Consistent and Efficient Reconstruction of Latent Tree Models

Myung Jin Choi, Vincent Y. F. Tan, Animashree Anandkumar, and Alan S. Willsky
Stochastic Systems Group,

Laboratory for Information and Decision Systems,
Massachusetts Institute of Technology.
Email: {myungjin,vtan,animakum,willsky}@mit.edu

***Abstract*—We study the problem of learning a latent tree graphical model where samples are available only from a subset of variables. We propose two consistent and computationally efficient algorithms for learning *minimal* latent trees, that is, trees without any redundant hidden nodes. Our first algorithm, *recursive grouping*, builds the latent tree recursively by identifying sibling groups. Our second and main algorithm, *CLGrouping*, starts with a pre-processing procedure in which a tree over the observed variables is constructed. This global step guides subsequent recursive grouping (or other latent-tree learning procedures) on much smaller subsets of variables. This results in more accurate and efficient learning of latent trees. We compare the proposed algorithms to other methods by performing extensive numerical experiments on various latent tree graphical models such as hidden Markov models and star graphs.**

I. Introduction

Latent tree models are tree-structured graphical models in which samples are only available from a subset of nodes known as the *observed nodes*. These samples are used for reconstructing the structure and the parameters of the tree. This problem has many applications, e.g., the inference of a phylogenetic or evolutionary tree from genetic data of extant species [5], inference of the network routing tree from end-to-end delay measurements at a subset of nodes [1] and building a hierarchical contextual tree to predict object co-occurrences in images [14]. Reconstructing general latent tree models is in general, formidable as finding the maximum-likelihood (ML) estimate of a latent tree is NP-hard [17].

There are three main contributions in this paper. Firstly, we propose two novel algorithms for general latent tree reconstruction under a unified approach for Gaussian and discrete distributions. Secondly, we provide provable guarantees for the proposed algorithms, viz., consistency, computational and sample complexities. Thirdly, our experimental results demonstrate the superiority of our approach for a wide variety of models including the hidden Markov model (HMM), the star and the complete tree models.[1]

As [15] points out, the identification of latent tree models has some built-in ambiguity, as there is an entire equivalence class of models in the sense that when all latent variables are marginalized out, each model in this class yields the same joint distribution over the observed variables. For example, we can take any latent tree and add another hidden variable as a leaf node connected to only one other (hidden or observed) node. Hence, much as one finds in fields such as state space dynamic systems (e.g., [13]), there is a notion of minimality that is required here, and our results are stated in terms of consistent learning of such minimal latent models.

Our first algorithm *recursive grouping* constructs the latent tree in a bottom-up fashion, by using *information distances* to add hidden nodes as neighbors to the existing nodes recursively. Our second algorithm *CLGrouping* (which stands for Chow-Liu grouping) includes a preprocessing step to group subsets of observed nodes, predicted to be "close" to one another in the latent tree, and then executes the recursive grouping algorithm on each of the smaller groups. When all the nodes are observed, the CLGrouping algorithm outputs the ML-estimate of the tree model [4].

The performance of our algorithms is naturally influenced by the structure and parameters of the underlying latent tree model. CLGrouping is significantly faster and more accurate than the recursive grouping algorithm, when the group sizes at the end of its pre-processing step are small. This occurs in the case of tree models with small maximum degree. In fact, for a fixed number of observed nodes, CLGrouping has optimal accuracy and running time for 3-HMM while the worst-case is the star model, and this is corroborated by our experiments.

We now provide an overview of the related work. Reconstructing latent tree models has been studied extensively in the context of learning phylogenetic trees. Efficient algorithms with provable performance guarantees are available (see [9], [5], [7]). However, the works mostly assume the trees are Cayley trees with only the leaves being observed. The most popular algorithm for constructing phylogenetic trees is the *neighbor-joining (NJ) method* by [18]. The algorithm proceeds by recursively pairing two nodes that are the closest neighbors in the true latent tree and introducing a hidden node as the parent of the two nodes. Many works also propose reconstructing latent trees using expectation maximization (EM) [8], [20]. However, these approaches suffer from the possibility of attaining local optima and thus no consistency guarantees can be provided.

[1] A r-*Cayley tree* is one where each non-leaf has constant degree r. A r-HMM is a r-Cayley tree in which non-leaf nodes form a chain. A r-*complete* tree is a r-Cayley tree where all leaves have the same graph distance from the root.

II. System Model and Preliminaries

Let $G = (W, E)$ be an undirected graph with node set $W = \{1, \dots, M\}$ and edge set $E \subset \binom{W}{2}$ and let $\mathrm{nbd}(i; G)$ and $\mathrm{nbd}[i; G]$ be the set of neighbors of node i and the *closed neighborhood* of i respectively, i.e., $\mathrm{nbd}[i; G] := \mathrm{nbd}(i; G) \cup \{i\}$. If an undirected graph does not include any loops, it is called a *tree*. For a tree $T = (W, E)$, the set of leaf nodes (nodes with degree 1), the maximum degree and the diameter, are denoted by $\mathrm{Leaf}(T)$ and $\Delta(T)$, and $\mathrm{diam}(T)$ respectively. The *path* between two nodes i and j in a tree $T = (W, E)$ is the set of edges connecting i and j and is denoted as $\mathrm{Path}((i, j); E)$. The *distance* between any two nodes i and j is the number of edges in $\mathrm{Path}((i, j); E)$. In an undirected tree, we can choose a *root node* arbitrarily, and define the parent-child relationships with respect to the root: for a pair neighboring nodes i and j, if i is closer to the root than j is, then i is called the *parent* of j, and j is called the *child* of i. Note that the root node does not have any parent, and for all other nodes in the tree, there exists exactly one parent. A set of nodes that share the same parent is called a *sibling* group. A *family* is the union of the siblings and the associated parent.

We consider latent trees with node set $W := V \cup H$, the union of V (with $m = |V|$), the set of observed nodes, and H, the set of latent (or hidden) nodes. The *effective depth* $\delta(T; V)$ (wrt V) is maximum distance of a hidden node to its closest observed node, i.e., $\delta(T; V) := \max_{i \in H} \min_{j \in V} |\mathrm{Path}((i, j); T)|$.

A. Latent Tree Model

An *undirected graphical model* [12] with respect to graph G corresponds to a family of multivariate probability distributions that factorize according to G. More precisely, a random vector $\mathbf{X} = (X_1, \dots, X_M)$, where each random variable X_i (i.e., variable at node i) takes values in the alphabet $\mathcal{X}$ is said to be *Markov* on G if for every node i, X_i is conditionally independent of other variables given its neighbors, i.e, $p(x_i | x_{\mathrm{nbd}(i;G)}) = p(x_i | x_{\setminus i})$, where $x_{\setminus i}$ denotes the set of variables excluding x_i. Let $\mathbf{x}^n := \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$ be i.i.d. samples drawn from p, a graphical model Markov on $T_p = (W, E_p)$. In our setup, the learner has access to the values only on the node set V, and we denote this set of sub-vectors containing only the elements in V, as $\mathbf{x}_V^n := \{\mathbf{x}_V^{(1)}, \dots, \mathbf{x}_V^{(n)}\}$.

We consider both Gaussian ($\mathcal{X} = \mathbb{R}$) and discrete ($\mathcal{X} = \{1, \dots, K\}$) graphical tree models. For a Gaussian distribution with a covariance matrix $\mathbf{\Sigma}$, the correlation coefficients between variables X_i and X_j is given by $\rho_{ij} := \mathbf{\Sigma}_{ij} / \sqrt{\mathbf{\Sigma}_{ii} \mathbf{\Sigma}_{jj}}$. Let $d_{i,j} := -\log |\rho_{ij}|$ be the *information distance* between X_i and X_j. Intuitively, if the information distance $d_{i,j}$ is large, then X_i and X_j are weakly correlated and vice-versa. Given the information distances $d_{i,j}$ on edges $(i, j) \in E_p$ of tree T_p, we can find the information distance between any two nodes $k, l \in W$ using the Markov property:

$$d_{k,l} = \sum_{(i,j) \in \mathrm{Path}((k,l); E_p)} d_{i,j}. \qquad (1)$$

For discrete random variables, let $\mathbf{J}^{ij}$ denote the joint probability matrix between X_i and X_j (i.e., $J_{ab}^{ij} = p(x_i = a, x_j = b), a, b \in \mathcal{X}$). Also let $\mathbf{M}^i$ be the diagonal marginal probability matrix of X_i (i.e., $M_{aa}^i = p(x_i = a)$). We can define the information distance as $d_{i,j} := -\log \frac{|\det \mathbf{J}^{\mathbf{ij}}|}{\sqrt{\det \mathbf{M}^{\mathbf{i}} \det \mathbf{M}^{\mathbf{j}}}}$, and the Markov property in (1) holds [11]. In the rest of the paper, we map the parameters of Gaussian and discrete distributions to an information distance matrix $\mathbf{d} = [d_{i,j}]$ to unify the analyses for both cases. Given samples of observed variables $\mathbf{x}_V^n$, we compute the maximum-likelihood (ML) estimates of the parameters, and use those parameters to compute the ML estimates of information distances between observed variables $\hat{\mathbf{d}} = [\hat{d}_{i,j}]_{i,j \in V}$.

B. Minimal Tree-extensions

Our goal is to recover the graphical model p, i.e., the latent tree structure and its parameters, given n i.i.d. samples of the observed variables $\mathbf{x}_V^n$. However, in general, there can be multiple models generating the same observed statistics. We consider the class of tree models where it is feasible to recover the model uniquely and provide necessary conditions for identifiability.

Firstly, we limit ourselves to the scenario where *all* the random variables (both observed and latent) take values on a common alphabet $\mathcal{X}$. Thus, in the Gaussian case, each hidden and observed variable is a univariate Gaussian. In the discrete case, each variable takes on values in the same finite alphabet $\mathcal{X}$. Note that the model may not be identifiable if some of the hidden variables are allowed to have arbitrary alphabets. As an example, consider a discrete latent tree model with binary observed variables ($K = 2$). A latent tree with the simplest structure (fewest number of nodes) is a tree in which all m observed binary variables are connected to one hidden variable. If we allow the hidden variable to take on 2^m states, then the tree can describe all possible statistics among the m observed variables, i.e., the joint distribution p_V can be arbitrary.[2]

A probability distribution $p_V(\mathbf{x}_V)$ is said to be *tree-decomposable* if it is the marginal of a tree-structured distribution $p(\mathbf{x}_V, \mathbf{x}_H)$, and p is said to be a *tree-extension* of p_V [15]. A latent tree $T_p = (W, E_p)$ is said to have *redundant* hidden node $h \in H$ if we can remove h and the marginal on the set of visible nodes V remains as p_V. A latent tree without redundant nodes is said to be *minimal*, and the corresponding distribution p is said to be a *minimal tree-extension* of p_V. The following conditions ensure that a latent tree is minimal [15]:

(C1) Each hidden variable has at least three neighbors (which can be either hidden or observed). Note that this ensures that all leaf nodes are observed. (C2) Any two variables in the tree model are neither perfectly dependent nor independent, i.e., $0 < l \leq d_{i,j} \leq u < \infty$ for all $(i, j) \in E_p$.

We assume throughout the paper that (C2) is satisfied for all probability distributions. Let $\mathcal{T}_{\geq 3}$ be the set of (latent)

[2]This follows from a elementary parameter counting argument.

trees satisfying (C1). We refer to $\mathcal{T}_{\geq 3}$ as the set of *minimal (or identifiable) latent trees*. Using marginalization operations, any non-minimal latent tree distribution can be reduced to a minimal latent tree model.

Proposition 1. (Minimal Tree-extensions) *[15] (i) For each tree-decomposable distribution p_V, there exists a minimal tree extension p Markov on tree $T_p \in \mathcal{T}_{\geq 3}$, which is unique up to renaming of the variables or their values. (ii) For Gaussian and binary distributions, if p_V is know exactly, the minimal tree extension p can be recovered. (iii) The structure of T_p is uniquely determined by the information distance matrix.*

We now define the notion of consistency.

Definition 1. (Consistency) *A reconstruction algorithm $\mathcal{A}$, which is a map of observed samples $\mathbf{x}_V^n$ to a tree $\hat{T}^n$ and a tree-structured graphical model $\hat{p}^n$, is* structurally consistent *if there exists a graph homomorphism h such that*

$$\lim_{n\to\infty} \Pr(\{\mathbf{x}_V^n : h(\hat{T}^n) \neq T_p\}) = 0. \quad (2)$$

Furthermore, we say that $\mathcal{A}$ is risk consistent *if to every $\epsilon > 0$,*

$$\lim_{n\to\infty} \Pr(\{\mathbf{x}_V^n : D(p \,||\, \hat{p}^n) > \epsilon\}) = 0, \quad (3)$$

where $D(p \,||\, \hat{p}^n)$ is the KL-divergence between the true distribution p and the estimated distribution $\hat{p}^n$.

In order to prove structural consistency, we analyze the output of algorithms given the exact information distances between observed nodes $\mathbf{d}$. It suffices to show that the algorithms output the correct latent tree T_p, since the ML estimates of information distance converge (in probability) to the true information distances. Once we reconstruct a tree structure and the associated information distance matrix, it is straightforward to recover the parameters of Gaussian distributions, so our algorithms are also risk consistent for Gaussian tree models. However, for discrete models, information distances between observed nodes are, in general, not sufficient to recover the parameters [2]. Therefore, we only prove structural consistency for discrete latent tree models.[3] Due to space constraints, most of the proofs of the main results can be found in the longer version of the paper [3].

III. Recursive Grouping

The main idea behind our first algorithm for reconstructing latent trees is to identify nodes that belong to the same family and to learn the tree structure by grouping these observed nodes recursively in a bottom-up fashion.

A. Testing Node Relationships

Define $\Phi_{ijk} := d_{i,k} - d_{j,k}$ to be the difference between the two information distances $d_{i,k}$ and $d_{j,k}$. The following result shows that we can identify leaf nodes, their observed parents, and whether a pair of leaf nodes are siblings (i.e., whether they share the same parent) using Φ_{ijk}.

[3]We can consider a two-step procedure for structure and parameter estimation: Firstly, we estimate the structure of the latent tree using the algorithms suggested in this paper. Subsequently, we use the Expectation Maximization (EM) algorithm [6] to infer the parameters.

Lemma 2. (Sibling Grouping) *The quantity Φ_{ijk} satisfies the following properties: (i) $\Phi_{ijk} = d_{i,j}$ for all $k \notin \{i, j\}$ if and only if i is a leaf node and j is its parent. (ii) $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j}$ for all $k, k' \notin \{i, j\}$ if and only if both i and j are leaves and they have the same parent, i.e., they belong to the same sibling group.*

Using the above lemma, we design tests to identify all (parent, leaf child) pairs and leaf-siblings among observed variables. We first compute Φ_{ijk} for all triples $i, j, k \in V$. For a pair of nodes $i, j \in V$, consider the set $\{\Phi_{ijk} : k \in V \setminus \{i, j\}\}$. If all the values in the set are equal to d_{ij}, then i is a leaf node and j is a parent of i. If the values are the same but not equal to d_{ij} or $-d_{ij}$, then i and j are siblings and both are leaf nodes.

When only the estimates of information distances $\hat{\mathbf{d}}$ are available, we relax the constraints for testing node relationships. Firstly, we only compute $\hat{\Phi}_{ijk} := \hat{d}_{i,k} - \hat{d}_{i,k}$ for those estimated distances $\hat{d}_{ij}$, $\hat{d}_{ik}$ and $\hat{d}_{jk}$ that are below a prescribed threshold $\tau > 0$, because longer information distance estimates (i.e., lower correlation estimates) are less accurate for a given number of samples. Thus, we use only reliable estimates of information distances for testing node relationships. For each pair of nodes (i, j) such that $\hat{d}_{ij} < \tau$, associate the set $\mathcal{K}_{ij} := \left\{k \in V\backslash\{i, j\} : \max\{\hat{d}_{ik}, \hat{d}_{jk}\} < \tau\right\}$. Secondly, we declare that nodes $i, j \in V$ are in the same family (i.e., parent-leaf child or leaf sibling pairs) if $\max_{k\in\mathcal{K}_{ij}} \widehat{\Phi}_{ijk} - \min_{k\in\mathcal{K}_{ij}} \widehat{\Phi}_{ijk} < \epsilon$ for another threshold $\epsilon > 0$. See [3] for more detailed descriptions of testing node relationships in a reliable way.

B. Recursive Grouping (RG) Algorithm

RG is a recursive procedure in which at each step, tests described in Section III-A are used to identify nodes that belong to the same family. Subsequently, RG introduces a parent node if a family of nodes (i.e., a sibling group) does not contain an observed parent. This newly introduced parent node corresponds to a hidden node in the original unknown latent tree. Once such a parent (i.e., hidden) node h is introduced, the information distances from h to all other observed nodes can be computed. More precisely, to compute statistics $d_{i,h}$ and $d_{j,h}$ from observed nodes i and j to their parent h, we solve the following linear equations: $d_{i,h} - d_{j,h} = d_{i,k} - d_{k,j}$ and $d_{i,h} + d_{j,h} = d_{i,j}$. Both equations can be readily verified by the Markov property in (1). For any other node k that is not a child of h, we can compute $d_{k,h}$ using a child i of h since $d_{k,h} = d_{i,k} - d_{i,h}$. Thus, we can regard h as an observed variable. Once a new layer of hidden variables has been introduced, we remove all child nodes, and build up the tree recursively. See Fig. 1 for an illustration of RG.

Theorem 3. (Consistency, Sample & Computational Complexity of Recursive Grouping) *(i) The recursive grouping algorithm is structurally consistent for all latent trees $T_p \in \mathcal{T}_{\geq 3}$. In addition, it is risk consistent for Gaussian latent trees. (ii) Assume that the effective depth is $\delta(T_p; V) = O(1)$. Then for every $\eta > 0$, there exists thresholds $\epsilon, \tau > 0$ such that*

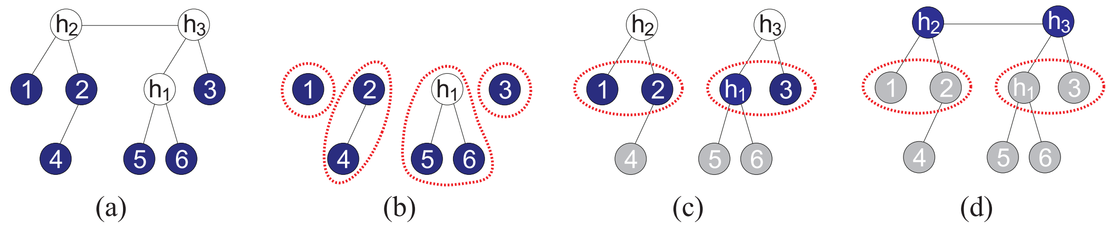

Fig. 1. An illustrative example of RG. Solid nodes indicate the active set at each iteration for which tests in Section III-A are applied. (a) Original latent tree. (b) Output after the first iteration of RG. Red dotted lines indicate the group of families. (c) Output after the second iteration of RG. Note that nodes 4,5, and 6 do not belong to the active set for the second iteration, and h_1, which was introduced in the first iteration, is now an active node. (d) Output after the third iteration of RG, which is same as the original latent tree.

if $n > C\log(m/\sqrt[3]{\eta})$ (for some $C > 0$), then the error probability in (2) *is bounded above by η. For Gaussian models, the error probability in* (3) *is bounded above by η as well. (iii) The computational complexity to recover the latent tree estimate $\hat{T}_p$ is $O(\mathrm{diam}(\hat{T}_p)m^3)$.*

A potential drawback of RG is that it involves computing $\hat{\Phi}_{ijk}$ for all observed triples, which may result in a high computational complexity. Indeed, from Theorem 3, the worst-case complexity is $O(m^4)$ which occurs when T_p, the true latent tree, is a hidden Markov model (HMM). This may be computationally prohibitive if m is large. In the next section, we design an algorithm which uses a *global* pre-processing step to reduce the overall complexity substantially, especially for trees with large diameters (of which HMMs are extreme examples).

IV. CLGrouping

In the previous section, we presented a recursive method to recover the structure of a latent tree. In this section, we improve the computational efficiency by performing a pre-processing clustering procedure before executing recursive grouping on smaller subsets of nodes. The pre-processing clustering procedure is based on the *Chow-Liu tree* [4] over the set of *observed* nodes V. We then obtain an estimate of the latent tree by recursively adding hidden variables to the Chow-Liu tree. For simplicity, we focus on Gaussian models first, and discuss the extension to discrete models in Section IV-D.

A. A Review of the of Chow-Liu Algorithm

The Chow-Liu tree $\widehat{T}_{\mathrm{CL}}$ is the ML tree with vertex set V given the samples $\mathbf{x}_V^n$. Denoting $\mathcal{D}(\mathcal{T}(V))$ as the set of distributions Markov on any tree spanning V, $\hat{p}_{\mathrm{CL}} := \mathrm{argmin}_{\nu\in\mathcal{D}(\mathcal{T}(V))} D(\hat{\mu}\,||\,\nu)$ is the corresponding tree distribution "closest" to the empirical distribution $\hat{\mu}(\mathbf{x}) := n^{-1}\sum_k \mathbf{1}\{\mathbf{x}=\mathbf{x}_k\}$. It was shown [4] that the tree that attains the minimum in $\hat{p}_{\mathrm{CL}}$ is given by

$$\widehat{T}_{\mathrm{CL}} = \underset{T=(V,E)\in\mathcal{T}}{\mathrm{argmax}} \sum_{e\in E} I(\hat{\mu}_e), \tag{4}$$

where (with abuse of notation) $I(\hat{\mu}_e)$ denotes the empirical mutual information between the two variables in the node pair e. That is, the Chow-Liu algorithm proceeds by first forming the empirical mutual information edge weights from the samples $\mathbf{x}_V^n$ and then searches for the tree that maximizes the sum of these weights. The parameters (node and pairwise marginals) are then estimated by ML.

B. Relationship between Chow-Liu and Latent Trees

Given the information distance estimates $\hat{\mathbf{d}}$ between the observed nodes in V, the Chow-Liu tree in (4) reduces to a minimum spanning tree (MST) problem with edge weights $\hat{d}$.

$$\widehat{T}_{\mathrm{CL}} = \mathrm{MST}(V;\hat{\mathbf{d}}) := \underset{T=(V,E)\in\mathcal{T}}{\mathrm{argmin}} \sum_{e\in E} \hat{d}_e. \tag{5}$$

The above result is due to the fact that the empirical mutual information $I(\hat{\mu}_e)$ is a monotonically decreasing function of the ML distance estimates $\hat{d}_e$.

When the latent tree T_p has no hidden variables, the Chow-Liu tree $\mathrm{MST}(V;\hat{\mathbf{d}})$ is the ML estimate of the tree structure and is consistent [4]. However, when there are hidden nodes, we need a transformation of $\mathrm{MST}(V;\hat{\mathbf{d}})$ to obtain a consistent estimate of T_p. To this end, we first analyze relationships between the latent tree and the Chow-Liu tree using the concept of the surrogate nodes defined as follows:

Definition 2. *(***Surrogate Node***) Given the latent tree $T_p = (W, E_p)$ and a node $i \in W$, the* surrogate node *of i wrt V is defined as $\mathrm{Sg}(i;T_p,V) := \mathrm{argmin}_{j\in V}\, d_{i,j}$.*

Note that if $i\in V$, then $\mathrm{Sg}(i;T_p,V)=i$ since $d_{i,i}=0$. When the tree and the observed set are unambiguous, the surrogate node of i will be denoted as $\mathrm{Sg}(i)$. Intuitively, the surrogate node of $h \in H$ is an observed node j with the strongest correlation or the smallest information distance $d_{h,j}$. We now relate the the original latent tree $T_p = (W, E_p)$ with the Chow-Liu tree given exact information distances $\mathrm{MST}(V;\mathbf{d})$.

Lemma 4. *(***Properties of Chow-Liu tree***) The Chow-Liu tree and surrogate nodes satisfy the following: (i) The surrogate nodes of any two neighboring nodes in E_p are neighbors in the Chow-Liu tree, i.e., for $i,j \in W$ with $\mathrm{Sg}(i) \neq \mathrm{Sg}(j)$,*

$$(i,j) \in E_p \Rightarrow (\mathrm{Sg}(i), \mathrm{Sg}(j)) \in \mathrm{MST}(V;\mathbf{d}). \tag{6}$$

(ii) The maximum degree of the Chow-Liu tree satisfies

$$\Delta(\mathrm{MST}(V;\mathbf{d})) \le \Delta(T_p)^{1+\frac{u}{l}\delta(T_p;V)}, \tag{7}$$

where l, u are the bounds on the information distances on edges in T_p.

For example, in Figure 2(a), a latent tree is shown with its corresponding surrogacy relationships, and Figure 2(b) shows the corresponding MST over the observed nodes.

C. CLGrouping Algorithm

In this section, we propose CLGrouping, which reconstructs *all* minimal latent trees consistently and efficiently. CLGrouping uses the properties of the Chow-Liu tree described in Lemma 4.

At a high-level, CLGrouping involves two distinct steps: Firstly, we construct the Chow-Liu tree $\mathrm{MST}(V;\hat{\mathbf{d}})$ over the set of observed nodes V. Secondly, we apply RG or NJ to reconstruct a latent subtree over the closed neighborhoods of every internal node in $\mathrm{MST}(V;\hat{\mathbf{d}})$. If RG (respectively NJ) is used, we term the algorithm CLRG (respectively CLNJ). CLRG proceeds as follows:

1) Construct the Chow-Liu tree $\mathrm{MST}(V;\hat{\mathbf{d}})$ as in (5). Set $T = \mathrm{MST}(V;\hat{\mathbf{d}})$.
2) Identify the set of internal nodes in $\mathrm{MST}(V;\hat{\mathbf{d}})$.
3) For each internal node i, let $\mathrm{nbd}[i;T]$ be its closed neighborhood in T and let $S = \mathrm{RG}(\mathrm{nbd}[i;T],\hat{\mathbf{d}})$ be the output of RG with $\mathrm{nbd}[i;T]$ as the set of input nodes.
4) Replace the subtree over node set $\mathrm{nbd}[i;T]$ in T with S. Denote the new tree as T.
5) Repeat steps 3 and 4 until all internal nodes have been operated on.

See Fig. 2 for an illustration of CLGrouping. The only difference between CLRG and CLNJ is Step 3 in which the subroutine NJ replaces RG. Note that in Step 3 that RG is only applied to a small subset of nodes which have been identified in Step 1 as possible neighbors in the true latent tree. This reduces the computational complexity of CLRG compared to RG. Let $|J| := |V \setminus \mathrm{Leaf}(\mathrm{MST}(V;\hat{\mathbf{d}}))| < m$ be the number of internal nodes in the MST. We now state one of the main results of this paper that the CLGrouping algorithm is consistent, and is also computationally efficient.

Theorem 5. *(***Consistency, Sample & Computational Complexity of CLGrouping***) (i) The CLGrouping algorithm is consistent for all minimal latent trees $T_p \in \mathcal{T}_{\geq 3}$. (ii) Assume that the effective depth is $\delta(T_p;V) = O(1)$. Then for every $\eta > 0$, there exists thresholds $\epsilon, \tau > 0$ such that if $n > C' \log(m/\sqrt[3]{\eta})$ (for some $C' > 0$), then the error probabilities in* (2) *and* (3) *are bounded above by η. (iii) The computational complexity to reconstruct the latent tree from distance estimates $\hat{\mathbf{d}}$ is $O(m^2 \log m + |J|\Delta(\mathrm{MST}(V;\hat{\mathbf{d}}))^3)$.*

Thus, the computational complexity of CLRG is low when the latent tree T_p has a small maximum degree and a small effective depth (such as the HMM) because (7) implies that $\Delta(\mathrm{MST}(V;\mathbf{d}))$ is also small. Indeed, we demonstrate in Section V that there is a significant speedup compared to applying RG over the entire observed node set V.

D. Extension to Discrete Models

For discrete models, $\mathrm{MST}(V;\hat{\mathbf{d}})$ is not necessarily same as the Chow-Liu tree, but all the relationships between $\mathrm{MST}(V;\mathbf{d})$ and the latent tree in Lemma 4 still hold. Thus, we first learn $\mathrm{MST}(V;\hat{\mathbf{d}})$ using the information distances as edge weights and apply CLGrouping to recover latent tree models. Theorem 5 holds for discrete models as well except for the risk consistency.

V. SIMULATIONS

In this section, we compare the performances of various latent tree learning algorithms. We generate samples from known latent tree structures with varying sample sizes, and apply the following learning algorithms: the neighbor-joining method (NJ) [18], the recursive grouping algorithm (RG), CLGrouping with NJ (CLNJ), and CLGrouping with RG (CLRG).

Figure 3 shows the three latent tree structures used in our simulations. The double-star has 2 hidden and 80 observed nodes, the HMM has 78 hidden and 80 observed nodes, and the 5-complete tree has 25 hidden and 81 observed nodes including the root node. For simplicity, we present simulation results only on Gaussian models but note that the behavior on discrete models is similar. All correlation coefficients on the edges ρ_{ij} were independently drawn from a uniform distribution supported on $[0.2, 0.8]$. The performance of each method is measured by averaging over 200 independent runs with different parameters. We use the following performance metrics to quantify the performance of each algorithm in Figure 4: (i) **Structure recovery error rate**: This is the proportion of times that the proposed algorithm fails to recover the true latent tree structure. Note that this is a very strict measure since even a single wrong hidden node or misplaced edge results in an error for the entire structure. (ii) **Robinson Foulds metric** [16]: This popular phylogenetic tree-distortion metric computes the number of graph transformations (edge contraction or expansion) needed to be applied to the estimated graph in order to get the correct structure. This metric quantifies the difference in the structures of the estimated and true models. (iii) **Error in the number of hidden variables**: We compute the average number of hidden variables introduced by each method and plot the absolute difference between the average estimated hidden variables and the number of hidden variables in the true structure. (iv) **KL-divergence** $D(p_V \,||\, \hat{p}^n_V)$: This is a measure of the distance between the estimated and the true models over the set of observed nodes V.[4]

We first note that from the structural error rate plots that the double star is the easiest structure to recover and the 5-complete tree is the hardest. In general, given the same number of observed variables, a latent tree with more hidden variables or larger effective depth (see Section II) is more difficult to recover.

For the double star, RG clearly outperforms all other methods. With only 1,000 samples, it recovers the true structure exactly in all 200 runs. On the other hand, CLGrouping

[4]We plot $D(p_V \,||\, \hat{p}^n_V)$ because if the number of hidden variables is estimated incorrectly, $D(p \,||\, \hat{p}^n)$ is infinite.

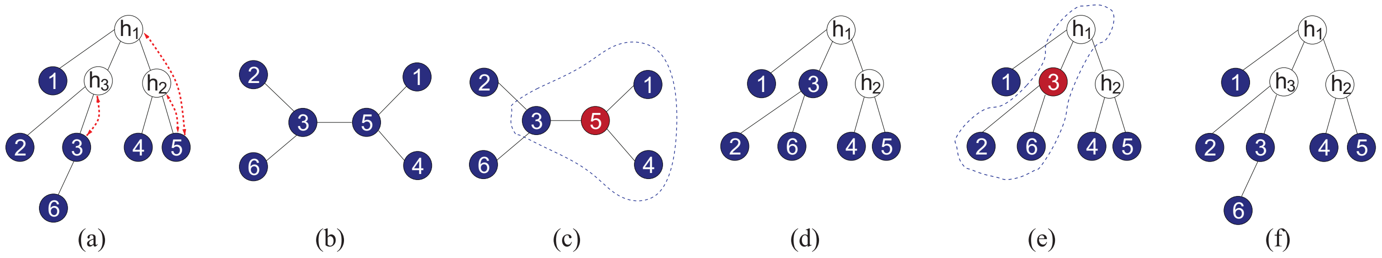

Fig. 2. Illustration of CLRG. The shaded nodes are the observed nodes and the rest are hidden nodes. The dotted lines denote surrogate mappings for the hidden nodes so for example, node 3 is the surrogate of h_3. (a) The original latent tree, (b) The Chow-Liu tree (MST) over the observed nodes V, (c) The closed neighborhood of node 5 is the input to RG, (d) Output after the first RG procedure, (e) The closed neighborhood of node 3 is the input to the second iteration of RG, (f) Output after the second RG procedure, which is same as the original latent tree.

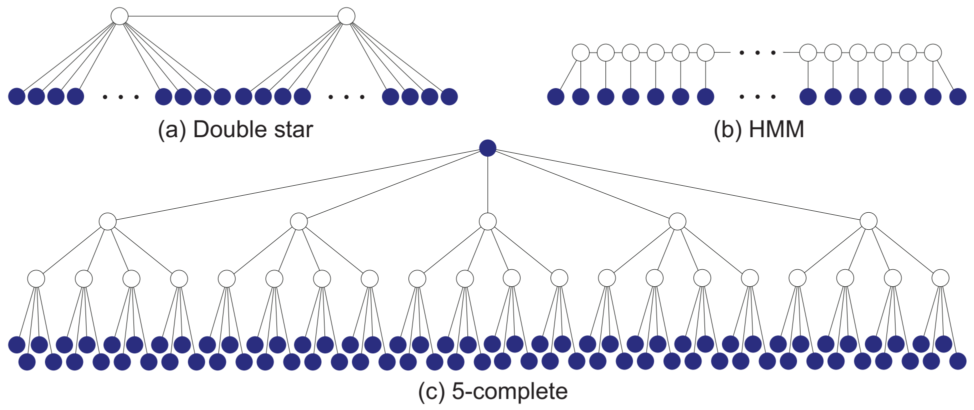

Fig. 3. Latent tree structures used in our simulations.

performs significantly better than RG for the HMM. There are two reasons for such performance differences. Firstly, for Gaussian distributions, it was shown [19] that given the same number of variables and their samples, the Chow-Liu algorithm is most accurate for a chain and least accurate for a star. Since the Chow-Liu tree of a latent double star graph is close to a star, and the Chow-Liu tree of a latent HMM is close to a chain, the Chow-Liu tree tend to be more accurate for the HMM than for the double star. Secondly, the internal nodes in the Chow-Liu tree of the HMM tend to have small degrees, so we can apply RG or NJ to a very small neighborhood, which results in a significant improvement in both accuracy and computational complexity.

Note that NJ is particularly poor at recovering the HMM structure. In fact, it has been shown that even if the number of samples grows polynomially with the number of observed variables (i.e., $n = O(m^B)$ for any $B > 0$), it is insufficient for NJ to recover HMM structures [10]. The 5-complete tree has two layers of hidden nodes, making it very difficult to recover the exact structure using any method. CLNJ has the best structure recovery error rate and KL divergence, while CLRG has the smallest Robinson-Foulds metric.

Table I shows the running time of each algorithm averaged over 200 runs and all sample sizes. All algorithms are implemented in MATLAB. As expected, we observe that CLRG is significantly faster than RG for HMM and 5-complete graphs. NJ is fastest, but CLNJ is also very efficient and leads to much more accurate reconstruction of latent trees.

	RG	NJ	CLRG	CLNJ
HMM	10.16	0.02	0.10	0.05
5-complete	7.91	0.02	0.26	0.06
Double star	1.43	0.01	0.76	0.20

TABLE I
AVERAGE RUNNING TIME OF EACH ALGORITHM IN SECONDS.

Based on the simulation results, we conclude that for a latent tree with a few hidden variables, RG is most accurate, and for a latent tree with a large diameter, CLNJ performs the best. A latent tree with multiple layers of hidden variables is more difficult to recover correctly using any method, and CLNJ and CLRG outperform NJ and RG.

VI. CONCLUSIONS

In this paper, we proposed algorithms to learn a latent tree model from the information distances of observed variables. Our first algorithm, recursive grouping, identifies sibling and parent-child relationships and introduces hidden nodes recursively. Our second algorithm, CLGrouping, first learns the Chow-Liu tree among observed variables and then applies latent-tree-learning subroutines such as recursive grouping or neighbor joining locally to each internal node in the Chow-Liu tree and its neighbors. These algorithms are structurally consistent (and risk consistent as well for Gaussian distributions), and have sample complexity logarithmic in the number

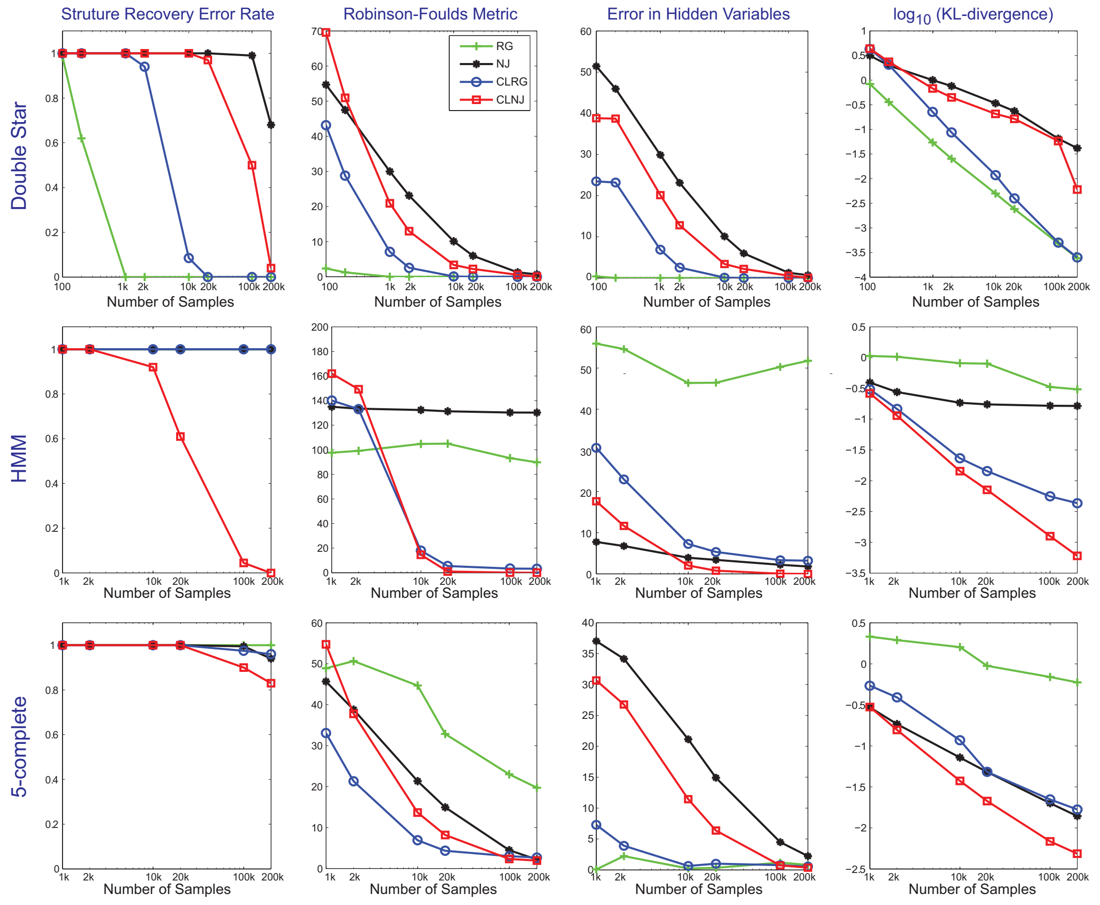

Fig. 4. Performance of RG, NJ, CLRG, and CLNJ for the latent trees shown in Figure 3.

of observed variables. The MATLAB implementation of our algorithms can be downloaded from the project webpage http://people.csail.mit.edu/myungjin/latentTree.html.

References

[1] S. Bhamidi, R. Rajagopal, and S. Roch, "Network delay inference from additive metrics," *To appear in Random Structures and Algorithms, Arxiv preprint math/0604367*, 2009.

[2] J. T. Chang and J. A. Hartigan, "Reconstruction of evolutionary trees from pairwise distributions on current species," in *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, 1991, pp. 254–257.

[3] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky, "Learning latent tree graphical models," arXiv:1009.2722v1.

[4] C. K. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Tran. on Information Theory*, vol. 3, pp. 462–467, 1968.

[5] C. Daskalakis, E. Mossel, and S. Roch, "Optimal phylogenetic reconstruction," in *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006, pp. 159–168.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[7] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge Univ. Press, 1999.

[8] G. Elidan and N. Friedman, "Learning hidden variable networks: The information bottleneck approach," *Journal of Machine Learning Research*, vol. 6, pp. 81–127, 2005.

[9] P. L. Erdös, L. A. Székely, S. M. A., and W. T. J., "A few logs suffice to build (almost) all trees: Part ii," *Theoretical Computer Science*, vol. 221, pp. 153–184, 1999.

[10] M. R. Lacey and J. T. Chang, "A signal-to-noise analysis of phylogeny estimation by neighbor-joining: Insufficiency of polynomial length sequences," *Mathematical Biosciences*, vol. 199, pp. 188–215, 2006.

[11] J. A. Lake, "Reconstructing evolutionary trees from dna and protein sequences: Parallnear distances," *Proceedings of the National Academy of Science*, vol. 91, pp. 1455–1459, 1994.

[12] S. Lauritzen, *Graphical models*. Clarendon Press, 1996.

[13] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, Models, and Applications*. Wiley, 1979.

[14] D. Parikh and T. H. Chen, "Hierarchical Semantics of Objects (hSOs)," in *ICCV*, 2007, pp. 1–8.

[15] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Network of Plausible inference*. Morgan Kaufmann, 1988.

[16] D. F. Robinson and L. R. Foulds, "Comparison of Phylogenetic Trees," *Mathematical Biosciences*, vol. 53, pp. 131–147, 1981.

[17] S. Roch, "A short proof that phylogenetic tree reconstruction by maximum likelihood is hard," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 3, no. 1, 2006.

[18] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Mol Biol Evol*, vol. 4, no. 4, pp. 406–425, Jul 1987.

[19] V. Y. F. Tan, A. Anandkumar, and A. S. Willsky, "Learning Gaussian tree models: Analysis of error exponents and extremal structures," *IEEE Transactions on Signal Processing*, vol. 58, no. 5, pp. 2701–2714, May 2010.

[20] N. L. Zhang and T. Kočka, "Efficient learning of hierarchical latent class models," in *ICTAI*, 2004.