The Role of Operation Granularity in Search-Based Learning of Latent Tree Models

Tao Chen[1], Nevin L. Zhang[2], and Yi Wang[3]

[1] Shenzhen Institute of Advanced Technology
Chinese Academy of Sciences
Shenzhen, China
{tao.chen}@siat.ac.cn

[2] Department of Computer Science & Engineering
The Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, Hong Kong
lzhang@cse.ust.hk

[3] Department of Computer Science
National University of Singapore
Singapore 117417, Singapore
wangy@comp.nus.edu.sg

Abstract. Latent tree (LT) models are a special class of Bayesian networks that can be used for cluster analysis, latent structure discovery and density estimation. A number of search-based algorithms for learning LT models have been developed. In particular, the HSHC algorithm by [1] and the EAST algorithm by [2] are able to deal with data sets with dozens to around 100 variables. Both HSHC and EAST aim at finding the LT model with the highest BIC score. However, they use another criterion called the cost-effectiveness principle when selecting among some of the candidate models during search. In this paper, we investigate whether and why this is necessary.

1 INTRODUCTION

Latent tree (LT) models are tree-structured Bayesian networks where internal nodes represent *latent variables* while leaf nodes represent *manifest variables*. In this paper we assume that all variables are categorical. LT models were previously known as hierarchical latent class models [3]. They are interesting for three reasons:

1. They relax the local independence assumption of latent class (LC) models [4] and hence provide a more general framework for cluster analysis of categorical data. LT analysis can find meaningful classes along multiple dimensions while the LC analysis always clusters data in one single way.
2. LT analysis can reveal latent structures behind data. Using LT models, researchers have found interesting latent structures from stocks data [5], marketing data [6] and medical data [7].

3. LT models are computationally simple to handle and at the same time can model complex interactions among manifest variables [8]. Those two properties make LT models a good tool for estimating joint distributions of discrete variables [9].

This paper is concerned with the induction of LT models from data. A number of search-based algorithms have been developed. Some of them can deal with data sets with only a few variables [3]. The first algorithm that can handle dozens of manifest variables was published in 2004 and is called HSHC (Heuristic Single Hill Climbing) [1]. Recently [2] have published another algorithm called EAST (Expansion, Adjustment, Simplification until Termination) which, in comparison with HSHC, is conceptually simpler, yet is more efficient and finds better models.

Both HSHC and EAST aim at finding the LT model with the highest BIC score. However, they use another criterion called the cost-effectiveness principle when selecting among some of the candidate models during search. This is to deal with the issue of *operation granularity*, which refers to the phenomenon that, in search-based learning of LT models, some operations might increase the complexity of the current model much more than other operations.

However, it is not well understood whether and why it is necessary to use the cost-effectiveness principle. In this paper we seek to clarify the issue through empirical investigation.

2 SEARCH-BASED LEARNING OF LT MODELS

Figure 1 (a) shows the structure of an example LT model. Let $\mathcal{D}$ be a data set on some manifest variables. To *learn an LT model* from $\mathcal{D}$ means to find a model m that maximizes the BIC score:

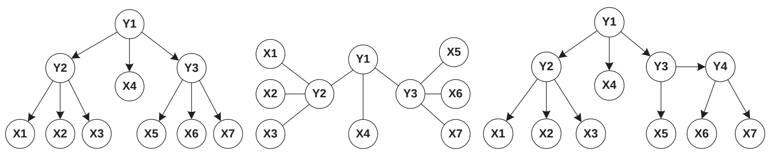
$$BIC(m|\mathcal{D}) = \max_{\theta} \log P(\mathcal{D}|m, \theta) - \frac{d(m)}{2} \log N,$$

where θ denotes the set of model parameters, $d(m)$ the number of independent parameters, and N the sample size. [4] It has been shown that edge orientations cannot be determined from data [3]. So we can learn only *unrooted latent tree models*, which are latent tree models with all directions on the edges dropped. An example is given in Figure 1 (b). From now on when we speak of LT models we always mean unrooted LT models unless it is explicitly stated otherwise.

2.1 SEARCH OPERATORS

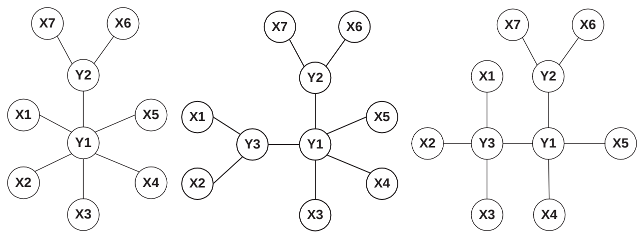
The basic building blocks of a search-based algorithm are the search operators. We adopt with minor modification five operators from [1]. They are: *state introduction (SI)*, *node introduction (NI)*, *node relocation (NR)*, *state deletion (SD)*,

[4] Geiger *et al.* [10] argue that the BICe score should be used instead of the BIC score when latent variables are present. However, the BICe score is currently impractical to use due to the lack of efficient methods for computing effective dimensions of models.



(a) m (rooted) (b) m (unrooted) (c) m'

Fig. 1. Rooted latent tree model, unrooted latent tree model and latent tree model obtained by the node introduction of Y4.



(a) m_1 (b) m_2 (c) m_3

Fig. 2. m_2 is a latent tree model obtained by applying NI to m_1. m_3 is a latent tree model by applying NR to m_2

and *node deletion (ND)*. Given an LT model and a latent variable in the model, the *SI* operator creates a new model by adding a new state to the domain of the variable. The *SD* operator does the opposite. The *NI* operator involves one latent node Y and two of its neighbors. It creates a new model by introducing a new latent node Y' to mediate the latent variable and the two neighbors. The cardinality of Y' is set to be that of Y. In m_1 of Figure 2 (a), introducing a new node Y_3 to mediate Y_1 and its neighbors X_1 and X_2 results in m_2. For the sake of computational efficiency, we do not consider introducing a new node to mediate Y and more than two of its neighbors. The *ND* operator is the opposite of NI. The *NR* operator involves two latent nodes Y_1 and Y_2 and a neighbor Z of Y_1. It creates a new model by relocating Z from Y_1 to Y_2, i.e. removing the link between Z and Y_1 and adding a link between Z and Y_2. In m_2 of Figure 2 (b), relocating X_3 from Y_1 to Y_3 results in m_3.

The search operators can be divided into three groups. The NI and SI operators make the current model more complex and hence are *expansion operators.* The ND and SD operators make the current model simpler and hence are *simplification operators.* The NR operator rearranges connections between the variables and hence is an *adjustment operator.*

2.2 A SEARCH PROCEDURE

Figure 3 presents a search procedure for learning LT model [2]. Called EAST0, the procedure adopts the grow-restructure-thin strategy that has emerged from

```
EAST0($m, \mathcal{D}$)
Repeat until termination:
    $m_1 \leftarrow$ expand($m, \mathcal{D}$).
    $m_2 \leftarrow$ adjust($m_1, \mathcal{D}$).
    $m_3 \leftarrow$ simplify($m_2, \mathcal{D}$).
    If $BIC(m_3|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return $m$;
    Else $m \leftarrow m_3$.

expand($m, \mathcal{D}$)
Repeat until termination:
    $m_1 \leftarrow \arg\max_{m' \in NI(m) \cup SI(m)} BIC(m'|\mathcal{D})$.
    If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return $m$.
    If $m_1 \in NI(m)$, $m \leftarrow$ enhanceNI($m_1, m, \mathcal{D}$);
    Else $m \leftarrow m_1$

adjust($m, \mathcal{D}$)
Repeat until termination:
    $m_1 \leftarrow \arg\max_{m' \in NR(m)} BIC(m'|\mathcal{D})$.
    If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return $m$;
    Else $m \leftarrow m_1$.

simplify($m, \mathcal{D}$)
Repeat until termination:
    $m_1 \leftarrow \arg\max_{m' \in ND(m)} BIC(m'|\mathcal{D})$.
    If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return $m$;
    Else $m \leftarrow m_1$.
Repeat until termination:
    $m_1 \leftarrow \arg\max_{m' \in SD(m)} BIC(m'|\mathcal{D})$.
    If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, break;
    Else $m \leftarrow m_1$.
```

Fig. 3. EAST0 is the basic version of the EAST search procedure. It does not handle operation granularity.

the literature on learning Bayesian networks without latent variables (e.g., [11]). The search procedure of EAST0 has a number of rounds. Every round is divided into three stages: expansion, adjustment and simplification. At the expansion stage, EAST0 searches with the expansion operators until the BIC score ceases to increase. To understand the motivation, recall that the BIC score consists of two terms. The first term is the maximized likelihood, which measures model fit. The second term penalizes model complexity. Our objective is to optimize the BIC score. Suppose we start with a model that does not fit the data at all. Then improving model fit is the first priority at the initial phase of the search. EAST0 does so by searching with the expansion operators.

Note the a subroutine named `enhanceNI` is called after each application of the NI operator. This is to compensate for the constraint imposed on the NI operator. Consider the model m_1 in Figure 2. We can introduce a new latent node Y_3 to mediate Y_1 and two of its neighbors, say X_1 and X_2, and thereby obtain the model m_2. However, we are not allowed to introduce a latent node to mediate Y_1 and more than two of its neighbors, say X_1, X_2 and X_3, and thereby obtain m_3. To remedy the situation we consider, after each application of the NI operator, enhancements to the operation. As an example, suppose we have just applied NI to m_1 and have obtained m_2. What we do next is to consider relocating the other neighbors of Y_1 in m_1, i.e. X_3, X_4, X_5 and Y_2, to the new latent variable Y_3. If it turns out to be beneficial to relocating X_3 but not the other three nodes, then we obtain the model m_3. We use `enhanceNI`$(m_2, m_1, \mathcal{D})$ to denote this subroutine.

After model expansion ceases to increase the BIC score, EAST0 enters the adjustment stage. At this stage, EAST0 repeatedly relocates nodes around in the model until it is no longer beneficial to do so. There are no definite conclusion on which of the likelihood term and the penalty term in the BIC score is improved. The intuition in the phase is that the search may make some mistakes in the expansion phase. For example, a variable is connected to a wrong parent. We correct such mistakes by relocating nodes. The BIC score is improved slightly in the resulting model. However, we come back to the right path to proceed. The adjustment stage is followed by the simplification stage. At this stage, EAST0 first repeatedly applies ND to the current model until the BIC score ceases to increase and then it does the same with SD. By considering ND and SD, we focus on the improvement of the penalty term in the BIC score. If model score is improved in any of the three stages, EAST0 repeats the circle with the best model obtained so far as the initial model.

2.3 COMPARISON WITH HSHC

EAST0 places no restriction on how far away a node can be relocated. Consequently it might need to evaluate a large number of candidate models, especially towards the end of the search process. Although expensive, unrestricted node relocation is necessary in EAST0 because model adjustment takes place after multiple expansion steps. Two nodes that should be neighbors might be located far away from each other after a series of NI operations and their enhancements. On the other hand, HSHC considers node relocation after each model expansion operation. It also considers more node relocations after a node has just been relocated. As such, it cannot afford to use unrestricted node relocation. So it restricts that a node can be relocated only one step away. Determining the pros and cons of the two options is another basic issue concerning search-based learning of LT models. In [12] we have shown that HSHC is more likely to end up at local maxima than EAST0. Hence we use EAST0 as the platform of study in this paper.

2.4 EFFICIENT MODEL EVALUATION

Each of the $\arg\max$ operators in EAST0 examines a list of candidate models and picks one of them as output. The BIC score is used as the objective function for the selection. To calculate the BIC score of a candidate model, one needs to maximize its likelihood function, which requires the expectation-maximization (EM) algorithm due to the presence of latent variables. EM is known to be computationally expensive. Hence it is prohibitive to compute the BIC scores of a large number of candidate models. Hence it is necessary to replace the BIC score with other objective functions that are easy to compute. In this section we present one such objective function. It is obtained from the BIC score by replacing its first term with what we call the maximum restricted loglikelihood.

Conceptually EAST0 works with unrooted LT models. In the implementation, however, we represent unrooted models as rooted models. Rooted LT models are BNs and their parameters are clearly defined. This makes it easy to see how the parameter composition of a candidate model m' is related to that of the current model m. Consider the models m and m' in Figure 1 (a) and (c). The latter is obtained from m by introducing a new latent variable Y_4 to mediate Y_3 and two of its neighbors X_6 and X_7. The two models share the parameters for describing the distributions $P(Y_1)$, $P(Y_2|Y_1)$, $P(X_1|Y_2)$, $P(X_2|Y_2)$, $P(X_3|Y_2)$,

$P(X_4|Y_1)$, $P(Y_3|Y_1)$ and $P(X_5|Y_3)$. On the other hand, the parameters for describing $P(Y_4|Y_3)$, $P(X_6|Y_4)$ and $P(X_7|Y_4)$ are peculiar to m' while those for describing $P(X_6|Y_3)$ and $P(X_7|Y_3)$ are peculiar to m.

We write the parameters of a candidate model m' as a pair (δ_1, δ_2), where δ_1 is the collection of parameters that m' shares with m. Similarly, we write the parameters of the current model m as a pair (θ_1, θ_2), where θ_1 is the collection of parameters that m shares with m'. Suppose we have computed the MLE (θ_1^*, θ_2^*) of the parameters of m. For a given value of δ_2, $(m', \theta_1^*, \delta_2)$ is a fully specified BN. In this BN, we can compute $P(\mathcal{D}|m', \theta_1^*, \delta_2) = \prod_{\mathbf{d} \in \mathcal{D}} P(\mathbf{d}|m', \theta_1^*, \delta_2)$. As a function of δ_2, this will be referred to as the *restricted likelihood function* of δ_2. The *maximum restricted loglikelihood*, or simply the *maximum RL*, of the candidate model m' is defined to be

$$\max_{\delta_2} \log P(\mathcal{D}|m', \theta_1^*, \delta_2).$$

The *restricted likelihood (RL)* method for model evaluation replaces the likelihood term in the BIC score of m' with its maximum RL and uses the resulting function to evaluate m'.

There is another method for efficient model evaluation. It first completes the data set $\mathcal{D}$ using the current model (m, θ^*), where θ^* is the MLE of the parameters of m. Then it uses the completed data set to evaluate candidate models. Hence we call it the *data completion (DC)* method. Determining the pros and cons of the RL and DC methods for efficient model evaluation is also a basic issue concerning search-based learning of LT models. In [12] we have shown that the RL method is more accurate than the DC method and it results in better models. Hence we use the RL method in this paper.

3 OPERATION GRANULARITY

Operation granularity refers to the phenomenon that some operations might increase the complexity of the current model much more than other operations. As an example, consider the situation where there are 100 binary manifest variables. Suppose the search starts with the LC model with one binary latent node Y. Applying the SI operator to the model would introduce 101 additional model parameters, while applying the NI operator to the model would increase the number of model parameters by only 2. The latter operation is clearly of much finer-grain than the former.

To deal with operation granularity, Zhang and Kočka [1] suggest that we choose between candidate models generated by NI and SI [5] using the so-called cost-effectiveness principle. Let m be the current model and m' be a candidate model. Define the *improvement ratio of* m' *over* m given data $\mathcal{D}$ to be

$$IR(m', m|\mathcal{D}) = \frac{BIC(m'|\mathcal{D}) - BIC(m|\mathcal{D})}{d(m') - d(m)}. \quad (1)$$

[5] Strictly speaking, operation granularity is also an issue for other search operators. However the issue is much less serious there and hence no special treatment is introduced for simplicity.

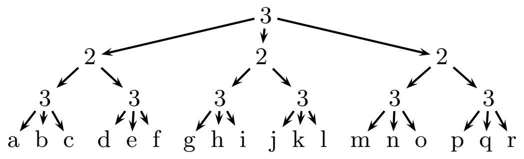


Fig. 4. One of the test models. Manifest nodes are labeled with variable names. All manifest variables have 3 states. Latent nodes are labeled with cardinalities.

It is the increase in model score per unit increase in model complexity. The *cost-effectiveness* principle states that among all candidate models generated by SI and NI, choose the one that has the highest improvement ratio. We use EAST to denote the algorithm that is the same as EAST0 except that it uses the cost-effectiveness principle for model selection in the expansion subroutine.

4 EMPIRICAL RESULTS

We have conducted experiments on synthetic data to compare EAST0 and EAST. The data were generated using three manually constructed LT models that contain 7, 12 and 18 manifest variables respectively. The structure of the 7-variable model is shown in Figure 1 (a) and that of the 18-variable model is shown in Figure 4. The structure of the 12-variable model is similar to that of the 18-variable model. The parameters were randomly generated. Three data sets of sizes 1k, 5k and 10k were sampled from each of the three models. The three data sets for the 7-variable model are denoted by $\mathcal{D}_7(1k)$, $\mathcal{D}_7(5k)$ and $\mathcal{D}_7(10k)$. Similar notations are used for other data sets.

We analyzed the data sets using EAST0 and EAST. [6] Both algorithms started with the LC model with a binary latent node. The quality of a learned model is measured by the empirical KL divergence of the model from the corresponding generative model, an approximation to the true KL divergence that was computed based on 5k testing data. We report results that are averages over 10 runs, along with the standard deviations. The results are given in Table 1.

For convenience we will refer to models obtained by the EAST and EAST0 algorithms as EAST and EAST0 models respectively. Consider the models learned for the data sets $\mathcal{D}_{18}(10k)$, $\mathcal{D}_{18}(5k)$ and $\mathcal{D}_{12}(10k)$. The divergences of the EAST models from the generative models are 0.0047, 0.0148 and 0.0032 respectively, while those of the EAST0 models are 0.0207, 0.0326 and 0.0079. The EAST models are significantly closer to the generative models than the EAST0 models. So the EAST algorithm reconstructed the generative distributions better than the EAST0 algorithm.

[6] Both EAST0 and EAST have two parameters μ and ν that control a subroutine for model evaluation. We have run EAST under several settings to determine the impact of the parameters. The results reported in this paper were obtained under the setting $\mu = 8$ and $\nu = 40$, the highest setting that we tried.

Table 1. Results of EAST0 and EAST on synthetic data. Quality of a learned model is measured by its empirical KL divergence from the generative model. The numbers are averages over 10 runs, along with the standard deviations in parenthesis. Highlighted in boldface are the cases where the models found by EAST0 are not as good as those found by EAST.

	$\mathcal{D}_7(1k)$			$\mathcal{D}_7(5k)$			$\mathcal{D}_7(10k)$		
	KL	time (mins)	steps	KL	time (mins)	steps	KL	time (mins)	steps
EAST0	.0287(4.3e-6)	.7(8.5e-3)	0(0)	.0101(4.8e-5)	6.3(0.1)	2.0(0.0)	.0058(8.6e-5)	8.4(0.6)	2.0(0.0)
EAST	.0287(4.2e-6)	.7(1.3e-2)	0(0)	.0101(4.5e-5)	7.1(0.1)	2.0(0.0)	.0057(1.0e-4)	8.4(0.3)	2.0(0.0)

	$\mathcal{D}_{12}(1k)$			$\mathcal{D}_{12}(5k)$			$\mathcal{D}_{12}(10k)$		
	KL	time (mins)	steps	KL	time (hrs)	steps	KL	time (hrs)	steps
EAST0	.1017(1.8e-2)	17.1(1.8)	12.1(0.7)	.0311(1.3e-2)	1.0(0.1)	14.1(2.1)	**.0079**(4.7e-3)	1.5(0.1)	13(1.1)
EAST	.0999(1.2e-2)	17.2(2.2)	12.0(1.0)	.0310(4.9e-5)	1.4(0.0)	19.6(0.5)	**.0032**(2.4e-4)	2.6(0.2)	20.8(1.1)

	$\mathcal{D}_{18}(1k)$			$\mathcal{D}_{18}(5k)$			$\mathcal{D}_{18}(10k)$		
	KL	time (hrs)	steps	KL	time (hrs)	steps	KL	time (hrs)	steps
EAST0	.1865(6.3e-6)	.6(.01)	20(0)	**.0326**(1.1e-2)	4.4(0.9)	22.4(4.6)	**.0207**(1.2e-2)	10.4(1.7)	24.4(4.0)
EAST	.1865(7.5e-6)	.7(.02)	20(0)	**.0148**(4.5e-3)	6.0(0.6)	33.8(1.7)	**.0047**(7.0e-4)	18.4(3.9)	37.2(0.8)

To give the reader a concrete feeling about the models, we present in Figure 6 the structures of an EAST model and an EAST0 model for the data set $\mathcal{D}_{18}(10k)$. We see that the structure of the EAST model is almost identical to that of the corresponding generative model, while the structure of the EAST0 model is quite different. So the EAST algorithm reconstructed the structure of the generative model much better than the EAST0 algorithm. Moreover the BIC score of the EAST model is -115108, which is significantly higher than -115500, the BIC score of the EAST0 model.

In summary, the EAST algorithm learned better models for $\mathcal{D}_{18}(10k)$, $\mathcal{D}_{18}(5k)$ and $\mathcal{D}_{12}(10k)$ than the EAST0 algorithm. On the other six data sets, the two algorithms found models of similar or the same quality. The only difference between EAST and EAST0 is that EAST deals with operation granularity using the cost-effectiveness principle, while EAST0 does not. So the results imply that it is indeed necessary to deal with operation granularity and the cost-effectiveness principle is an effective method for doing so. We investigate the reasons in the next section.

5 PERFORMANCE DIFFERENCE EXPLAINED

We explain the differences in performance between EAST0 and EAST in four steps.

5.1 EARLY USE OF SI OPERATIONS

Let us examine one run of EAST0 and one run of EAST on the $\mathcal{D}_{18}(10k)$ data set. Figure 5 (a) shows the change in model complexity over the search steps.

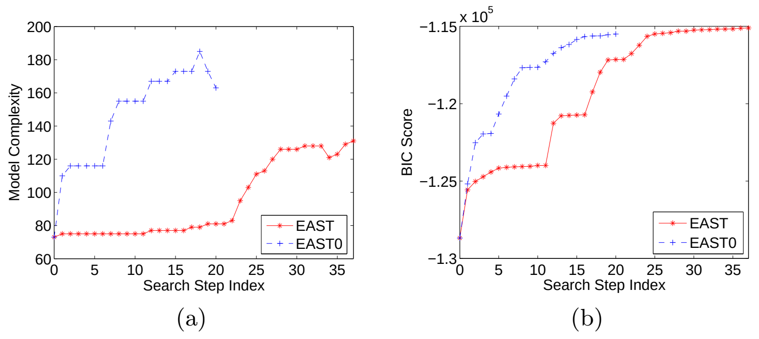


Fig. 5. Search processes on $\mathcal{D}_{18}(10k)$: Model complexity and score as functions of search steps.

We see that the curve for EAST0 increases quickly early in the search process. There are two big jumps, one at Step 1 and another at Step 7. A trace of the search path reveals that those are the steps where EAST0 applied SI operations. In contrast the curve for EAST increases slowly and has smaller jumps. The first jump occurred at Step 23. It was the first time where EAST applied the SI operator.

So EAST0 applied SI operations much earlier than EAST. This is true not only for the runs examined above, but also true for all runs on $\mathcal{D}_{18}(10k)$ and for all runs on $\mathcal{D}_{12}(5k)$, $\mathcal{D}_{12}(10k)$ and $\mathcal{D}_{18}(5k)$. We see from Table 1 that EAST0 took fewer steps than EAST to terminate on the four data sets. This is exactly because EAST0 took large steps by applying SI operation early, while EAST did not.

In general, EAST0 tends to apply SI operations early while EAST tends to apply them late. Here are the reasons. First, as discussed in Section 3, SI operations are often of larger-grain than NI operations at the early stage of search. Second, although a large-grain operation increases the penalty term of the BIC score more than a small-grain operation, it often increases the likelihood term even more early in the search process because model fit is usually poor at that time. So there are often SI operations with higher scores than NI operations at the early stage of search. Third, EAST0 selects model based on the BIC score. It would choose SI operations if they have higher scores than NI operations. Hence it often chooses SI operations over NI operations early in the search process. Fourth, EAST is not only concerned with the increase in model score, but also its “cost”, i.e., the increase in model complexity. It tries to achieve maximum increase in model score with minimum increase in model complexity. As such, it is less likely than EAST0 to choose large-grain operations and hence tends to choose SI operations later than EAST0.

5.2 FAT LATENT VARIABLES

Three models from the aforementioned search path of EAST0 are shown at the top of Figure 6. We see that a latent variable with 4 states was created at Step 7, after the second SI operation. Since the variable has more states than latent variables in the generative model, we call it a *fat latent variable.* [7] EAST0 introduced three other fat latent variables later. One of them was subsequently deleted and the cardinality of another was reduced by one. Two fat latent variables remain in the final model. On the other hand, EAST introduced no fat latent variables at all.

We say that a latent node is *well connected* if it is connected to, in a relative sense, a large number of manifest variables. To account for the interactions among those manifest variables, or even part of the interactions, the latent node usually needs to have a large number of states. If its cardinality is currently low, an increase in its cardinality would greatly improve model fit and hence greatly increase model score. Hence SI operations, if applied at all, tend to be applied on well connected latent nodes. Since search starts from LC model, there is only a few (maybe only one) well connected latent nodes at the early stage of search. EAST0 tends to apply SI operations at the early stage of search. Hence EAST0 tends to repeatedly increase the cardinality of well connected latent nodes and consequently tends to create fat latent variables. On the other hand, EAST usually applies SI operation late in the search process. At that time, a latent node is usually connected to a small number of manifest variables. Hence there is less chance to create fat latent variables.

5.3 LOCAL MAXIMA

Fat latent nodes do not necessarily lead to local maxima, but they do sometimes. Use m_{EAST0} to denote the final model obtained by EAST0. Its BIC score is -115500. In contrast the BIC score of the model found by EAST is -115108. So m_{EAST0} in Figure 6 is a local maxima. In the following we argue that one reason for EAST0 to be trapped at this local maxima is because Z is a fat latent variable.

A comparison of m_{EAST0} with the generative model suggests that it might be beneficial to introduce a new latent node to mediate Z and its neighbors j and l. However, EAST0 did not do this. The reasons, we argue, are as follows. In the generative model the interactions among the manifest variables m, n and o are accounted for by a latent variable with 3 states. However, the variable Z in m_{EAST0} has 4 states. This is more than enough to account for the interactions among m, n and o. In addition, it also accounts for, to some extent, interaction between j and l and correlations between the two groups of variables. Consequently there is no strong reason to introduce a new latent node.

[7] Intuitively, a fat latent variable is one whose cardinality is larger than necessary. For the case without a generative model, it can be defined with respect to the model that has the highest BIC score.

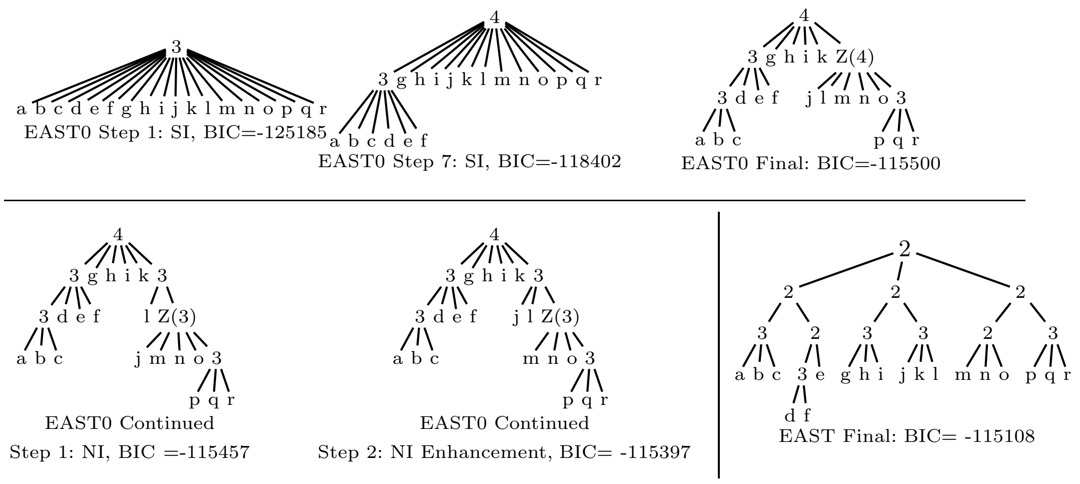


Fig. 6. Top Row: Three models from a search path of EAST0 on the data set $\mathcal{D}_{18}(10k)$. Bottom Row (Left): Continued search by EAST0 from the model that is obtained from the final EAST0 model by decreasing the cardinality of Z by 1. The search went on for 10 steps. Only two steps are shown here. Bottom Row (Right): Final model obtained by EAST.

To verify our argument, we decreased the cardinality of Z to 3 and searched further using EAST0 starting from the resulting model. The trace is shown at the bottom row of Figure 6. We see that, as expected, a new latent node was indeed introduced between Z and its neighbors j and l. The local maximum was escaped. Furthermore the new node is placed close to manifest variables g, h, i and k. This is ideal because in the generative model j and l are close to those variables.

Why didn't EAST0 decrease the cardinality Z as we did manually then? This is because Z is directly connected not only to m, n and o, but also j and l. To account for the interactions among m, n and o, three states are sufficient. However, to account for the correlations among all those 5 variables, three states are not sufficient. As a matter of fact, if the cardinality of Z is decreased from 4 to 3, the BIC score also decreases from -115,500 to -115,537. So this is a deadlock situation. To separate j and l from Z by introducing a new node, we need to first reduce the cardinality of Z; and to reduce the cardinality of Z, we need to first reduce the number of manifest variables connected to it. This deadlock would not have occurred had fat latent variables not been introduced.

5.4 IMPACT OF PROBLEM SIZE AND SAMPLE SIZE

Putting together the arguments presented above suggests that EAST0 is more easily than EAST to get trapped at local maxima. This explains why the models it found in our experiments are sometimes not as good as those found by EAST. One thing remains unexplained: Why the differences occurred only on the data sets $\mathcal{D}_{12}(10k)$, $\mathcal{D}_{18}(5k)$ and $\mathcal{D}_{18}(10k)$ that are more complex than the other data sets in terms of sample size and the number of manifest variables.

To understand the phenomenon, let o_{SI} be an SI operation and o_{NI} be an NI operation. Suppose the o_{SI} is of larger grain than o_{NI}. Then it increases the penalty term of the BIC score more than the latter. Further suppose that o_{SI} increases the likelihood term more than o_{NI}. Then which operation EAST0 ends up choosing depends on how the two terms balance out. It is well known that the likelihood term increases linearly with sample size, while the penalty term increase logarithmically. Therefore as the sample size increases, EAST0 would be more and more likely to choose o_{SI} over o_{NI}. This implies that EAST0 would be more and more likely to apply SI operations early in the search process and hence is more and more easily to get trapped at local maxima. This explains why, in our experiments, EAST0 performed worse and worse as we move from $\mathcal{D}_{18}(1k)$ to $\mathcal{D}_{18}(5k)$ and then to $\mathcal{D}_{18}(10k)$.

In our experiment, EAST0 performed worse and worse as we move from $\mathcal{D}_{7}(10k)$ to $\mathcal{D}_{12}(10k)$ and then to $\mathcal{D}_{18}(10k)$. We explain the phenomenon as follows. When the number of manifest variables increases, SI operations would be of larger and larger grain relative to NI operations. Consequently, EAST0 would be more and more likely to apply SI operations early in the search process and hence is more and more easily and get trapped at local maxima.

6 CONCLUSIONS

We have shown that operation granularity often results in local maxima if not dealt with. This is because that, at the early stage of search, SI operations are usually of larger grain than NI operations and often have higher BIC scores. If one simply uses BIC for model selection, then one tends to apply SI operations early, which often leads to fat latent variables, which in turn might result in local maxima. When the cost-effectiveness principle is used for model selection, on the other hand, SI operations are applied much latter. This reduces the chance of creating fat latent variables and hence the chance of local maxima.

One might suggest that we deal with operation granularity by introducing additional search operators. After reading Section 5.3 one might, for instance, suggest a composite search operator that first reduces the cardinality of a latent variable and then introduces a new latent node. This would complicate algorithm design and would significantly increase the complexity of the search process. In contrast the cost-effectiveness principle is a simple and yet effective way to deal with the issue.

Acknowledgements Research on this work was supported by Hong Kong Research Grants Council GRF Grant #622408, the National Basic Research Program of China (aka the 973 Program) under project 2011CB505101 and the Shenzhen New Industry Development Fund #CXB201005250021A.

References

1. Zhang, N., Kočka, T.: Efficient learning of hierarchical latent class models. In: Proc. of the 16th IEEE International Conference on Tools with Artificial Intelligence. (2004)
2. Chen, T., Zhang, N., Wang, Y.: Efficient model evaluation in the search-based approach to latent structure discovery. In: Proc. of 4th European Workshop on Probabilistic Graphical Model. (2008)
3. Zhang, N.: Hierarchical latent class models for cluster analysis. J. Mach. Learn. Res. **5** (2004)
4. Lazarsfeld, P., Henry, N.: Latent structure analysis. Houghton Mifflin, Boston (1968)
5. Elidan, G., Friedman, N.: Learning hidden variable networks: the information bottleneck approach. J. Mach. Learn. Res. **6** (2005)
6. Zhang, N.: Discovery of latent structures: Experience with the coil challenge 2000 data set. ICCS (2007)
7. Zhang, N., Yuan, S., Chen, T., Wang, Y.: Latent tree models and diagnosis in traditional chinese medicine. Artificial Intelligence in Medicine (42) (2008)
8. Pearl, J.: Probabilistic reasoning in intellegent systems. Morgan Kaufmann, San Mateo (1988)
9. Wang, Y., Zhang, N.L., Chen, T.: Latent tree models and approximate inference in bayesian networks. Journal of Artificial Intelligence Research **32**(879-900) (2008)
10. Geiger, D., Heckerman, D., Meek, C.: Asymptotic model selection for directed networks with hidden variables. In: Proc. of the 12th Conference on Uncertainties in Artificial Intelligence. (1996)
11. Chickering, D.M.: Optimal structure identification with greedy search. J. Mach. Learn. Res. **3** (2002)
12. Chen, T.: Search-based learning of latent tree models. PhD dissertation, The Hong Kong University of Science and Technology, Department of Computer Science and Engineering (2009)