Quartet-Based Learning of Hierarchical Latent Class Models: Discovery of Shallow Latent Variables

Tao Chen and Nevin L. Zhang
Department of Computer Science
The Hong Kong University of Science & Technology
Hong Kong, China
{*csct,lzhang*}*@cs.ust.hk*

Abstract

Hierarchical latent class (HLC) models are tree-structured Bayesian networks where leaf nodes are observed while internal nodes are hidden. The currently most efficient algorithm for learning HLC models can deal with only a few dozen observed variables. While this is sufficient for some applications, more efficient algorithms are needed for domains with, e.g., hundreds of observed variables. With this demand in mind, we explore quartet-based methods. The basic idea comes from phylogenetic tree reconstruction: One first learn submodels for quartets — groups of four observed variables— and then derive an overall model from those quartet submodels. As the first step in the new direction, this paper assumes that there is a way to find the "true" submodel for any quartet and investigate how to identify shallow latent variables efficiently by using the minimum number of quartet submodels. By shallow latent variables, we mean latent variables that are connected to at least one observed variable.

1 Introduction

Hierarchical latent class (HLC) models (Zhang 2004) are tree-structured Bayesian networks where variables at leaf nodes are observed and hence are called *manifest variables*, while variables at internal nodes are hidden and hence are called *latent variables*. In this paper, we will use the terms "nodes" and "variables" interchangeably. HLC models were first identified by Pearl (1988) as a potentially useful class of models and were first systematically studied by Zhang (2004) as a framework to alleviate the disadvantages of LC models for clustering. As a tool for cluster analysis, HLC Models produce more meaningful clusters than latent class models and they allow multi-way clustering at the same time. As a tool for probabilistic modeling, they can model high-order interactions among observed variables and help one to reveal interesting latent structures behind data. They also facilitate unsupervised profiling.

Several algorithms for learning HLC models have been proposed. Among them, the heuristic single hill-climbing (HSHC) algorithm developed by Zhang and Kočka (2004) is by far the most efficient. HSHC has been used to successfully analyze the CoIL Challenge 2000 data set (van der Putten and van Someren 2004), which consists of 42 manifest variables and 5,822 records, and a data set about traditional Chinese medicine (TCM), which consists of 35 manifest variables and 2,600 records.

There are infinitely many possible HLC models for a given set of manifest variables. Zhang (2004) has argued that it suffices to consider what are called regular HLC models. The number of regular models for a given set of manifest variables is finite. However, it is super-exponential in the number of manifest variables. This makes it computationally challenging to learn HLC models. HSHC took 98 hours to analyze the aforementioned TCM data set on a top-end PC, and 121 hours to analyze the CoIL Challenge 2000 data set. It is clear that HSHC will not be able to analyze data sets with hundreds of variables. As a matter of fact, the TCM data set originally consists of some 70 variables. Only 35 were selected for analysis out of the consideration of computational complexity.

Aimed at developing algorithms more efficient than currently available, we explore quartet-based methods. The basic idea is to: (1) construct submodels for some quartets, and (2) make use of those quartet submodels when inferring an overall model. The quartet submodels contain information about the overall model. For example, if two manifest variables do not share a common latent parent in one quartet submodel, then they should not share a common latent parent in the overall model.

This paper is the first step in the new direction. A latent variable in an HLC model is *shallow* if it is connected to at least one manifest variable. We assume that there is a way to find the true (with respect to the generative model) submodel for any quartet and study the following problem:

> How to identify all the shallow latent variables by using the minimum number of quartet submodels?

We show that the shallow latent variables can be identified in $O(kn^2)$ time using $O(kn^2)$ quartet submodels, where n is the number of manifest variables and k is the number of shallow latent variables.

HSHC can be used to construct quartet submodels. There is no guarantee that HSHC can always find true quartet submodels, especially when the sample size is small. In future work, we will study the problem of identifying shallow latent variables without assuming the ability to always construct true quartet submodels; and we will study the problem of discovering "deep" latent variables.

2 Related work

Our work is related to work in two other research areas. The first is research on the latent variable graphs (LVGs) (Spirtes *et al.* 2000). An LVG is a two-layered Bayesian network where nodes on the upper level are latent while those on the lower level are observed. There are arcs among latent nodes, arcs among observed nodes, and arcs from latent nodes to observed nodes. But there are not any arcs from observed nodes to latent nodes.

Linear LVGs (LLVGs) are LVGs where all the variables are continuous, and each observed variable depends linearly on its parents via a regression equation. Silva *et al.* (2003) studied the problem of identifying the latent nodes of LLVGs. A two stage approach is proposed. In the first stage, they start with the complete graph over the observed nodes, and delete edges from it based on Tetrad constraints (Spirtes *et al.* 2000). In the second stage, they identify cliques in the graph, and introduce a latent node for each clique. The complexity of the first stage is $O(n^6)$, where n is the number of observed variables, while that of the second stage is exponential in the size of the largest clique. To validate their approach, Silva *et al.* (2003) conducted experiments that involve a few to about one dozen observed variables.

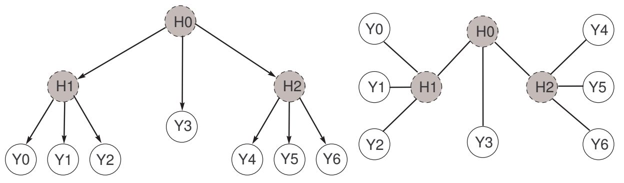

Figure 1: An example HLC model and the corresponding unrooted HLC model. The H_i's are latent variables and the Y_j's are manifest variables.

In comparison with work on LLVGs, phylogenetic tree (PT) reconstruction is more closely related to our work. The probabilistic models of phylogenetic trees (Durbin *et al.* 1998) can be viewed as special HLC models where (1) each latent node has exactly two children; (2) all variables have four states, namely A, C, G, and T; and (3) the conditional probability table of each variable has only one parameter, i.e. edge length.

The most prominent algorithm for PT reconstruction is neighbor-joining(NJ) (Saitou and Nei 1987). It stands out in terms of both efficiency and model quality (St. John *et al.* 2003). However, it cannot be generalized to HLC models because it depends on a concept of distance. The maximum-likelihood method performs well in terms of model quality, but it does not scale up well. The SEMPHY method by Friedman *et al.* (2002) partially alleviates the problem using the idea of structural EM. HSHC can be viewed as the counter-part of SEMPHY for HLC models. Quartet-based methods are another class of methods proposed to address the scalability of the maximum-likelihood method. They assume that all $O(n^4)$ quartet submodels have been precomputed and aim at finding the overall tree that matches the structures of the most number of submodels.

Quartet-based methods for PT reconstruction cannot be used for learning HLC models without modification. There are two reasons. First, when there are hundreds of variables, it is computationally expensive to learn all $O(n^4)$ submodels. Second, we need to take into consideration the effect of "erroneous" quartet submodels, i.e., submodels learned while inconsistent with the true overall model. These are two major research issues in developing quartet-based methods for HLC models. In current stage we assume that all submodels learned are true and then focus on the first issue.

3 HLC Models and Problem Statement

Figure 1 shows an example HLC model (left diagram). Zhang (2004) has argued that it is impossible to de-

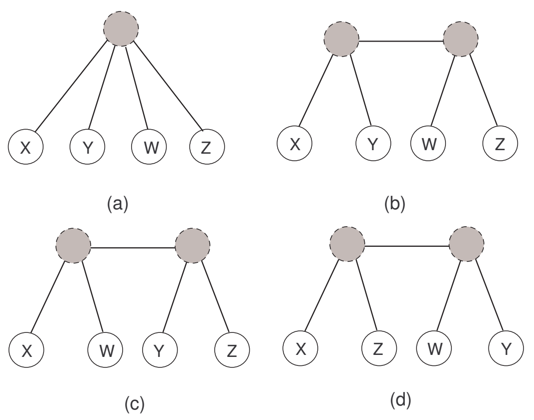

Figure 2: Four possible regular HLC structures for the case of four manifest variables. The fork in (a) is denoted by $[XYWZ]$, the dogbones in (b), (c), and (d) respectively by $[XY|WZ]$, $[XW|YZ]$, and $[XZ|WY]$.

termine, from data, the orientations of edges in an HLC model. One can learn only *unrooted HLC models*, which are HLC models with all directions on the edges dropped. [1] The picture on the right hand side of Figure 1 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. From now on when we speak of HLC models we always mean unrooted HLC models unless it is explicitly stated otherwise.

In this paper, we will use the term *HLC structure* to refer to the set of nodes in an HLC model and the connections among them. An HLC structure is *regular* if it does not contain latent nodes of degree 2. Starting from an irregular HLC structure, we can obtain a regular structure by connecting the two neighbors of each latent node of degree 2 and then remove that node. This process is known as *regularization.*

Let $\mathbf{Y}$ the set of manifest variables of an HLC model/structure. Then we say that the model/structure is a *model/structure on* $\mathbf{Y}$. There are four possible regular HLC structures on a set of four manifest variables $\{X, Y, W, Z\}$. The structure shown in Figure 2.(a) is a *fork*, which contains only one latent node. The structures shown in Figure 2.(b),(c) and (d) are *dogbones*, which consist of two latent nodes.

Now let $\mathcal{S}$ be a general regular HLC structure and let $\mathbf{Q}$=$\{X, Y, W, Z\}$ be a set of four manifest variables in $\mathcal{S}$. We will refer to $\mathbf{Q}$ as an *quartet.* The *restriction* of $\mathcal{S}$ on $\mathbf{Q}$ is obtained from $\mathcal{S}$ by deleting all the nodes and edges not in the shortest paths between any pair of variables in $\mathbf{Q}$. Applying regularization to the resulting HLC structure, we obtain the *quartet sub-structure* for $\mathbf{Q}$, which will be denoted by $\mathcal{S}|_{\mathbf{Q}}$. It is clear that $\mathcal{S}|_{\mathbf{Q}}$ is a regular HLC structure on $\mathbf{Q}$. Hence it is either the fork $[XYWZ]$, or one of the doghones $[XY|WZ]$, $[XW|YZ]$, and $[XZ|WY]$.

Consider the HLC structure in Figure 1 (right). The quartet sub-structure for $\{Y_0, Y_1, Y_2, Y_3\}$ is the fork $[Y_0Y_1Y_2Y_3]$, while that for $\{Y_0, Y_1, Y_3, Y_4\}$ is the dogbone $[Y_0Y_1|Y_3Y_4]$, and that for $\{Y_0, Y_3, Y_4, Y_5\}$ is the dogbone $[Y_0Y_3|Y_4Y_5]$.

In an HLC model/structure, a *shallow* latent node is one that is adjacent to at least one manifest node. Two manifest nodes are *siblings* if they are adjacent to the same (shallow) latent node. All manifest nodes adjacent to a given shallow latent node constitute a *sibling cluster.* In the HLC structure shown on the right of Figure 1, there are 3 shallow latent variables H_0, H_1, and H_2, and there are 3 sibling clusters, namely $\{Y_3\}$, $\{Y_0, Y_1, Y_2\}$, and $\{Y_4, Y_5, Y_6\}$.

Let $\mathcal{M}$ be an HLC model with a regular structure $\mathcal{S}$. Suppose that $\mathbf{D}$ is a collection of i.i.d samples drawn from $\mathcal{M}$. Each record in $\mathbf{D}$ contains values for the manifest variables, but not for the latent variables. Let $\texttt{QML}(\mathbf{D}, \mathbf{Q})$ be a routine that takes data $\mathbf{D}$ and a quartet $\mathbf{Q}$ as inputs, and returns an HLC structure on the quartet $\mathbf{Q}$.[2] In this paper, we make the following assumption that relates QML to $\mathcal{S}$:

Assumption 1 *For any quartet* $\mathbf{Q}$, $\texttt{QML}(\mathbf{D}, \mathbf{Q})$ *returns the quartet sub-structure* $S|_{\mathbf{Q}}$.

In other words, we assume that QML always finds true quartet sub-structures.

Under Assumption 1, we investigate how to identify all the shallow latent nodes in $\mathcal{S}$ using QML. A shallow latent node is defined by its relationship with its manifest neighbors. Hence to identify all the shallow latent nodes means to identify all the sibling clusters in $\mathcal{S}$. We present an algorithm for identifying the sibling clusters in $\mathcal{S}$ that calls the routine QML $O(kn^2)$ times, where k is the number of shallow latent nodes and n is the number of manifest variables.

4 Technical Preparations

Let $\mathcal{S}$ be a regular HLC structure. In this section, we prove several results on the relationships among sib-

[1]We nonetheless still begin with HLC models because they are easier to understand for some readers and edge orientation can sometimes be decided through model interpretation.

[2]QML stands for Quartet Model Learner. One would expect a quartet model learner to return a model. In this paper, we use only the structure of the model.

ling clusters in $\mathcal{S}$, quartet sub-structures of $\mathcal{S}$, and topological properties of $\mathcal{S}$ itself. First of all, this lemma follows readily from the definition of quartet sub-structures.

Lemma 1 *Let* $\mathbf{Q}$=$\{X, Y, W, Z\}$ *be a quartet in a regular HLC structure* $\mathcal{S}$.

1. *The quartet sub-structure* $\mathcal{S}|_\mathbf{Q}$ *is the dogbone* $[XY|WZ]$ *if and only if, in* $\mathcal{S}$*, the shortest path joining* X *and* Y *is disjoint from that joining* W *and* Z.

2. *The quartet sub-structure* $\mathcal{S}|_\mathbf{Q}$ *is the fork* $[XYWZ]$ *if and only if, in* $\mathcal{S}$*, all the shortest paths joining pairs of nodes in* $\mathbf{Q}$ *intersect at one common latent node.* □

This proposition relates sibling clusters to quartet sub-structures.

Proposition 1 *Let* $\mathbf{Q}$=$\{X, Y, W, Z\}$ *be a quartet in a regular HLC structure* $\mathcal{S}$*. If the quartet sub-structure* $\mathcal{S}|_\mathbf{Q}$ *is the dogbone* $[XY|WZ]$*, then* X/Y *cannot be from the same sibling cluster as* W/Z.

Proof: We know from Lemma 1 (1) that the shortest path joining X and Y is disjoint from the path joining W and Z. This implies that the latent neighbor of X/Y is different from that of W/Z, and hence they cannot be in the same sibling cluster. □

Let e be an edge in an HLC structure $\mathcal{S}$. To *split* $\mathcal{S}$ at edge e means to remove the edge. This renders the structure disconnected and results in two connected components. So, the structure $\mathcal{S}$ is split at e into the two connected components.

In an HLC structure, an *internal edge* is one that connects two latent nodes. This lemma follows readily from the definition of regularity.

Lemma 2 *Consider splitting an HLC structure* $\mathcal{S}$ *at an edge* e*. If* $\mathcal{S}$ *is regular and* e *is an internal edge, then each of the two resultant connected components contain at least 2 manifest nodes.* □

This proposition relates quartet sub-structures to split.

Proposition 2 *Suppose we have split a regular HLC structure* $\mathcal{S}$ *at an internal edge into two connected component* $\mathcal{S}_1$ *and* $\mathcal{S}_2$*. Let* X *and* Y *be two manifest nodes from* $\mathcal{S}_1$*, while* W *and* Z *be two manifest nodes from* $\mathcal{S}_2$*. Then the quartet sub-structure* $\mathcal{S}|_\mathbf{Q}$ *for the quartet* $\mathbf{Q}$=$\{X, Y, W, Z\}$ *is the dogbone* $[XY|WZ]$.

Proof: $\mathcal{S}_1$ is a connected tree. Hence there must be at least one path within $\mathcal{S}_1$ that joins X and Y. Let P_1 be the shortest among all such paths. Then P_1 must also be the shortest path in $\mathcal{S}$ that joins X and Y. In other words, the shortest path in $\mathcal{S}$ that joins X and Y lies within $\mathcal{S}_1$. Similarly, the shortest path in $\mathcal{S}$ that joins W and Z lies within $\mathcal{S}_2$. Those two shortest paths are obviously disjoint from each other. The proposition therefore follows from Lemma 1 (1). □

5 Discovering Shallow Latent Variables: The Principle

In order to compute the sibling cluster that contains a given manifest node X, we need to answer this question for each of the other manifest nodes Z: Which quartet sub-structures do we need in order to determine whether Z is a sibling of X? The following theorem provides one answer.

Theorem 1 *Suppose* X*,* W*, and* Z *be three different manifest nodes in a regular HLC structure* $\mathcal{S}$*. Then* X *is not a sibling of* W/Z *if and only if there exists another manifest node* Y *such that*

1. *The quartet sub-structure* $\mathcal{S}|_\mathbf{Q}$ *for the quartet* $\mathbf{Q}$=$\{X, Y, W, Z\}$ *is a dogbone, and*

2. X *is not a sibling of* W/Z *in the dogbone.*

Proof: We will prove the theorem only for Z. If it is true for Z, then it is also true for W by symmetry.

The if-part is true because of Proposition 1.

To prove the only-if part, consider the shortest path joining X and Z in $\mathcal{S}$. Because X and Z are not siblings, the path must contain at least two latent nodes, and hence an internal edge. Let e be an internal edge on the path. Split $\mathcal{S}$ at e into two connected components $\mathcal{S}_1$ and $\mathcal{S}_2$. Without losing generality, suppose X is in $\mathcal{S}_1$ and Z is in $\mathcal{S}_2$, and suppose W is in $\mathcal{S}_2$. According to Lemma 2, there are at least two manifest nodes $\mathcal{S}_1$. Let Y be a manifest node in $\mathcal{S}_1$ other than X. By Proposition 2, we know the quartet sub-structure $\mathcal{S}|_\mathbf{Q}$ for the quartet $\mathbf{Q}$=$\{X, Y, W, Z\}$ is the dogbone $[XY|WZ]$. In this dogbone, X and Z are not siblings. The theorem is therefore proved. □

6 Discovering Shallow Latent Variables: The Algorithm

Assumption 1 states that we can obtain, from the data set $\mathbf{D}$, quartet sub-structures of the generative model $\mathcal{M}$ using the routine `QML`. Theorem 1 tells us which quartet sub-structures we need in order to find all the siblings of a manifest node X of $\mathcal{M}$. Putting these two

```
Procedure FindSC($\mathbf{D}$):
1. Let $\mathbf{Y}$ be the set of variables in $\mathbf{D}$.
2. Let $\mathcal{L} \leftarrow \emptyset$. Pick $X \in \mathbf{Y}$.
3. Loop for ever,
4.     $\mathbf{C} \leftarrow$ FindOneSC($X, \cup\mathcal{L}, \mathbf{Y}, \mathbf{D}$).
5.     $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{C}\}$.
6.     if $\mathbf{Y} \setminus (\cup\mathcal{L}) \neq \emptyset$, pick $X \in \mathbf{Y} \setminus (\cup\mathcal{L})$.
7.     else return $\mathcal{L}$.
```

```
Procedure FindOneSC($X, \mathbf{N}, \mathbf{Y}, \mathbf{D}$):
1. Pick $W \in \mathbf{Y}\backslash\{X\}$, $i \leftarrow 1$.
2. For each $Z \in \mathbf{Y}\backslash\{X, W\}$,
3.     if $Z \in \mathbf{N}$, continue.
4.     For each $Y \in \mathbf{Y}\backslash\{X, W, Z\}$,
5.         if QML($\mathbf{D}, \{X, Y, W, Z\}$) $= [XW|YZ]$,
6.             $\mathbf{N} \leftarrow \mathbf{N} \cup \{Y, Z\}$,
7.             if ($i > 1$) break.
8.         else if QML($\mathbf{D}, \{X, Y, W, Z\}$) $= [XY|WZ]$,
9.             $\mathbf{N} \leftarrow \mathbf{N} \cup \{W, Z\}$, break.
10.        else if QML($\mathbf{D}, \{X, Y, W, Z\}$) $= [XZ|WY]$,
11.            $\mathbf{N} \leftarrow \mathbf{N} \cup \{W, Y\}$. EndFor
12.    $i \leftarrow i + 1$. EndFor
13. Return $\mathbf{Y} \setminus \mathbf{N}$.
```

Figure 3: $\mathbf{D}$ is an i.i.d data set sampled from an HLC model $\mathcal{M}$ that has a regular structure. Under Assumption 1, the Algorithm FindSC finds all the sibling clusters in $\mathcal{M}$.

together, we get an algorithm, named FindOneSC and shown in Figure 3, for computing the sibling cluster in $\mathcal{M}$ that contains X.

In addition to the manifest variable X, the algorithm has three other inputs: $\mathbf{D}$ is the data set, $\mathbf{Y}$ is the set of variables in $\mathbf{D}$, or equivalently the set of manifest variables in $\mathcal{M}$. $\mathbf{N}$ is a set of manifest nodes known not to be siblings of X. It is empty when FindOneSC is called for the first time (from FindSC).

The algorithm checks many quartets. The variable X appears in all of them. In other words, X is a "standing member" of all quartets. At Line 1, another standing member W is picked. So, all the quartets will have two standing members, X and W.

There are two for-loops. In the outer for-loop, the algorithm examines each manifest node Z one by one and determines whether it is a sibling of X. To do so, it first checks whether Z is already known not to be a sibling of X (Line 3). If this is the case, it skips the rest of the for-loop and moves on to the next manifest node.

Otherwise, it considers, in the inner for-loop, each quartet that consists of X, W, Z, and a fourth manifest node Y. It computes the sub-structure for the quartet using the routine QML. If the quartet sub-structure is a dogbone and X and Z are not siblings of the dogbone, then, by Assumption 1 and Theorem 1, Z is not a sibling of X in $\mathcal{M}$ and is hence added to the set $\mathbf{N}$ (Lines 6, 9).

If the quartet sub-structure turns out to be the dogbone $[XW|YZ]$, we know not only that Z is not a sibling of X, but also that Y is not a sibling of X. For this reason, Y is also added to $\mathbf{N}$ at Line 6. For the same reason, W is added to $\mathbf{N}$ at Line 9 and both W and Y are added to $\mathbf{N}$ at Line 11.

There is the question of whether W is a sibling of X. To determine this, we let the inner for-loop continue, for the first Z examined, even after we already learn that Z is not a sibling of X (Line 7). According to Theorem 1, if W is not a sibling of X, either the condition at Line 8 or that at Line 10 will be eventually satisfied, and hence W will be eventually added to $\mathbf{N}$.

After the outer for-loop, $\mathbf{N}$ contains all non-siblings of X. Hence $\mathbf{Y}\backslash\mathbf{N}$ contains X and all its siblings. It is the sibling cluster that contains X.

Note that FindOneSC uses only dogbones. It does not uses forks at all.

The procedure FindSC($\mathbf{D}$) shown at the top of Figure 3 returns the list of all the sibling clusters of $\mathcal{M}$. To begin with, it arbitrarily pick a variable X (Line 2), and computes the sibling cluster that contains X by calling FindOneSC (Line 4). Then it picks another variable X outside the cluster (Line 6) and repeats the process, and so on. It terminates when there are no nodes outside the clusters that have already been identified (Lines 6, 7).

In the algorithm, $\mathcal{L}$ stands for the list of clusters, and $\cup\mathcal{L}$ is the union of all the clusters in $\mathcal{L}$. When FindOneSC is first called, its second argument $\cup\mathcal{L}$ is the empty set, indicating that we do not yet know any non-siblings of X. In subsequent calls, however, $\cup\mathcal{L}$ is no longer empty. It consists of all nodes in all the clusters that have already been identified. Since X is outside of those clusters, all nodes in $\cup\mathcal{L}$ are non-siblings of X.

Putting all the arguments presented above together, we have proved the following theorem:

Theorem 2 *Let $\mathcal{M}$ be an HLC model with a regular structure. Suppose that $\mathbf{D}$ is a collection of i.i.d samples drawn from $\mathcal{M}$. Let* QML($\mathbf{D}, \mathbf{Q}$) *be a routine that takes the data $\mathbf{D}$ and a quartet $\mathbf{Q}$ as inputs, and returns an HLC structure on the quartet $\mathbf{Q}$. If* QML *satisfies Assumption 1, then* FindSC($\mathcal{D}$) *returns the list of all sibling clusters in $\mathcal{M}$.* $\square$

Let n and k be the numbers of manifest nodes and sibling clusters respectively. It is clear that the number of calls that FindOneSC makes to QML is no more than $(n-2)(n-3)$.[3] In practice, the number could be much lower because of Lines 3, 7, and 9. The total number of calls that FindSC makes to QML is no more than $k(n-2)(n-3)$, which simplifies to $O(kn^2)$.

7 An Example

We next illustrate the algorithm FindSC using the HLC model shown on the right hand side of Figure 1.

Suppose Y_0 is first picked to be X at Line 2 of FindSC and hence FindOneSC is invoked at Line 4 to compute the sibling cluster that contains Y_0. Within FindOneSC, suppose Y_1 is picked to be W at Line 1.

- The outer for-loop first determines whether Y_2 is a sibling of Y_0. To do so, it computes, using QML, the quartet substructures for quartets of the form $\{Y_0, Y_1, Y_2, Y\}$, where Y runs from Y_3 to Y_6. It turns out that all those quartet substructures are forks. Hence nothing is added to the set $\mathbf{N}$.
- The outer for-loop then determines whether Y_3 is a sibling of Y_0. To do so, it computes the quartet substructures for quartets of the form $\{Y_0, Y_1, Y_3, Y\}$, where Y runs from Y_2, Y_4, Y_5, to Y_6. It turns out that the quartet substructure for $\{Y_0, Y_1, Y_3, Y_4\}$ is the dogbone $[Y_0Y_1|Y_3Y_4]$. Hence Y_3 is not a sibling of Y_0, and is added to the set $\mathbf{N}$. As a by-product, we also know that Y_4 is not a sibling of Y_0, and it is also added to $\mathbf{N}$. Note that in the case, Y_5 and Y_6 are not examined in the inner for-loop.
- The outer for-loop continues to determine whether Y_5 and Y_6 are siblings of Y_0. It turns out that those nodes are not siblings of Y_0, and hence are added to the set $\mathbf{N}$.
- Finally, FindOneSC returns the set of nodes not in $\mathbf{N} = \{Y_3, Y_4, Y_5, Y_6\}$. Hence, what it returns is $\{Y_0, Y_1, Y_2\}$, which is indeed the sibling clusters that contains Y_0.

The control is then passed back to FindSC. Suppose Y_3 is picked to be X at Line 6. Then FindOneSC is called at Line 4 to compute the sibling cluster that contains Y_3. Note that the second argument now is $\{Y_0, Y_1, Y_2\}$ instead of the empty set. Within FindOneSC, the initial value of $\mathbf{N}$ is $\{Y_0, Y_1, Y_2\}$. Suppose Y_4 is picked to be W at Line 1.

- The outer for-loop first examines Y_0, Y_1, and Y_2. Since those nodes are in $\mathbf{N}$ already, nothing is done.
- The outer for-loop then determines whether Y_5 is a sibling of Y_3. To do so, it computes the quartet substructures for quartets of the form $\{Y_3, Y_4, Y_5, Y\}$, where Y runs from Y_0, Y_1, Y_2, to Y_6.
- The quartet substructure for $\{Y_3, Y_4, Y_5, Y_0\}$ is the dogbone $[Y_0Y_3|Y_4Y_5]$. Hence Y_5 is not a sibling of Y_3, and is added to the set $\mathbf{N}$. As a by-product, we also know that Y_4 is not a sibling of Y_3, and it is also added to $\mathbf{N}$. Note that in the case, Y_1, Y_2, and Y_6 are not examined in the inner for-loop.
- The outer for-loop continues to determine whether Y_6 are siblings of Y_3. It turns out that this node is not a sibling of Y_3, and hence is added to the set $\mathbf{N}$.
- Finally, FindOneSC returns the set of nodes not in $\mathbf{N} = \{Y_0, Y_1, Y_2, Y_4, Y_5, Y_6\}$. Hence, what it returns is $\{Y_3\}$, which is indeed the sibling clusters that contains Y_3.

We leave it to the reader to complete the rest of this example.

8 Conclusions

How to learn HLC models efficiently is an interesting research problem. We are investigating quartet-based approaches. This paper reports our first result in this new direction. We assume that there is a routine that can correctly learn the structures of the quartet submodels of the generative model, and we show how the shallow latent nodes of the generative model can be identified by calling the routine $O(kn^2)$ times. In the immediate future, we will relax the assumption, and will study how to discover "deep" latent nodes.

Acknowledgements

Research on this work was supported by Hong Kong Grants Council Grant #622105 and HKUST Grant DAG04/05.EG11. We thank Haipeng Guo and Yi Wang for valuable discussions.

[3]For readability, we have QML at Lines 5, 8, and 10. This does not mean QML is called three times in each pass through the inner for-loop. In actual implementation, one needs to call it only once. There are three cases depends on the type of the structure that it returns.

References

[1] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge University Press.

[2] Friedman, N., Ninio, M., Pe'er, I., and Pupko, T. (2002).A structural EM algorithm for phylogenetic inference. *Journal of Computational Biology*, 9:331-353.

[3] Pearl, J.(1988). *Probabilistic Reasoning in Intelligent Systems: Netwroks of Plausible Inference.* Morgan Kaufmann Publishers, Palp Alto.

[4] Saitou, N. and Nei, M.(1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution.* 4: 406-425.

[5] Silva, R., Scheines, R., Glymour, C. and Spirtes, P.(2003). Learning measurement models for unobserved variables. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03).*

[6] Spirtes, P., Glymour, C. and Scheines, R.(2000). *Causation, Prediction and Search.* Cambridge University Press.

[7] St. John, K., Warnow, T., Moret, B.M.E. and Vawter, L.(2003) Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. *Journal of Algorithms* 48(1): 173-193.

[8] van der Putten, P. and van Someren, M. (2004). A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. *Machine Learning*, Kluwer Academic Publishers, 57, 177-195.

[9] Zhang, N.L.(2004). Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research.* **5**, 697-723.

[10] Zhang, N.L. and Kočka, T. (2004). Efficient learning of hierarchical latent class models. In *Proc. of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-2004).*