Hierarchical Inference of Unicast Network Topologies Based on End-to-End Measurements

Meng-Fu Shih and Alfred O. Hero, III, *Fellow, IEEE*

***Abstract*—In this paper, we address the problem of topology discovery in unicast logical tree networks using end-to-end measurements. Without any cooperation from the internal routers, topology estimation can be formulated as hierarchical clustering of the leaf nodes based on pairwise correlations as similarity metrics. Unlike previous work that first assumes the network topology is a binary tree and then tries to generalize to a nonbinary tree, we provide a framework that directly deals with general logical tree topologies. A hierarchical algorithm to estimate the topology is developed in a recursive manner by finding the best partitions of the leaf nodes level by level. Our simulations show that the algorithm is more robust than binary-tree based methods.**

***Index Terms*—Graph-based clustering, mixture models, network tomography, topology estimation.**

I. Introduction

THE infrastructure of a packet network is composed of switching devices (as nodes) and communication channels (as links). It is constantly changing due to devices going online and offline, and the corresponding routing table updates. The topology information of the infrastructure can be revealed by packet routes across the entire network. Tools such as Traceroute trace a packet route by collecting responses from all the switching devices on the route. This kind of cooperation from the network has a negative impact on network performance and security, and such cooperation is likely to become more difficult in the future. Due to this reason the problem of discovering the network topology based only on end-to-end measurements has been of great interest [1]–[8]. This type of problems belongs to the research category called *network tomography*.

Ratnasamy *et al.* [1] and Duffield *et al.* [2] pioneered work in discovery of multicast network topologies. They specifically targeted the identification of the network's logical tree structure. By sending multicast probes from the root node of the tree to a pair of the leaf nodes, one can estimate the successful transmission rate on the shared portion of the probe paths, called the shared path, based on end-to-end loss. Those rate estimates were used by the *deterministic binary tree classification algorithm* (DBT) [2], [3] to construct a binary logical tree in a bottom-up manner. The extension to a general tree is basically done by pruning the links with loss rates less than some heuristically selected threshold. The DBT algorithm has also been extended to use other metrics such as packet delays [2], [3].

Topology estimation in unicast networks was investigated by Castro *et al.* [4]–[6]. They invented a method of probing, called *sandwich probes*, to estimate the queueing delay on the shared path from the root to two of the leaf nodes. Castro *et al.* also proposed a binary tree construction algorithm similar to DBT, called the *agglomerative tree algorithm* (ALT), which modifies DBT to account for the variability of the measurements through the spread of its probability density function (pdf) [6]. The special case of Gaussian-distributed measurements was previously called the *likelihood-based binary tree algorithm* (LBT) [5]. To compensate for the greedy behavior of the ALT, causing it to reach a local optimum in many cases, as well as to extend the result to general trees without using a threshold, they introduced a Monte Carlo Markov chain (MCMC) method to generate a sequence of tree candidates by birth (node insertion) and death (node deletion) transitions [6]. The tree candidate that gives the highest likelihood is adopted as the estimate of the topology.

In this paper, we propose a general method for estimation of unicast network topologies. As in [2] and [6], we focus on the estimation of logical tree structure of the network. The key to our approach is a formulation of the problem as a hierarchical clustering of the leaf nodes based on a set of measured pairwise similarities. The similarity of a pair of leaf nodes can be represented by some metric function associated with the path from the root to the nearest common ancestor of the two leaf nodes. We investigate three different types of similarity metrics that can be estimated from end-to-end measurements: queueing delay using sandwich probes, delay variance using packet pairs, and loss rate also using packet pairs. We modify the likelihood model for the pairwise similarities in [5] and [6] to include a prior distribution on the nearest common ancestor node of each pair of the leaf nodes. This results in a finite mixture model with every mixture component corresponding to a distinct internal node. A penalized maximum likelihood (PML) is developed using a minimum message length (MML) type of penalty for model order selection. An expectation-maximization (EM) algorithm can be used for unsupervised estimation of the mixture model parameters by maximizing the PML. Topology estimation is then performed by a top-down search for the best partitions of the leaf nodes. The first step is to construct a complete graph whose edge weights are derived from the mixture model estimate. Then a partition algorithm is applied to cluster the vertices based on the edge weights.

Our contribution in this paper includes 1) the use of hierarchical topology likelihood with finite mixture models and MML

Manuscript received May 24, 2005; revised June 21, 2006. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Meng-Fu Shih. This research was partially supported by the National Science Foundation under Grant CCR-0325571.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: mfshih@gmail.com; hero@eecs.umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2006.890830

1053-587X/$25.00 © 2007 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

model order penalties; 2) the top-down recursive partitioning of the leaf nodes, which directly yields a general logical tree without using thresholds or Monte Carlo methods; 3) the estimation of leaf node partitions using graph-based clustering and unsupervised learning of the finite mixture models; and 4) the intelligent search of the partition likelihood surface using graph clustering procedures.

The performance of our algorithm is compared with the DBT and LBT using Matlab model simulation under a wide range of conditions on the magnitudes and variances of the similarity estimates. The results show that our algorithm generally achieves a lower error, as measured by tree edit distance [11] to the true topology, and a higher percentage of correctly estimated trees. The three candidate probing schemes are evaluated on an ns-2[1] simulated network. Monte Carlo simulations show the queueing delay metric measured by sandwich probes have the best performance when the network load is light. For a moderate load, the delay variance metric measured by packet pair probes provides the most reliable similarity estimate for the leaf nodes. When the network is congested with heavy traffic the loss rate metric measured by packet pair probes generates the most accurate topology estimates. We also use tree edit distance as a metric to define distributions of topology estimates. This idea is illustrated by a network simulated in ns-2.

This paper is organized as follows. In Section II, we set up the logical tree network model. The probing methods and the associated similarity metrics are also introduced. In Section III, we derive the finite mixture model for the end-to-end similarity measurements. Based on this model, we define the partition and hierarchical topology likelihoods. In Section IV, we illustrate the *hierarchical topology estimation algorithm* (HTE), which recursively partitions the leaf nodes based on graph connectivity. In Section V, we conduct comprehensive simulations in Matlab and ns-2 to evaluate the performance of our algorithm with different probing methods and under various network environments. Section VI provides the conclusion and discusses future work. For more detailed derivations and more simulation studies than what could be presented in this paper the reader is referred to [12].

II. Background

A. Problem Formulation

Our work focuses on the problem of estimating *logical tree* network structures given end-to-end statistics measured by probes sent from the root to the leaf nodes. We assume there is no information provided by the internal devices of the network. A directed logical tree $T = (\mathbf{V}, \mathbf{E})$ is defined by two sets of objects: $\mathbf{V}$ as the set of nodes, and $\mathbf{E}$ as the set of directed links. We let the root be defined as node 0, $\mathbf{V}_i$ be the set of internal nodes and $\mathbf{V}_r$ be the set of leaf nodes. The root is the only node having a single child node, while all internal nodes have at least two child nodes. We adopt the convention to number the links by their child end nodes. The topology estimation problem is illustrated in Fig. 1, where the topology on the right is an example of a logical tree.

[1]http:www-mash.cs.berkeley.edu/ns/ns.html

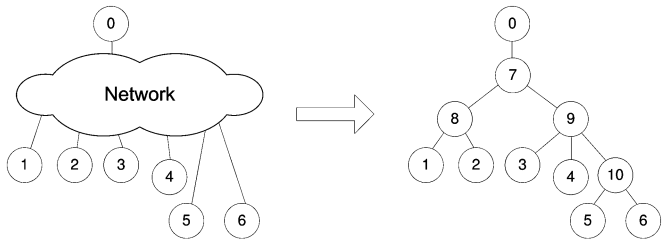


Fig. 1. Illustration of the topology estimation problem.

We also define the following useful notation. For a node $v \in \mathbf{V} \setminus \{0\}$, let $\mathrm{par}(v)$ be the parent node of v. Then $c(v) = \{v' \in \mathbf{V} : \mathrm{par}(v') = v\}$ denotes the set of children of v. The nodes in $c(v)$ are *sibling nodes* because they share the same parent. $c(v)$ can be formulated as the union of two disjoint sets: the set of leaf node children $c_r(v) = c(v) \cap \mathbf{V}_r$ and the set of internal node children $c_i(v) = c(v) \cap \mathbf{V}_i$. Let $V_{ic} = \{v \in \mathbf{V}_i : c_i(v) \neq \phi\}$ represent the set of internal nodes whose children are not all leaf nodes. Let $v_1 \prec v_2$ denote v_1 being a descendant of v_2. We define $d(v) = \{v' \in \mathbf{V}_r : v' \prec v\}$ be the set of descendant leaf nodes of v.

Topology estimation can be formulated as *hierarchical clustering* of the leaf nodes in which each group of nodes may be recursively partitioned into subgroups [6], [7]. Each leaf node itself is also considered as a cluster, called a *trivial cluster*. Hierarchical clustering relies on a measure of pairwise information to partition the input objects [13]. The objects in one (sub)cluster must be more similar to each other than to those in the remaining (sub)clusters. Suppose the similarity between a pair of leaf nodes (i, j) can be expressed by some quantitative measure $\gamma_{i,j}$, called similarity metric. Assume that $\gamma_{i,j} = \gamma_{j,i}$ and $\gamma_{i,i} = \infty$. Given a partition of leaf nodes, we define the *intracluster* similarities as those between two leaf nodes in the same cluster, and the *intercluster* similarities as those between two leaf nodes in two different clusters [6].

In general, if the clusters are good, the intercluster similarities should be smaller than the intracluster ones. Define C as a hierarchical clustering of the leaf nodes. We propose to define a *similarity clustering tree* $T_s(\mathbf{C})$ as follows. The root node in $T_s(\mathbf{C})$ corresponds to the top-level partition in C, and is associated with the set of all intercluster similarities for that partition. Each cluster containing two or more leaf nodes corresponds to a child node of the root and is associated with the set of all inter-subcluster similarities. This process is repeated recursively for all the partitions having nontrival clusters. The set of similarities associated with a node in $T_s(\mathbf{C})$ is called a *similarity set*. A similarity set is called *trivial* if all the intercluster similarities in the set are between two trivial clusters, otherwise it is *nontrivial*. All the $\gamma_{i,j}$'s in the same set are assumed to be equal, and they always have greater values than those associated with the parent node in $T_s(\mathbf{C})$. Fig. 2 shows hierarchical clustering C for the leaf nodes in Fig. 1 and the similarity clustering tree $T_s(\mathbf{C})$. It is easy to verify that $T_s(\mathbf{C})$ is a bijective mapping from C to a tree graph, which means topology estimation is also equivalent to hierarchical grouping of the pairwise similarities. This property will be the key to the development of our algorithm.

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

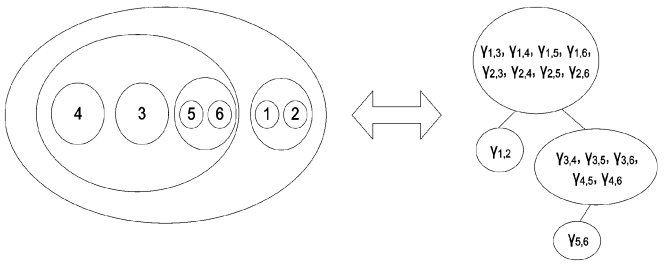


Fig. 2. Hierarchical clustering $\mathbf{C}$ of the leaf nodes in Fig. 1 (left) and the corresponding similarity clustering tree $T_s(\mathbf{C})$ (right).

In topology estimation, the concept of *metric-induced network topology* (MINT) introduced by Bestavros *et al.* [10] provides a framework for defining the similarity metrics. Under the MINT framework a metric is defined which is used to capture the similarity between all measurement pairs, e.g., covariance between measured delays. Any pair of leaf nodes that are connected to the source (root) through common links will have approximately equal similarity according to the metric. Different network topologies will usually generate different clusters of leaf pairs having almost identical similarities. The metric thus induces a virtual network topology that is associated with the actual topology. Note that each node in the similarity clustering tree corresponds to a unique internal node in the topology. This internal node is the nearest common ancestor shared by each pair of the leaf nodes in the similarity set. This implies the following connection between the MINT and the similarities. Define $p_{i,j}$ as the directed path from node i to j for $j \prec i$. To simplify the notation we let $p_i = p_{0,i}$ for $i \in \mathbf{V} \setminus \{0\}$. Let $a(i,j)$ be the nearest common ancestor of leaf node i and j. Then each $p_{a(i,j)}$ is uniquely mapped to a similarity set in $T_s(\mathbf{C})$, which includes $\gamma_{i,j}$. Hence, we can define $\gamma_{i,j}$ as the metric function for $p_{a(i,j)}$ [5].

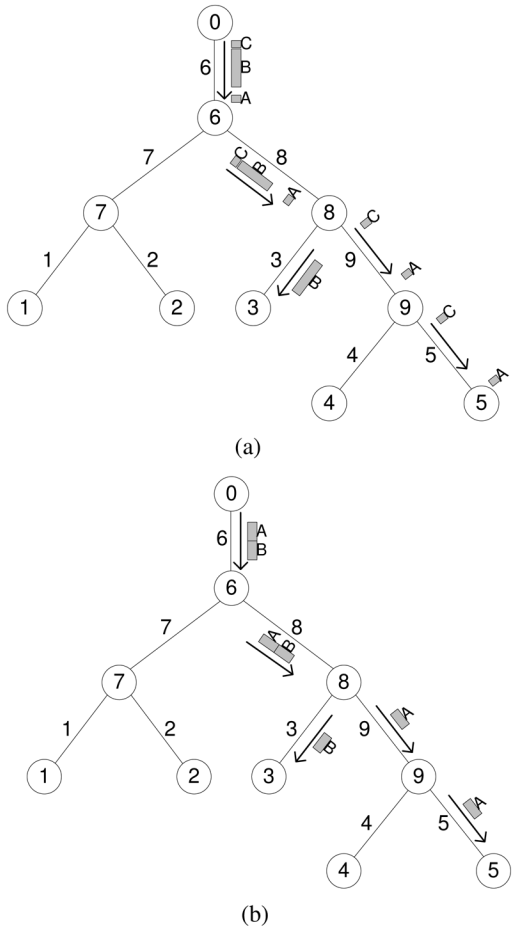


Fig. 3. Example for (a) sandwich probes and (b) packet pair probes. The probe tree $t_{3,5}$ is defined by the routes of the probe packets, which consists of links 3, 5, 6, 8, and 9.

B. End-to-End Unicast Probing Schemes

In this section, we discuss three possible schemes of unicast probing and induced similarity metrics that can be used for topology discovery. We assume the network topology and the traffic routing remain unchanged during the entire probing session. In our modeling we also assume the following statistical properties on the network environment:

A1) **spatial independence**: the packet delays over different links are independent;

A2) **temporal independence and stationarity**: the packet delays over a link are identically and independently distributed (i.i.d.).

We also define the binary logical tree formed by the union of path p_i and $p_j, i,j \in \mathbf{V}_r$, as a *probe tree* and denote it by $t_{i,j}$. Note that $p_{a(i,j)}$ is the intersection of p_i and p_j, and is called the *shared path* of $t_{i,j}$.

Sandwich probes were invented by Castro *et al.* in [5] for the similar purpose of topology estimation. Each probe contains three time-stamped packets: two small packets and one big packet *sandwiched* between the two small ones. The small packets are sent to one of the two leaf nodes, while the large packet is sent to the other [see Fig. 3(a)]. The queueing delay of the second small packet caused by the large packet can be considered as a metric on the shared path.

A *packet pair probe* consists of two closely spaced small packets. Both packets are sent from the root node but routed to two different leaf nodes [see Fig. 3(b)]. We need an additional assumption for packet pair probes:

A3) **delay consistency**: the queueing delays of the packet pair are identical with probability 1 when they travel along the shared path.

The first type of metric that can be retrieved from the packet pairs is delay variance [10], [14]. The independence assumption A1) implies the (queueing) delay over the shared path has a variance equal the end-to-end delay covariance of the two packets. For each probe tree we need N_{cov} end-to-end delay measurements to obtain one sample of the delay variance over the share path. The second type of metric can be computed from the packet pair probes is packet loss rate. Here we extend assumption A1)–A3) by interpreting packet losses as infinite delays. Similar to the delay variance metric, the packet loss rate on the shared path can be estimated by end-to-end loss rates [2], and one needs N_{loss} packet pairs to compute a single metric sample.

Theoretically, the sandwich probes are expected to have the best performance in a lightly loaded network environment

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

because the queueing delay is mainly caused by the large middle packet [8]. The best situation for packet pair probes with delay variance metrics is a moderately loaded network because the background traffic produces sufficient variation on packet queueing delays. Lastly, one can expect packet loss rates provide the most sufficient information to identify the topology for highly congested networks. Similar comparisons showing how different types of metrics perform with different traffic load in multicast networks can be found in [15].

III. Hierarchical Topology Likelihood Using Finite Mixture Models

A. Finite Mixture Model for Similarity Estimation

To establish a simple and unified framework for specifying metrics based on either packet delay or loss, we adopt the following strategy. First, observe that the metric samples $\hat{\gamma}_n^{(i,j)}$ estimated from data along probe tree $t_{i,j}$ are i.i.d according to A1) and A2). If we average every N_{norm} samples $\hat{\gamma}_n^{(i,j)}$, the result will be approximately Gaussian distributed when N_{norm} is large, according to the central limit theorem (CLT). We call the averaged samples *normalized similarity samples*, denoted by $\bar{\gamma}_n^{(i,j)}$. $\bar{\gamma}_n^{(i,j)}$ are also i.i.d. for all n.

Secondly, we find if $a(i,j) = a(k,l) = v$, then $\bar{\gamma}_n^{(i,j)}$ and $\bar{\gamma}_n^{(k,l)}$ have the same mean $\mu_v = \gamma_{i,j}$. As the variances of $\bar{\gamma}_n^{(i,j)}$ and $\bar{\gamma}_n^{(k,l)}$ go to 0 linearly as $N_{\text{norm}} \to \infty$, it is easy to show that $\bar{\gamma}_n^{(i,j)} \to \gamma_{i,j}, \bar{\gamma}_n^{(k,l)} \to \gamma_{i,j}$, and $|\bar{\gamma}_n^{(i,j)} - \bar{\gamma}_n^{(k,l)}| \to 0$ in probability (hence in distribution) as $N_{\text{norm}} \to \infty$. Since $\sqrt{N_{\text{norm}}}\bar{\gamma}_n^{(i,j)}$ and $\sqrt{N_{\text{norm}}}\bar{\gamma}_n^{(k,l)}$ both converges to Gaussian when $N_{\text{norm}} \to \infty$, we make the following approximation assumption:

A4) **consistency of similarity distributions**: let $\bar{\gamma}^{(i,j)}$ be the average of N_{norm} i.i.d. metric samples $\hat{\gamma}^{(i,j)}$; as $N_{\text{norm}} \to \infty, \bar{\gamma}^{(i,j)}$ and $\bar{\gamma}^{(k,l)}$ are approximately equal in (Gaussian) distribution if $a(i,j) = a(k,l)$, for $i,j,k,l \in \mathbf{V}_r$ and $i \neq j, k \neq l$.

The magnitude of the variance difference between $\hat{\gamma}_n^{(i,j)}$ and $\hat{\gamma}_n^{(k,l)}$ affects the minimum value of N_{norm} to be used. As $|\text{Var}(\bar{\gamma}^{(i,j)}) - \text{Var}(\bar{\gamma}^{(k,l)})| = |\text{Var}(\hat{\gamma}_n^{(i,j)}) - \text{Var}(\hat{\gamma}_n^{(k,l)})|/N_{\text{norm}}, N_{\text{norm}}$ should increase linearly with $|\text{Var}(\hat{\gamma}_n^{(i,j)}) - \text{Var}(\hat{\gamma}_n^{(k,l)})|$ in order to keep the approximation valid. According to our simulations (see Section V), a typical minimum value for N_{norm} is around 20–25 to achieve accurate topology estimates.

Let the set of normalized similarity samples for $\gamma_{i,j}$ be $\mathbf{\Gamma}_{i,j} = \{\bar{\gamma}_n^{(i,j)}\}_n$, and let $\mathbf{\Gamma} = \{\mathbf{\Gamma}_{i,j}\}_{(i,j)}$. We define $N_{i,j} = |\mathbf{\Gamma}_{i,j}|$ and $N = \sum_{i<j} N_{i,j}$. When v is an internal node, we denote $T(v) = \{t_{i,j} : a(i,j) = v, i < j, i,j \in \mathbf{V}_r\}$ as the set of probe trees whose branches split v and let $N_T(v) = |T(v)|$. Also let $N_T = \binom{|\mathbf{V}_r|}{2}$ be the total number of probe trees in tree T. Suppose the set of internal nodes V_i is known and the cardinality $|T(v)|$ is given for each $v \in \mathbf{V}_i$. A reasonable prior distribution of $a(i,j)$ is $\alpha(v) = P(a(i,j) = v) = (N_T(v)/N_T)$ for $v \in \mathbf{V}_i$. Given $f(\mathbf{\Gamma}_{i,j}|a(i,j) = v) = \phi(\mathbf{\Gamma}_{i,j}; \theta_v)$, where ϕ denotes the Gaussian pdf, this induces a *finite mixture model* $f(\mathbf{\Gamma}_{i,j}) = \sum_{v \in \mathbf{V}_i} \alpha(v)\phi(\mathbf{\Gamma}_{i,j}; \theta_v)$ (see, e.g., [16]). A finite mixture model $f(x)$ is generally expressed as the convex combination of probability density functions: $f(x) = \sum_{m=1}^{k} \alpha_m h_m(x)$, where $0 \leq \alpha_m \leq 1, \sum_{m=1}^{k} \alpha_m = 1$, and h_m is an arbitrary pdf for $m = 1, \ldots, k$. The α_m's are called the *mixing probabilities*, and the h_m's are the *mixture components*. k is the number of mixture components in the model, often referred as the *model order* of $f(x)$. If the h_m's are all Gaussian (with different parameters), then $f(x)$ is a Gaussian mixture. Given the mixture models for the similarities $\mathbf{\Gamma}_{i,j}$, the distribution of $\mathbf{\Gamma}$ becomes

$$f_{\text{FM}}(\mathbf{\Gamma}) = \prod_{\substack{i,j \in \mathbf{V}_r \\ i<j}} \sum_{v \in \mathbf{V}_i} \alpha(v)\phi(\mathbf{\Gamma}_{i,j}; \theta_v). \tag{1}$$

Note that the model order in (1) equals the number of the internal nodes of the tree. Each mixture component $\phi(\cdot; \theta_v)$ corresponds to a unique internal node v and $\mathbf{\Gamma}_{i,j}$ is contributed by $\phi(\cdot; \theta_v)$ if $a(i,j) = v$. The key difference between the models in [6] and (1) is that in $f_{\text{FM}}(\mathbf{\Gamma})$ the common parent node $a(i,j)$ is distributed according to some discrete prior instead of being a deterministic value. This relaxation leaves the prior, along with other parameters in the model, to be determined, e.g., by unsupervised estimation of the mixture model. It can be achieved using the EM algorithm [17], [18]. To discover the topology, the similarity sets in $T_s(\mathbf{C})$ are determined by associating each $a(i,j)$ with the component that contributes $\mathbf{\Gamma}_{i,j}$.

B. MML Penalized Likelihood for the Mixture Model

Likelihood-based estimation of the parameter $\mathbf{\Theta}$ in the mixture model

$$f_{\text{FM}}(\mathbf{\Gamma}; \mathbf{\Theta}) = \prod_{\substack{i,j \in \mathbf{V}_r \\ i<j}} \sum_{m=1}^{k} \alpha_m \phi(\mathbf{\Gamma}_{i,j}; \theta_m) \tag{2}$$

for $\mathbf{\Theta} = (k, \alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k)$ falls in the category of *missing data* problems. To avoid the complication of optimizing the α_m's over discrete values, we assume $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ is continuously distributed over the region $0 \leq \alpha_m \leq 1, \sum_{m=1}^{k} \alpha_m = 1$. For a given model order k (k also denotes the number of internal nodes), the *unobserved data* in our case is $\{a(i,j)\}$, which indicates the contributing component for each $\mathbf{\Gamma}_{i,j}$. Define the unobserved indicator function $Z_m^{(i,j)}$ for $m = 1, \ldots, k$ by $Z_m^{(i,j)} = 1$ if $\mathbf{\Gamma}_{i,j}$ is contributed by the mth component, and $Z_m^{(i,j)} = 0$ otherwise. Along with the observed data $\mathbf{\Gamma}$, the set $\{\mathbf{\Gamma}, \{Z_m^{(i,j)}\}\}$ is called the *complete data*. The maximum-likelihood (ML) estimate of $\mathbf{\Theta}$ with a given k can be obtained by using the EM algorithm, which generates a sequence of estimates with nondecreasing likelihoods [17], [18].

However, when k is unknown this becomes a model selection problem and the ML criterion can cause an *overfitting problem* in which a higher model order k generally results in a higher likelihood. A strategy to balance the model complexity and the goodness of data fitting is to add model order penalties to the likelihood [19]. We adopt a criterion called MML [19] to derive the penalty function. MML has been widely used in unsupervised learning of mixture models [9], [17], [18]. The incomplete data penalized log likelihood is expressed as [17]

$$\tilde{\mathcal{L}}(\mathbf{Y}; \mathbf{\Theta}) \stackrel{\text{def}}{=} \log f(\mathbf{\Theta}) + \log f(\mathbf{Y}|\mathbf{\Theta}) - \frac{1}{2}\log|\mathbf{I}(\mathbf{\Theta})| - \frac{c}{2}(1 + \log \kappa_c) \tag{3}$$

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

for observed data $\mathbf{Y}$ and parameter set $\mathbf{\Theta}$, where $I(\mathbf{\Theta})$ is the Fisher information matrix (FIM) associated with $\mathbf{Y}$, $|\cdot|$ denotes the determinant operator, c is the dimension of $\mathbf{\Theta}$, and κ_c is the so-called *optimal quantizing lattice constant for* $\Re^c$, meaning the multidimensional parameters are assumed to be quantized using optimal quantizing lattices [18], [19].

For a given model order k, our choice for the prior distributions of the parameters follows the least informative priors in [18]. The mixing probabilities in $\boldsymbol{\alpha}$ have a uniform prior $f(\boldsymbol{\alpha}) = (k-1)!$ for $0 \leq \alpha_m \leq 1, \forall m = 1, \ldots, k$, and $\sum_{m=1}^{k} \alpha_m = 1$. For f_{FM} being a Gaussian mixture, $\theta_m = (\mu_m, \sigma_m^2)$ for $m = 1, \ldots, k$. The prior for σ_m is uniform between 0 and σ_p, where σ_p is the standard deviation of the entire population $\mathbf{\Gamma}$. So we have $f(\sigma_m) = (1)/(\sigma_p)$ for $0 \leq \sigma_m \leq \sigma_p$. We also take a uniform prior for μ_m distributed within one standard deviation σ_p of μ_p, where μ_p is the mean of the population $\mathbf{\Gamma}$. Therefore, $f(\mu_m) = 1/2\sigma_p$, for $\mu_p - \sigma_p \leq \mu_m \leq \mu_p + \sigma_p$. The prior for the model order k is assumed uniform between two predetermined bounds $k_{\min}$ and $k_{\max}$. With the assumption that the parameters are independent, we have $f(\mathbf{\Theta}) = ((k-1)!)/(2^k \sigma_p^{2k})$.

In general, it is difficult to derive a closed form for the FIM of finite mixture models with more than one component. The authors of [18] suggested replacing the determinant of the FIM by the product of the determinant of the FIM for each component times the FIM determinant for the mixing probabilities. Hence, $|\mathbf{I}_{\mathrm{FM}}(\mathbf{\Theta})| \approx |\mathbf{I}_0(\boldsymbol{\alpha})| \times \prod_{m=1}^{k} |\mathbf{I}_m(\theta_m)|$, where $\mathbf{I}_0(\boldsymbol{\alpha})$ is the FIM for $\boldsymbol{\alpha}$ and $\mathbf{I}_m(\theta_m)$ is the FIM for the mth component with parameter θ_m. $\mathbf{I}_m(\theta_m)$ can be expressed as $\mathbf{I}_m(\theta_m) = \sum_{i,j \in \mathbf{V}_r, i<j} \mathbf{I}_m^{(i,j)}(\theta_m)$, where $I_m^{(i,j)}(\theta_m)$ is the FIM associated with $\mathbf{\Gamma}_{i,j}$ for the mth component, i.e.,

$$I_m^{(i,j)}(\theta_m) = \begin{bmatrix} \frac{\alpha_m N_{i,j}}{\sigma_m^2} & 0 \\ 0 & \frac{2\alpha_m N_{i,j}}{\sigma_m^2} \end{bmatrix}.$$

Therefore, we have $|\mathbf{I}_m(\theta_m)| = (2\alpha_m^2 N^2)/(\sigma_m^4)$.

To determine the FIM for $\boldsymbol{\alpha}$, one can view the $\boldsymbol{\alpha}$ as being the parameters of a multinomial distribution, which selects N_T $a(i,j)$'s from k internal nodes with the probability of choosing the mth internal node being α_m, where N_T is the total number of probe trees. Hence, $|\mathbf{I}_0(\boldsymbol{\alpha})| = (N_T)/(\prod_{m=1}^{k} \alpha_m)$. As ordering of the components is irrelevant, the factorial term $\log(k!)$ can be removed from the MML expression (3). We also approximate κ_c by the one-dimensional constant $\kappa_1 = (1/12)$ as in [18], [19]. Substituting the terms above into (3), we have

$$\begin{aligned} \mathcal{L}_p(\mathbf{\Gamma}; \mathbf{\Theta}) = & \log f_{\mathrm{FM}}(\mathbf{\Gamma}; \mathbf{\Theta}) \\ & + \log \frac{(k-1)!}{2^k \sigma_p^{2k}} + \log(k!) \\ & - \frac{1}{2} \log N_T - k(\log \sqrt{2} + \log N) \\ & - \frac{1}{2} \sum_{m=1}^{k} \log \alpha_m + \sum_{m=1}^{k} \log \sigma_m^2 - \frac{3k}{2}(1 - \log 12). \end{aligned} \tag{4}$$

One can use the EM algorithm to maximize (4) over $\mathbf{\Theta}$ [8], [18].

C. Hierarchical Topology Likelihood

Equation (4) is difficult to use directly for topology estimation due to identifiability problems. Recall that each internal node in the topology corresponds to a unique component in the finite mixture model. Consider the example in Fig. 1 once again. If $\gamma_{1,2} = \gamma_{5,6}$, the estimates $\bar{\gamma}_n^{(1,2)}$ and $\bar{\gamma}_n^{(5,6)}$ become indistinguishable, and the two mixture components erroneously merge to a single one. To overcome this problem, we propose a hierarchical definition of the topology likelihood which recursively evaluates each partition likelihood and hierarchically clusters the leaf node pairs.

Consider a group of leaf nodes G. Let $\boldsymbol{\gamma}(\mathbf{G}) = \{\gamma_{i,j} : i < j, i, j \in \mathbf{G}\}$ be the set of pairwise similarity metrics for $\mathbf{G}$, and $\mathbf{\Gamma}(\mathbf{G}) = \{\mathbf{\Gamma}_{i,j} : i < j, i, j \in \mathbf{G}\}$ be the normalized samples of $\boldsymbol{\gamma}(\mathbf{G})$. Let $\mathbf{K} = \{K_1, \ldots, K_D\}$ be a partition of $\mathbf{G}$ where $\mathbf{K}_d, d = 1, \ldots, D$ are disjoint subsets of G. Without loss of generality, let $K_1, \ldots, K_{D'}$ be the clusters containing two or more leaf nodes, and $K_{D'+1}, \ldots, K_D$ be single-node clusters. According to the monotonicity property, the intercluster $\gamma_{i,j}$'s share the smallest value in $\boldsymbol{\gamma}(\mathbf{G})$. Hence, the set of all intercluster $\mathbf{\Gamma}_{i,j}$'s, denoted by $\mathbf{\Gamma}_0(\mathbf{K})$, obey a Gaussian distribution that has the smallest mean over $\mathbf{\Gamma}(\mathbf{G})$. This means for the finite mixture model of $\mathbf{\Gamma}(\mathbf{G})$, the component with the smallest mean contributes $\mathbf{\Gamma}_0(\mathbf{K})$. We call this component the *intercluster component* of $f_{\mathrm{FM}}(\mathbf{\Gamma}(\mathbf{G}))$ and let $\mathbf{\Theta}_0(\mathbf{K})$ denote its parameter set. Let K_l be a cluster with two or more leaf nodes. The set of intracluster $\mathbf{\Gamma}_{i,j}$'s in K_l, denoted by $\mathbf{\Gamma}_l(\mathbf{K})$, also follows a finite mixture model. Let $\mathbf{\Theta}_l(\mathbf{K})$ be the mixture parameter set for $f_{\mathrm{FM}}(\mathbf{\Gamma}_l(\mathbf{K}))$. If all the subclusters in K_l are trivial, $f_{\mathrm{FM}}(\mathbf{\Gamma}_l(\mathbf{K}))$ degenerates to a single component density function. We define the *penalized partition likelihood* as

$$\begin{aligned} \mathcal{L}_k(\mathbf{\Gamma}(\mathbf{G}); \mathbf{K}, \mathbf{\Theta}(\mathbf{K})) \stackrel{\mathrm{def}}{=} & \mathcal{L}_p(\mathbf{\Gamma}_0(\mathbf{K}); \mathbf{\Theta}_0(\mathbf{K})) \\ & + \sum_{l=1}^{D'} \mathcal{L}_p(\mathbf{\Gamma}_l(\mathbf{K}); \mathbf{\Theta}_l(\mathbf{K})) \end{aligned} \tag{5}$$

where $\mathbf{\Theta}(\mathbf{K}) = (\mathbf{\Theta}_0(\mathbf{K}), \ldots, \mathbf{\Theta}_{D'}(\mathbf{K}))$. This motivates the following *hierarchical topology likelihood* for a logical tree $T = (\mathbf{V}, \mathbf{E})$:

$$\begin{aligned} \mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T)) = & \sum_{v \in \mathbf{V}_{ic}} \mathcal{L}_k(\mathbf{\Gamma}(d(v)); \mathbf{K}(v) \\ & \mathbf{\Theta}(\mathbf{K}(v)) \,|\, \{\mathbf{K}(v') : v \prec v', v' \in \mathbf{V}_{ic}\}) \end{aligned} \tag{6}$$

where $\mathbf{K}(v) = \{d(v') : v' \in c_i(v)\} \cup c_r(v)$ denotes the partition specified by the child nodes of v. The evaluation of $\mathcal{L}_T$ mimics exactly the construction of the similarity clustering tree. Each $v \in \mathbf{V}_{ic}$ corresponds to a unique node in $T_s(\mathbf{C})$ that is associated with a nontrivial similarity set.

IV. Topology Estimation Algorithm

A. Hierarchical Topology Estimation Algorithm

We propose a greedy algorithm to estimate the logical tree topology using a top-down approach that partitions the leaf nodes recursively. First, we use $f_{\mathrm{FM}}(\mathbf{\Gamma})$ to find the most coarse partition specified by the sibling nodes in $c(v)$ for v being the root node's child, then we determine if there exists any finer subpartition within each cluster. This process is repeated until no finer partitions are found. Fig. 4(a) shows an example that illustrates how the partition of nodes 6–11 identifies two internal nodes. This iterative procedure is greedy because in each iteration it focuses on finding the optimal partition within the current cluster of the leaf nodes without considering

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

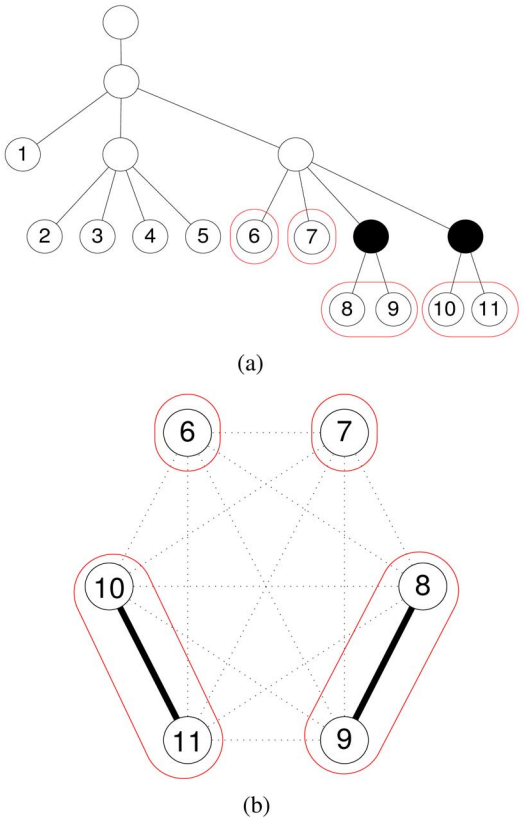


Fig. 4. Illustration of the hierarchical topology estimation. (a) Partition of nodes 6–11 identifies the two shaded internal nodes. (b) Graph-based partition of nodes 6–11. The solid edges have weights ≈ 1, and the dotted edges have weights ≈ 0.

other clusters or any possible subpartitions in the subsequent iterations.

The key to the hierarchical topology estimation algorithm is to find the partition of the leaf nodes. Our algorithm is motivated by the following observation. First, we label the component having the smallest mean in $f_{\mathrm{FM}}(\mathbf{\Gamma}(\mathbf{G}))$ by component 1 for some subset of leaf nodes $G \subseteq \mathbf{V}_r$. Assume there is no estimation error in the mixture model. In this ideal case, component 1 is the intercluster component supported exactly by all the intercluster $\mathbf{\Gamma}_{i,j}$'s. Then, the conditional mean $\omega_1^{(i,j)} = E[Z_1^{(i,j)}|\mathbf{\Gamma}(\mathbf{G}); \hat{\mathbf{\Theta}}] \approx 1$ if (i,j) is an intercluster pair and $\omega_1^{(i,j)} \approx 0$ otherwise, because $\omega_1^{(i,j)}$ can be viewed as a conditional mean estimator (CME) of the indicator function $Z_1^{(i,j)}$. Consider an undirected complete graph H whose vertices are the leaf nodes in G such that there exists an edge between every pair of the vertices. If we specify a weight $\varpi_{i,j} = \text{wieght}(e_{i,j}) = 1 - \omega_1^{(i,j)}$ to every edge $e_{i,j}$, one can easily find that a vertex in H strongly connects only to its peers in the same cluster. This implies that the partition of the leaf nodes can be estimated based on the edge weights of H. The partition in Fig. 4(a) estimated using graph edge weights is depicted in Fig. 4(b).

Basically, any graph-based clustering algorithm for weighted graphs could work for our purpose. Here, we describe a simple algorithm proposed in [20], the highly connected subgraph (HCS) algorithm. Let $H = (\mathbf{V}_H, \mathbf{E}_H)$ be a graph both undirected and weighted, where V_H is the set of vertices and E_H is the set of edges. Every edge e in E_H has a nonnegative real weight $\varpi(e)$. A *cut* in a graph is defined as a set of edges whose removal results in a disconnected graph. The total weight of the edges in a cut S is called the *cut weight* of S, denoted by $|S|$. A *minimum cut* (mincut) is a cut with the minimum weight. The weight of a mincut is called the *connectivity* of the graph, denoted by $\text{conn}(H)$.

The key definition for HCS is the following: A graph H with $n > 1$ vertices is called *highly connected* if $\text{conn}(H) > (n/2)$. In our case, a highly connected complete graph always has an average weight of the edges greater than $(1/2)$ [8]. The HCS algorithm requires a subroutine $\text{MINCUT}(H)$ which accepts graph H as the input and returns $(S, H', \bar{H}')$, where S is a mincut of H that divides H into two disjoint subgraphs H' and $\bar{H}'$. The problem of finding a mincut for a connected graph is one of the classical subjects in graph theory [22], [23]. The HCS algorithm recursively applies subroutine MINCUT to partition the graph until all the connected subgraphs are highly connected. The details can be found in [20].

In order to apply the HCS algorithm to our problem, we define a new indicator function as follows. Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be two clusters in a partition of the leaf node set $\mathbf{G} \subseteq \mathbf{V}_r$. Suppose the finite mixture model estimate for $\mathbf{\Gamma}(\mathbf{G})$ is $f_{FM}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$, which has k components. Let $\mathbf{B} \subseteq \{1, \ldots, k\}$ be a subset of the components. Define $\mathbf{\Gamma}_{\mathbf{A}_1,\mathbf{A}_2} = \{\mathbf{\Gamma}_{i,j} : i \in \mathbf{A}_1, j \in \mathbf{A}_2\}$ to be the set of i.i.d. normalized samples for the intercluster similarities between $\mathbf{A}_1$ and $\mathbf{A}_2$. Let $f_{\mathbf{B}}(\cdot) = \sum_{m \in \mathbf{B}} \alpha_m \phi(\cdot; \theta_m)$ denote the composite component formed by $\mathbf{B}$. Then, we define $Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)}$ as an indicator function of $\bar{\gamma}_n^{(i,j)} \in \mathbf{\Gamma}_{\mathbf{A}_1,\mathbf{A}_2}$, for $n = 1, \ldots, N_{i,j}, i \in \mathbf{A}_1, j \in \mathbf{A}_2$, such that $Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)} = 1$ if $\bar{\gamma}_n^{(i,j)}$ is contributed by the composite component $f_{\mathbf{B}}$, and $Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)} = 0$ otherwise. Then $Z_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)} = \{Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)}\}_{i,j,n}$ is i.i.d. with mean $E[Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)}] = \eta_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)}$. According to the definitions above, we define an edge $(\mathbf{A_1}, \mathbf{A_2})$ connecting two clusters A_1, A_2 with weight defined by

$$\varpi_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)} = 1 - \frac{1}{N_{\mathbf{A}_1,\mathbf{A}_2}} \times \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} \sum_{n=1}^{N_{i,j}} E \times \left[Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)} \mid \bar{\gamma}_n^{(i,j)}; \mathbf{\Theta} \right] \tag{7}$$

where $N_{\mathbf{A}_1,\mathbf{A}_2} = \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} N_{i,j}$.

B. Precluster Algorithm

The MINCUT procedure in the HCS algorithm can be computationally demanding when there are many vertices in the complete graph [23]. One way to reduce the complexity is to precluster the vertices in H into groups which are obviously in the same cluster. For a set of leaf nodes $G \subseteq \mathbf{V}_r$ and its corresponding finite mixture estimate $f_{\mathrm{FM}}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$, we assume the intercluster component is identified as a composite component $B \subseteq \{1, \ldots, k\}$, where k is the mixture model order. If there is no estimation error, B includes only the component having the smallest mean, called component 1 for simplicity. A simple way

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

to determine if a leaf node j resides in a different cluster from i is to check whether the edge weight between them is less than $(1/2)$. Define the set of *foreign leaf nodes* for node i with respect to B as $\mathbf{F_B}(i) = \{j : \varpi_{\mathbf{B}}^{(\{i\},\{j\})} < (1/2), j \in \mathbf{G}\backslash\{i\}\}, \forall i \in \mathbf{G}$. $\mathbf{F_B}(i)$ contains all possible nodes that are not in the same cluster as i. Then, we group nodes i_1 and i_2 in the same cluster if and only if $\mathbf{F_B}(i_1) = \mathbf{F_B}(i_2)$.

When there exists significant error in the finite mixture model estimates, it is possible that component 1 may not be correctly estimated as the intercluster component. Two possible situations may occur. First, a mixture model estimate with too fine resolution could decompose the intercluster component into several subcomponents. Second, an estimate with too-coarse resolution could merge the intercluster component with the intracluster ones. This would lead to an unsufficiently rich set of components to accurately reconstruct the topology resulting in an overly fine clustering (too many clusters) of the leaf nodes. These two situations are likely to occur especially when the number of samples are limited.

C. Progressive Search Algorithm

We deal with the first situation by a *progressive search* method. Let the estimated components in $f_{\mathrm{FM}}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$ be sorted by ascending order of their means, i.e., component 1 has the smallest mean and component k has the largest. The search starts with treating $B_1 = \{1\}$ as the intercluster component and estimating a preclustering $K_p(\mathbf{B}_1)$. Then expand the component subset to $B_2 = \{1, 2\}$ and estimate another preclustering $K_p(\mathbf{B}_2)$. Repeat this procedure until $B_k = \{1, \ldots, k\}$, which includes all the components in $f_{\mathrm{FM}}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$. Then, we select the preclusterings with the least number of clusters as the *preclustering estimates*, i.e., $\hat{\mathbf{K}}_p = \{\mathbf{K}_p(\mathbf{B}_i) : |\mathbf{K}_p(\mathbf{B}_i)| \leq |\mathbf{K}_p(\mathbf{B}_j)|, \forall i, j \in \{1, \ldots, k\}, i \neq j\}$.

A complete graph can be drawn from each preclustering estimate. Now, the vertices may represent clusters of leaf nodes. Let $H(\mathbf{K}_p(\mathbf{B}_i))$ be the complete graph whose vertices represent the clusters in $K_p(\mathbf{B}_i)$ and edge weights are computed using B_i. Then, the graphs $H(\mathbf{K}_p(\mathbf{B}_i))$ for $K_p(\mathbf{B}_i) \in \hat{\mathbf{K}}_p$ are used as inputs of the HCS algorithm. The output with the highest $\mathcal{L}_k$ is adopted as the *HCS clustering estimate*, denoted by $\hat{\mathbf{K}}$. It is possible that multiple B_i's derive the same $\hat{\mathbf{K}}$. We specify the smallest set as $\hat{\mathbf{B}}$ and use it in the following postmerge algorithm.

D. Postmerge Algorithm

To address the second situation described at the end of Section IV-B, we propose a postmerge algorithm to deal with overly fine clusters. Given the optimal HCS clustering $\hat{\mathbf{K}}$, we form the complete graph $H(\hat{\mathbf{K}})$ using $\hat{\mathbf{B}}$ as the intercluster component set. Let $v_{\mathbf{A}}$ denote the vertex for cluster $\mathbf{A}$. For each cluster A, we define its closest cluster $c(\mathbf{A})$ such that $v_{c(\mathbf{A})}$ has the strongest connection to $v_{\mathbf{A}}$, i.e., $c(\mathbf{A}) = \mathrm{argmax}_{\mathbf{A}' \in \hat{\mathbf{K}}\backslash\mathbf{A}} \varpi_{\hat{\mathbf{B}}}^{(\mathbf{A},\mathbf{A}')}$. Then, for each $A \in \hat{\mathbf{K}}$, we get a new partition by merging A and $c(\mathbf{A})$. If the highest likelihood obtained by merging a pair of clusters is greater than that of $\hat{\mathbf{K}}$, we update $\hat{\mathbf{K}}$ by the corresponding new partition.

TABLE I
HIERARCHICAL TOPOLOGY ESTIMATION ALGORITHM

```
Input: {0}, V_r, Γ ;     Output: T = (V, E)

Initialize T with only one internal node
Finish ← false, I ← the set of internal nodes
while (~Finish) do {
    for all i ∈ I do {
        Estimate f_FM(Γ(d(i)); Θ̂_i)
        K̂_p ← Pre-clustering/progressive search algorithm.
        K̂, B̂ ← HCS(H(K_p))
        if (|K̂| > 1) do {K̂ ← Post-merge algorithm} }
    I ← the set of all new found internal nodes
    if I = φ do {Finish ← true}
    else {update T by inserting the new internal nodes} }
```

This process is repeated until no improvement is made by any pairwise merge.

We would like to point out that all the preclustering, progressive search and postmerge algorithms are heuristic. However, unlike the thresholds used in [2] and [3], they are based on the probability model and likelihood function. Furthermore, our algorithms are all deterministic instead of Monte Carlo, so the convergence problem of simulated methods can be avoided. The complete HTE algorithm is summarized in Table 1.

E. Asymptotic Performance of the HTE Algorithm

We discuss the asymptotic performance of our algorithm as the estimated similarity samples tend to the true similarities. Under the assumption of i.i.d. $\{\hat{\gamma}_n^{(i,j)}\}$, one scenario to increase the accuracy of the normalized samples is to collect more measurements. As $N_{\mathrm{norm}} \to \infty$, we have $\bar{\gamma}^{(i,j)} \to \gamma_{i,j}$ in probability, according to the strong law of large numbers (SLLN). Let $\gamma_{\min} = \min_{(i,j)} \gamma_{i,j}$ be the intercluster similarity. As $N_{\mathrm{norm}} \to \infty$, the indicator function $Z_1^{(i,j)}$ also converges in probability to a random variable whose value is 1 if $\mathbf{\Gamma}_{i,j} = \gamma_{\min}$, and is 0 otherwise. Hence, $\omega_1^{(i,j)} = E[Z_1^{(i,j)} | \mathbf{\Gamma}_{i,j}] \to 1$ in probability if leaf nodes i and j are in different clusters, otherwise $\omega_1^{(i,j)} \to 0$. This also indicates the complete graph converges in probability to a graph in which vertices i and j are connected with a unity edge weight if and only if leaf nodes i and j are in the same cluster; otherwise, they are not connected at all (edge weight is 0). Furthermore, in this limiting graph, a subgraph formed by leaf nodes in the same cluster is always highly connected. Then, the partition output by the HCS algorithm will converge to the correct one in probability without applying the progressive search and the postmerge algorithms. Finally, all the above implies the topology estimated by the HTE algorithm converges to the correct topology in probability as $N_{\mathrm{norm}} \to \infty$.

V. COMPUTER SIMULATIONS

A. Matlab Model Simulation

First, we simulated a small network with the simple nonbinary virtual topology shown in Fig. 5. The simulations were im-

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

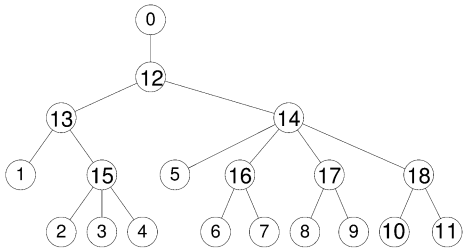


Fig. 5. Logical tree topology for the network used in computer simulations.

plemented in Matlab, and for each pair of leaf nodes, we generated 200 similarity samples as follows. Given a pair of leaf nodes (i,j), a similarity sample $\hat{\gamma}_n^{(i,j)}$ was obtained by the sum of randomly generated metric samples over all the links in the path $p_{a(i,j)}$. Each metric sample for a link was generated according to a Gaussian distribution with a randomly generated mean γ_l and a standard deviation σ_l proportional to the mean. Note that the true link metric was specified by γ_l. γ_l was generated according to a uniform distribution over a region centered at η with width equal to β. The standard deviation σ_l was obtained by multiplying γ_l with a positive factor ρ.

We implemented the proposed HTE algorithm with an averaging factor $N_{\text{norm}} = 20$ to compute the normalized similarity samples, which means for each probe tree there were ten samples of $\bar{\gamma}^{(i,j)}$. We also implemented two other topology discovery algorithms: the LBT algorithm [5], [6] and the DBT algorithm [2], [3]. The latter was originally designed for multicast networks, but can also be directly applied to unicast networks. Both algorithms estimated a binary tree given the similarity samples. A second stage was applied to generalize the binary tree by pruning the links whose metric estimates were smaller than a threshold δ. Defining $\hat{\mu}_{\text{link}}$ and $\hat{\sigma}_{\text{link}}$ to be the empirical mean and standard deviation of the estimated link metrics from the DBT or LBT over all the links, respectively, we set δ to be $\hat{\mu}_{\text{link}} - \hat{\sigma}_{\text{link}}$.

The performance of the algorithms was evaluated in terms of the tree edit distance [11] between the estimated tree and the true topology. The tree edit distance is analogous to the edit distance between two strings. A mapping from logical tree T_1 to T_2 is defined as a set of basic editing operations that allows us to transform T_1 to T_2. The basic editing operations are as follows: *replacement*—relabel a node; *insertion*—insert a node; and *deletion*—delete a nonroot node. If a mapping includes R replacements, I insertions, and D deletions, then the cost of the mapping is given by $rR + iI + dD$, where r is the cost of a replacement, i is the cost of an insertion, and d is the cost of a deletion. The set of costs is called *unit cost* if $r = i = d = 1$, which is adopted here. The tree edit distance between T_1 and T_2 is then defined as the cost of a minimum-cost mapping between them.

In the model experiment, we tested the proposed HTE algorithm, along with the DBT and LBT. We fixed the range of the uniform distribution for each link metric to the region [2], [6]. The scale factor ρ for sample standard deviation varied from 1 to 10 for link 16 and 17 and was fixed at $(1/\sqrt{2})$ for the others.

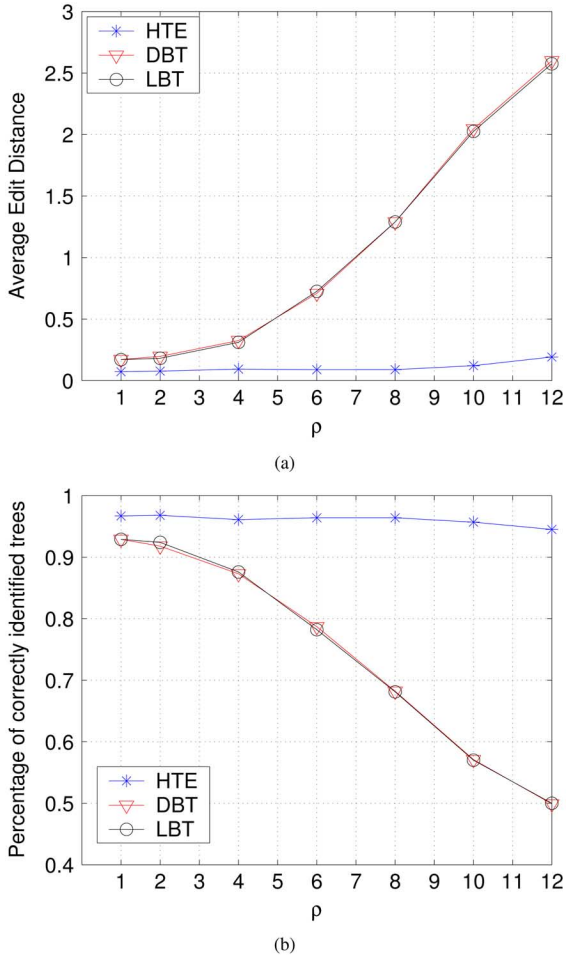


Fig. 6. (a) Average tree edit distance and (b) percentage of correctly identified trees versus the proportional factor ρ for link 16 and 17 in the Matlab simulation.

The result is illustrated in Fig. 6. Each data point was averaged from the outcomes of 1000 independent simulations. As the accuracy of topology estimation decreased with increasing ρ, HTE exhibited a minor loss in its performance while DBT and LBT both suffered from a serious degradation in their estimation capability.

Although all the three algorithms are greedy in the sense that they depend on local information to construct the topology, the DBT and LBT are both agglomerative algorithms that repeat clustering the two most similar leaf nodes in each iteration [2], [6]. This indicates they depend on a small region of the parameter space to make local decisions on the topology. On the other hand, the HTE algorithm finds a local optimum over a larger region of the parameter space which specifies the partition of a (sub)set of the leaf nodes. Therefore, the HTE estimates are generally closer to the global optimal topology than the other two algorithms.

B. NS Simulation

For a more practical environment we used ns-2 to simulate the network in Fig. 5. Two types of links were used: the links attached to the leaf nodes were assigned with bandwidth 1 Mb/s and latency 1 ms and the others were assigned with bandwidth

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

TABLE II
PARAMETERS SPECIFYING THE NUMBER OF PROBES USED IN ns SIMULATION

N_1	5	7	9	11	13	15	15	15	15	15	15	15	15	15	15	15	25	25	25	25
N_{norm}	5	5	5	5	5	5	7	9	11	13	15	15	15	15	15	15	10	15	20	25
N_2	10	10	10	10	10	10	10	10	10	10	10	12	14	16	18	20	20	20	20	20

N_1	25	25	25	25	25
N_{norm}	30	35	40	45	50
N_2	20	20	20	20	20

2 Mb/s and latency 2 ms. Each link was modelled by a first-in first-out (FIFO) queue with buffer size being 50 packets long. Cross traffic was also generated by ns-2 to simulate various network conditions. The cross traffic comprises 10% UDP streams and 90% TCP flows in terms of the bandwith utilization. The UDP streams had constant bit rates but a random noise was added to the scheduled packet departure time. The TCP flows were bursty processes with Pareto ON–OFF models. We tested the three probing schemes described in Section II: queueing delay using sandwich probes, delay variance using packet pairs, and loss rate also using packet pairs. The packet size in a packet pair probe was set to 10 B. The sizes of the large and the small packets in a sandwich probe were 500 and 10 B, respectively. The probes were sent by UDP streams with Poisson departure. The departure interval had a mean equal to eight times the transmission delay on the outgoing link of the root node. The destination was randomly selected for each probe. The parameters specifying the number of probes used in ns simulation can be found in Table II, where $N_{\rm cov} = N_{\rm loss} = N_1$ for packet pair probes, and N_2 is the total number of normalized similarity samples.

We compared the three probing schemes under three different network conditions. Each was averaged over 30 independent simulations using the HTE algorithm. Fig. 7 shows the performance in a lightly loaded, moderately loaded, and heavily loaded network, respectively. The horizontal axes denote the number $N_1 N_{\rm norm}$ of similarity estimates used in each simulation. The notation $(a/b/c)$ in the titles denotes the average condition for the whole network, where a is the average packet delay, b is the packet delay variance, and c is the packet drop rate over all the links. The legends for queueing delay, delay variance, and loss rate similarity metrics were marked by "Sandwich," "Cov," and "Loss," respectively.

As predicted in Section II, the sandwich probes provided the most reliable topology estimate in a lightly loaded network. We found some of the links had very small packet delay variances and hence could not be identified using delay variance metrics. A similar situation occurred for loss rate similarities since packet drop is rare. In a moderately loaded network, each link queue provided enough delay variation to perform topology estimation using packet pair delay variances. Fig. 7(b) shows the packet pair delay variances achieved the best performance. The error distance still converged to zero for sandwich probes, but the convergence rate was slower due to the noise introduced by background traffic. However, the packet drop rates for some links were still too low for loss rate estimates to converge. For a heavily loaded network, each link had a substantial packet drop probability which made the loss rate the most reliable similarity metric. Although the link delay variances were large, the performance of the delay variance suffered since the number of successfully received probes was significantly reduced. The sandwich probes had the worst performance due to both high packet drop rate and high delay variance. Note that some data points in the "Sandwich" and "Cov" curves were missing because, in those cases, most of the probes were lost, and there were not sufficient samples to estimate the topology.

The tree edit distance provides a way to describe the distribution of the topology estimates. To illustrate, we simulated a larger network in ns-2, whose topology is shown in Fig. 8(a). The bandwidth and latency for the internal links which are not attached to the leaf nodes were assigned 5 Mb/s and 5 ms, respectively. The edge links at the leaf nodes had bandwidth set to 1 or 2 Mb/s, and latency set to 1 or 2 ms. Similar cross traffic as before was generated to establish a light load condition. Here, we used sandwich probes to collect similarity estimates via delay differences. For each probe tree, 200 similarity estimates were collected, and we set $N_{\rm norm} = 10$ to obtain 20 normalized samples for the HTE algorithm.

For a total of M independent simulations we defined the *median topology* as the topology estimate obtained from the median of the similarity samples over all the simulations. Then the distribution of the topology estimates can be described by the one-sided probability mass function (pmf) of the tree edit distance between the estimate and the median topology. We obtained a median topology identical to the true network from 30 independent simulations. We say that this topology estimate is *median unbiased*. The topology estimate distribution is shown in Fig. 8(b) as the pmf of the edit distance to the median topology.

Finally we would like to address the effect on the performance of our model when the spatio and temporal independence assumptions in A1) and A2) are violated in the ns-2 simulations. Recall that link packet loss is considered as infinite packet queueing delay over the link. The spatial and temporal independence assumptions of delays were violated since ns-2 was configured to simulate bursty TCP background flows that cross multiple links. Our experimental results demonstrate that the proposed HTE algorithm is relatively insensitive to such violations. Indeed, the proposed finite mixture models for the normalized similarity samples are able to accurately cluster end-to-end delay samples even though the similarity estimates are computed under assumptions A1) and A2).

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

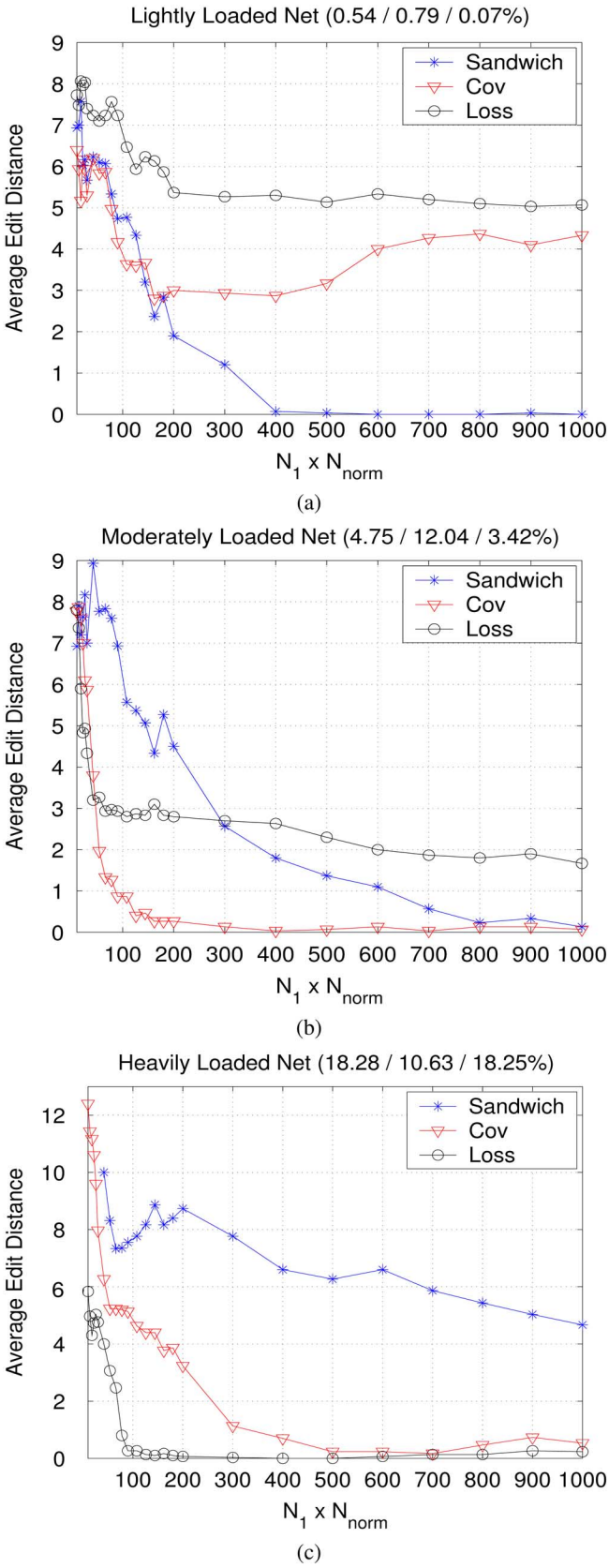


Fig. 7. Error tree distance versus the number of normalized similarity samples $N_1 N_{\text{norm}}$ for the three probing schemes.

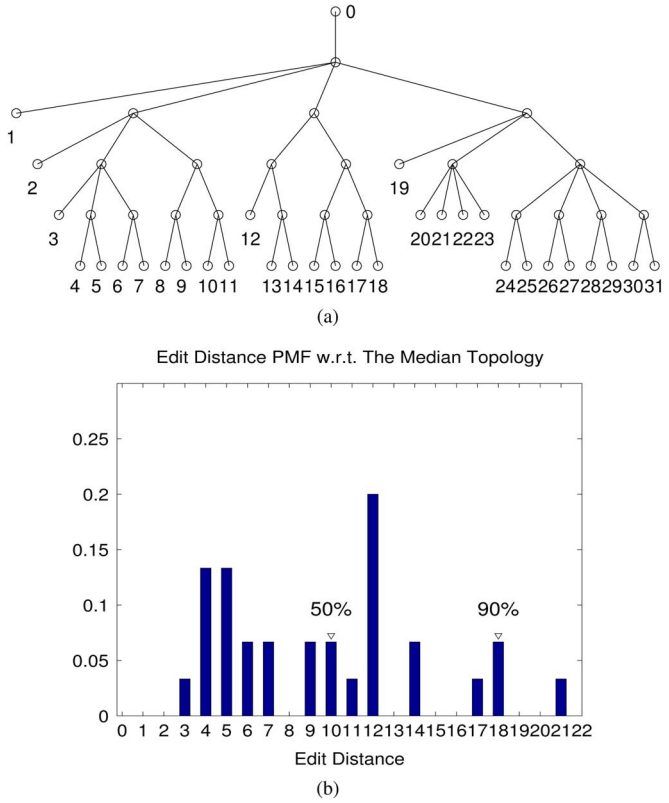


Fig. 8. (a) True topology used in ns simulation to illustrate the distribution of the topology estimates. (b) The pmf of the tree edit distance with respect to the median topology for the estimates.

VI. Conclusion and Future Work

Estimation of the logical tree topology from end-to-end unicast measurements of the network was investigated. We formulated the problem as hierarchical clustering of the leaf nodes based on pairwise similarities. A new finite mixture model was proposed for the similarity estimates, and a penalized likelihood using MML-type penalty was derived for model selection. Topology estimation was achieved by recursively finding the best partitions of the leaf nodes to expose internal node structure. We derived from the finite mixture estimate a complete graph whose vertices are the leaf nodes. A simple clustering algorithm based on the graph edge weights was then applied to partition the leaf nodes. We used Matlab and ns simulations to demonstrate the performance of the proposed algorithm.

Future work could focus on the use of hybrid probing schemes which consist of multiple types of probes. Our work could also be extended to include multiple probing sources, such as in [24]. Extensive real network experiments should be implemented in the future to compare to real network topologies.

References

[1] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," presented at the IEEE INFOCOM 1999, New York, NY, Mar. 1999.

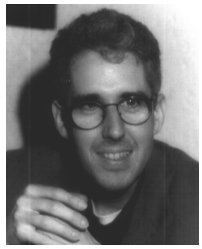
[2] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *IEEE Trans. Inf. Theory*, vol. 48, pp. 26–45, Jan. 2002.

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

[3] N. G. Duffield, J. Horowitz, and F. Lo Presti, "Adaptive multicast topology inference," presented at the IEEE INFOCOM 2001, Anchorage, AL, Apr. 2001.
[4] M. Coates, R. Castro, and R. Nowak, "Maximum likelihood network topology identification from edge-based unicast measurements," presented at the ACM Sigmetric, Marina Del Rey, CA, Jun. 2002.
[5] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Internet tomography: Recent developments," *Stat. Sci.*, vol. 19, no. 3, pp. 499–517, 2003.
[6] R. Castro, M. Coates, and R. Nowak, "Likelihood based hierarchical clustering," *IEEE Trans. Signal Processing*, vol. 52, no. 8, pp. 2308–2321, Aug. 2004.
[7] M. Shih and A. Hero, "Network topology discovery using finite mixture models," presented at the IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP) 2003, Montreal, QC, Canada, May 2003.
[8] M. Shih and A. Hero, "Topology discovery on unicast networks: A hierarchical approach based on end-to-end measurements," EECS Dept., Univ. of Michigan, Ann Arbor, MI, CSPL Tech. Rep. TR-357, Mar. 2005.
[9] M. Shih and A. O. Hero III, "Unicast-based inference of network link delay distributions with finite mixture models," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2219–2228, Aug. 2003.
[10] A. Bestavros, J. Byers, and K. Harfoush, "Inference and labeling of metric-induced network topologies," presented at the IEEE INFOCOM 2002, New York, NY, Jun. 2002.
[11] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM J. Comput.*, vol. 18, pp. 1245–1262, 1989.
[12] M. Shih, "Unicast Internet tomography," Ph.D. dissertation, EECS Dept., Univ. of Michigan, Ann Arbor, MI, 2005.
[13] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering : A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.
[14] N. G. Duffield and F. Lo Presti, "Network tomography from measured end-to-end delay covariance," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 978–992, Dec. 2004.
[15] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from end-to-end measurements," presented at the Proc. IP Traffic Measurement, Modeling, Management, Monterey, CA, Sep. 2000.
[16] G. McLachlan and D. Peel, *Finite mixture models*. New York: Wiley, 2000.
[17] M. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 381–396, Mar. 2002.
[18] J. J. Oliver, R. A. Baxter, and C. S. Wallace, "Unsupervised learning using MML," in *Proc. 13th Int. Conf. Machine Learning (ICML)*, 1996, pp. 364–372.
[19] A. D. Lanterman, "Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model order estimation," *Int. Stat. Rev.*, vol. 69, pp. 185–212, Aug. 2001.
[20] E. Hartuv and R. Shamir, "A clustering algorithm based on graph connectivity," *Inf. Process. Lett.*, vol. 76, no. 4–6, pp. 175–181, Dec. 2000.
[21] M. Brinkmeier, "Communities in graphs," Technical University Ilmenau, Ilmenau, Germany, Tech. Rep., 2002 [Online]. Available: http://eiche.theoinf.tu-ilmenau.de/~mbrinkme/documents/communities.pdf, [Online]. Available:
[22] H. Nagamochi and T. Ibaraki, "Linear time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph," *Algorithmica*, vol. 7, no. 5–6, pp. 583–596, 1992.
[23] M. Stoer and F. Wagner, "A simple min-cut algorithm," *J. ACM*, vol. 44, no. 4, pp. 585–591, Jul. 1997.
[24] M. Rabbat, R. Nowak, and M. Coates, "Multiple source, multiple destination network tomography," presented at the IEEE INFOCOM, Hong Kong, Mar. 2004.

Meng-Fu Shih was born in Taipei, Taiwan, R.O.C., in 1972. He received the B.S. degree from Tatung Institute of Technology, Taipei, Taiwan, R.O.C. in 1994 and the M.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C. in 1996, both in electrical engineering, and the Ph.D. degree in electrical engineering (systems) from the University of Michigan, Ann Arbor, in 2005.

Since October 2005, he has been with KLA-Tencor Corporation, San Jose, CA, where he is a Research Scientist. His research interests include optimization, statistical signal processing, estimation, and detection.



Alfred O. Hero, III (S'79–M'80–SM'96–F'98) received the B.S. (*summa cum laude*) from Boston University, Boston, MA, in 1980 and the Ph.D. degree from Princeton University, Princeton, NJ, in 1984, both in electrical engineering.

Since 1984, he has been with the University of Michigan, Ann Arbor, where he is a Professor in the Department of Electrical Engineering and Computer Science and, by courtesy, in the Department of Biomedical Engineering and the Department of Statistics. He has held visiting positions at I3S University of Nice, Sophia-Antipolis, France (2001); Ecole Normale Supérieure de Lyon (1999), Lyon, France; Ecole Nationale Supérieure des Télécommunications, Paris, France (1999); Scientific Research Labs of the Ford Motor Company, Dearborn, MI (1993); Ecole Nationale Superieure des Techniques Avancees (ENSTA), Paris, France; Ecole Superieure d'Electricite, Paris, France (1990); and the Massachusetts Institute of Technology Lincoln Laboratory, Cambridge, MA (1987–1989). His recent research interests have been in areas including inference for sensor networks, bioinformatics, and statistical signal and image processing.

Dr. Hero is a member of Tau Beta Pi, the American Statistical Association (ASA), the Society for Industrial and Applied Mathematics (SIAM), and the U.S. National Commission (Commission C) of the International Union of Radio Science (URSI). He received the IEEE Signal Processing Society Meritorious Service Award in 1998, the IEEE Signal Processing Society Best Paper Award in 1998, the IEEE Third Millennium Medal, and a 2002 IEEE Signal Processing Society Distinguished Lecturership. He is currently President of the IEEE Signal Processing Society (2006–2008).

Authorized licensed use limited to: University of Michigan Library. Downloaded on May 5, 2009 at 11:02 from IEEE Xplore. Restrictions apply.