce the number of models examined at each step of search from $O(n^4k)$ to $O(n^2)$. This is achieved via a new algorithm called *single hill-climbing (SHC)*. Two technical aspects of SHC are particularly interesting: 1) The set of search operators that it employs is the result of our efforts in finding an appropriate tradeoff between algorithmic complexity and model quality. 2) It selects among model based on, instead of BIC score itself, improvement in BIC score per unit increase in model complexity. As we will see, BIC score does not work for SHC.

Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)
1082-3409/04 $20.00 © 2004 **IEEE**

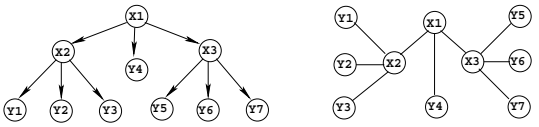

Figure 1. An example HLC model and the corresponding unrooted HLC model. The X_i's are latent variables and the Y_j's are manifest variables.

The second step to reduce the complexity of DHC is to apply the idea of structural EM [2] to SHC so that exact EM is required only once for each step of search. The result is an algorithm called *heuristic SHC (HSHC)*. Structural EM completes data using the current model and uses the completed data set to evaluate candidate models. Candidate models generated by SHC contain latent variables that are non-existent in the current model or are different from the corresponding variables in the current model. Therefore, the application of Structural EM to SHC is not straightforward. We have found approximate ways to evaluate the candidate models that turn out to be equivalent to performing hypothesis tests for conditional independence and for goodness-of-fit of LC models.

We have conducted empirical studies to determine whether HSHC can find good models for real-world applications in a timely fashion. In particular, HSHC was used to analyze the COIL Challenge 2000 [7] data. The data set consists of 86 variables and 5822 records. After preprocessing, we were left with 42 mostly binary variables. HSHC took 121 hours to finish the analysis and it produced an interesting model.

2 HLC models

A *hierarchical latent class (HLC) model* is a Bayesian network where (1) the network structure is a rooted tree and (2) the variables at the leaf nodes are observed and all the other variables are not. An example HLC model is shown in Figure 1 (left diagram). In this paper, we use the terms "node" and "variable" interchangeably. The observed variables are referred to as *manifest variables* and all the other variables as *latent variables*. A *latent class (LC) model* is an HLC model where there is only one latent node. We usually write an HLC model as a pair $M = (m, \theta)$, where θ is the collection of parameters. The first component m consists of the model structure and cardinalities of the variables. We will sometimes refer to m also as an HLC model. When it is necessary to distinguish between m and the pair (m, θ), we call m an *uninstantiated HLC model* and the pair (m, θ) an *instantiated HLC model*.

Two instantiated HLC models $M{=}(m, \theta)$ and $M'{=}(m', \theta')$ are *marginally equivalent* if they share the same manifest variables $Y_1, Y_2, \ldots, Y_n$ and

$$P(Y_1, \ldots, Y_n|m, \theta) = P(Y_1, \ldots, Y_n|m', \theta'). \quad (1)$$

An uninstantiated HLC models m *includes* another m' if for any parameterization θ' of m', there exists parameterization θ of m such that (m, θ) and (m', θ') are marginally equivalent, i.e. if m can represent any distributions over the manifest variables that m' can. If m includes m' and vice versa, we say that m and m' are *marginally equivalent*. Marginally equivalent (instantiated or uninstantiated) models are *equivalent* if they have the same number of independent parameters. One cannot distinguish between equivalent models using penalized likelihood scores.

Let X_1 be the root of an HLC model m. Suppose X_2 is a child of X_1 and it is a latent node. Define another HLC model m' by reversing the arrow $X_1 \rightarrow X_2$. In m', X_2 is the root. The operation is hence called *root walking*; the root has walked from X_1 to X_2. Root walking leads to equivalent models [8]. This implies that it is impossible to determine edge orientation from data. We can learn only *unrooted HLC models*, which are HLC models with all directions on the edges dropped. Figure 1 also shows an example unrooted HLC model. An unrooted HLC model represents a class of HLC models. Members of the class are obtained by rooting the model at various nodes. Semantically it is a Markov random field on an undirected tree. The leaf nodes are observed while the interior nodes are latent. The concepts of marginal equivalence and equivalence can be defined for unrooted HLC models in the same way as for rooted models. From now on when we speak of HLC models we always mean unrooted HLC models unless it is explicitly stated otherwise.

Let $|X|$ stand for the cardinality of a variable X. For a latent variable Z in an HLC model, enumerate its neighbors as $X_1, X_2, \ldots, X_k$. An HLC model is *regular* if for any latent variable Z, $|Z| \leq \prod_{i=1}^{k} |X_i| / \max_{i=1}^{k} |X_i|$, and when Z has only two neighbors, strict inequality holds and one of the neighbors must be a latent node. Note that this definition applies to both instantiated and uninstantiated models.

Given an irregular instantiated model m, there exists a regular model that is marginally equivalent to m and has fewer independent parameters [8]. The process of obtaining the regular model is called *regularization*. It is evident that if penalized likelihood is used for model selection, the regularized model is always preferred over m itself.

3 The SHC algorithm

Assume that there is a collection $\mathbf{D}$ of i.i.d samples on a number of manifest variables generated by an unknown regular HLC model. SHC aims at reconstructing the regular unrooted HLC models that corresponds to the generative model. It does so by hill-climbing in the space of all

unrooted regular HLC models for the given manifest variables. For this paper, we assume that the BIC score is used to guide the search. The BIC score of a model m is:

$$BIC(m|\mathbf{D}) = logP(\mathbf{D}|m, \theta^*) - \frac{d(m)}{2} logN$$

where θ^* is the ML estimate of model parameters, $d(m)$ is the dimension of m, i.e. the number of independent parameters, and N is the sample size.

The overall strategy of SHC is similar to that of greedy equivalence search (GES), an algorithm for learning Bayesian network structures in the case when all variables are observed [4]. It begins with the simplest HLC model, i.e. the LC model, and works in two phases. In Phase I, SHC *expands* models by introducing new latent nodes and additional states for existing nodes. The aim is to improve the likelihood term of the BIC score. In Phase II, SHC *retracts* models by deleting latent nodes or states of latent nodes. The aim is to reduce the penalty term of the BIC score, while keeping the likelihood term more or less the same. If model quality is improved in Phase II, SHC goes back to Phase I and the process repeats itself.

Search operators: SHC hill-climbs using five search operators, namely State Introduction, Node Introduction, Node Relocation, State Deletion, and Node Deletion. The first three operators are used in Phase I and the rest are used in Phase II.

Given an HLC model and a latent variable in the model, *State Introduction (SI)* creates a new model by adding a state to the state space of the variable. Clearly, the new model includes the old one.

Node Introduction (NI) involves one latent node X and two of its neighbors. It creates a new model by introducing a new latent node X' to mediate X and the two neighbors. The new node has the same number of states as X. Consider the HLC model m_1 in Figure 2. Applying NI to the latent node X and its neighbors Y_1 and Y_2 results in the model m_2. The new node X_1 has the same state space as X. For the sake for computational efficiency, we do not consider introducing a new node to mediate X and more than two of its neighbors. Without this restriction, the number of candidate models the operator could produce is exponential in the number of neighbors. Also note that NI is disallowed when X has only two neighbors. In this case, it would create a latent node that is also a leaf node. In HLC models, only manifest nodes can be leaves.

Let m' be a model obtained from another model m via NI. Then m' includes m. Note that if we set the cardinality of the new node X' to a number smaller than $|X|$, then m' would no longer include m. On the other hand, if we set $|X'|$ to a number larger than $|X|$, then m' would be more complex than necessary.

The two operators discussed so far introduce new ingredients to a model. *Node Relocation (NR)*, the next operator,

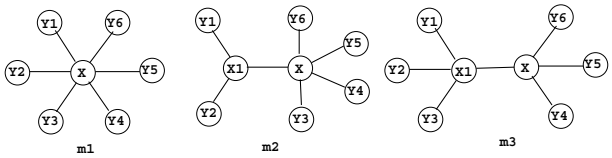

Figure 2. The NI and NR operators.

re-arranges connections among existing nodes. It involves two neighboring latent nodes X_1 and X_2 and a neighbor Z of X_1 that is different from X_2. It creates a new model by relocating Z to X_2, i.e. removing the link between Z and X_1 and adding a link between Z and X_2. Consider the HLC model m_2 in Figure 2. Relocating Y_3 from X to X_1 results in model m_3. Note that a node is allowed to be relocated only "one step away". This is for the sake of computational efficiency and, judging from our experience, more flexibility does not seem necessary. Also note that if the latent node X_1 has only two neighbors, relocating Z to X_2 would make the latent node X_1 a leaf node. In this case, we simply remove X_1.

State Deletion (SD) is the opposite of SI. Given an HLC model and a latent node, it creates a new model by deleting a state of the latent node. It is not applicable if the latent node has only two states. *Node Deletion (ND)* is the opposite of NI. It involves two neighboring latent node X and X' in an HLC model. It creates a new model by deleting X and making all neighbors of X other than X' neighbors of X'. If model m' is obtained by applying State or Node Deletion to model m, then m includes m'.

All of the five operators might lead to the violation of the regularity constraints. We therefore follow each operator immediately with a regularization step.

Model selection: Given a data set, our task is to find a model that fits the data well and has low complexity. It is possible to achieve perfect fit to data using an LC model where the latent node has a high cardinality. That is to use the model with the lowest structure complexity and high variable complexity. Here *variable complexity* refers to the number of values latent variables can have, while *structure complexity* refers to the number of nodes and links among them. Clearly, we need to find a balance between variable complexity and structure complexity so that the overall model complexity is low.

The SI operator increases variable complexity, while the NI operator increases structure complexity. To find an appropriate tradeoff between the two aspects of model complexity, SHC starts with the model that has the lowest variable complexity and the lowest structure complexity, i.e. the LC model where the latent node has only two states. At each step in Phase I, it generates candidate models by applying the SI, NI, and NR operators on the current model and selects one of the candidate models to seed the next step of

step. The key question is which candidate model to select.

A naive answer to pick the one with the highest BIC score. This strategy does not work. We have observed in experiments that SHC often does not leave LC models when models are selected based on BIC score itself: It kept increasing the cardinality of the only latent node, but did not introduce new latent nodes.

To understand the phenomenon, consider the first step, where the current model is the LC model with a binary latent node. Suppose there are n manifest variables. Then, SI would introduce $n+1$ additional parameters, while NI would introduce only 2 additional parameters. More parameters means better fit to data, which in turn implies that the loglikelihood term of the BIC score is higher. At this early stage of search, the improvement in the loglikelihood term is more drastic than the increase in the penalty term. Consequently, the (unique) model generated by SI is likely to have a higher score than models produced by NI and is likely to be chosen, resulting in another LC model where the latent node has three states. In one of our experiments, the initial LC model has a BIC score of -128,684. The model generated by NI from the initial LC model has a BIC score of -125,185, while the best model generated by NI has a BIC score of only -125,569. Repeating the arguments, we see that SI is likely to be applied again in the next step, and again in the step after, and so on.

As an analogy, we compare candidate models to investment alternatives and their BIC scores to earnings (dividends minus transaction costs) of investment alternatives. When deciding one particular investment, if one aims at maximizing the earning, then one would prefer alternatives that involve large sums of fund to those that involves small sums of fund. Compared with small investments, large investments usually have higher earnings despite higher transaction costs. Similarly, if we aim at maximizing the BIC score, then we would prefer more complex candidate models than less complex one. At the beginning of SHC, the likelihood term of the BIC score increases faster with model complexity than the penalty term. Consequently, more complex candidate models usually have higher BIC scores with less complex ones.

Maximizing the earnings of each investment transaction does not bring about maximum total earnings for the finite fund that one has. A better strategy is to maximize earnings ratio, i.e. earning over the amount of investment. We apply this idea to model selection. Let m be the current model and m' be a candidate model. Define the *unit improvement of m' over m given* $\mathbf{D}$ to be

$$U(m', m|\mathbf{D}) = \frac{BIC(m'|\mathbf{D}) - BIC(m|\mathbf{D})}{d(m') - d(m)}. \quad (2)$$

It is the increase in model score per unit increase in model complexity. SHC selects models using the following *cost-effectiveness* principle: *Among all candidate models, choose the one that has the highest unit improvement over the current model.*

The NR operator does not necessarily increase the number of model parameters. Care must be taken when comparing models it produces with models generated by other operators. At each step, SHC first considers candidate models produced by the NR operator. Among them, those models whose BIC scores are not higher than that of the current model are discarded. If there are, among the remaining, models that have fewer parameters than the current model, the one with the highest BIC score is chosen and search moves to the next step immediately without considering other operators. If all the remaining models have more parameters than the current model, on the other hand, then they are compared with other candidate models using the cost-effectiveness principle.

Model selection in Phase II is straightforward and is based on model score.

Pseudo code: We now give the pseudo code for the SHC algorithm. The input to the algorithm is a data set $\mathbf{D}$ on a list of manifest variables. Records in $\mathbf{D}$ do not necessarily contain values for all the manifest variables. The output is an unrooted HLC model. Model parameters are optimized using the EM algorithm. Given a model m, the collections of candidate models the search operators produce will be respectively denoted by $NI(m)$, $SI(m)$, $NR(m)$, $ND(m)$, and $SD(m)$.

```
SHC(D)
Let m be the LC model with a binary latent node.
Repeat until termination:
  m'=PhaseI(m, D).
  If BIC(m'|D)≤BIC(m|D), return m. Else m=m'.
  m'=PhaseII(m, D).
  If BIC(m'|D)≤BIC(m|D), return m. Else m=m'.
PhaseI(m, D)
Repeat until termination:
  Remove from NR(m) all models m'
    s.t. BIC(m'|D)≤BIC(m|D),
  If there is m'∈NR(m) s.t. d(m')≤d(m)
    m=m' and continue.
  Find m' in NR(m)∪NI(m)∪SI(m)
    that maximizes U(m', m|D).
  If BIC(m'|D)≤BIC(m|D), return m. Else m=m'.
PhaseII(M, D)
Repeat until termination:
  Find the model m' in ND(m)∪SD(m)
    that maximizes BIC(m'|D).
  If BIC(m'|D)≤BIC(m|D), return m. Else m=m'.
```

4 The heuristic SHC algorithm

At each step of search, SHC generates a set of candidate models, evaluates each of them, and selects the best one.

Before a model can be evaluated, its parameters must be optimized. Due to the presence of latent variables, parameters are optimized using the EM algorithm. EM is known to be computationally expensive. SHC runs EM on each candidate model and is hence still inefficient.

The same problem confronts hill-climbing algorithms for learning general Bayesian networks from data with missing values. There, structural EM was proposed to reduce the number of calls to EM [2]. The idea is to complete the data, or fill in the missing values, using the current model and then evaluate the candidate models based on the completed data. Parameter optimization based on the completed data does not require EM at all. EM is called only once at the end of each iteration to optimize the parameters of the best candidate model.

In this section, we apply the idea of structural EM to SHC. The main technical issue that we need to address is that the variables in the candidate models can differ slightly different from those in the current model. Hence there might be a slight mismatch between the completed data and the candidate models. In a candidate model generated by the NI operator, for instance, there is one new variable that does not appear in the current model. The completed data contain no values for the new variable.

How should we evaluate a candidate model based on a slightly mismatched data set? The answer depends how the candidate model is generated, i.e. by which operator. We divide all the candidate models into several groups, with one group for each operator. Models in a group are compared with each other based on the completed data and the best one is selected. Thereafter a second model selection process is invoked to choose one from the best models of the groups. This second process is the same as the model selection process in SHC, except that there is only one candidate model for each operator. In this phase, parameters of models are optimized using EM.

4.1 Model selection in Phase I

In the next 3 subsections, we discuss how to select among candidate models generated by each of the search operators used in Phase I. Here are some notations that we will use. We use m to denote the current model and θ^* to denote the ML estimate of the parameters of m based on the data set $\mathbf{D}$. The estimate was computed using EM at the end of the previous step. Let P_m be the joint probability represented by the instantiated model (m, θ^*).

Completing the data set $\mathbf{D}$ using the model (m, θ^*), we get a data set that contain values for all variables in m. Denote the completed data set by $\mathbf{D}_m$. Let $\mathbf{V}$ the set of variables in m. $\mathbf{D}_m$ induces an empirical distribution over $\mathbf{V}$, which we denote by $\hat{P}(\mathbf{V})$. In the following, we will need to refer to the quantities of $\hat{P}(\mathbf{X})$ and $\hat{P}(\mathbf{X}|\mathbf{Y})$, where $\mathbf{X}$ and $\mathbf{Y}$ are subsets of variables. Such quantities are computed from the model (m, θ^*) and the original data set $\mathbf{D}$. They are not obtained from the completed data set $\mathbf{D}_m$. In fact, we never explicitly compute the completed data set. It is introduced only for conceptual clarity.

4.2 Selecting among models generated by NI

Consider a candidate model m' in $NI(m)$. Suppose it is obtained from m by introducing a latent variable H to mediate the interactions between a node A and two of its neighbors B and C. Define

$$U_{NI}(m', m|\mathbf{D}_m) = \frac{N \sum_{A,B,C} \hat{P}(A,B,C) log \frac{\hat{P}(B,C|A)}{\hat{P}(B|A)\hat{P}(C|A)}}{d(m') - d(m)}. \quad (3)$$

This is the criterion that we use to select among models in $NI(m)$; We select the one that maximizes the quantity.

The criterion is intuitively appealing. Consider the term on the numerator of the expression on the right hand side of (3). Except for a constant factor of 2, it is the G-squared statistic, based on $\mathbf{D}_m$, for testing the hypothesis that B and C are conditionally independent given A. The larger the term, the further away B and C are from being independent given A, and the more improvement in model quality the NI operation would bring about. So selecting the candidate model m' in $NI(m)$ that maximizes $U_{NI}(m', m|\mathbf{D}_m)$ amounts to choosing the way to apply NI operator that would result in the largest increase in model quality per complexity unit.

The U_{NI} criterion is also a natural approximation of the cost-effectiveness principle.We explain why in the rest of this subsection.

The cost-effectiveness principle states that we should choose the model m' in $NI(m)$ that maximizes:

$$U(m', m|\mathbf{D}_m) = \frac{BIC(m'|\mathbf{D}_m) - BIC(m|\mathbf{D}_m)}{d(m') - d(m)}. \quad (4)$$

For technical convenience, root m and m' at node A. It is well known that $BIC(m|\mathbf{D}_m)$ is given by

$$N \sum_{X \in \mathbf{V}} \sum_{X, pa(X)} \hat{P}(X, pa(X)) log \hat{P}(X|pa(X) - \frac{d(m)}{2} logN,$$

where $pa(X)$ stands for the set of parents of X in m.

The challenge is to compute term $BIC(m'|\mathbf{D}_m)$. Let θ'^* be the ML estimate of the parameters of m' based on $\mathbf{D}_m$. Use $P_{m'}$ to denote the joint probability represented by parameterized model (m', θ'^*). Consider a variable X in m' that is not B , C, or H. The parents of X in m' are the same as those in m. Moreover, the variable and it parents are observed in the data set $\mathbf{D}_m$. Hence we have $P_{m'}(X|pa(X)) = \hat{P}(X|pa(X))$. Consequently, $BIC(m'|\mathbf{D}_m)$ is given by

$$log P(\mathbf{D}_m|m', \theta'^*) - \frac{d(m')}{2} logN$$

$$= N \sum_{X \in \mathbf{V} \setminus \{B,C\}} \sum_{X, pa(X)} \hat{P}(X, pa(X)) log \hat{P}(X|pa(X)) + N \sum_{A,B,C} \hat{P}(A,B,C) log P_{m'}(B,C|A) - \frac{d(m')}{2} log N$$

where $P_{m'}(B,C|A) = \sum_H P_{m'}(H|A) P_{m'}(B|H) P_{m'}(C|H)$. One can estimate the conditional probability distributions $P_{m'}(H|A)$, $P_{m'}(B|H)$, and $P_{m'}(C|H)$ from $\mathbf{D}_m$. But this requires running EM because $\mathbf{D}_m$ does not contain values for H.

Not wanting to run EM, we seek an approximation for the second term on the right hand side of the above equation. We choose to approximate it with the maximum value possible, i.e. $N \sum_{A,B,C} \hat{P}(A,B,C) log \hat{P}(B,C|A)$. This leads to the following approximation of $U(m', m|\mathbf{D}_m)$

$$\frac{N \sum_{A,B,C} \hat{P}(A,B,C) log \frac{\hat{P}(B,C|A)}{\hat{P}(B|A)\hat{P}(C|A)} - \frac{d(m')-d(m)}{2} log N}{d(m') - d(m)}.$$

Simplifying this expression and deleting terms that do not depend on m', we get the term on the right hand side of (3).

4.3 Selecting among models generated by SI

Consider a candidate model m' in $SI(m)$. Suppose it is obtained from the current model m by introducing a new state to a variable A. Use A' to denote this modified variable in m'. The completed data set $\mathbf{D}_m$ contains values for A. Since A' differs from A, $\mathbf{D}_m$ cannot be directly used to evaluate m'. We hence remove the values of A from $\mathbf{D}_m$ and denote the resulting new data set by $\mathbf{D}_{m,A}$.

Let B_1, ..., B_k be the neighbors of A (A') in m (m'). Define $U_{SI}(m', m|\mathbf{D}_{m,A})$ by

$$\frac{N \sum_{B_1,...,B_k} \hat{P}(B_1, \ldots, B_k) log \frac{\hat{P}(B_1,...,B_k)}{P_m(B_1,...,B_k)}}{d(m') - d(m)}. \quad (5)$$

This is the heuristic we use to select among models generated by the SI operator.

This selection criterion is intuitively appealing. In m, consider the LC model formed by A and its neighbors B_1, ..., B_k. With respect to $\mathbf{D}_{m,A}$, A is latent and the B_i's are observed. Except for a constant factor of 2, the term on the numerator of the expression on the right hand side of (5) is the G-Squared statistic for testing the goodness-of-fit of the LC model. If the term is small, then the B_i's are close to being mutually independent of each other given A. Most of the observed correlations among them are accounted for by the latent variable A. If the term is large, on the other hand, the B_i's are far from being mutually independent given A. There are significant correlations among them that are not accounted for by the variable A. Introducing a new state to A will help to capture those correlations and hence significantly increase model fit.

It is clear that the complexity of computing $U_{SI}(m', m|\mathbf{D}_{m,A})$ is exponential in k. For the sake of efficiency, the criterion is not implemented exactly. Rather, we use the following approximation:

$$\frac{2N \sum_{i,j:i \neq j} \sum_{B_i,B_j} \hat{P}(B_i, B_j) log \frac{\hat{P}(B_i,B_j)}{P_m(B_i,B_j)}}{(k-1)(d(m') - d(m))}. \quad (6)$$

In the rest of this subsection, we explain how $U_{SI}(m', m|\mathbf{D}_{m,A})$ follows from the cost-effectiveness measure

$$U(m', m|\mathbf{D}_{m,A}) = \frac{BIC(m'|\mathbf{D}_{m,A}) - BIC(m|\mathbf{D}_{m,A})}{d(m') - d(m)}. \quad (7)$$

Let $\mathbf{V}_A$ the set of variables in $\mathbf{D}_{m,A}$. It is the same as the set of variables in m' except for A'. Let θ'^*_A be the ML estimate of parameters of m' based on $\mathbf{D}_{m,A}$. Use $P_{m',A}$ to denote the joint probability distributions represented by the parameterized model (m', θ'^*_A). For technical convenience, root m at node A and m' at node A'. Then $BIC(m'|\mathbf{D}_{m,A})$ is given by

$$N \sum_{X \in \mathbf{V}_A, pa(X) \neq A'} \sum_{X, pa(X)} \hat{P}(X, pa(X)) log \hat{P}(X|pa(X)) + N \sum_{B_1,...,B_k} \hat{P}(B_1, \ldots, B_k) log P_{m',A}(B_1, \ldots, B_k) - \frac{d(m')}{2} log N.$$

To obtain $P_{m',A}(B_1, \ldots, B_k)$, one needs to run EM to estimate $P_{m'A}(A)$ and $P_{m',A}(B_i|A)$. Not wanting to run EM, we approximate the second term in the above expression with $N \sum_{B_1,...,B_k} \hat{P}(B_1, \ldots, B_k) log \hat{P}(B_1, \ldots, B_k)$, which is an upper bound of the term.

Let θ^*_A is the ML estimate of parameters of m based on $\mathbf{D}_{m,A}$. Use $P_{m,A}$ to denote the joint probability distribution represented the parameterized model (m, θ^*_A). Then $BIC(m|\mathbf{D}_{m,A})$ is given by

$$N \sum_{X \in \mathbf{V}_A, pa(X) \neq A'} \sum_{X, pa(X)} \hat{P}(X, pa(X)) log \hat{P}(X|pa(X)) + N \sum_{B_1,...,B_k} \hat{P}(B_1, \ldots, B_k) log P_{m,A}(B_1, \ldots, B_k) - \frac{d(m)}{2} log N.$$

To obtain $P_{m,A}(B_1, \ldots, B_k)$, one needs to run EM. Not wanting to run EM, we approximate it using the same joint probability $P_m(B_1, \ldots, B_k)$ in the parameterized model (m, θ).

The above two approximations lead to the following approximation of $BIC(m'|\mathbf{D}_{m,A}) - BIC(m|\mathbf{D}_{m,A})$:

$$N \sum_{B_1,...,B_k} \hat{P}(B_1, \ldots, B_k) log \frac{\hat{P}(B_1,...,B_k)}{P_m(B_1,...,B_k)} - \frac{d(m')-d(m)}{2} log N. \quad (8)$$

Substituting this for $BIC(m'|\mathbf{D}_{m,A}) - BIC(m|\mathbf{D}_{m,A})$ in (7), simplifying the resulting expression, and removing terms that does not depend on m', we obtain the right hand side of (5).

4.4 Selecting among models generated by NR

Consider a candidate model m' in $NR(m)$. Model m' consists of the same variables as m and hence the application of structural EM is straightforward in this case. To be more specific, suppose m' is obtained from m by relocating a neighbor B of a node A to another neighbor C of A. For technical convenience, assume m is rooted at A. Then $BIC(m'|\mathbf{D}_m) - BIC(m|\mathbf{D}_m)$ is given by

$$\sum_{A,B,C} \hat{P}(A,B,C) log \frac{\hat{P}(B|C)}{P_m(B|A)} - \frac{d(m') - d(m)}{2} logN. \quad (9)$$

Among all models in $NR(m)$, we choose the one for which this difference is the largest.

4.5 Model selection in Phase II

Consider a candidate model m' in $ND(m)$. Suppose it is obtained from m by deleting a latent node B. Suppose the neighbors of B in m are $A, C_1, \ldots, C_r$ and suppose the C_i's are made neighbors of A in m'. For technical convenience, assume m is rooted at A. Then $BIC(m'|\mathbf{D}_m) - BIC(m|\mathbf{D}_m)$ is given by

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} \sum_{A,C_i} \hat{P}(A,C_i) log \hat{P}(C_i|A) \\ &- \textstyle\sum_{i=1}^{m} \sum_{B,C_i} \hat{P}(B,C_i) log P_m(C_i|B)] \\ &- \textstyle\sum_{A,B} \hat{P}(A,B) log P_m(B|A) - \frac{d(m')-d(m)}{2} logN. \end{aligned} \quad (10)$$

Among all models in $ND(m)$, we choose the one for which this difference is the largest.

We do not have good heuristics for selecting among models in $SD(m)$ based on $\mathbf{D}_m$. So we proceed with them in the straightforward way. Parameters of all these candidate models are optimized by running EM. Their BIC scores are then computed. The one with the highest BIC score is chosen.

4.6 A generalization

The algorithm outlined so far has a natural generalization. For a given operator, instead of choosing the best model we can choose the top K best models for some number K. This top-K scheme reduces the chance of getting trapped in local maxima. In general, the larger the K, the smaller the probability of encountering local maxima. On the other hand, larger K also implies running EM on more models and hence longer computation time. In practice, one can start with K being 1 and increase it gradually as long as time permits.

4.7 Local EM

By applying the idea of structural EM, we have substantially reduced the number of calls to EM. Nonetheless we still need to run EM on a number of models at each step of search. Within the top-K scheme, we need to run EM on K models for each of the operators except for SD. For SD, we need to run EM on n_h models, where n_h is the number of latent nodes in the current model. To achieve further speedup, we replace all those calls to EM with calls to a more efficient procedure that we refer to as local EM.

Parameters of the current model m were estimated at the end of the previous search step. Each candidate model m' generated at the current search step differs from m only slightly. The idea of *local EM* is to optimize the conditional probability distributions (CPDs) of only a few variables in m', while keeping those of other variables the same as in m. If m' is obtained from m by adding a state to or deleting a state from a variable A, then only the CPD's that involve A are optimized. If m' is obtained from m by introducing a latent node H to separate a node A from two of its neighbors, then only the CPD's that involve A and H are optimized. If m' is obtained from m by relocating a node B from A to C, then only the CPD's that involve A and C are optimized. Finally, if m' is obtained from m by deleting a node B and making all neighbors except for one, which we denote by A, neighbors of A, then only the CPD's that involve A are optimized.

Obviously, model parameters provided by local EM deviate from those provided by EM. To avoid accumulation of deviations, we run EM once at the end of each search step on the model that is selected as the best at that step.

5 Empirical results

This section reports experiments designed to determine whether HSHC can learn models of good quality and how efficient it is. In all the experiments, EM and local EM were configured as follows. To estimate all/some of the parameters for a given uninstantiated/partially instantiated model m, we first randomly generated 64 sets of parameters for the model, resulting in 64 initial fully instantiated models [1]. One EM/local EM iteration was run on all models and afterwards the worst 32 models were discarded. Then two EM/local EM iterations were run on the remaining 32 models and afterwards the worst 16 models were discarded. This process was continued until there was only one model. On this model, EM/local EM were terminated either if the increase in loglikelihood fell below 0.01 or the total number of iterations exceeded 500.

[1] At the end of each search step, we ran EM on the model selected in that step. This model was used as one of the initial instantiated models. Hence only 63 initial models were actually generated randomly.

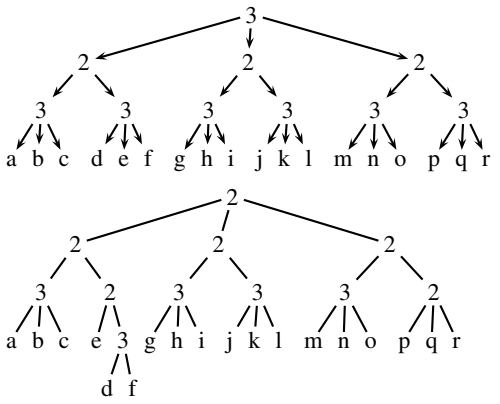

Figure 3. TOP: One of the test models. Manifest nodes are labelled with their names. All manifest variables have 3 states. Latent nodes are labelled with their cardinalities. BOTTOM: The unrooted HLC model reconstructed from data by HSHC3.

5.1 Experiments with synthetic data

We used 5 generative models, consisting of 6, 9, 12, 15, and 18 manifest variables respectively. One of the models is shown in Figure 3. Parameters were randomly generated except we ensured that each conditional distribution has a component with mass larger than 0.8. We also ensured that, in every conditional probability table, that the largest components of different rows are not all at the same column. A data set of 10,000 records were sampled for each model. We then ran both SHC and HSHC to reconstruct the generative models from the data sets. HSHC was tested on all the 5 data sets, while SHC was tested on only 3, i.e. those sampled from the 3 simplest generative models. For HSHC, we let the K in the top-K scheme run from 1 to 3. So we in fact tested three versions of the algorithm. We will refer to them using HSHC1, HSHC2, and HSHC3. The algorithms were implemented in Java and all experiments were run on a Pentium 4 PC with a clock rate of 2.26 GHz.

To measure the quality of the learned models, a testing set $\mathbf{D}_t$ of 5,000 records were sampled from each generative model. The log score $logP_l(\mathbf{D}_t)$ of each learned model and the log score $logP_o(\mathbf{D}_t)$ of the corresponding original model were computed. Let N_t be the number records in $\mathbf{D}_t$ in general. Note that as N_t goes to infinity the average log score difference $(logP_o(\mathbf{D}_t) - logP_l(\mathbf{D}_t))/N_t$ tends to $KL(P_o, P_l)$, the KL divergence of the probability distribution of manifest variables in the learned model from that of manifest variables in the original model. We hence refer to it as *empirical KL divergence*. It is a good measure of the quality of the learned model.

The empirical divergences between the learned models and the original models are shown in Figure 4. We see that

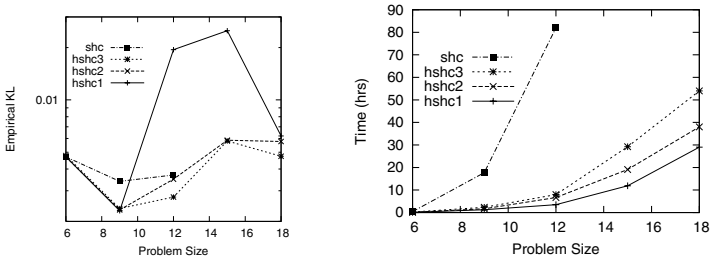

Figure 4. Model quality and time statistics.

some of the models reconstructed by HSHC1 are of poor quality in two of the five cases. However, all the models reconstructed by HSHC2 and HSHC3 match the generative models extremely well in terms of distribution over the manifest variables. The structures of these models are either identical or very similar to the structures of the generative models. The structure of the model produced by HSHC3 for the model shown at the top of 3 is shown at the bottom of the same figure. The two structures are very similar.

Time statistics are shown in Figure 4. We see that HSHC is much more efficiently than SHC and it scales up fairly well. In another experiment involving the HLC model shown in Figure 1, SHC was 22 times faster than DHC.

5.2 Experiments with real-world data

The training set of the COIL Challenge 2000 data consists of 5,822 customer records. Each records consists of 86 attributes, containing socio-demographic information (Attributes 1-43) and insurance product ownerships (Attributes 44-86). The task is to learn a model and use it to predict who would buy caravan (mobile home) insurance polices.

For our experiments, we selected Attributes 4,5,43 and 44-86. Attribute values were merged so that there are at least 30 cases for each value. Thereafter, the attributes have 2 to 9 values. In the resultant data set, there are fewer than 10 cases where Attributes 50, 60, 71 and 81 take nonzero values. Those attributes were therefore excluded from further analysis. This leaves us with 42 attributes.

We analyzed the data set using HSHC. Again the top-K scheme was used, with K being 1, 5, 10, and 20. The running times and the BIC scores of the resulting models are shown in the following table:

K	1	5	10	20
Time (hrs)	51	99	121	169
BIC	-52,522	-51,625	-51,465	-51,592

The best model was found in the case of K=10. We denote the model by M_{10}. The structure of the model is shown in Figure 5, which is interesting in a number of ways. For example, in the data set, there is a pair of variables for each type of insurance policy, standing respectively for the amount of contribution and the number of policies; In M_{10},

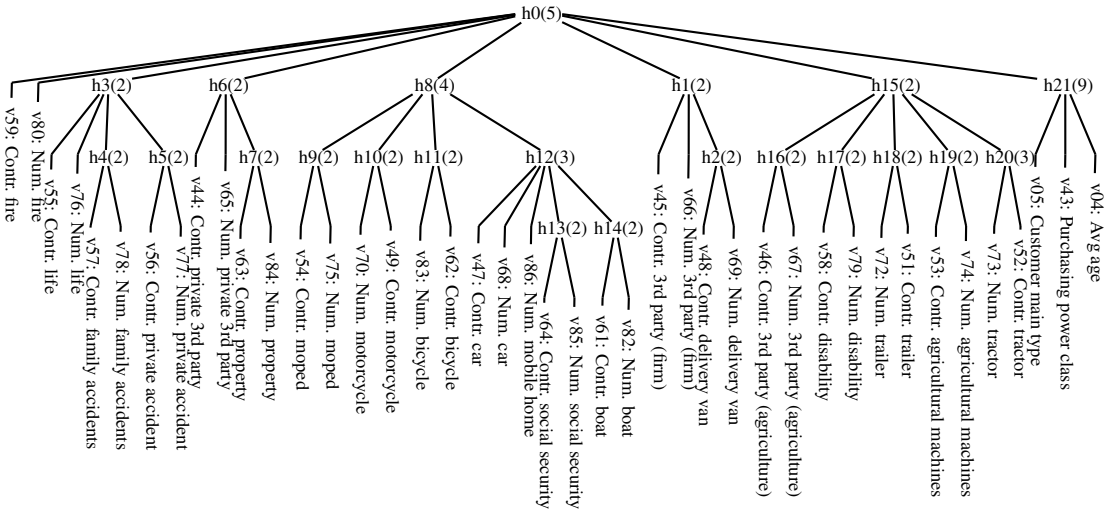

Figure 5. HLC model found for the COIL Challenge 2000 data.

there is a latent variable for each such pair. Moreover, all agriculture-related attributes are under one latent variable, namely h_{15}; all vehicle-related attributes are under the latent variable h_8; and so on. One might argue the model structure is not completely in accord with the structure of the domain. We have considered a number of possible improvements. None of them have given rise to a better model.

In COIL Challenge 2000, there is a test set of 4,000 that contains 238 caravan policy owns. The prediction task requires participants to identify a subset of 800 that contains as many caravan policy owners as possible. Random selection results in 42 policy owners, while the best entry contains 121. Selection based on M_{10} gives 110. This is good performance considering that HSHC aims at optimizing BIC score rather than classification error.

For the sake of comparison, we also ran GES on the training set to obtain a Bayesian network without latent variables. The network is intuitively less appealing than M_{10}, as one can naturally imagine. When it is used to guide the prediction task, only 83 policy owners were selected. More details will be given in a longer version of this paper.

6 Conclusions

We have developed a search-based algorithm for learning HLC models. The algorithm is not fast enough for online applications, but it can used in applications such as COIL Challenge 2000 to analyze data offline.

Our results with the COIL Challenge 2000 data should have given the reader a hint about why learning HLC models is interesting from the viewpoint of application. Good results have also been obtained in an application in traditional Chinese medicine. More details will be given in separate papers.

Acknowledgements

We thank Finn V. Jensen and Gytis Karciauskas for valuable discussions. Research was partially supported Hong Kong Research Grants Council under grant HKUST6088/01E and Grant Agency of the Czech Republic under Grant 201/02/1269.

References

[1] D. Connolly (1993). Constructing hidden variables in Bayesian networks via conceptual learning. *ICML-93*, 65-72.

[2] N. Friedman (1997). Learning belief networks in the presence of missing values and hidden variables. *ICML-97*, 125-133.

[3] P. F. Lazarsfeld, and N. W. Henry(1968). *Latent structure analysis*. Boston: Houghton Mifflin.

[4] C. Meek (1997). *Graphical models: Selection causal and statistical models.* Ph.D. Thesis, Carnegie Mellon University.

[5] J. Pearl (pearl88). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* Morgan Kaufmann Publishers, Palo Alto.

[6] G. Schwarz (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2), 461-464.

[7] P. van er Putten and M. van Someren (eds.) (2002). COIL Challenge 2000: The insurance company case. Sentient Machine Research, Amsterdam.

[8] N. L. Zhang (2002). Hierarchical latent class models for cluster analysis. *AAAI-02*, 230-237.