Topology Discovery on Unicast Networks: A Hierarchical Approach Based on End-to-End Measurements

Meng-Fu Shih, Alfred O. Hero III
Department of Electrical Engineering and Computer Science
University of Michigan at Ann Arbor
email: mfshih@umich.edu, hero@eecs.umich.edu

March 15, 2005

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 pair-wise correlations as similarity metrics. We investigate three types of similarity metrics: queueing delay measured by sandwich probes, delay variance measured by packet pairs, and loss rate measured also by packet pairs. Unlike previous work which first assumes the network topology is a binary tree and then tries to generalize to a non-binary tree, we provide a framework which directly deals with general logical tree topologies. Based on our proposed finite mixture model for the set of similarity measurements we develop a penalized hierarchical topology likelihood that leads to a natural clustering of the leaf nodes level by level. A hierarchical algorithm to estimate the topology is developed in a similar manner by finding the best partitions of the leaf nodes. Our simulations show that the algorithm is more robust than binary-tree based methods. The three types of similarity metrics are also evaluated under various network load conditions using `ns-2`.

1 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 on-line 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, 2, 3, 4, 5, 6].

Estimation of internal network statistics or parameters using end-to-end information belongs to the research category called *network tomography* [7]. This type of network inference strongly resembles a standard *inverse problem* dealing with estimated internal parameters of systems from external measurements. Network tomog-

raphy approaches have been applied to the estimation of link statistics such as packet drop rates [8, 9, 10] and delay distributions [11, 12, 13].

[1] and [3] provided pioneer work in discovery of multicast network topologies. They specifically targeted the identification of the network's logical tree structures. A logical network structure is an abstraction from the physical topology which shows only the nodes that differentiate multiple paths. In other words, a node with a single ingress and single egress link is absorbed into the link connecting its neighbors. 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 based on end-to-end loss. Those rate estimates were used by the *deterministic binary tree classification algorithm* (DBT) [2, 3, 4] to construct a binary logical tree in a bottom-up mannar. This agglomerative algorithm selects the pair of leaf nodes which suffer the most severe loss on the shared path, and connects them to a new parent node. The parent node is then treated as a leaf node which represents the original pair of leaf nodes. The loss rates on the two newly added links are also computed. This process is repeated until only one leaf node is left, which becomes the root of the binary tree. To our knowledge all the agglomerative estimation algorithms for binary trees use similar methodology to the DBT [5]. 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 was also extended to use other metrics such as packet delays [3, 4].

The fundamental idea of topology inference falls in the framework of *metric-induced network topologies* (MINT) [14]. Every path is associated with a metric function, such as the successful transmission rate or the average delay over the path. The identifiability of the topology using DBT is then guaranteed by the monotonicity of this function. A metric is monotone if for any path its value is strictly less (such as successful transmission rates) or greater (such as delays) than that of a subpath. Such monotonicity depends on the probing scheme which estimates the shared path metric from the edge measurements. For example, when end-to-end packet loss is measured from multicast probes, packet loss rate on the shared path from root to a pair of leaf nodes can be estimated. The loss rate then has increasing mononicity given every link on the shared path has non-zero packet drop rate.

Topology estimation in unicast networks was investigated by Castro *et al.* [5, 15, 16]. They invented a method of probing, called *sandwich probes*, in which each probe sends a large packet, destined to one of the two selected leaf nodes, between two small packets, destined to the other. The second small packet always queues behind the large one until they seperate from each other. Its corresponding queueing delay over the shared probe path becomes the path metric. Under a light load assumption, this metric value equals the delay difference between the two small packets. 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) [5]. The special case of Gaussian distributed measurements was previously called the *likelihood-based binary tree algorithm* (LBT) [15]. 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 [5, 16]. The tree candidate which 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 [3] and [16] we focus on estimation of the logical tree structure of the network. Unlike previous work in which a binary tree is first estimated and then extended to a general tree using heuristic thresholds or Monte-Carlo methods, our method accomplishes direct estimation of a general binary or non-binary logical tree. 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 pair-wise similarities. Each internal node specifies a unique cluster of descendant leaf nodes which share the internal node as their common ancester. Each leaf node itself is also considered as a single-node cluster. The clusters specified by sibling nodes, i.e., nodes having the same parent, define a partition for the set of descendant leaf nodes specified by the sibling nodes' parent. Given a partition we call a pair of leaf nodes *intra-cluster* if each node in the pair is in the same cluster, otherwise they are called *inter-cluster* [5].

The similarity of a pair of leaf nodes can be represented by an MINT metric function associated with the path from the root to the nearest common ancestor of the two leaf nodes [5, 16]. Any probing scheme that produces accurate estimates of a discriminating similarity metric can be applied here. 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 pair-wise similarities used in [5, 15, 16] to include a prior distribution on the nearest common ancestor node of each pair of the leaf nodes. It results in a finite mixture model with every mixture component corresponding to a distinct internal node. An unsupervised PML-EM algorithm is developed to estimate the mixture model parameters using an MML-type of penalty for excessively high model order selection.

For the set of descendant leaf nodes specified by an internal node and the partiton defined by the internal node's children, the mixture component with the smallest mean corresponds to the inter-cluster pairs of leaf nodes. We call this component the *inter-cluster component*. To define the likelihood of a partition we express the set of leaf node pairs as the union of the set of inter-cluster pairs and the set of intra-cluster pairs for each

cluster. The partition likelihood is then formulated as the product of individual intra-cluster likelihoods times the inter-cluster likelihood. Each intra-cluster likelihood obeys a finite mixture model and the inter-cluster likelihood is determined by the inter-cluster component. A new topology likelihood is then hierarchically formulated as the product of each (conditional) partition likelihood resulting from the tree topology.

Topology estimation is performed by a recursive search for the best partitions of the leaf nodes from the top to the bottom of the logical tree. The partition algorithm utilizes a clustering procedure based on the connectivity of a complete graph derived from the finite mixture model of the leaf node similarities. To reduce the complexity in the graph-based clustering we propose a pre-clustering step which aggregates only the highly similar leaf nodes into a group. We also suggest a progressive search algorithm for the inter-cluster component, which is the key element for deriving the complete graph. If the mixture model estimate is too coarse it is possible to merge the inter-cluster component with others and produce an overly fine partition. We propose to use a post-merging algorithm to solve this problem.

To summarize we list the major differences of our algorithm from the DBT and ALT algorithms: (1) the use of hierarchical topology likelihood with finite mixture models and MML model order penalties; (2) the top-down recursive partitioning of the leaf nodes which directly yields a general logical tree for the topology estimate without using heuristic thresholds or Monte-Carlo methods; (3) the estimation of leaf node partitions using graph-based clustering and unsupervised learning of the finite mixture models; (4) the intelligent search of the partition likelihood surface using the 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 distance to the true topology and a higher percentage of correctly estimated trees when the distance is measured by a graph edit distance metric specialized to trees, called tree edit distance [17, 18, 19, 20]. The three candidate probing schemes are evaluated on a `ns-2` simulated network using our algorithm. The Monte-Carlo simulation shows the queueing delay metric measured by sandwich probes have the best performance when the network load is light. For a moderate load situation 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 graph edit distance to define the distribution of topology estimates and illustrate the idea with a network simulated in `ns-2` [21].

This paper is organized as follows. In Section 2 we set up the logical tree network model. The probing methods and the associated similarity metrics are also introduced. In Section 3 we derive the finite mixture

model for the end-to-end similarity measurements. Based on this model we define the partition likelihood and the hierarchical topology likelihood that will be used in the estimation algorithm. In Section 4 we provide the PML-EM estimation algorithm for the proposed finite mixture model. Then we illustrate the *hierarchical topology estimation algorithm* (HTE) which recursively partitions the leaf nodes based on graph connectivity. In Section 5 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 that violate the spatio temporal stationarity assumptions. Section 6 provides the conclusion and discusses future work.

2 Background

2.1 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 the root and the leaf nodes are the only accessible nodes to the estimator. There is no cooperation from routers or other devices on interior links of the network. We do not exploit any prior information on the number of the internal nodes or their interconnections.

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. $\mathbf{V}$ can be expressed as the union: $\mathbf{V} = \{0\} \cup \mathbf{V}_i \cup \mathbf{V}_r$, where the root is defined as node 0, $\mathbf{V}_i$ denotes the set of internal nodes and $\mathbf{V}_r$ is the collection of leaf nodes. We let the root be the only node having a single child node, while all internal nodes have at least two children and the leaf nodes have none. Thus in order to translate a real network into a logical tree we have to ignore all of the internal devices which have single inbound and outbound links, and connect their parent and child nodes by virtual links. Such devices are not identifiable from the end-to-end measurements.

We also define the following useful notations regarding a logical tree $T = (\mathbf{V}, \mathbf{E})$. For a node $v \in \mathbf{V} \setminus \{0\}$, let $par(v)$ be the parent node of v. Then $c(v) = \{v' \in \mathbf{V} : par(v') = v\}$ denotes the set of children of $v \in \mathbf{V}$. 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 $\mathbf{V}_{ic} = \{v \in \mathbf{V}_i : c_i(v) \neq \phi\}$ represent the set of internal nodes whose children are not all leaf nodes. We define recursively the composite function $par^n(v) = par(par^{n-1}(v))$ with $par^1(v) = par(v)$. So we say v_1 is a descendant of v_2, denoted by $v_1 \prec v_2$, if and only if $par^n(v_1) = v_2$ for some $n \in \mathbb{N}$. Let $d(v) = \{v' \in \mathbf{V}_r : v' \prec v\}$ be the set of descendant leaf nodes of $v \in \mathbf{V}$.

We adopt the following labelling scheme for the logical tree. The leaf nodes are labelled from 1 to R, where

$R = |\mathbf{V}_r|$ is the total number of leaf nodes. The labels for the internal nodes start from $R+1$ to $R+|\mathbf{V}_i|$. The links are numbered corresponding to their child end nodes, i.e., link l connects node l to $par(l)$. The topology estimation problem is illustrated in Figure 1, where the topology on the right is an example of a logical tree.

Topology estimation can be formulated as *hierarchical clustering* of the leaf nodes in which each group of nodes may be recursively partitioned into subgroups [5, 6]. The corresponding tree is known as a *dendrogram*. Each internal node v in a dendrogram specifies a unique cluster of leaf nodes $d(v)$. The clusters specified by children nodes in $c(v)$, i.e., $\{d(v') : v' \in c(v)\}$, define a partition of $d(v)$. Each leaf node itself is also considered as a cluster, called a *trivial cluster*. For example, the children of node 7 in Figure 1 divide up the leaf nodes into two groups: {1,2} and {3,4,5,6}, where the second group contains subgroups {3}, {4}, and {5,6} according to the set of child nodes $c(9)$.

Hierarchical clustering has been widely studied in many areas such as database classification and machine learning [22, 23, 24, 25, 26]. It relies on a measure of pair-wise information, e.g., similarity or inter-object distance, to partition the input objects. 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 each pair of leaf nodes can be expressed by some quantitative measure γ, called a similarity metric. Let $\gamma_{i,j}$ be the similarity metric value associated with a specific pair of leaf nodes (i, j), $i \neq j$. Assume that $\gamma_{i,j} = \gamma_{j,i}$ and $\gamma_{i,i} = \infty$. The latter indicates that an object is the most similar to itself. Given a partition of leaf nodes, define the *intra-cluster* similarities as those between two leaf nodes in the same cluster, and the *inter-cluster* similarities as those between two leaf nodes in two different clusters [5]. Figure 2 shows an example of inter-cluster and intra-cluster pairs of leaf nodes with respect to the partition specified by internal nodes 6 and 7 in the logical tree to the left. The intra-cluster pairs are denoted by 'A' and the inter-cluster pairs are denoted by 'B' in the table to the right.

In general, if the clusters are good, the inter-cluster similarities should be smaller than the intra-cluster ones. Define $\mathbf{C}$ as a hierarchical clustering of the leaf nodes, i.e., a set of groups of leaf nodes into clusters and subclusters, along with the set of $\gamma_{i,j}$ for all leaf node pairs. We propose to define a *similarity clustering tree* $T_s(\mathbf{C})$ given a hierarchical clustering $\mathbf{C}$ as follows. The root node in $T_s(\mathbf{C})$ corresponds to the top-level partition in $\mathbf{C}$, and is associated with the set of all inter-cluster 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 non-trival 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 inter-cluster similarities in the set are between two trivial clusters, otherwise it is *non-trivial*. All the $\gamma_{i,j}$'s in the same set are assumed to be equal, and they are always greater valued than those associated with

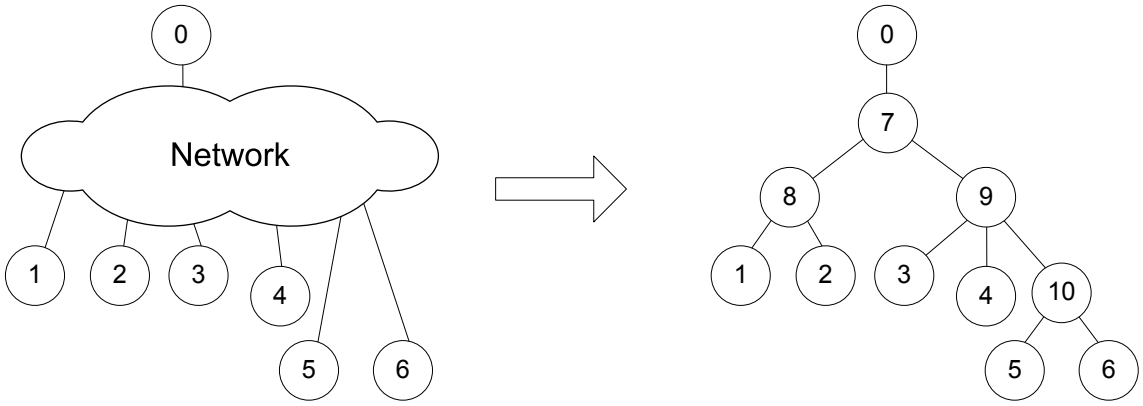
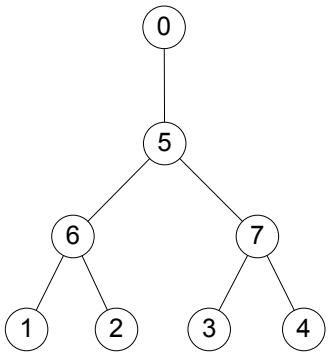


Figure 1: Illustration of the topology estimation problem. The logical tree (right) has $\mathbf{V}_r = \{1,2,3,4,5,6\}, \mathbf{V}_i = \{7,8,9,10\}$, and $\mathbf{E} = \{1,2,3,4,5,6,7,8,9,10\}$, where the links are numbered after their child end nodes.

the parent node of $T_s(\mathbf{C})$. Figure 3 shows the hierarchical clustering $\mathbf{C}$ for the leaf nodes in Figure 1 and the similarity clustering tree $T_s(\mathbf{C})$. The trivial similarity sets in $T_s(\mathbf{C})$ are $\{\gamma_{1,2}\}$ and $\{\gamma_{5,6}\}$. It is easy to verify that $T_s(\mathbf{C})$ is a bijective mapping from $\mathbf{C}$ to a tree graph, which means topology estimation is also equivalent to hierarchical grouping of the pair-wise similarities. This property will be the key to the development of our algorithm.



j \ i	1	2	3	4
1	A	A	B	B
2	A	A	B	B
3	B	B	A	A
4	B	B	A	A

Figure 2: Illustration of inter-cluster and intra-cluster pairs of leaf nodes with respect to the partition specified by internal node 6 and 7 in the tree to the left. The *intra-cluster* pairs are denoted by 'A' and the *inter-cluster* pairs are denoted by 'B' in the table to the right.

In topology estimation, the concept of *Metric-Induced Network Topology* (MINT) introduced by Bestavros *et al.* [14] provides a framework for defining the similarity metrics. Each path in an MINT is associated with a metric function. Note that each node in the similarity clustering tree corresponds to a unique internal node in the topology which is the nearest common ancester shared by each pair of the leaf nodes in the associated similarity set. It implies the following connection between the MINT and the similarities. Define $p_{i,j}$ as the directed path from node i to j for $j \prec i$, which is expressed as the union of the links intersecting the path. To

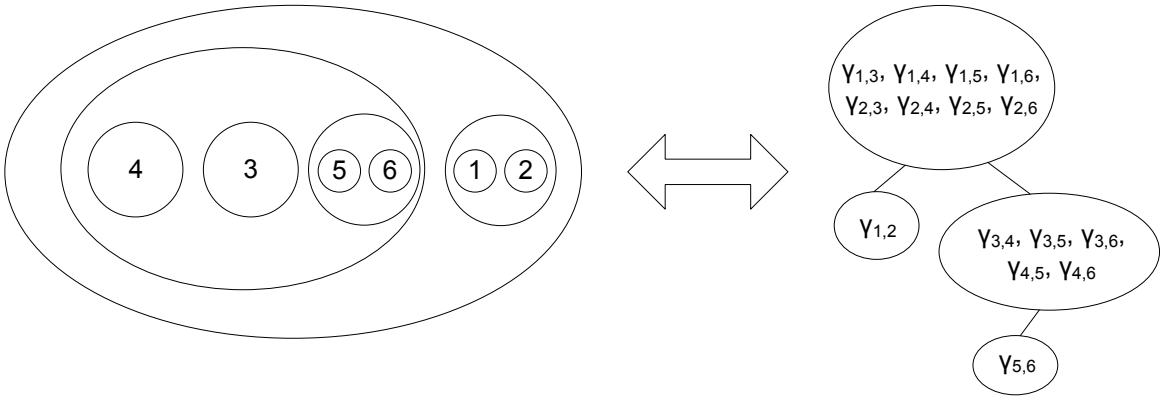


Figure 3: The hierarchical clustering $\mathbf{C}$ of the leaf nodes in Figure 1 (left) and the corresponding similarity clustering tree $T_s(\mathbf{C})$ (right).

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 $p_{a(i,j)}$ in the MINT [15, 16]. Moreover, one can observe that a deeper node in the similarity clustering tree $T_s(\mathbf{C})$ always corresponds to a longer $p_{a(i,j)}$. This means $\gamma_{i,j}$ must obey a (increasing) monotonicity property with respect to the length (number of hops) of the path $p_{a(i,j)}$. The properties of $\gamma_{i,j}$ are summarized as follows [14, 16]:

P1) **Monotonicity**: $\gamma_{i,j} < \gamma_{k,l}$ if $p_{a(i,j)}$ is a proper subpath of $p_{a(k,l)}$, for $i,j,k,l \in \mathbf{V}_r$ and $i \neq j$, $k \neq l$.

P2) **Consistency**: $\gamma_{i,j} = \gamma_{k,l}$ if $a(i,j) = a(k,l)$, for $i,j,k,l \in \mathbf{V}_r$ and $i \neq j$, $k \neq l$.

It is easy to verify that P1) and P2) are sufficient conditions for finding a unique similarity clustering tree based on the set of $\gamma_{i,j}$'s.

2.2 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 link delays of a packet over different links on its path are independent. The link delays of different packets on different links are also independent.

A2) **Temporal Independence and Stationarity**. The delays of different packets over the same link are identically and independently distributed (i.i.d.).

In our `ns-2` simulations we verify that these assumptions are not critical for satisfactory performance of the model. All of the probing schemes can be implemented in a similar way to estimate $\gamma_{i,j}$ and extract the topology. Assume each probe contains multiple unicast packets which are sent from the root node to one of the two (randomly) selected leaf nodes i and j. All the packets share the same traffic route until they diverge toward different destinations. To make it easy to convey the idea, we define the binary logical tree formed by the union of path p_i and p_j as a *probe tree*, denoted by $t_{i,j}$. There are a total of $N_T = \binom{R}{2}$ probe trees in T. Note that $p_{a(i,j)}$ is the intersection of p_i and p_j, and is called the *shared path* of probe tree $t_{i,j}$. The two branches in $t_{i,j}$ are $p_{a(i,j),i}$ and $p_{a(i,j),j}$. In Figure 4 we depict an example of a probe tree $t_{3,5}$ for sandwich probes discussed below. The shared path $p_{a(3,5)}$ includes link 6 and 8. The branch $p_{a(3,5),5}$ includes link 9 and 5, and $p_{a(3,5),3}$ is link 3. An estimate of $\gamma_{i,j}$ is computed from the end-to-end statistics collected by the probe packets, denoted by $\hat{\gamma}_n^{(i,j)}$ for the nth sample for $\gamma_{i,j}$. Moreover, unlike delay tomography, the probe source and the receivers do not need to have sychronized system clocks because a constant shift in the mean of delay measurements doesn't affect the estimate for any type of the similarity metric that is introduced below. [11, 12, 13].

2.2.1 Sandwich Probes

Sandwich probes were invented by Castro *et al.* in [15] 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. In Figure 4 the small packets A and C are sent to node 5 and the large packet B is sent to node 3. According to [15] a little space is inserted between packets A and B to reduce the possibility of packet C catching up to packet A after it seperates from B. The idea of sandwich probing is that the second small packet always queues behind the large packet until they seperate from each other at node $a(i,j)$, suppose the probe is sent along $t_{i,j}$, so the additional queueing delay suffered by the second small packet can be considered as a metric on $p_{a(i,j)}$. In the example shown in Figure 4, when the network is free from any background traffic packet B causes (deterministic) queueing delays for packet C on link 6 and 8, while packet A encounters no delay at all in any link queue. With a non-random transmission and processing delay for each link, the end-to-end delay difference between A and C is exactly the sum of the queueing delays of C at link 6 and 8. It is obvious that in this ideal case the delay difference satisfies P1) because it sums positive queueing delays over the links on $p_{a(i,j)}$, and it also satisfies P2) as long as the size of the large packet remains fixed.

When there is background traffic in the network, the queueing delays become random. The delay difference measurement can be approximated as its deterministic value in a load-free network (the signal) plus a zero mean

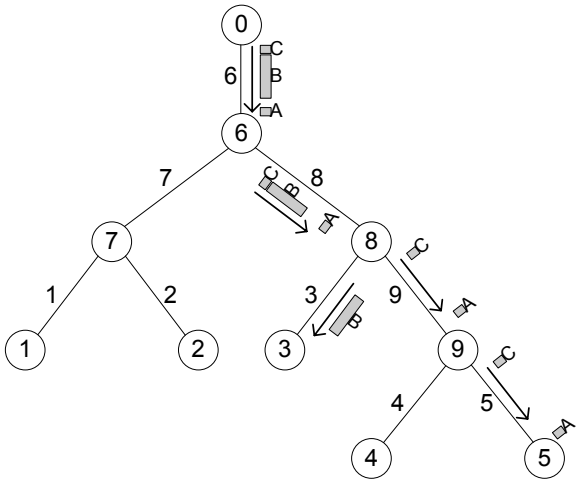


Figure 4: A sandwich probe example. The small packets A and C are sent to node 5 and the large packet B is sent to node 3. The probe tree $t_{3,5}$ is defined by the routes of the probe packets, which consists of links 3, 5, 6, 8, and 9. The metric $\gamma_{3,5}$ is the end-to-end delay difference between A and C.

random noise. It is reasonable to assume the noise has zero mean because the variability in queueing delays causes the two small packets to be further or closer to each other with approximately equal likelihood. The noise might also be assumed to be i.i.d. due to the independence and stationarity assumptions. The performance of the topology estimate will scale with the signal-to-noise ratio (SNR) defined as the square of the signal over the noise variance. As the network load grows the variance of each queue size increases and the SNR reduces accordingly. A possible failure in this situation occurs when the second small packet in the sandwich catches up to the first small one after it jettisoning the large packet. Although the queueing delay variance may drop in a saturated network where the link queues remain nearly full all the time (when the queue capacities are fixed in bytes), other failures could occur. For example, the large packet could be discarded by the network before it reaches the branching point of the probe tree. Besides, heavy packet loss may prevent us collecting enough samples in a short period of time for which the network can be considered stationary. Therefore, one can expect the sandwich probes to have the best performance in a lightly-loaded environment.

2.2.2 Packet Pair Probes

Packet pair techniques have been previously applied to flow control and bottleneck bandwidth estimation in networks [27, 28, 29]. Recently packet pairs were also used to perform in network loss and delay tomography [9, 12, 13]. A packet pair probe consists of two closely-spaced packets, generally with the same small size. They are both sent from the root node but routed to two different leaf nodes. Figure 5 depicts an example of packet pair probe. The main assumption justifying the use of packet pair probes is the following:

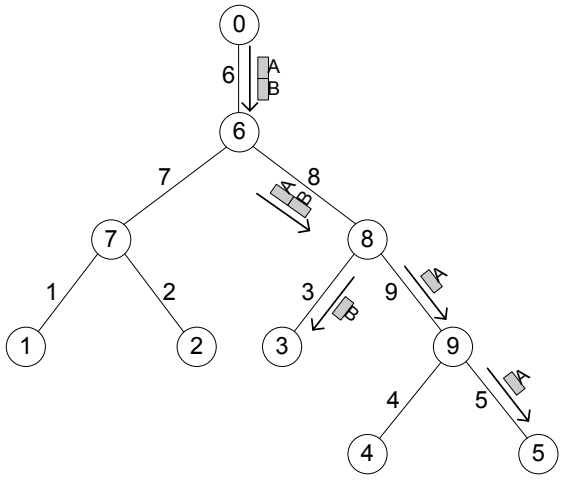


Figure 5: A packet pair probe example. Packet A and B are sent back-to-back from the root node to node 5 and 3, respectively. We assume they stay together and encounter identical delays on shared path which includes link 6 and 8.

A3) **Delay Consistency**: the queueing delays of the two packets in a packet pair probe are identical with probability 1 when they travel along the shared path.

Practically speaking, even if the two packets stay close to each other until their paths diverge, the second packet still tends to have larger queueing delays than the first one on the shared path due to possible intervening packets from other traffic flows (cross traffic). Such effects can be reduced by randomly ordering the two packets [30]. Suppose the packet pair probes are sent through the probe tree $t_{i,j}$. Let the queueing delay over the shared path $p_{a(i,j)}$ be $x_{0,i} = x_{0,a(i,j)} + x_i$ for the packet sent to node i and $x_{0,j} = x_{0,a(i,j)} + x_j$ for the packet sent to node j, where $x_{0,a(i,j)}$ denotes the queueing delay over $p_{a(i,j)}$ for the first packet in the probe. The random ordering of the two packets makes x_i and x_j have approximately the same mean, i.e., the mean of $x_i - x_j = 0$. Under the stated statistical assumptions the difference $x_{0,i} - x_{0,j}$ can be modelled as an i.i.d noise process.

The first type of metric that can be retrieved from the packet pairs is delay variance. The independence assumption A1) implies that the (queueing) delay over a path has a variance equal the sum of the delay variance for each link. This variance is also monotonically increasing with respect to the number of links in the path as long as each link introduces non-zero variance to the packet delay. The consistency property can be achieved if the delay variance over $p_{a(i,j)}$ can be calculated from end-to-end delays of packet pairs sent through $t_{i,j}$. Let $\mathbf{Y}_n^{(i,j)} = (Y_{1,n}^{(i,j)}, Y_{2,n}^{(i,j)})$ be the end-to-end delays of the nth packet pair sent along the probe tree $t_{i,j}$, where $Y_{1,n}^{(i,j)}$ and $Y_{2,n}^{(i,j)}$ denote the delays toward i and j, respectively. We also define $X_{0,n}^{(i,j)}$, $X_{1,n}^{(i,j)}$ and $X_{2,n}^{(i,j)}$ as the delays of the nth packet pair over path $p_{a(i,j)}$, $p_{a(i,j),i}$, and $p_{a(i,j),j}$, respectively. From assumptions A1)–A3) we know $X_{0,n}^{(i,j)}, X_{1,n}^{(i,j)}, X_{2,n}^{(i,j)}$ are independent for all n. So the variance of $X_{0,n}^{(i,j)}$ can be computed by

$Var(X_{0,n}^{(i,j)}) = Cov(X_{0,n}^{(i,j)} + X_{1,n}^{(i,j)}, X_{0,n}^{(i,j)} + X_{2,n}^{(i,j)}) = Cov(Y_{1,n}^{(i,j)}, Y_{2,n}^{(i,j)})$. This relationship is depicted in Figure 6.

For each probe tree $t_{i,j}$ we need N_{cov} end-to-end delay measurements to obtain a sample of the delay variance over $p_{i,j}$ using the unbiased covariance estimator

$$\hat{s}_n^{(i,j)} = \frac{\sum_{n_0=(n-1)N_{cov}+1}^{nN_{cov}} \left(y_{1,n_0}^{(i,j)} - \bar{y}_{1,n}^{(i,j)}\right)\left(y_{2,n_0}^{(i,j)} - \bar{y}_{2,n}^{(i,j)}\right)}{N_{cov} - 1}, \tag{1}$$

where $\bar{y}_{q,n}^{(i,j)} = \frac{1}{N_{cov}} \sum_{n_0=(n-1)N_{cov}+1}^{nN_{cov}} y_{q,n_0}^{(i,j)}$, for $q = 1, 2$. Experience has shown that N_{cov} is usually around 20 or 30. This means that the number of packet pairs needed to probe the network should be at least as large as that of the sandwich probes multiplied by an order, provided the same number of metric samples are collected. However, this does not necessarily imply the time length of the probing session or the overhead of the network load is also as large or larger. The reason is that the size of a sandwich probe is generally larger than the packet pair size. A typical packet pair probe consists of packets containing a few tens of bytes. However, a sandwich probe normally contains hundreds of bytes due to the need for a large middle packet. Indeed the packet in the middle of a sandwich probe needs to be large enough to induce distinguishable queueing delays. Its size requirement will become higher and higher as the network speed of communication becomes increasingly overprovisioned to accommodate larger and larger bursts of user traffic. Furthermore, the probing rate needs to be controlled to keep a minimal impact on the network. When the rate is held fixed for packet pair and sandwich schemes, it takes an order magnitude longer time to send the same number of sandwich probes as packet pair probes.

For packet pair probing the identifiability of a link in the network relies on the relative magnitude of the queueing delay variance in each link. Hence, one can expect packet pair probing to be ineffective for lightly-loaded environments where there is little background traffic to provide sufficient variation in queueing delays. On the other hand, when a link is congested, the high packet drop rate causes a similar problem for the sandwich probes. Therefore the best situation for packet pair probes with delay variance metrics is a moderately-loaded network.

The second type of metric we propose to use with the packet pair is packet loss rate. Assumption A1)-A3) can be extended by interpreting packet losses as infinite delays. It implies for a probe tree $t_{i,j}$ the packet pair either both survive (with finite delays) or both get dropped (with infinite delays) on the shared path $\gamma_{a(i,j)}$. For simplicity we do not consider the case where their loss events are correlated (with a correlation coefficient less than 1). Coates and Nowak [9] studied this correlated case in network loss tomography and our methods can be extended to this scenario once a suitable similarity metric has been identified.

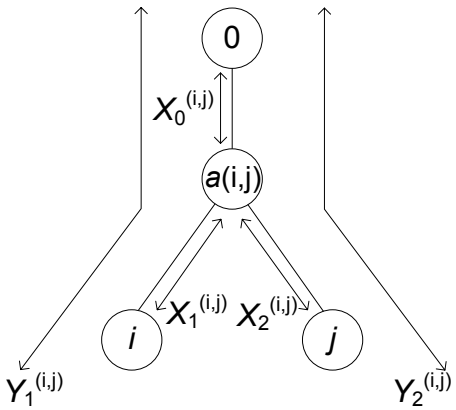


Figure 6: A *logical probe tree* $t_{i,j}$ is shown to illustrate the computation of the delay variance over $p_{a(i,j)}$ from end-to-end delays. Due to the independence of $X_0^{(i,j)}$, $X_1^{(i,j)}$, and $X_2^{(i,j)}$, $Var(X_0^{(i,j)}) = Cov(Y_1^{(i,j)}, Y_2^{(i,j)})$.

Figure 7 shows a probe tree $t_{i,j}$ with packet loss rates $q_0^{(i,j)}, q_1^{(i,j)}$, and $q_2^{(i,j)}$ over path $p_{a(i,j)}$,$p_{a(i,j),i}$, and $p_{a(i,j),j}$, respectively. Define the following probabilities:

$$\begin{aligned} u_0^{(i,j)} &= P(\{Y_{1,n}^{(i,j)} < \infty\} \wedge \{Y_{2,n}^{(i,j)} < \infty\}) \\ &= (1 - q_0^{(i,j)})(1 - q_1^{(i,j)})(1 - q_2^{(i,j)}) \\ u_1^{(i,j)} &= P(\{Y_{1,n}^{(i,j)} = \infty\} \wedge \{Y_{2,n}^{(i,j)} < \infty\}) \\ &= (1 - q_0^{(i,j)}) q_1^{(i,j)} (1 - q_2^{(i,j)}) \\ u_2^{(i,j)} &= P(\{Y_{1,n}^{(i,j)} < \infty\} \wedge \{Y_{2,n}^{(i,j)} = \infty\}) \\ &= (1 - q_0^{(i,j)})(1 - q_1^{(i,j)}) q_2^{(i,j)}. \end{aligned} \tag{2}$$

Let $\hat{u}_0^{(i,j)}, \hat{u}_1^{(i,j)}, \hat{u}_2^{(i,j)}$ be empirical estimates of $u_0^{(i,j)}, u_1^{(i,j)}, u_2^{(i,j)}$, respectively. Then the packet loss rate over $p_{a(i,j)}$ can be estimated by

$$\hat{q}_0^{(i,j)} = 1 - \frac{(\hat{u}_0^{(i,j)} + \hat{u}_1^{(i,j)})(\hat{u}_0^{(i,j)} + \hat{u}_2^{(i,j)})}{\hat{u}_0^{(i,j)}}, \qquad \text{for } \hat{u}_0^{(i,j)} \neq 0. \tag{3}$$

Eqn. (3) implies the consistency property for the loss metric, and the monotonicity property holds provided each link has non-zero packet drop probability. Similar to the covariance estimator in (1), one needs to collect N_{loss} packet pairs for a single sample of $\hat{q}_0^{(i,j)}$. Unlike the previously discussed two schemes, packet loss now provides sufficient information to identify the topology for highly congested networks.

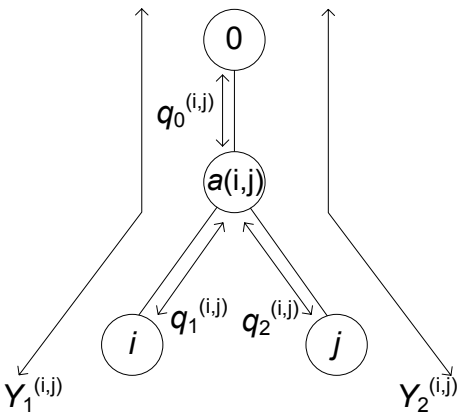


Figure 7: A *logical probe tree* $t_{i,j}$ is shown to illustrate the estimation of the packet loss rate over $p_{a(i,j)}$ from end-to-end loss rates.

3 Hierarchical Topology Likelihood Using Finite Mixture Models

3.1 Finite Mixture Model for Similarity Estimation

Modelling the behavior and dynamics of today's network has been an active research area for decades [31, 32, 33, 34, 35]. It is a difficult task due to the large-scale heterogeneity and non-stationarity of the network environment. Several delay models in the network has been proposed based on theoretical analysis and/or real network data (see, e.g., [36, 37, 38]). Most of them are theoretically justified for an M/M/1 or other queueing system. Others are heuristically justified based on measurements of a given type of the network. Some combined elements of both theoretical and experimental modelling. So far there is no evidence known to us that any existing delay model will be applicable to all traffic conditions. An exception is modelling packet loss in a stationary network. It is widely accepted to use Bernoulli process as the packet loss model for packet loss in stationary environments [8, 9, 10].

To establish a simple and unified framework for metrics based on either packet delay or loss, we adopt the following strategy. Firstly observe that the metric samples $\hat{\gamma}_n^{(i,j)}$ estimated from data along probe tree $t_{i,j}$, e.g., the sandwich delay difference, packet pair delay covariance in (1), or packet loss rate in (3), 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 (C.L.T.). We call the averaged samples *normalized similarity samples*, denoted by $\bar{\gamma}_n^{(i,j)}$ for the nth sample for $\gamma_{i,j}$. Note that to compute one normalized sample it requires N_{norm} sandwich probes, $N_{norm}N_{cov}$ packet pairs using covariance metric, or $N_{norm}N_{loss}$ packet pairs using loss rate metric. $\bar{\gamma}_n^{(i,j)}$ are also i.i.d. for all n. Secondly, we find if $a(i,j) = a(k,l) = v$, then $\hat{\gamma}_n^{(i,j)}$ and $\hat{\gamma}_n^{(k,l)}$ have the same mean as $\mu_v = \gamma_{i,j}$. However, there is no guarantee that their variances are also the same. As one averages $\hat{\gamma}_n^{(i,j)}$ and $\hat{\gamma}_n^{(k,l)}$ their variances decrease linearly by a factor of N_{norm} as does

the variance of the difference between them. With N_{norm} being large, as required by the C.L.T., this difference becomes negligible. We will make the large N_{norm} assumption that $\bar{\gamma}_n^{(i,j)}$ and $\bar{\gamma}_n^{(k,l)}$ have approximately identical Gaussian distribution with mean μ_v and variance σ_v^2. We state this approximation formally as a Lemma:

Lemma 1. Let $\bar{\gamma}^{(i,j)}$ be the average of N_{norm} i.i.d. metric samples $\hat{\gamma}^{(i,j)}$. As $N_{norm} \to \infty$, $\bar{\gamma}^{(i,j)}$ and $\bar{\gamma}^{(k,l)}$ become equal in (Gaussian) distribution if $a(i,j) = a(k,l)$, for $i,j,k,l \in \mathbf{V}_r$ and $i \neq j$, $k \neq l$.

A simple model can be drawn from Lemma 1 as follows. Let the set of normalized similarity samples for $\gamma_{i,j}$ be $\mathbf{\Gamma}_{i,j} = \{\bar{\gamma}_n^{(i,j)}\}_n$, and let $\mathbf{\Gamma} = \{\mathbf{\Gamma}_{i,j}\}_{(i,j)}$. Also define $N_{i,j} = |\mathbf{\Gamma}_{i,j}|$ and $N = \sum_{i<j} N_{i,j}$. For an internal node v in the logical tree network, we define $T(v) = \{t_{i,j} : a(i,j) = v,\ i < j,\ i,j \in \mathbf{V}_r\}$ as the set of probe trees whose branches split at node v. Let $N_T(v) = |T(v)|$. The set of normalized similarity samples for $T(v)$ is $\mathbf{\Gamma}(v) = \{\mathbf{\Gamma}_{i,j} : t_{i,j} \in T(v)\}$. Let $N(v) = |\mathbf{\Gamma}(v)|$. According to Lemma 1 the samples in $\mathbf{\Gamma}(v)$ are i.i.d. realizations of a Gaussian distribution with parameters $\theta_v = (\mu_v, \sigma_v^2)$. Therefore a simple model for $\mathbf{\Gamma}$ can be formulated by

$$f_S(\mathbf{\Gamma}) = \prod_{v \in \mathbf{V}_i} \phi(\mathbf{\Gamma}(v); \theta_v), \tag{4}$$

where ϕ denotes the Gaussian pdf and $\phi(\mathbf{\Gamma}(v);\theta_v) = \prod_{(i,j):t_{i,j} \in T(v)} \prod_{n=1}^{N_{i,j}} \phi(\bar{\gamma}_n^{(i,j)};\theta_v)$. For the example in Figure 1, $f_S(\mathbf{\Gamma}) = \phi(\mathbf{\Gamma}(7);\theta_7)\phi(\mathbf{\Gamma}(8);\theta_8)\phi(\mathbf{\Gamma}(9);\theta_9)\phi(\mathbf{\Gamma}(10);\theta_{10})$, where $\mathbf{\Gamma}(7) = \{\mathbf{\Gamma}_{1,3}, \mathbf{\Gamma}_{1,4}, \mathbf{\Gamma}_{1,5}, \mathbf{\Gamma}_{1,6}, \mathbf{\Gamma}_{2,3}, \mathbf{\Gamma}_{2,4}, \mathbf{\Gamma}_{2,5}, \mathbf{\Gamma}_{2,6}\}$, $\mathbf{\Gamma}(8) = \{\mathbf{\Gamma}_{1,2}\}$, $\mathbf{\Gamma}(9) = \{\mathbf{\Gamma}_{3,4}, \mathbf{\Gamma}_{3,5}, \mathbf{\Gamma}_{3,6}, \mathbf{\Gamma}_{4,5}, \mathbf{\Gamma}_{4,6}\}$, and $\mathbf{\Gamma}(10) = \{\mathbf{\Gamma}_{5,6}\}$. Eqn. (4) is similar to the model used in [5]. The problem in topology estimation is then to simultaneously determine the $\mathbf{V}_i$, $\{a(i,j)\}$, and $\{\theta_v\}$ which maximize the likelihood. Although Eqn. (4) is simple and straightforward, it is not suitable for efficient estimation algorithms. One must determine the topology, i.e., $\mathbf{V}_i$ and $\{a(i,j)\}$, before maximizing the likelihood with respect to $\{\theta_v\}$ because $\phi(\cdot;\theta_v)$ is only evaluated at $\mathbf{\Gamma}(v)$. Therefore the global maximum likelihood (ML) estimate can only be found by either exhaustive or Monte-Carlo search [5].

Herein we provide an alternative model which will be used later to develop an unsupervised estimation algorithm. Suppose the set of internal nodes $\mathbf{V}_i$ is known and the cardinality $|T(v)|$ is provided for each $v \in \mathbf{V}_i$, but there is no knowledge of the common parent node $a(i,j)$ for any pair of leaf nodes (i,j). A reasonable prior distribution of $a(i,j)$ is $\alpha(v) = P(a(i,j) = v) = \frac{N_T(v)}{N_T}$ for $v \in \mathbf{V}_i$. Given $f(\mathbf{\Gamma}_{i,j}|a(i,j) = v) = \phi(\mathbf{\Gamma}_{i,j};\theta_v)$ we have $f(\mathbf{\Gamma}_{i,j}) = \sum_{v \in \mathbf{V}_i} \alpha(v)\phi(\mathbf{\Gamma}_{i,j};\theta_v)$, which is known as a finite mixture model (see, e.g., [39]). 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

$$f_{FM}(\mathbf{\Gamma}) = \prod_{\substack{i,j\in\mathbf{V}_r \\ i<j}} \sum_{v\in\mathbf{V}_i} \alpha(v)\phi(\mathbf{\Gamma}_{i,j};\theta_v). \tag{5}$$

Note that the model order for the similarities estimated from each probe tree $t_{i,j}$ 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 $a(i,j) = v$. For the example in Figure 1, using the mixture model gives $f_{FM}(\mathbf{\Gamma}) = \prod_{i,j\in\{1,...,6\},i<j} \left[\frac{8}{15}\phi(\mathbf{\Gamma}_{i,j};\theta_7) + \frac{1}{15}\phi(\mathbf{\Gamma}_{i,j};\theta_8) + \frac{5}{15}\phi(\mathbf{\Gamma}_{i,j};\theta_9) + \frac{1}{15}\phi(\mathbf{\Gamma}_{i,j};\theta_{10})\right]$. The key difference between the models in (4) and (5) is that in $f_{FM}(\mathbf{\Gamma})$ 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 by unsupervised estimation of the mixture model, which can be implemented using the expectation-maximization (EM) algorithm [40, 41]. To discover the topology, the similarity sets in $T_s(\mathbf{C})$ are determined by associating each $a(i,j)$ with the component that the probability of $\mathbf{\Gamma}_{i,j}$ being contributed by it is the maximum over all the components in $f_{FM}(\mathbf{\Gamma})$.

3.2 The MML Penalized Likelihood for The Mixture Model

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

$$f_{FM}(\mathbf{\Gamma};\mathbf{\Theta}) = \prod_{\substack{i,j\in\mathbf{V}_r \\ i<j}} \sum_{m=1}^{k} \alpha_m\phi(\mathbf{\Gamma}_{i,j};\theta_m), \tag{6}$$

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 optimzing the α_m's over discrete values in the EM algorithm, we assume $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_k)$ is continuously distributed over the region $0 \le \alpha_m \le 1, \sum_{m=1}^{k} \alpha_m = 1$. 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 ML estimate of $\mathbf{\Theta}$ with a given k can be obtained by using the EM algorithm [40, 42] which generates a sequence of estimates with nondecreasing likelihoods.

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 [43]. We adopt a criterion called Minimum Message Length (MML) [44] to derive the penalty function.

MML has been widely used in unsupervised learning of mixture models [12, 40, 41, 45]. The incomplete data penalized log-likelihood is generally expressed as [40]

$$\tilde{\mathcal{L}}(\mathbf{Y};\boldsymbol{\Theta}) \stackrel{\text{def}}{=} \log f(\boldsymbol{\Theta}) + \log f(\mathbf{Y}|\boldsymbol{\Theta}) - \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\Theta})| - \frac{c}{2}(1+\log\kappa_c) \tag{7}$$

for observed data $\mathbf{Y}$ and parameter set $\boldsymbol{\Theta}$, where $\mathbf{I}(\boldsymbol{\Theta})$ is the Fisher information matrix (FIM) associated with $\mathbf{Y}$, $|\cdot|$ denotes the determinant operator, c is the dimension of $\boldsymbol{\Theta}$, and κ_c is the so-called *optimal quantizing lattice constant for* $\Re^c$.

For a given model order k our choice for the prior distributions of the parameters follows the approach of [41]. The mixing probabilities in $\boldsymbol{\alpha}$ have a uniform prior:

$$f(\boldsymbol{\alpha}) = (k-1)! \quad \text{for} \quad 0 \le \alpha_m \le 1,\ \forall m = 1,\ldots,k, \text{ and } \sum_{m=1}^{k} \alpha_m = 1.$$

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 $\boldsymbol{\Gamma}$:

$$f(\sigma_m) = \frac{1}{\sigma_p} \quad \text{for} \quad 0 \le \sigma_m \le \sigma_p.$$

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 $\boldsymbol{\Gamma}$:

$$f(\mu_m) = \frac{1}{2\sigma_p} \quad \text{for} \quad \mu_p - \sigma_p \le \mu_m \le \mu_p + \sigma_p.$$

The prior for the model order k is assumed uniform between two pre-determined bounds k_{min} and k_{max}. It is a constant and can be ignored. With the assumption that the parameters are independent, we have

$$f(\boldsymbol{\Theta}) = \frac{(k-1)!}{2^k \sigma_p^{2k}}. \tag{8}$$

In general it is difficult to derive a closed form for the FIM of finite mixture models with more than one components. The authors of [41] suggested to replace 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

$$|\mathbf{I}_{FM}(\boldsymbol{\Theta})| \approx |\mathbf{I}_0(\boldsymbol{\alpha})| \times \prod_{m=1}^{k} |\mathbf{I}_m(\theta_m)|, \tag{9}$$

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 having associated parameters θ_m. $\mathbf{I}_m(\theta_m)$ can be expressed as

$$\mathbf{I}_m(\theta_m) = \sum_{\substack{i,j \in \mathbf{V}_r \\ i<j}} \mathbf{I}_m^{(i,j)}(\theta_m), \tag{10}$$

where $\mathbf{I}_m^{(i,j)}(\theta_m)$ is the FIM associated with $\mathbf{\Gamma}_{i,j}$ for the mth component:

$$\mathbf{I}_m^{(i,j)}(\theta_m) = \begin{bmatrix} \frac{\alpha_m N_{i,j}}{\sigma_m^2} & 0 \\ 0 & \frac{2\alpha_m N_{i,j}}{\sigma_m^2} \end{bmatrix}. \tag{11}$$

Therefore

$$|\mathbf{I}_m(\theta_m)| = \frac{2\alpha_m^2 N^2}{\sigma_m^4}. \tag{12}$$

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

$$|\mathbf{I}_0(\boldsymbol{\alpha})| = \frac{N_T}{\prod_{m=1}^k \alpha_m}. \tag{13}$$

We do not require a particular ordering on the components, so the corresponding factorial term in the MML expression (7) can be removed. Since there are a total of $k!$ possible permutations for the components, the likelihood penalty is then decreased by $\log(k!)$. Besides, we also approximate κ_c by $\kappa_1 = \frac{1}{12}$ as in [41]. Substituting (8)-(13) into (7) we have

$$\begin{aligned} \mathcal{L}_p(\mathbf{\Gamma};\mathbf{\Theta}) &= \log f_{FM}(\mathbf{\Gamma};\mathbf{\Theta}) + \log\frac{(k-1)!}{2^k\sigma_p^{2k}} + \log(k!) - \frac{1}{2}\log N_T - k\left(\log\sqrt{2} + \log N\right) \\ &\quad -\frac{1}{2}\sum_{m=1}^k \log\alpha_m + \sum_{m=1}^k \log\sigma_m^2 - \frac{3k}{2}(1-\log 12). \end{aligned} \tag{14}$$

The details of the algorithm maximizing (14) over $\mathbf{\Theta}$ will be provided in the next section.

3.3 The Hierarchical Topology Likelihood

Eqn. (14) is difficult to use directly for topology estimation due to identifiability problems even with the over-modelling MML penalties. Recall that each internal node in the topology corresponds to a unique component in the finite mixture model. Consider the example in Figure 1 once again. If $\gamma_{1,2} = \gamma_{5,6}$, the estimates $\bar{\gamma}_n^{(1,2)}$ and $\bar{\gamma}_n^{(5,6)}$ admit identical Gaussian distribution using an approximation similar to Lemma 1. Hence we are not able to find the correct set of internal nodes because the two mixture components 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 into groups having common similarities.

First consider a group of leaf nodes $\mathbf{G}$. Let $\boldsymbol{\gamma}(\mathbf{G}) = \{\gamma_{i,j} : \ i < j, \ i,j \in \mathbf{G}\}$ be the set of pair-wise 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 $K_d, \ d = 1, \ldots, D$ are disjoint subsets of $\mathbf{G}$ which may contain

subclusters. 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 inter-cluster $\gamma_{i,j}$'s share the smallest value in $\boldsymbol{\gamma}(\mathbf{G})$. Hence the set of all inter-cluster $\boldsymbol{\Gamma}_{i,j}$'s, denoted by $\boldsymbol{\Gamma}_0(\mathbf{K})$, obey a Gaussian distribution which has the smallest mean over $\boldsymbol{\Gamma}(\mathbf{G})$. This means for the finite mixture model of $\boldsymbol{\Gamma}(\mathbf{G})$ the component with the smallest mean contributes $\boldsymbol{\Gamma}_0(\mathbf{K})$. We call this component the *inter-cluster component* of $f_{FM}(\boldsymbol{\Gamma}(\mathbf{G}))$ and let $\boldsymbol{\Theta}_0(\mathbf{K})$ denote its parameter set. All other components are contributed by the intra-cluster $\boldsymbol{\Gamma}_{i,j}$'s. Let K_l be a cluster with two or more leaf nodes, i.e., $l \in \{1, \ldots, D'\}$. The set of intra-cluster $\boldsymbol{\Gamma}_{i,j}$'s in K_l, denoted by $\boldsymbol{\Gamma}_l(\mathbf{K})$, also follows a finite mixture model by itself. Let $\boldsymbol{\Theta}_l(\mathbf{K})$ be the mixture parameter set for $f_{FM}(\boldsymbol{\Gamma}_l(\mathbf{K}))$. If K_l has exactly two leaf nodes or all the subclusters in K_l are trivial, i.e., single leaf node subclusters, $f_{FM}(\boldsymbol{\Gamma}_l(\mathbf{K}))$ degenerates to a single component density function. We define the penalized *partition likelihood* as:

$$\mathcal{L}_k(\boldsymbol{\Gamma}(\mathbf{G}); \mathbf{K}, \boldsymbol{\Theta}(\mathbf{K})) \stackrel{\text{def}}{=} \mathcal{L}_p(\boldsymbol{\Gamma}_0(\mathbf{K}); \boldsymbol{\Theta}_0(\mathbf{K})) + \sum_{l=1}^{D'} \mathcal{L}_p(\boldsymbol{\Gamma}_l(\mathbf{K}); \boldsymbol{\Theta}_l(\mathbf{K})), \tag{15}$$

where $\boldsymbol{\Theta}(\mathbf{K}) = (\boldsymbol{\Theta}_0(\mathbf{K}), \ldots, \boldsymbol{\Theta}_{D'}(\mathbf{K}))$. For example, the partition specified by the children of node 7 in Figure 1 divides up $\mathbf{G}_1 = \{1,2,3,4,5,6\}$ into two clusters $\mathbf{K}_1 = \{K_{1,1}, K_{1,2}\} = \{\{1,2\}, \{3,4,5,6\}\}$ with $\boldsymbol{\Gamma}(\mathbf{G}_1) = \boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0(\mathbf{K}_1) = \boldsymbol{\Gamma}(7)$, $\boldsymbol{\Gamma}_1(\mathbf{K}_1) = \boldsymbol{\Gamma}(8)$, and $\boldsymbol{\Gamma}_2(\mathbf{K}_1) = \boldsymbol{\Gamma}(9) \cup \boldsymbol{\Gamma}(10)$. Hence

$$\mathcal{L}_k(\boldsymbol{\Gamma}(\mathbf{G}_1); \mathbf{K}_1, \boldsymbol{\Theta}(\mathbf{K}_1)) = \sum_{i=0}^{2} \mathcal{L}_p(\boldsymbol{\Gamma}_i(\mathbf{K}_1); \boldsymbol{\Theta}_i(\mathbf{K}_1)). \tag{16}$$

Similarly for cluster $K_{1,2}$, which has a subclustering $\mathbf{K}_2 = \{K_{2,1}, K_{2,2}, K_{2,3}\} = \{\{5,6\}, \{3\}, \{4\}\}$ due to common parent node 9, we can also formulate its (conditional) partition likelihood as

$$\mathcal{L}_k(\boldsymbol{\Gamma}(\mathbf{G}_2); \mathbf{K}_2, \boldsymbol{\Theta}(\mathbf{K}_2) | \mathbf{K}_1) = \mathcal{L}_p(\boldsymbol{\Gamma}_0(\mathbf{K}_2); \boldsymbol{\Theta}_0(\mathbf{K}_2)) + \mathcal{L}_p(\boldsymbol{\Gamma}_1(\mathbf{K}_2); \boldsymbol{\Theta}_1(\mathbf{K}_2)), \tag{17}$$

where $\mathbf{G}_2 = \{3,4,5,6\}$, $\boldsymbol{\Gamma}(\mathbf{G}_2) = \boldsymbol{\Gamma}_2(\mathbf{K}_1)$, $\boldsymbol{\Gamma}_0(\mathbf{K}_2) = \boldsymbol{\Gamma}(9)$, and $\boldsymbol{\Gamma}_1(\mathbf{K}_2) = \boldsymbol{\Gamma}(10)$. Recall that the similarity sets associated with a similarity clustering tree $T_s(\mathbf{C})$ are sets of inter-(sub)cluster similarities. Comparing (16) and (17) to the $T_s(\mathbf{C})$ in Figure 3, the inter-cluster term in each partition likelihood represents the likelihood of the normalized samples for a unique similarity set associated with $T_s(\mathbf{C})$.

This example motivates the following *hierarchical topology likelihood* for the logical tree in Figure 1:

$$\mathcal{L}_T(\boldsymbol{\Gamma}; T, \boldsymbol{\Theta}(T)) = \mathcal{L}_k(\boldsymbol{\Gamma}(\mathbf{G}_1); \mathbf{K}_1, \boldsymbol{\Theta}(\mathbf{K}_1)) + \mathcal{L}_k(\boldsymbol{\Gamma}(\mathbf{G}_2); \mathbf{K}_2, \boldsymbol{\Theta}(\mathbf{K}_2) | \mathbf{K}_1), \tag{18}$$

where $\boldsymbol{\Theta}(T) = \{\boldsymbol{\Theta}(\mathbf{K}_1), \boldsymbol{\Theta}(\mathbf{K}_2)\}$. $\mathcal{L}_T$ recursively sums up the partition likelihoods whose inter-cluster terms correspond to non-trival similarity sets. Those $\mathcal{L}_k$'s corresponding to trivial similarity sets are not included

because they already appear as intra-cluster terms in (18). For a logical tree $T = (\mathbf{V}, \mathbf{E})$ the general formula for the hierarchical topology likelihood is

$$\mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T)) = \sum_{v \in \mathbf{V}_{ic}} \mathcal{L}_k \left(\mathbf{\Gamma}(d(v)); \mathbf{K}(v), \mathbf{\Theta}(\mathbf{K}(v)) \mid \{ \mathbf{K}(v') : v \prec v',\ v' \in \mathbf{V}_{ic} \} \right), \tag{19}$$

where $\mathbf{K}(v) = \{d(v') : v' \in c_i(v)\} \cup c_r(v)$ denotes the partition specified by child nodes in $c(v)$.

The evaluation of $\mathcal{L}_T$ mimics exactly the construction of the similarity clustering tree with each $v \in \mathbf{V}_{ic}$ corresponding to a unique node in $T_s(\mathbf{C})$ associated with a non-trivial similarity set. It may seem that estimation using the hierarchical topology likelihood will lose the ability to perform unsupervised learning because one needs to specify the hierarchical clustering of the leaf nodes before evaluating $\mathcal{L}_T$. This is not true because the clustering itself is performed using the parameter estimates of the finite mixture model. Note that the partition $\mathbf{K}(v)$ for $v \in \mathbf{V}_{ic}$ is uniquely determined by the inter-cluster similarity set corresponding to v, which can be inferred by the inter-cluster component in $f_{FM}(\mathbf{\Gamma}(d(v)))$ because that component is supported by the normalized samples of the particular similarity set. The details of the clustering algorithm will be provided in the next section. Before explaining the algorithm we would like to make a comment about our model. So far the model is restricted to Gaussian mixture components due to the C.L.T. approximation. In fact, (5) also holds for non-Gaussian ϕ's as long as $\bar{\gamma}^{(i,j)}$ and $\bar{\gamma}^{(k,l)}$ are approximately equal in distribution when $a(i,j) = a(k,l)$. This non-Gaussian mixture approach may be useful in the small sample regime, e.g., where the $\hat{\gamma}_{i,j}$ metric might be approximated by a multivariate Gamma distribution. In that case our model can be extended accordingly, which may be useful when new probing and/or approximation methods are proposed in the future.

4 Topology Estimation Algorithm

4.1 Estimation of The Finite Mixture Model

We first illustrate the estimation of the proposed finite mixture model $f_{FM}(\mathbf{\Gamma}; \mathbf{\Theta})$ in (6) using the EM algorithm. Lets assume the model order k is known for the time being. As discussed in the previous section the complete data for our model is $\mathbf{U} = \{\mathbf{\Gamma}, \{Z_m^{(i,j)}\}\}$, which has log-likelihood (without model order penalties) equal

$$\mathcal{L}_c(\mathbf{U}; \mathbf{\Theta}) = \sum_{\substack{i,j \in \mathbf{V}_r \\ i<j}} \sum_{m=1}^{k} Z_m^{(i,j)} \left[\log \alpha_m + \sum_{n=1}^{N_{i,j}} \log \phi(\bar{\gamma}_n^{(i,j)}; \theta_m) \right]. \tag{20}$$

The E-step in the t-th iteration computes the conditional expectation

$$\begin{aligned} Q(\mathbf{\Theta};\hat{\mathbf{\Theta}}^{(t)}) &= E\left[\mathcal{L}_c(\mathbf{U};\mathbf{\Theta})|\mathbf{\Gamma}=\mathbf{g};\hat{\mathbf{\Theta}}^{(t)}\right] \\ &= \sum_{\substack{i,j\in\mathbf{V}_r \\ i<j}}\sum_{m=1}^{k}\left[\omega_m^{(i,j)}\log\alpha_m + \sum_{n=1}^{N_{i,j}} Q_{m,n}^{(i,j)}(\theta_m)\right], \end{aligned} \tag{21}$$

where

$$\begin{aligned} \omega_m^{(i,j)} &= E\left[Z_m^{(i,j)}|\mathbf{\Gamma}=\mathbf{g};\hat{\mathbf{\Theta}}^{(t)}\right] \\ &= \frac{P(Z_m^{(i,j)}=1,\mathbf{\Gamma}_{i,j}=\mathbf{g}_{i,j};\hat{\mathbf{\Theta}}^{(t)})}{P(\mathbf{\Gamma}_{i,j}=\mathbf{g}_{i,j};\hat{\mathbf{\Theta}}^{(t)})} \\ &= \frac{\hat{\alpha}_m^{(t)}\prod_{n=1}^{N_{i,j}}\phi(g_n^{(i,j)};\hat{\theta}_m^{(t)})}{\sum_{m'=1}^{k}\hat{\alpha}_{m'}^{(t)}\prod_{n=1}^{N_{i,j}}\phi(g_n^{(i,j)};\hat{\theta}_{m'}^{(t)})} \end{aligned} \tag{22}$$

and

$$Q_{m,n}^{(i,j)}(\theta_m) = E\left[Z_m^{(i,j)}\log\phi(\bar{\gamma}_n^{(i,j)};\theta_m)|\bar{\gamma}_n^{(i,j)}=g_n^{(i,j)};\hat{\mathbf{\Theta}}^{(t)}\right]. \tag{23}$$

The M-step maximizes (21) with respect to $\mathbf{\Theta}$:

$$\alpha_m^* = \frac{\sum_{i,j\in\mathbf{V}_r,i<j}\omega_m^{(i,j)}}{N_T} \tag{24}$$

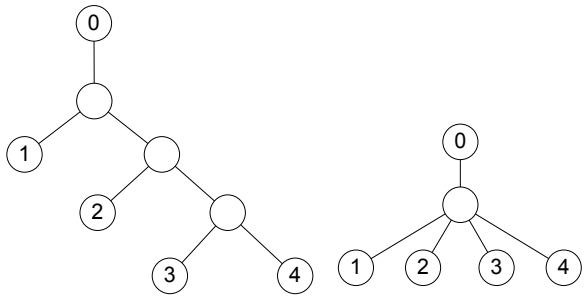
$$\theta_m^* = \text{argmax}_{\theta_m}\sum_{\substack{i,j\in\mathbf{V}_r \\ i<j}}\sum_{n=1}^{N_{i,j}} Q_{m,n}^{(i,j)}(\theta_m). \tag{25}$$

When ϕ is Gaussian the M-step for $\theta_m=(\mu_m,\sigma_m^2)$ becomes

$$\mu_m^* = \frac{\sum_{i,j\in\mathbf{V}_r,i<j}\left[\sum_{n=1}^{N_{i,j}} g_n^{(i,j)}\right]\omega_m^{(i,j)}}{\sum_{i,j\in\mathbf{V}_r,i<j} N_{i,j}\cdot\omega_m^{(i,j)}} \tag{26}$$

$$\sigma_m^* = \frac{\sum_{i,j\in\mathbf{V}_r,i<j}\left[\sum_{n=1}^{N_{i,j}}\left(g_n^{(i,j)}-\mu_m^*\right)^2\right]\omega_m^{(i,j)}}{\sum_{i,j\in\mathbf{V}_r,i<j} N_{i,j}\cdot\omega_m^{(i,j)}}. \tag{27}$$

The optimal model order k can be determined as follows. We assume k_{min} and k_{max} are the lower and upper bounds for k, respectively, which are determined according to some a priori information. If such information is unavailable, when $|\mathbf{V}_r|=n$ the maximum possible value for k is $n-1$, which corresponds to a binary (sub)tree with dangling leaf nodes (see Figure 8(a)), and the minimum possible value for k is 1, which corresponds to the case where all the leaf nodes are siblings (see Figure 8(b)). For each $k\in[k_{min},k_{max}]$ we obtain the maximum likelihood estimate of the parameters $\hat{\mathbf{\Theta}}_k^*$ using the EM algorithm, and compute the penalized log-likelihood $\mathcal{L}_p(\mathbf{\Gamma};\hat{\mathbf{\Theta}}_k^*)$ by (14). The optimal model order k^* is then chosen as the one which achieves the maximum penalized likelihood.



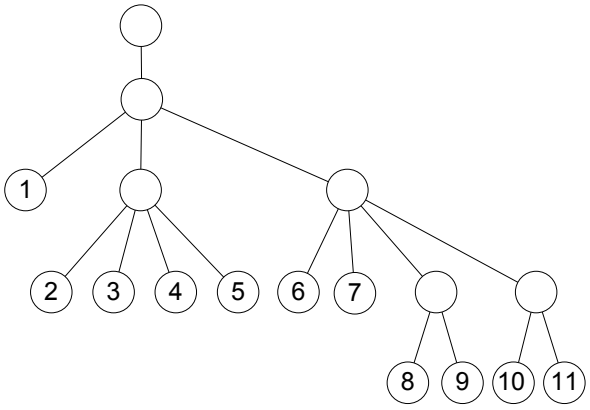
(a) (b)

Figure 8: Illustration of logical tree topologies corresponding to mixture models with maximal possible model order $k_{max} = 3$ (a) and minimal possible model order $k_{min} = 1$ (b) when there are 4 leaf nodes in the network.

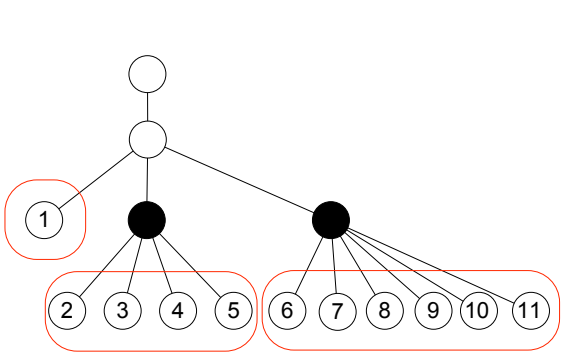
4.2 Hierarchical Topology Estimation Algorithm

We propose a greedy algorithm to estimate the logical tree topology hierarchically. It is a top-down approach that partitions the leaf nodes recursively. First we use $f_{FM}(\mathbf{\Gamma})$ 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. Figure 9 shows an example which illustrates how the algorithm works. Figure 9(a) is the true topology of the network. The estimator first identifies 3 clusters in $\mathbf{V}_r$: $\{\{1\}, \{2, 3, 4, 5\}, \{6, 7, 8, 9, 10, 11\}\}$, which are the 3 encircled groups in Figure 9(b). The two shaded vertices are the internal nodes found by this initial clustering. In the next iteration the estimator determines there is no subpartition existing in the second cluster, but the third cluster includes 4 subclusters. These subclusters are illustrated in Figure 9(c) and they define two more internal nodes in the topology which are depicted again as two shaded vertices. 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 other clusters or any possible subpartitions in the subsequent iterations.

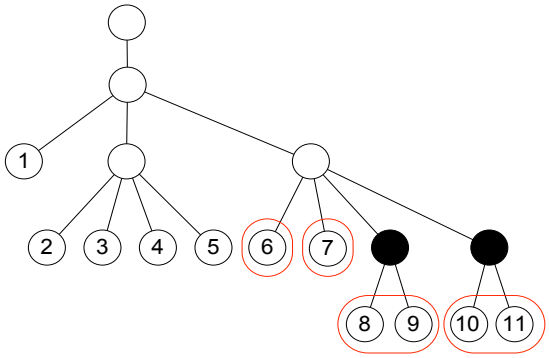
The key to the hierarchical topology estimation algorithm is to find the partition of the leaf nodes. Our algorithm is motivated by the following observation. First we label the component having the smallest mean in $f_{FM}(\mathbf{\Gamma}(\mathbf{G}))$ by component 1 for some subset of leaf nodes $\mathbf{G} \subseteq \mathbf{V}_r$. Assume there is no estimation error in the mixture model. In this ideal case component 1 is the inter-cluster component supported exactly by all the inter-cluster $\mathbf{\Gamma}_{i,j}$'s. Then the conditional mean in (22) for $m = 1$ has a value $\omega_1^{(i,j)} \approx 1$ if (i, j) is an inter-cluster 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 function $Z_1^{(i,j)}$ indicating whether $\mathbf{\Gamma}_{i,j}$ is contributed by the inter-cluster component. Consider an undirected complete graph H whose vertices are the leaf nodes in $\mathbf{G} \subseteq \mathbf{V}_r$ such that there exists an edge between every pair of the



(a)



(b)



(c)

Figure 9: Illustration of the hierarchical topology estimation. (a) depicts the true topology. (b) and (c) are the first and second iterations in the estimation process, respectively. The encircled groups of leaf nodes are the clusters found by the estimator, which are used to identify the shaded internal nodes.

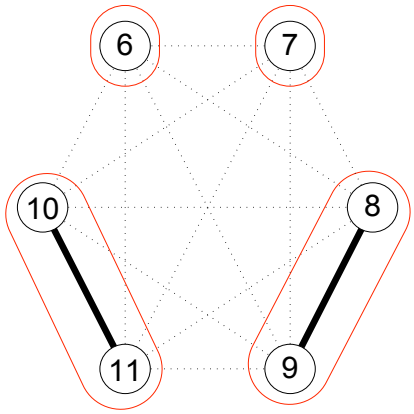


Figure 10: The partition of leaf nodes $\{6, 7, 8, 9, 10, 11\}$ in Figure 9(c) based on graph connectivity. The solid edges have weights ≈ 1, and the dotted edges have weights ≈ 0. The vertices enclosed by the same circle denote a cluster of the leaf nodes.

vertices. If we specify a weight

$$\varpi_{i,j} = \text{weight}(e_{i,j}) = 1 - \omega_1^{(i,j)} \tag{28}$$

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 from the connectivity of H. The partition in Figure 9(c) using graph connectivity is depicted in Figure 10.

Clustering algorithms using graph-theoretic approaches can be found in many research areas such as the clustering of gene expression data and the information retrieval in the WWW [46, 47, 48, 49, 50]. Basically any graph-based clustering algorithm for weighted graphs could work for our purpose. Here we describe a simple algorithm proposed in [48], the Highly Connected Subgraph (HCS) algorithm. It was originally developed for unweighted graphs where all the edge weights equal to one, but can be easily generalized for weighted graphs. Let $H = (\mathbf{V}_H, \mathbf{E}_H)$ be a graph both undirected and weighted, where $\mathbf{V}_H$ is the set of vertices and $\mathbf{E}_H$ is the set of edges. Every edge e in $\mathbf{E}_H$ has a nonnegative real weight $\varpi(e)$. The degree of a vertex v, $deg(v)$, is the number of edges connected to v. 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 a minimum weight. The weight of a mincut is called the *connectivity* of the graph, denoted by $conn(H)$. Note that the mincut of a graph may not be unique, especially when the graph is unweighted.

The key definition for HCS is the following: A graph H with $n > 1$ vertices is called *highly connected* if $conn(H) > \frac{n}{2}$. An important property of a highly connected complete graph can be obtained by modification of Theorem 1 in [48], which is provided below:

Table 1: The HCS algorithm.

```
HCS(H)
begin
    (S, H', H̄') ← MINCUT(H)
    if (H is highly connected)
        return (H)
    else
        HCS(H')
        HCS(H̄')
    end if
end
```

Theorem 5.1. The average weight of the edges connected to a vertex in a highly connected complete graph H is larger than $\frac{1}{2}$.

Proof. Let n be the number of vertices in H. Since H is a complete graph, every vertex has a degree equal to $n-1$. Suppose there exists a vertex v such that the average weight of the edges connected to v is less than or equal to $\frac{1}{2}$. Denote those edges by $e_1, \ldots, e_{n-1}$. Then the cut weight of $S = \{e_1, \ldots, e_{n-1}\}$ is

$$|S| = \sum_{i=1}^{n-1} \text{weight}(e_i) \leq \frac{n-1}{2} < \frac{n}{2},$$

which contradicts the fact that H is highly connected.

The HCS algorithm requires a subroutine 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. It has many applications in, for example, circuit design and communication networks. Several algorithms for solving the mincut problem can be found in the literature [51, 52, 53, 54, 55, 56, 57]. The HCS algorithm is summarized in Table 1 [48].

Before applying the HCS algorithm to topology discovery, we would like to discuss several detailed improvements which can be made to our hierarchical estimation algorithm. In order to make it more compatible with the HCS algorithm 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 inter-cluster 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 for $\bar{\gamma}_n^{(i,j)} \in \mathbf{\Gamma}_{\mathbf{A}_1,\mathbf{A}_2}$, where $i \in \mathbf{A}_1$, $j \in \mathbf{A}_2$, $n = 1, \ldots, N_{i,j}$, 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. $Z_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)} = \{Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)}\}_{i,j,n}$ is i.i.d. and assumed to have mean $E[Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)}] = \eta_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)}$.

4.3 Robust Edge Weights for Complete Graphs

The probability $\omega_1^{(i,j)}$ used by edge weights $\varpi_{i,j}$ in (28) is not a robust estimator for $Z_1^{(i,j)}$. Indeed, a single poor estimate in the inter-cluster $\mathbf{\Gamma}_{i,j}$ could make $\omega_1^{(i,j)} \approx 0$ because the numerator in (22) is α_1 times the product of each $\phi(\bar{\gamma}_n^{(i,j)}; \theta_1)$ for $\bar{\gamma}_n^{(i,j)} \in \mathbf{\Gamma}_{i,j}$. To overcome this problem we propose to replace $\omega_1^{(i,j)}$ by the average of the conditional mean estimates of $Z_1^{(i,j,n)}$ with each estimate using only single $\bar{\gamma}_n^{(i,j)}$ in $\mathbf{\Gamma}_{i,j}$. As contrasted to the original definition in (22), this estimate is smoothed over all the samples in $\mathbf{\Gamma}_{i,j}$. The general form of the new edge weight between two clusters of leaf nodes $\mathbf{A}_1$ and $\mathbf{A}_2$ with respect to a composite inter-cluster component $f_{\mathbf{B}}$ is then given by

$$\begin{aligned} \varpi_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)} &= 1 - \frac{1}{N_{\mathbf{A}_1,\mathbf{A}_2}} \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} \sum_{n=1}^{N_{i,j}} E\left[Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)} \mid \bar{\gamma}_n^{(i,j)}; \mathbf{\Theta} \right] \\ &= 1 - \frac{1}{N_{\mathbf{A}_1,\mathbf{A}_2}} \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} \sum_{n=1}^{N_{i,j}} \frac{\sum_{m \in \mathbf{B}} \hat{\alpha}_m \phi(\bar{\gamma}_n^{(i,j)}; \hat{\theta}_m)}{\sum_{m'=1}^{k} \hat{\alpha}_{m'} \phi(\bar{\gamma}_n^{(i,j)}; \hat{\theta}_{m'})}, \end{aligned} \tag{29}$$

where $N_{\mathbf{A}_1,\mathbf{A}_2} = \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} N_{i,j}$. Since $E\left[Z_{i,j,n}^{(\mathbf{B},\mathbf{A}_1,\mathbf{A}_2)} \mid \bar{\gamma}_n^{(i,j)}; \mathbf{\Theta} \right]$ is i.i.d. for $\bar{\gamma}_n^{(i,j)} \in \mathbf{\Gamma}_{\mathbf{A}_1,\mathbf{A}_2}$, according to the Weak Law of Large Numbers (W.L.L.N.) we have

$$\lim_{N_{\mathbf{A}_1,\mathbf{A}_2} \to \infty} P\left(\left| \varpi_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)} - (1 - \eta_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)}) \right| < \epsilon \right) = 1, \tag{30}$$

where ϵ is an arbitrarily small positive number. Note that the edge weights always fall in the region $[0, 1]$.

4.4 Pre-cluster Algorithm

The MINCUT procedure in the HCS algorithm can be computationally demanding when there are many vertices in the complete graph. For example, the algorithm proposed in [56] has an overall run time complexity of $\mathcal{O}(|\mathbf{V}_H||\mathbf{E}_H| + |\mathbf{V}_H|^2 \log |\mathbf{V}_H|)$ for any graph H. One way to reduce the complexity is to pre-cluster the vertices in H into groups which are obviously in the same cluster.

For a set of leaf nodes $\mathbf{G} \subseteq \mathbf{V}_r$ and finite mixture estimate $f_{FM}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$, we assume the inter-cluster component in $f_{FM}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$ is identified as the composite component formed by components in the subset $\mathbf{B} \subseteq \{1, \ldots, k\}$, where k is the model order of $f_{FM}(\mathbf{\Gamma}(\mathbf{G}); \mathbf{\Theta})$. If there is no estimation error in

$f_{FM}(\mathbf{\Gamma}(\mathbf{G});\mathbf{\Theta})$, $\mathbf{B}$ includes only the component having the smallest mean, denoted by component 1 for simplicity. A simple way 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 $\frac{1}{2}$. Define the set of *foreign leaf nodes* for node i with respect to $\mathbf{B}$ as

$$\mathbf{F_B}(i) = \left\{ j : \varpi_{\mathbf{B}}^{(\{i\},\{j\})} < \frac{1}{2},\ j \in \mathbf{G} \backslash \{i\} \right\}, \quad \forall i \in \mathbf{G}. \tag{31}$$

$\mathbf{F_B}(i)$ contains all possible nodes which are not in the same cluster as i. Then we simply 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 inter-cluster component. Two possible situations may occur: a mixture model estimate with too fine resolution could decompose the inter-cluster components into several subcomponents among which component 1 is only one of them; or, an estimate with too coarse resolution could merge the inter-cluster component with other intra-cluster components and result in an overly fine clustering (too many clusters) of the leaf nodes.

4.5 Progressive Search Algorithm

We deal with the first situation by a *progressive search* method. Let the estimated components in $f_{FM}(\mathbf{\Gamma}(\mathbf{G});\mathbf{\Theta})$ be sorted by ascending order of their means, i.e., component 1 has the least mean and component k has the largest. The search starts with applying $\mathbf{B}_1 = \{1\}$ to form the inter-cluster component and estimating a pre-clustering $\mathbf{K}_p(\mathbf{B}_1)$ using (31). Then expand the component subset to $\mathbf{B}_2 = \{1, 2\}$ and estimate another pre-clustering $\mathbf{K}_p(\mathbf{B}_2)$. Repeat this procedure until $\mathbf{B}_k = \{1, \ldots, k\}$ which includes all the components in $f_{FM}(\mathbf{\Gamma}(\mathbf{G});\mathbf{\Theta})$. Then we select the pre-clusterings with the least number of clusters as the *pre-clustering estimates*:

$$\hat{\mathbf{K}}_p = \{\mathbf{K}_p(\mathbf{B}_i) : |\mathbf{K}_p(\mathbf{B}_i)| \leq |\mathbf{K}_p(\mathbf{B}_j)|,\ \forall i, j \in \{1, \ldots, k\},\ i \neq j\}. \tag{32}$$

A complete graph can be drawn from each pre-clustering estimate with its vertices now representing the clusters which may contain two or more leaf nodes in one cluster. The proposed robust edge weights in (29) can be applied to this new definition of vertices. Let $H(\mathbf{K}_p(\mathbf{B}_i))$ be the complete graph whose vertices represent the clusters in $\mathbf{K}_p(\mathbf{B}_i)$ and edge weights are computed using $\mathbf{B}_i$. Then the graphs $H(\mathbf{K}_p(\mathbf{B}_i))$ for $\mathbf{K}_p(\mathbf{B}_i) \in \hat{\mathbf{K}}_p$ are used as inputs of the HCS algorithm and the output with the highest $\mathcal{L}_k$ is adopted as the *HCS clustering estimate* , denoted by $\hat{\mathbf{K}}$. Since it is possible that $\hat{\mathbf{K}}$ can be produced from multiple HCS inputs using different $\mathbf{B}_i$'s, we specify the smallest component subset $\mathbf{B}_i$ used to obtain $\hat{\mathbf{K}}$ as $\hat{\mathbf{B}}$ and use it in the following post-merge algorithm.

4.6 Post-merge Algorithm

To address the second situation described at the end of Section 4.4 we propose a post-merge 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 inter-cluster component set. For every cluster $\mathbf{A}$ represented by a vertex $v_{\mathbf{A}}$ in the graph we define its closest cluster $c(\mathbf{A})$ as the cluster represented by the strongest vertex connected to $v_{\mathbf{A}}$, i.e.,

$$c(\mathbf{A}) = \underset{\mathbf{A}' \in \hat{\mathbf{K}} \setminus \mathbf{A}}{\operatorname{argmax}} \varpi_{\hat{\mathbf{B}}}^{(\mathbf{A},\mathbf{A}')}. \tag{33}$$

Then for each $\mathbf{A} \in \hat{\mathbf{K}}$ we get a new partition by merging $\mathbf{A}$ and $c(\mathbf{A})$ into one cluster, and evalute the penalized partition likelihood. If the highest likelihood obtained by merging a pair of clusters is greater than the likelihood of $\hat{\mathbf{K}}$, we update $\hat{\mathbf{K}}$ by the corresponding new partition and repeat the process until no improvement is made by any pair-wise merge. Note that when a pair of clusters are the closest to each other their respective merges result in the same partition. The number of merges needed to be tested when there are M clusters in the partition is lower bounded by $\lceil \frac{M}{2} \rceil$ and upper bounded by $M - 1$. The complete algorithm for hierarchical topology estimation is summarized in Table 2, where the finite mixture model $f_{FM}(\mathbf{\Gamma}(d(i)))$ for an internal node $i \in \mathbf{V}_i$ has estimated parameter set $\hat{\mathbf{\Theta}}_i = \{k_i, \alpha_{i,1}, \ldots, \alpha_{i,k_i}, \theta_{i,1}, \ldots, \theta_{i,k_i}\}$. As for the progressive search algorithm, we assume $\theta_{i,1}, \ldots, \theta_{i,k_i}$ have means in ascending order, i.e., $\mu_{i,n_1} \leq \mu_{i,n_2}$ if and only if $n_1 < n_2$. Note that except for the first iteration, each $f_{FM}(\mathbf{\Gamma}(d(i)); \hat{\mathbf{\Theta}}_i)$ is estimated in the previous iteration when the partition likelihood is computed.

5 Computer Simulations

Both `matlab` and `ns-2` simulations were performed to test our modelling and estimation procedures. First we describe our `matlab` simulations whose purpose is to establish the ability to estimate topology from simulated similarity metrics with additive noise. Then an `ns-2` simulation is performed to test the algorithm when similarities are estimated from probe data.

5.1 MATLAB Model Simulation

First we simulated a small network with the simple non-binary virtual topology shown in Figure 11. The simulations were implemented 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 $p_{a(i,j)}$ using `matlab`. Each metric sample for a link was

Table 2: Hierarchical Topology Estimation (HTE) algorithm.

Input: $\{0\}, \mathbf{V}_r, \mathbf{\Gamma}$
Output: $T = (\mathbf{V}, \mathbf{E})$

Finish $\leftarrow$ false
$N \leftarrow |\mathbf{V}_r| + 1$
$\mathbf{V} \leftarrow \{0, N\} \cup \mathbf{V}_r$, $\mathbf{E} \leftarrow \{(0, N)\} \cup \{(N, v) : v \in \mathbf{V}_r\}$
$T \leftarrow (\mathbf{V}, \mathbf{E})$
$\mathbf{I} \leftarrow \{N\}$, $N \leftarrow N + 1$
Estimate $f_{FM}(\mathbf{\Gamma}(d(N)); \hat{\mathbf{\Theta}}_N)$
while ($\sim$Finish) do
  $\mathbf{I}' \leftarrow \phi$, Finish $\leftarrow$ true
  for all $i \in \mathbf{I}$ do

    for $m = 1$ to k_i do
      $\mathbf{B}_m \leftarrow \{1, \ldots, m\}$
      $\mathbf{F}_{\mathbf{B}_m}(j) \leftarrow \left\{ l : \varpi_{\mathbf{B}_m}^{(\{j\},\{l\})} < \frac{1}{2},\ l \in d(i) \setminus \{j\} \right\},\ \forall j \in d(i)$
      form pre-clustering $\mathbf{K}_p(\mathbf{B}_m)$
    end for
    $\hat{\mathbf{K}}_p \leftarrow \{\mathbf{K}_p(\mathbf{B}_m) : |\mathbf{K}_p(\mathbf{B}_m)| \leq |\mathbf{K}_p(\mathbf{B}_n)|,\ \forall m, n \in \{1, \ldots, k_i\},\ m \neq n\}$
    $\mathbf{M} \leftarrow \{m : \mathbf{K}_p(\mathbf{B}_m) \in \hat{\mathbf{K}}_p\}$

    for all $m \in \mathbf{M}$ do
      $\hat{\mathbf{K}}_m \leftarrow \text{HCS}(H(\mathbf{K}_p(\mathbf{B}_m)))$
      Compute $\mathcal{L}_k(\mathbf{\Gamma}(d(i)); \hat{\mathbf{K}}_m, \mathbf{\Theta}(\hat{\mathbf{K}}_m))$
    end for
    $\hat{\mathbf{K}} \leftarrow \text{argmax}_{\hat{\mathbf{K}}_m, m \in \mathbf{M}} \mathcal{L}_k(\mathbf{\Gamma}(d(i)); \hat{\mathbf{K}}_m, \mathbf{\Theta}(\hat{\mathbf{K}}_m))$
    $\hat{\mathbf{B}} \leftarrow \text{argmin}_{\mathbf{B}_m, m: \hat{\mathbf{K}}_m = \hat{\mathbf{K}}} |\mathbf{B}_m|$

    if ($|\hat{\mathbf{K}}| > 1$) do
      stop $\leftarrow$ false
      while ($\sim$stop) do
        for all $\mathbf{A} \in \hat{\mathbf{K}}$ do
          $c(\mathbf{A}) \leftarrow \text{argmax}_{\mathbf{A}' \in \hat{\mathbf{K}} \setminus \mathbf{A}} \varpi_{\hat{\mathbf{B}}}^{(\mathbf{A}, \mathbf{A}')}$
          $\mathbf{K}_{\mathbf{A}} \leftarrow \left(\hat{\mathbf{K}} \setminus \{\mathbf{A}, c(\mathbf{A})\} \right) \cup (\mathbf{A} \cup c(\mathbf{A}))$
          Compute $\mathcal{L}_k(\mathbf{\Gamma}(d(i)); \mathbf{K}_{\mathbf{A}}, \mathbf{\Theta}(\mathbf{K}_{\mathbf{A}}))$
        end for
        $\mathbf{K}_{new} \leftarrow \text{argmax}_{\mathbf{K}_{\mathbf{A}}, \mathbf{A} \in \hat{\mathbf{K}}} \mathcal{L}_k(\mathbf{\Gamma}(d(i)); \mathbf{K}_{\mathbf{A}}, \mathbf{\Theta}(\mathbf{K}_{\mathbf{A}}))$
        if $\left(\mathcal{L}_k(\mathbf{\Gamma}(d(i)); \mathbf{K}_{new}, \mathbf{\Theta}(\mathbf{K}_{new})) > \mathcal{L}_k(\mathbf{\Gamma}(d(i)); \hat{\mathbf{K}}, \mathbf{\Theta}(\hat{\mathbf{K}})) \right)$ do
          $\hat{\mathbf{K}} \leftarrow \mathbf{K}_{new}$
        else
          stop $\leftarrow$ true
        end if
      end while
    end if

```
        if (|K̂| > 1) do
            for all A ∈ K̂ do
                if (|A| > 1) do
                    V ← V ∪ {N}
                    E ← (E \ {(i, j) : j ∈ A}) ∪ {(i, N)} ∪ {(N, j) : j ∈ A}
                    if (|A| > 2) do
                        I′ ← I′ ∪ {N}
                        Finish ← false
                    end if
                    N ← N + 1
                end if
            end for
        end if
    end for
    I ← I′
end while
```

generated according to a Gaussian distribution with a randomly generated mean γ and a standard deviation σ_l proportional to the mean. Note that the true link metric was specified by $\boldsymbol{\gamma}$. γ_l was generated according to a uniform distribution over a region centered at η with width equal to β, i.e., $\boldsymbol{\gamma} \sim [\eta - \frac{\beta}{2}, \eta + \frac{\beta}{2}]$. The standard deviation σ_l was obtained by multiplying γ_l with a positive factor ρ.

We implemented the proposed hierarchical topology estimator (HTE) with an averaging factor $N_{norm} = 20$ to compute the normalized similarity samples, which means for each probe tree there were 10 samples of $\bar{\gamma}^{(i,j)}$. We also implemented to other topology discovery algorithms: the LBT algorithm [5, 15] and the DBT algorithm [2, 4]. The latter was originally designed for multicast networks, but can also be directly applied to unicast networks. For a pair of the leaf nodes (i, j) in the DBT algorithm we used the average of similarity samples $\{\hat{\gamma}_n^{(i,j)}\}_n$ as the metric. Since here the pair-wise similarities of the leaf nodes were designed to be sums of additive link metrics, the metric estimate for each link computed by the DBT algorithm became the difference between the similarity metric values whose corresponding shared paths differ by the specific link (see Figure 1 in [4] for comparison). 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 $\hat{\mu}_{link}$ and $\hat{\sigma}_{link}$ 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 $\hat{\mu}_{link} - \hat{\sigma}_{link}$.

The performance of the algorithms was evaluated in terms of the graph edit distance between the estimated tree and the true topology. The problem of comparing and matching trees has diverse applications in areas such as compiler design and pattern recognition [59]. Several distance metrics between a pair of trees had been proposed in the past decades, such as the edit distance [17, 18, 19, 20, 60, 61, 62], the alignment distance [63],

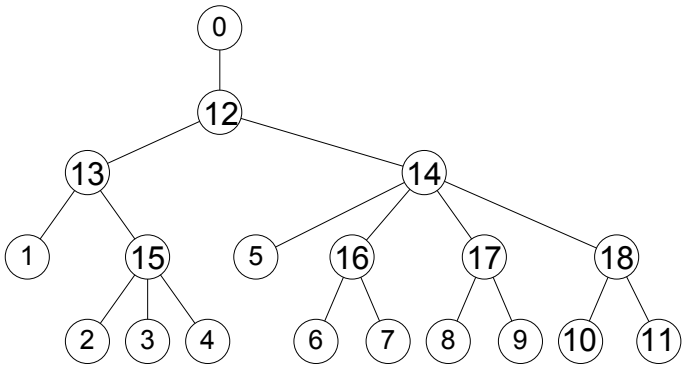


Figure 11: The logical tree topology for the network used in computer simulations in Section 5.

the isolated-subtree distance [64], the top-down distance [65] and the bottom-up distance [59]. We adopted the simple edit distance as our performance metric. The tree edit distance is analogous to the edit distance between two strings which is given by the least-cost sequence of elementary operations that transforms one string into the other. Let T be a rooted tree. T is a *labelled tree* if each node is given a symbol from a finite alphabet. T is called an *ordered tree* if a left-to-right order among siblings is assigned. The basic editing operations for an ordered tree are: **Replacement** – relabel a node v in T. If the label remains unchanged, it is an *identical replacement*, otherwise it is a *non-identical replacement*; **Insertion** – insert a node v in T. If v' is the parent of v in T, v becomes the parent of the children of v whose consecutive subsequence is replaced by v; **Deletion** – delete a non-root node v in T. If v' is the parent of v in T, the children of v become a subsequence of nodes which are inserted in the place of v in the left-to-right order of the children of v'.

A mapping from T_1 to T_2 is defined as a set of operations which allows to transform T_1 to T_2. If a mapping M includes R non-identical replacements, I insertions, and D deletions, then the cost of mapping M is given by $rR + iI + dD$, where r is the cost of a non-identical replacement, i is the cost of an insertion, d is the cost of a deletion, and the cost of identical replacement is ususally 0. The set of costs is called *unit cost* if $r = i = d = 1$. The graph edit distance between T_1 and T_2 is then defined as the cost of a minimum-cost mapping between T_1 and T_2. Figure 12 illustrates a mapping example from T_1 to T_2 which has 6 identical replacements, 1 non-identical replacement, 1 insertion and 3 deletion. Under the assumption of unit cost, the cost of the mapping is 5. Algorithms computing the edit distance between two ordered trees can be found in [18, 19]. The graph edit distance problem for unordered trees has been shown to be NP-complete [66].

To transform the true and estimated topology into ordered trees we applied the following rule. Recall that we label the leaf nodes by $1, \ldots, |\mathbf{V}_r|$, where $\mathbf{V}_r$ is the set of leaf nodes. Define the score of a leaf node to be the number specified by the leaf node's label, known to the estimator. The score of an internal node was given by the average score over the descendant leaf nodes of the internal node. Then the left-to-right order among

sibling nodes was determined by the ascending order of the node scores. For example, in Figure 1 the score of node 8 is 1.5 and the score of node 9 is 4.5. So the left-to-right order is $(8, 9)$. Throughout the computer experiments we adopted the unit cost of deletion, insertion, and non-identical replacement.

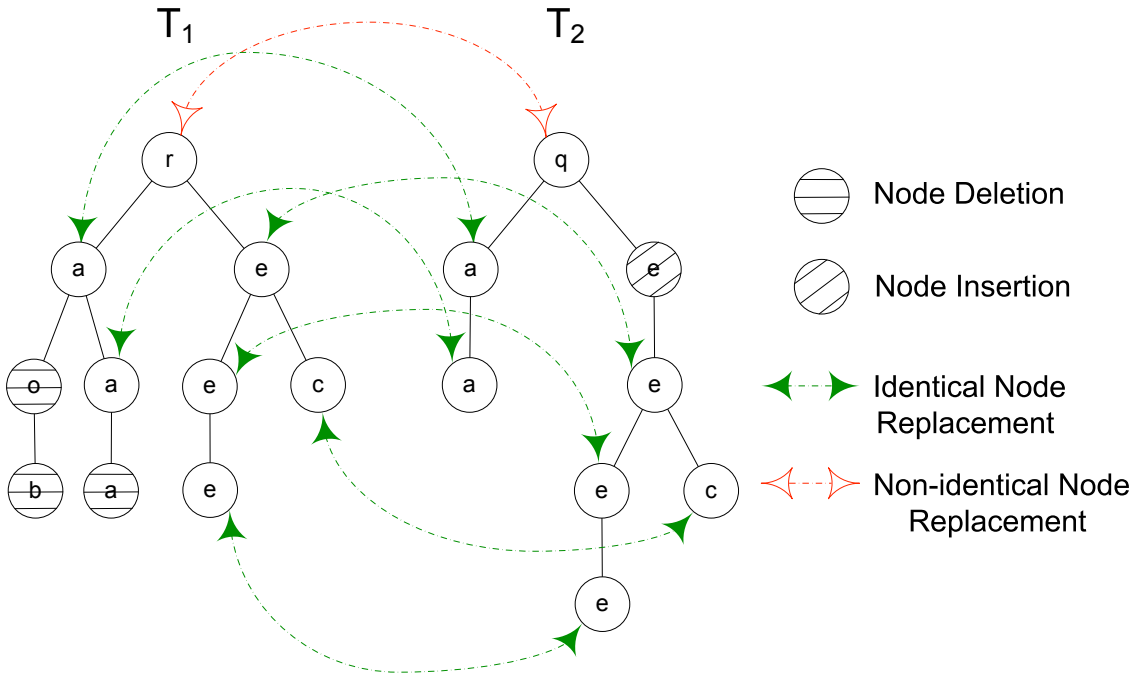


Figure 12: Illustration of the editing operations between two ordered trees.

We conducted two experiments using the proposed HTE algorithm, along with the DBT and LBT. These experiments were intended to clarify the advantages and drawbacks of each method and should not be considered as a thorough comparison of performance. In the first experiment we let each γ obey the same uniform distribution with a fixed width β equal 2 and a mean η varied from 3 to 13. We also fixed the noise variance scale factor $\rho = \frac{1}{\sqrt{2}}$. The result is illustrated in Figure 13. Each data point was obtained from 1000 independent simulations. The magnitudes of the link metrics determine the distance between any two different components in the finite mixture model. The separability among different internal nodes increases as the link metrics become large. Compared to DBT and LBT the HTE generally achieved a lower average edit distance to the true topology and a higher percentage of correctly identified trees. The performances of DBT and LBT are quite close to each other. They still provided reasonably accurate estimates as long as the variance of the similarity estimates remained small.

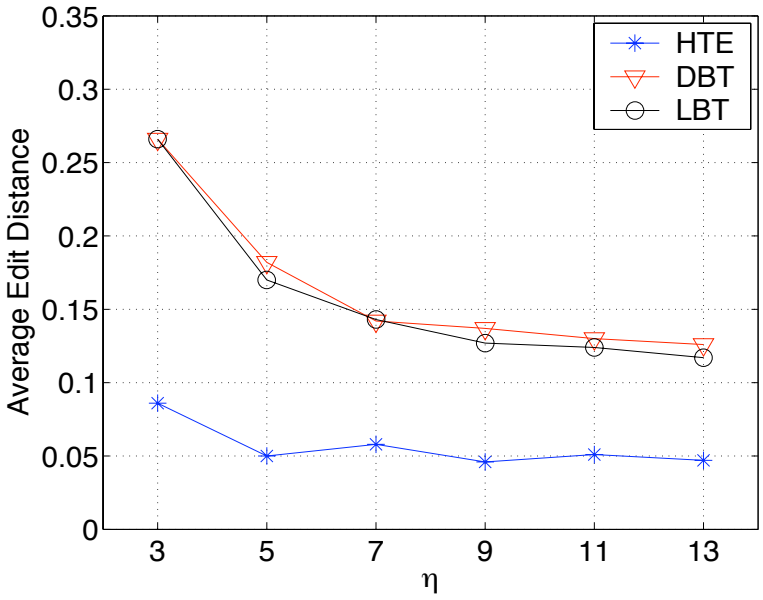
In the second experiment we fixed the range of the uniform distribution for each link metric to $[2, 6]$. The scale factor ρ for sample standard deviations was varied from 1 to 10 for link 16 and 17, and fixed at $\frac{1}{\sqrt{2}}$ for the others. The result is illustrated in Figure 14. Each data point was averaged from the outcomes of 1000 independent simulations. As the accuracy of topology estimation decreased with the increasing ρ, HTE

exhibited a minor loss in its performance while DBT and LBT both suffered from a serious degradation in the estimation capability.

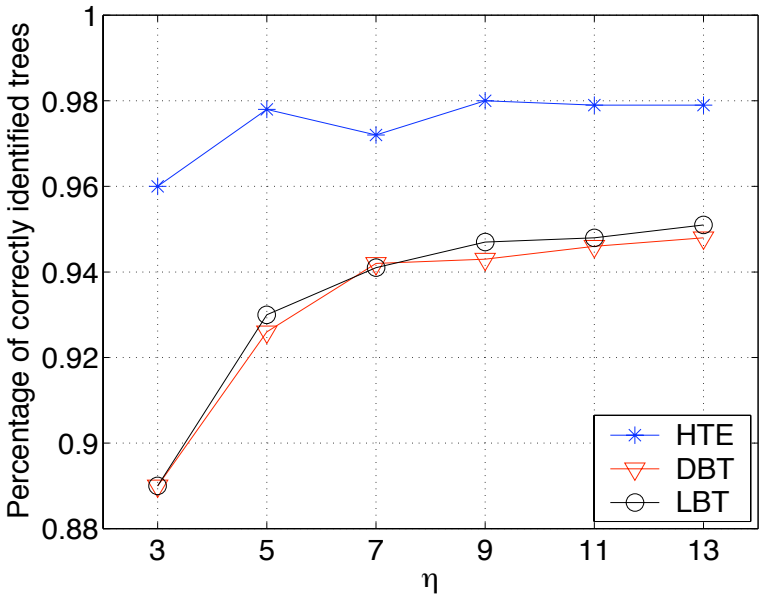
The outperforming of our algorithm is mainly due to the following reason. Although all the algorithms of HTE, LBT, and DBT are greedy in the sense that they all depend on local information to construct the topology, HTE is much less greedy than LBT and DBT. Recall that in the introduction we explained the DBT and LBT both being agglomerative algorithms which repeat the process of selecting the two most similar leaf nodes, connecting them to a parent node, and treating the new parent node as a leaf node which represents the selected pair of original leaf nodes. It means that each iteration in the DBT or LBT algorithm focuses on a small region in the topology parameter space which determines only the two most similar subclusters of the leaf nodes to be combined into one cluster. On the other hand, our HTE algorithm tries to find a local optimum over a larger region of the parameter space in each level of the hierarchical topology estimation which decides the partition of a (sub)set of the leaf nodes. This implies the estimates obtained from the HTE algorithm are generally closer to the global optimal topology than those estimated by the DBT or LBT algorithm.

5.2 NS Simulation

For a more practical environment we used `ns-2` to simulate the network in Figure 11. Two types of links were used: the links attached to the leaf nodes were assigned with bandwidth 1Mbps and latency 1ms and the others were assigned with bandwidth 2Mbps and latency 2ms. Each link was modelled by an 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 2: queueing delay using sandwich probes, delay variance using packet pairs, the loss rate also using packet pairs. The packet size in a packet pair probe was set to 10 bytes. The size of the large packet and the small packets in a sandwich probe was 500 bytes and 10 bytes, respectively. The probes were sent by UDP streams with Poisson departure. The departure interval had a mean equal to 8 times the transmission delay of a single probe on the outgoing link of the root node. The pair of leaf node destinations was randomly selected for each probe. The simulation was repeated until at least N probes were sent through every probe tree. $N = N_1 N_{norm}$ for sandwich probes and $N = N_1 N_{norm} N_{cov}$ or $N = N_1 N_{norm} N_{loss}$ for packet pair probes, where N_1 is the specified number of normalized similarity samples expected to be collected at each probe tree. Here we let $N_2 = N_{cov} = N_{loss}$. The parameters specifying the number of probes used in `ns-2`

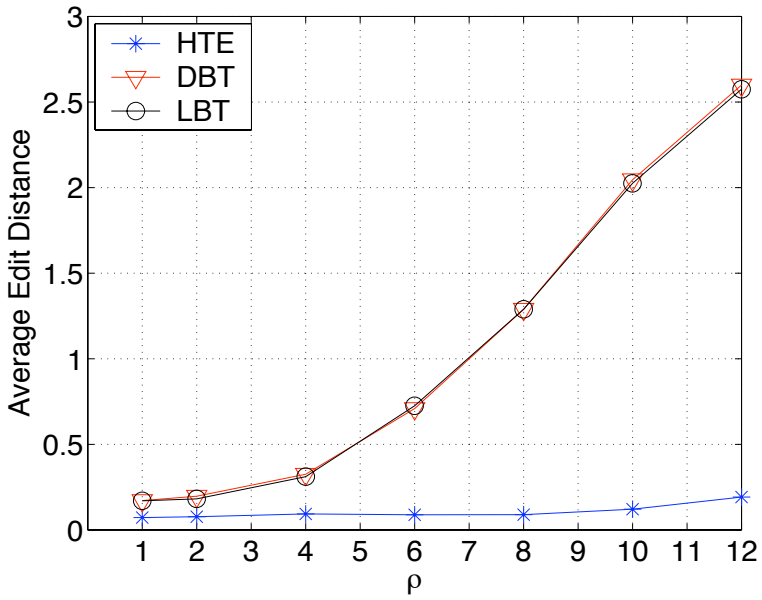


(a)

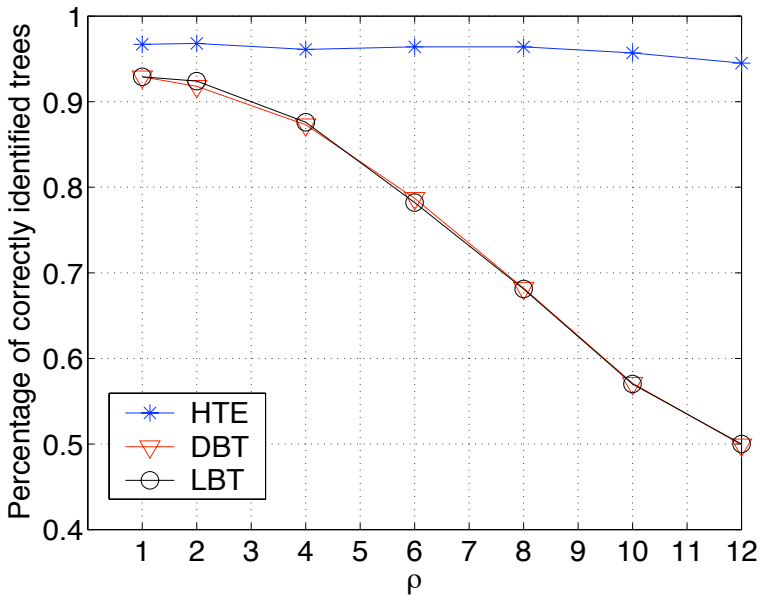


(b)

Figure 13: Average graph edit distance (a) and percentage of correctly identified trees (b) versus the mean of uniformly distributed link metrics in model simulation. The range of the uniform distribution was fixed at 2. The Gaussian samples of a link metric had standard deviation equal to the link metric times $\frac{1}{\sqrt{2}}$. The DBT and LBT algorithms were also implemented for comparison with the proposed HTE.



(a)



(b)

Figure 14: Average graph edit distance (a) and percentage of correctly identified trees (b) versus the proportional factor ρ for link 16 and 17 in model simulation. The true link metrics were uniformly distributed in $[2, 6]$. The DBT and LBT algorithms were also implemented for comparison with the proposed HTE algorithm.

Table 3: The 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
N_{norm}	5	5	5	5	5	5	7	9	11	13	15	15	15	15	15	15
N_2	10	10	10	10	10	10	10	10	10	10	10	12	14	16	18	20

N_1	25	25	25	25	25	25	25	25	25
N_{norm}	10	15	20	25	30	35	40	45	50
N_2	20	20	20	20	20	20	20	20	20

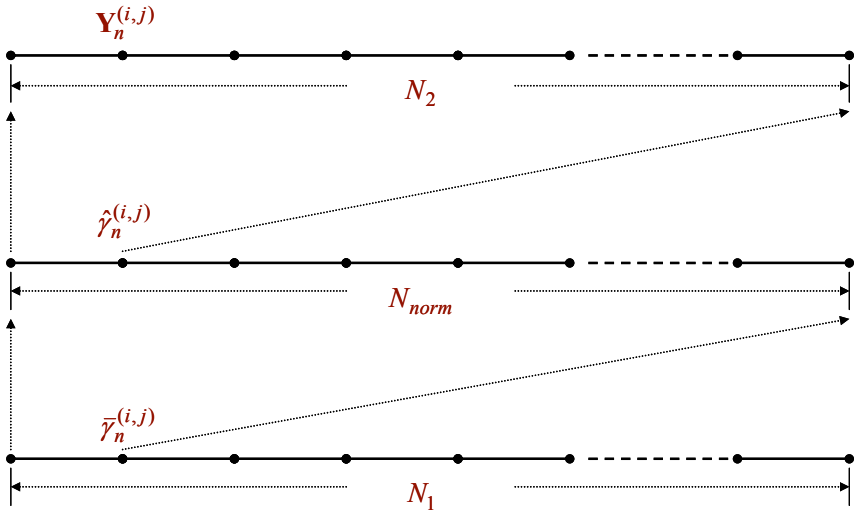


Figure 15: Illustration of the computation of N_1 normalized similarity samples. Each normalized sample is the average of N_{norm} similarity estimates. For packet pair probes each similarity estimate is obtained using N_2 measurements. In this figure it shows the delay measurement $\mathbf{Y}_n^{(i,j)}$ of packet pair probes.

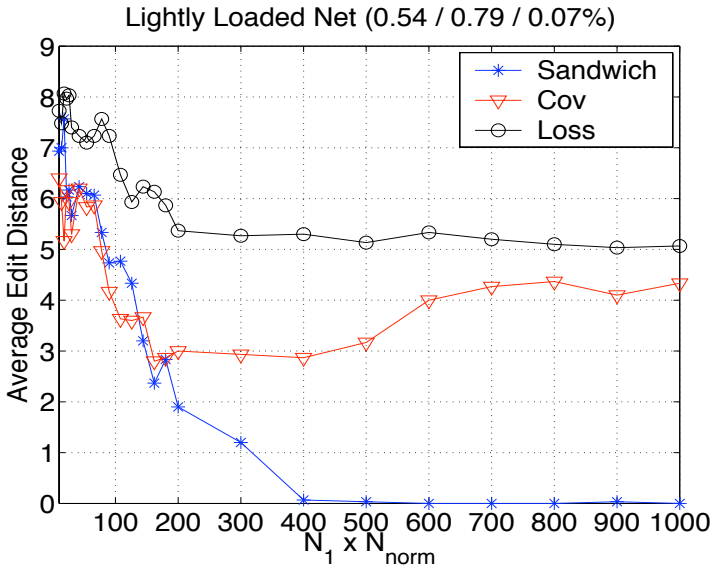
simulation are listed in Table 3. Each column gives a set of parameters for a simulation. Figure 15 shows how to collect N_1 normalized similarity samples $\bar{\gamma}_n^{(i,j)}$ for each probe tree. Each normalized sample was the average of N_{norm} similarity estimates $\hat{\gamma}_n^{(i,j)}$, which were estimated from end-to-end delay differences, delay covariances, or packet drop rates. For packet pair probes each similarity estimate was obtained using N_2 measurements. If any probe was dropped for a set of N_2 packet pairs, the similarity estimate was computed with the remaining probes. If too many probes were lost to obtain a reliable estimate, we simply discarded it and averaged the remaining estimates to get an approximately normalized Gaussian sample. Similar rules applied to the sandwich probes for computing the normalized samples.

We first compared the three probing schemes under three different network conditions. Each graph edit distance between the estimated tree and the true topology was averaged over 30 independent simulations using the proposed hierarchical topology estimator. Figure 16 (a),(c),(e) show the performance in a lightly-loaded, moderately-loaded, and heavily-loaded network respectively, where the horizontal axes denote the number $N_1 N_{norm}$ of similarity estimates used in each simulation. The average packet delay (ms), packet delay standard

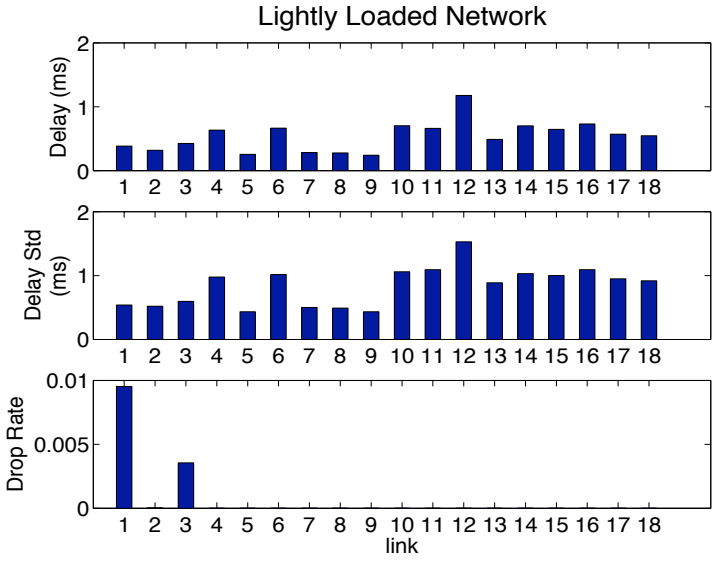
deviation (ms), and packet drop rate for each link is plotted in Figure 16 (b),(d),(f). The notation $(a/b/c)$ in the titles of Figure 16 (a),(c),(e) denotes the average condition for the whole network in which a, b, and c are the average packet delay, packet delay variance, and packet drop rate over all the links, respectively. Throughout the experiments in `ns-2` we used this notation to indicate the load condition of the network. The legends for queueing delay, delay variance, and loss rate similaritie metrics were marked by 'Sandwich', 'Cov', and 'Loss', respectively.

As predicted in Section 2 the sandwich probes provided the most reliable topology estimate in a lightly-loaded network. From Figure 16(b) we found some of links had very small packet delay variances and hence could not be identified using delay variance metrics. A similar situation occured for packet drop rates for light loading since the drop packet rate was very small. The curves for packet pair probing with delay covariance and dropped packet measurements do not converge, as shown in Figure 16(a). In a moderately-loaded network each link queue provided enough delay variation to perform topology estimation using packet pair delay covariance. Figure 16(c) shows the packet pair delay covariances achieved the best performance. The average edit distance to the true topology still converged to zero for sandwich probes, but with a slower rate due to the noise introduced by delay variations. The curve for packet pairs using loss rate similarity did not converge to zero. This is because of identifiability problems on the links which did not experience any dropped packets. For a heavily-loaded network, each link had a substantial packet drop probability which made the loss rate similarities contain the most reliable information about the network structure. Although the delay variance on each link increased in the heavily-loaded case compared to the moderate loading, the packet pair delay covariance method performed worse since the number of successfully received probes was significantly reduced due to packet losses. The sandwich probes suffered from both the high packet drop rate and high delay variance. Hence, sandwich probing gave the least reliable estimates for this heavily loaded situation. Note that some data points in the 'Sandwich' and 'Cov' curves were missing because for those cases almost every probe failed to pass through the network, leaving insufficient samples to perform topology estimation. To assess the influence of ambient traffic on the similarity estimates, we also plot the graph edit distance curves for each probing scheme seperately in Figure 17.

The edit distance provides us a metric to describe the distribution of the topology estimates. To illustrate, we simulated a larger network in `ns-2`, whose topology is shown as the top tree in Figure 18. The bandwidth and latency for the internal links which are not attached to the leaf nodes were assigned 5Mbps and 5ms, respectively. The edge links at the leaf nodes had bandwidth equal to 1Mbps or 2Mbps, and latency equal to 1ms or 2ms. Similar cross traffic as before was generated to establish a light load condition. Here we

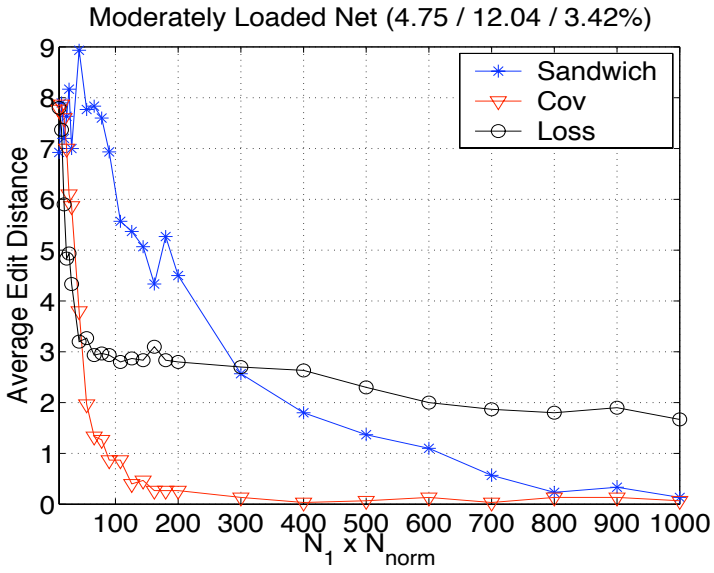


(a)

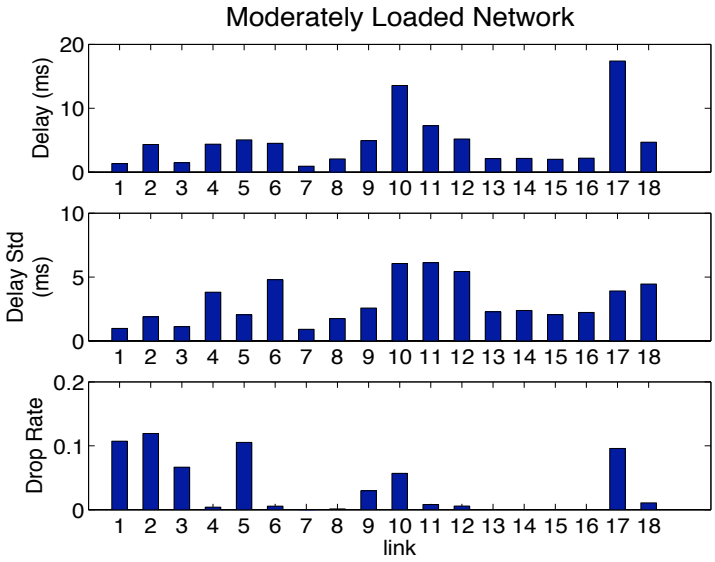


(b)

Figure 16: (a),(c),(e) The average graph edit distance between the estimated tree and the true topology versus the number of normalized similarity samples $N_1 N_{norm}$ in a lightly, moderately, and heavily loaded network simulated in `ns`, respectively, for the three probing schemes introduced in Section 2. (b),(d),(e) The average condition for each link queue in the lightly, moderately, and heavily loaded network, respectively. The units for delay and delay standard deviation (Delay Std) are both ms.

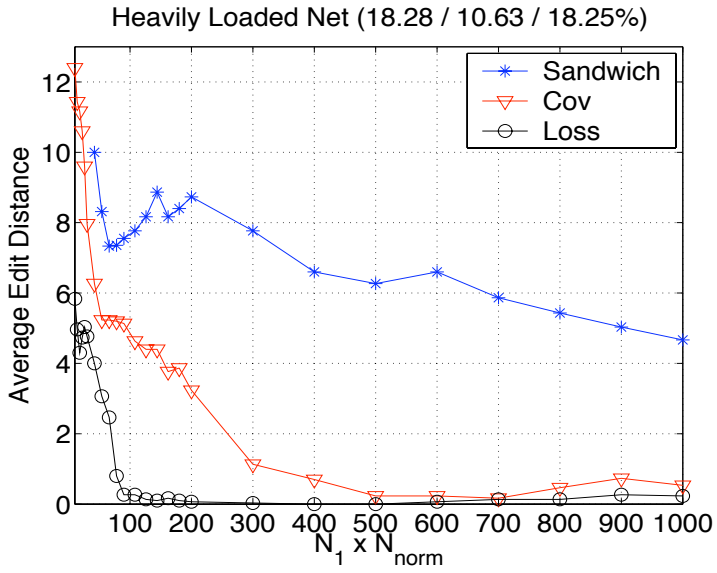


(c)

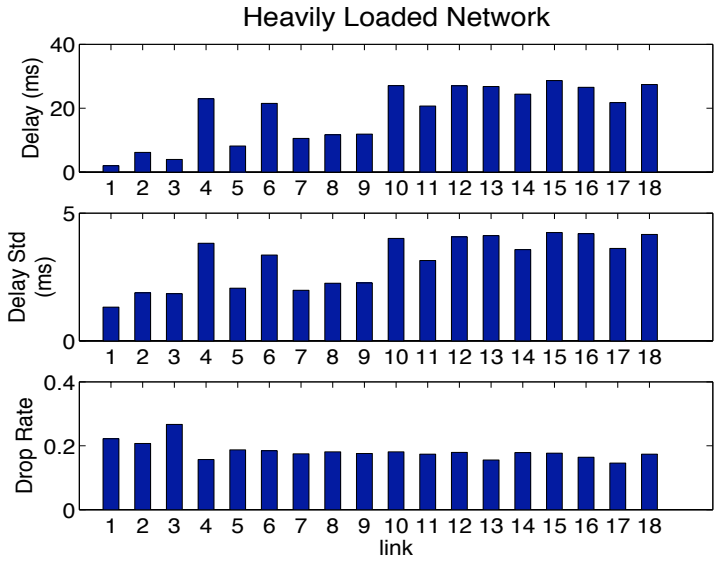


(d)

Figure 16 (continued)

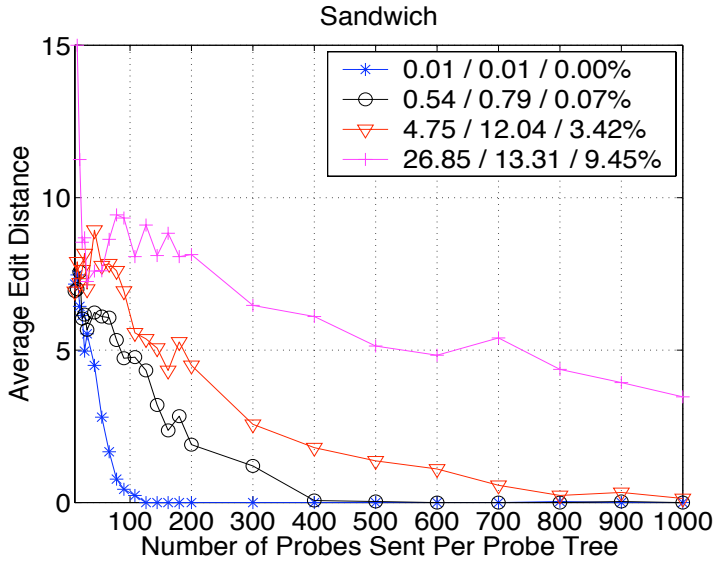


(e)

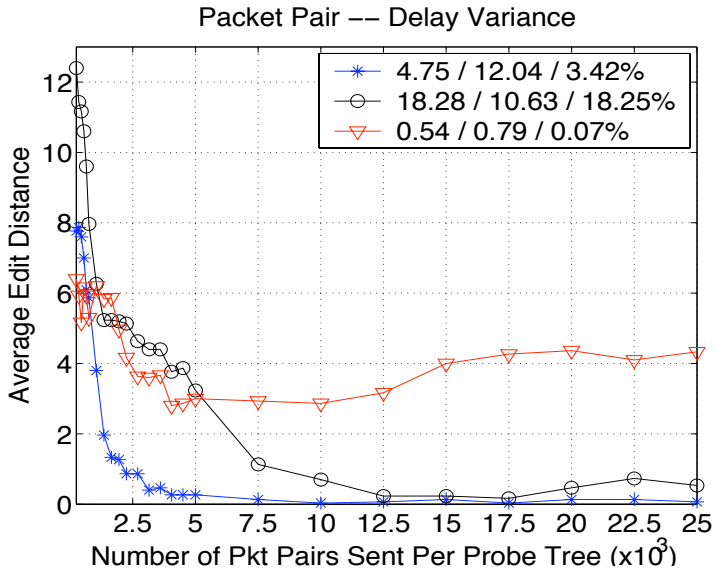


(f)

Figure 16 (continued)

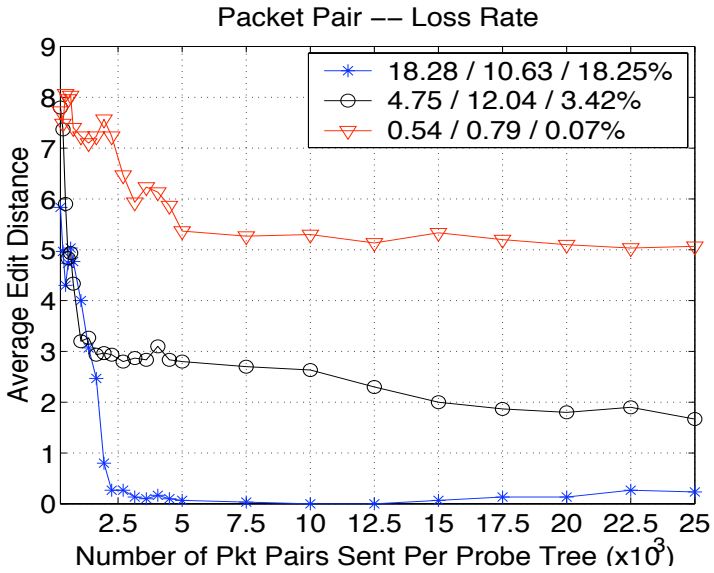


(a)



(b)

Figure 17: The performance of HTE using delay difference measured by sandwich probes (a), delay covariance measured by packet pairs (b) and loss rate measured by packet pairs (c) under various network conditions. The vertical axes show the average graph edit distance between the estimated tree and the true topology over 30 independent simulations. The horizontal axes are the number of normalized similarity samples $N_1 N_{norm}$ used in each simulation.



(c)

Figure 17 (continued)

used sandwich probes to collect similarity estimates via delay differences. For each probe tree 200 similarity estimates were collected and we set $N_{norm} = 10$ to obtain 20 normalized samples for the HTE algorithm.

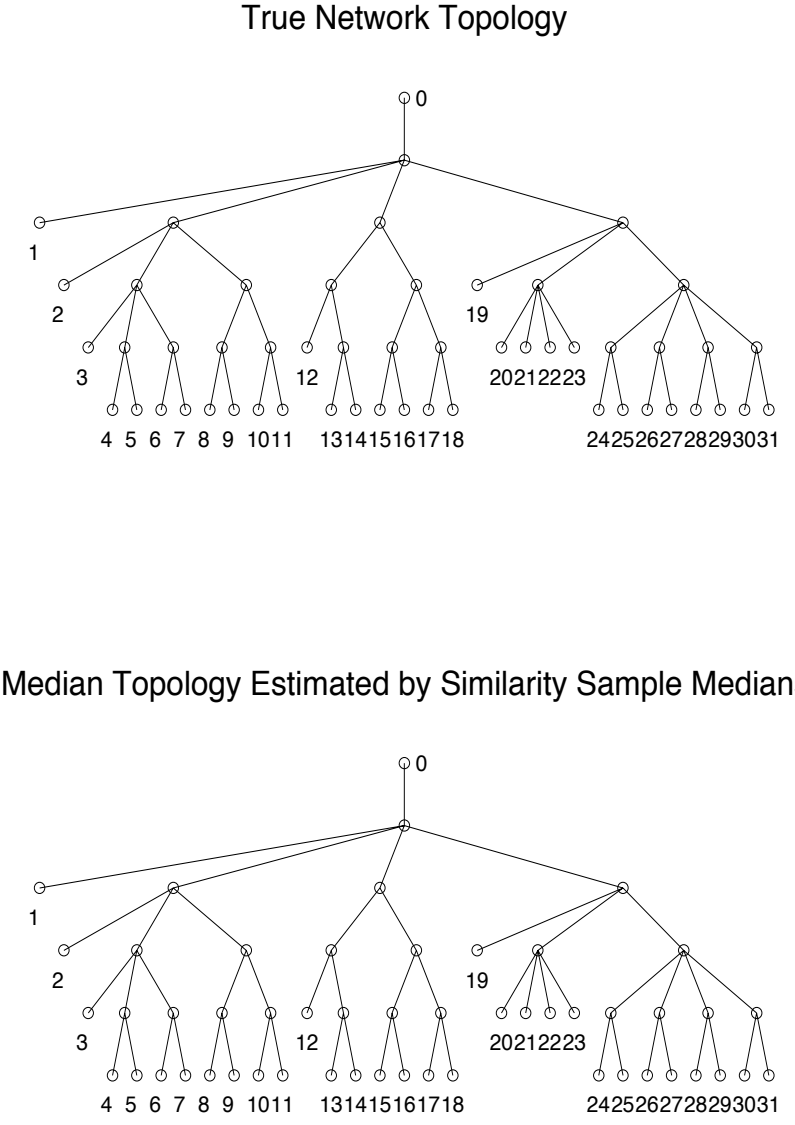
For a total of M independent simulations we defined the *median topology* as the topology estimate obtained from the median of the similarity samples over all the M simulations. For example, given a probe tree the first normalized similarity sample used for the estimation of the median topology equals the median of the first normalized samples for the probe tree over the M simulations. Then the distribution of the topology estimates can be described by the one-sided pmf of the graph edit distance between the estimate and the median topology. The median topology obtained from 30 independent simulations on the top tree network in Figure 18 is shown in the bottom of the same figure, and the corresponding distribution of the topology estimates is shown in Figure 19 as the pmf for the edit distance to the median topology. For these simulations the median topology was equal to the true topology. We will say that the topology estimate is *median unbiased*. The edit distances corresponding to the 50th and 90th percentiles are also indicated. Some examples of the estimated topology along with their distances to the median topology are depicted in Figure 20. The top tree shows one of the closest estimates to the median topology. The tree in the middle corresponds to the 50th percentile of the edit distance, and the tree in the bottom has the largest distance to the median topology over all the estimates. One can observe that the error distances came mainly from the failure to estimate fine subclusters of the leaf nodes at the bottom of the tree.

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 equivalent to infinite packet queueing delay over the link. The spatial and temporal independence assumptions of delays are violated since `ns-2` was configured to simulate bursty TCP background data 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).

6 Conclusion and Future Work

The estimation of the logical tree topology from end-to-end unicast measurements of the network was investigated. We established a general framework for this problem that requires no assumptions on the logical tree structure. The topology estimation problem was transformed into a hierarchical clustering problem for grouping the leaf nodes based on pair-wise similarities. We proposed to use a similarity clustering tree to describe the hierarchical clustering for the leaf nodes in terms of that for the pair-wise similarities, and we established a one-to-one mapping between the similarity clutsering tree and the network topology. We described three possible similarity metrics along with the probing schemes used to estimate these metrics: sandwich probes measuring the queueing delay; packet pair probes measuring the delay variance; and packet pair probes measuring the loss rate. A new finite mixture modelling approach was proposed for clustering the similarity estimates using a prior pmf on the nearest common ancester of each pair of leaf nodes. The penalized likelihood approach using MML-type penalty was also derived for model selection, which was used in an unsupervised EM algorithm for the estimation of the finite mixture model. The key to the use of the mixture model is the association of mixture components attached to the similarity $\{\gamma_{i,j}\}$ of leaf node pairs i and j belong to different clusters, called inter-cluster components. Based on the observation that the component having the smallest mean contains information important for splitting the leaf node clusters, we defined a new recursive partition likelihood as a clustering metric. The hierarchical topology likelihood was then formulated as the product of all the conditional partition likelihoods.

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 with the vertices being the leaf nodes to be partitioned. A robust estimator was proposed for the indicator function showing whether a pair of leaf nodes belong to two different clusters, based on the inter-cluster component of the finite mixture model.



(a)

Figure 18: The true topology (upper) used in `ns` simulation to illustrate the distribution of the topology estimates. The median topology over 30 independent simulations is also shown in the bottom which is identical to the true topology, and thus our topology estimator is *median unbiased*.

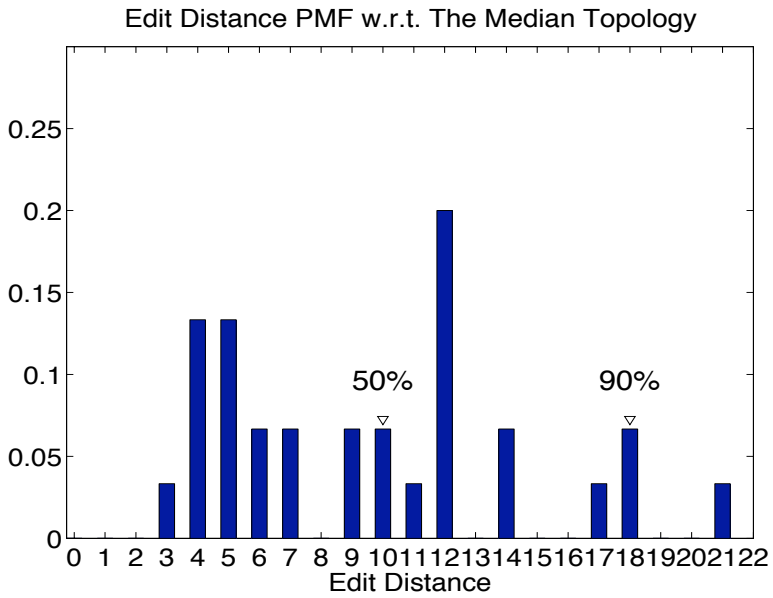
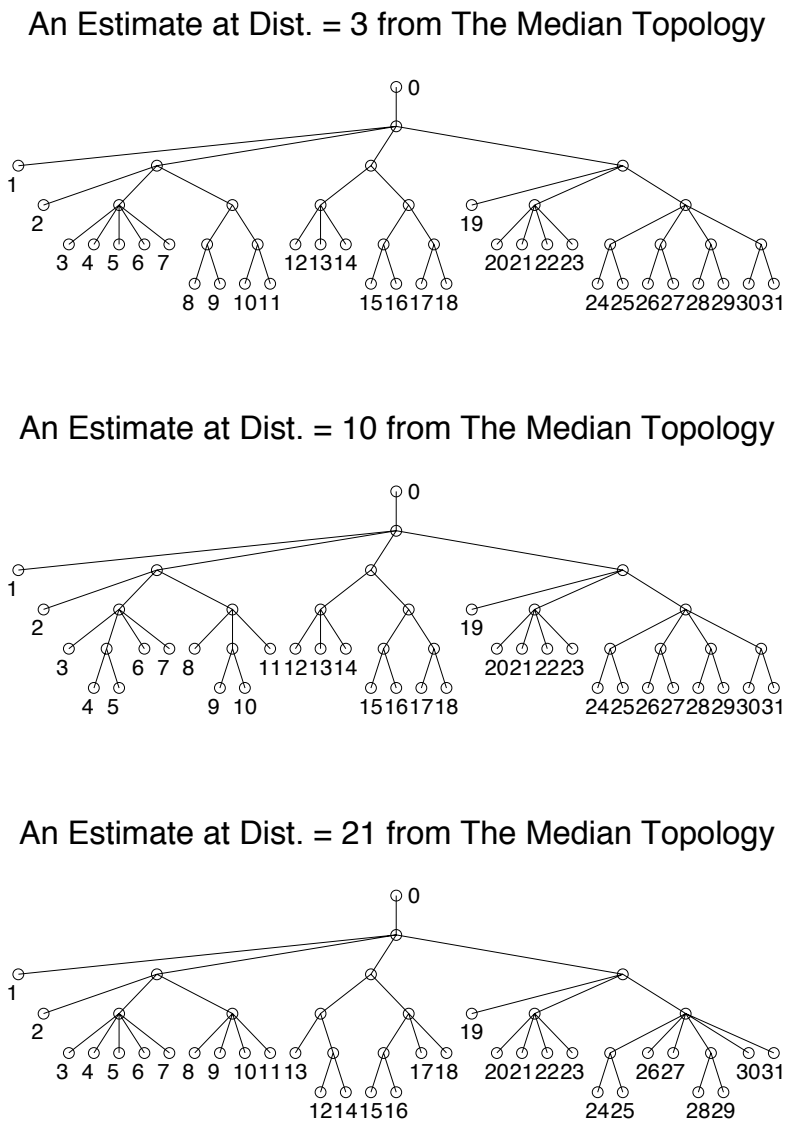


Figure 19: The pmf of the graph edit distance with respect to the median topology for the estimates from the `ns` simulation using the network in Figure 18. The locations of the 50th and 90th percentiles are also indicated.

The estimator was used to compute the edge weights indicating the pair-wise similarities. A simple clustering algorithm based on the graph connectivity was then applied to partition the leaf nodes. To reduce the complexity of the graph-based clustering when the number of leaf nodes is large we proposed a pre-clustering algorithm to reduce the vertices in the graph. When there is estimation error in the finite mixture model, the inter-cluster component could be decomposed into several small components or be merged with other components. The former problem was solved by a pre-cluster algorithm which progressively includes components, starting with the one having the smallest mean. The latter was overcome by a post-merge algorithm which tests various ways to merge pairs of closely similar clusters.

We used `matlab` model simulations on a small network to demonstrate the proposed algorithm and compared it with the DBT and LBT algorithms. The proposed algorithm achieved a lower error edit distance to the true topology and higher percentage of correctly estimated trees than the DBT and LBT, under various conditions on the magnitudes and variances of the similarity estimates. Our algorithm outperformed the DBT and LBT algorithms because we adopted a less greedy approach in finding the optimal topology based on end-to-end observations. For a more practical situation we simulated the same network in `ns-2` and tested the performance of the three similarity metrics along with the corresponding probing schemes using the proposed algorithm. The Monte-Carlo simulation results showed the delay difference measured by the sandwich probes had the best performance when the network load is light. For a moderate load situation the delay variance metric using packet pair probes provided the most reliable estimates for the leaf node similarities. When there was a good chance for packets to be discarded by a congested network the loss rate measured by packet pair



(a)

Figure 20: Examples of the estimated topology along with their tree edit distance to the median topology for the `ns` simulated network in Figure 18.

probes generated the topology estimates with the lowest error distance. We also defined the median topology of topology estimates obtained from Monte-Carlo experiments. Median topology was used to describe the distribution of topology estimates by the pmf of its graph edit distance to every topology estimate. The idea was illustrated by Monte-Carlo `ns-2` simulations on a larger network using sandwich probes.

Future work could focus on the use of hybrid probing schemes, which might better adapt to all the possible conditions in a network. The similarity samples would be a multi-dimensional vector including the measurements from different types of probes. This work could also be extended to better estimate topology of the network by sending the probes from multiple sources, as in [67, 68, 69]. Extensive real network experiments should be implemented in the future to compare to ground truth real network topologies.

References

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

[2] N.G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley, "Multicast topology inference from end-to-end measurements," *ITC Seminar on IP traffic, Measurement and Modelling*, Monterey, CA, Sep. 2000.

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

[4] N.G. Duffield, J. Horowitz, F. Lo Presti, "Adaptive multicast topology inference," *IEEE Infocom 2001*, Anchorage, Alaska, April 2001.

[5] R. Castro, M. Coates, R. Nowak, "Likelihood based hierarchical clustering," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2308-2321, Aug. 2004.

[6] M. Shih, A. Hero, "Network topology discovery using finite mixture models," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP) 2003*, Montreal, Canada, May 2003.

[7] Y. Vardi, "Network tomography: estimating source-destination traffic intensities from link data," *J. Am. Stat. Assoc.*, vol.91, pp.365-377, 1996.

[8] R. Cáceres, N. G. Duffield, J. Horowitz, D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. on Information Theory*, vol.45, no.7, Nov. 1999.

[9] M. Coates, R. Nowak, "Network loss inference using unicast end-to-end measurement," *ITC Conf. on IP Traffic, Modelling and Management*, Monterey, CA., Sep. 2000.

[10] A. Ziotopoulos, A.O. Hero III, W. Wasserman, "Estimation of network link loss rates via chaining in multicast trees," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP) 2001*, Salt Lake City, Utah, May 2001.

[11] F. Lo Presti, N.G. Duffield, J. Horowitz, D. Towsley, "Multicast-based inference of network-internal delay distributions," *IEEE/ACM Trans. on Networking*, vol. 10, pp.761-775, Dec. 2002.

[12] M. Shih and A.O. Hero III, "Unicast-based inference of network link delay distributions with finite mixture models," *IEEE Trans. on Signal Processing*, vol. 51, pp. 2219-2228, Aug. 2003.

[13] Y. Tsang, M. Coates and R. Nowak, "Network delay tomography," *IEEE Trans. on Signal Processing*, vol. 51, pp. 2125-2136, Aug. 2003.

[14] A. Bestavros, J. Byers, K. Harfoush, "Inference and labeling of metric-induced network topologies," *IEEE Infocom 2002*, New York, NY., June 2002.

[15] M. Coates, R. Castro, R. Nowak, "Maximum likelihood network topology identification from edge-based unicast measurements", *ACM Sigmetric*, Marina Del Rey, CA., June 2002.

[16] R. Castro, M. Coates, G. Liang, R. Nowak, B. Yu, "Network tomography: recent developments," to appear in *Statistical Science*. [Online]. Available: http://stat-www.berkeley.edu/users/binyu/ps/cny.pdf

[17] K.-C. Tai, "The tree-to-tree correction problem," *J. ACM*, vol. 26, no. 3, pp. 422-433, 1979.

[18] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM Journal of Computing*, vol. 18, pp. 1245-1262, 1989.

[19] D. Shasha and K. Zhang, "Fast algorithms for the unit cost editing distance between trees," *Journal of Algorithms*, vol. 11, pp.581-621, 1990.

[20] K. Zhang, R. Statman, and D. Shasha, "On the editing distance between unordered labeled trees," *Inform. Process. Lett.*, vol. 42, no. 3, pp. 133-139, 1992.

[21] *ns-2* - Network Simulator. [Online].
Available: http:www-mash.cs.berkeley.edu/ns/ns.html.

[22] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms which use cluster centres," *The Computer Journal*, vol. 26, pp. 354-359, 1984.

[23] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," *Proceedings of the 14th International Conference on Machine Learning*, pp. 170-178, Nashville, Tennessee, July 1997.

[24] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data clustering : a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, 1999.

[25] R.A. Mollineda and E.Vidal, "A relative approach to hierarchical clustering," *Pattern Recognition and Applications, Frontiers in Artificial Intelligence and Applications*, vol. 56, pp. 19-28, IOS Press, 2000.

[26] S. Guha, R. Rastogi, and K. Shim, "ROCK: a robust clustering algorithm for categorical attributes," *Information Systems*, vol. 25, no. 5, pp.345-366, 2000.

[27] K. Lai, M. Baker, "Measuring link bandwidths using a deterministic model of packet delay" *ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000.

[28] A. Pásztor and D. Veitch, "The packet size dependence of packet-pair like methods," *Proceedings of IWQoS*, Miami Beach, Florida, 2002.

[29] S. Keshav, "A control-theoretic approach to flow Control," *Proceedings of ACM SIGCOMM*, 1991.

[30] M. Coates, R. Nowak, "Network tomography for internal delay estimation ", *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP) 2001*, Salt Lake City, Utah, May 2001.

[31] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, 1994.

[32] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz, "The changing nature of network traffic: scaling phenomena," *ACM Computer Communication Review*, vol. 28, pp. 5-29, Apr. 1998.

[33] B. K. Ryu and A. Elwalid, "The importance of long-range dependence of VBR video traffic in ATM traffic engineering: myths and realities," *ACM Computer Communication Review*, vol. 26, pp. 3-14, Oct. 1996.

[34] M. Grossglauser and J. Bolot, "On the relevance of long range dependence in network traffic," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp.629-640, Oct. 1999.

[35] J. Yeo and A. Agrawala, " Multiscale analysis for wireless LAN traffic characterization," Technical Report CS-TR-4571 and UMIACS-TR-2004-16, Dept. of Computer Science, Univ. of Maryland, Mar. 2004. [Online]. Available: http://citeseer.ist.psu.edu/641388.html

[36] H. Fuks; A.T. Lawniczak, and S. Volkov, "Packet delay in models of data networks," *ACM Transactions on Modelling and Simulations*, vol. 11, pp. 233-250, 2001.

[37] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi and C. Diot, "Analysis of measured single-hop delay from an operational backbone network," *Proceedings of IEEE INFOCOM 2002*, vol. 21, no. 1, pp. 535 - 544, June 2002.

[38] O. Østerbø, "Models for end-to-end delay in packet networks," Scientific Report, Telenor, Jan. 2003. [Online].
Available: http://www.telenor.no/fou/publisering/rapp03/R_4_2003l.pdf.

[39] G. McLachlan, D. Peel, "Finite mixture models," New York: John Wiley & Sons, 2000.

[40] M. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 381-396, Mar. 2002.

[41] J.J. Oliver, R.A. Baxter, C.S. Wallace, "Unsupervised learning using MML," *Proceedings of the 13th International Conference on Machine Learning (ICML)*, 1996.

[42] A.P. Dempster, N.M. Laird, D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society (B)*, vol. 39, no. 1, pp. 1-38, 1977.

[43] A.D. Lanterman, "Schwarz, Wallace, and Rissanen,: intertwining themes in theories of model order estimation," *Int'l Statistical Rev.*, vol. 69, pp. 185-212, Aug. 2001.

[44] C. Wallace and P. Freeman, "Estimation and inference via compact coding," *Journal of the Royal Statistical Society (B)*, vol. 49, no. 3, pp. 241-252, 1987.

[45] R.A. Baxter and J.J. Oliver, "Finding overlapping components with MML," *Statistics and Computing*, vol. 10. no. 1, pp. 5-16, Jan. 2000.

[46] D.W. Matula, "K-components, clusters, and slicings in graphs," *SIAM J. Appl. Math.*, vol. 22, no. 3, pp. 459-480, 1972.

[47] D.W. Matula, "Graph theoretic techniques for cluster analysis algorithms," *Classification and Clustering*, pp. 95-129, Academic Press, 1977.

[48] E. Hartuv and R. Shamir, "A clustering algorithm based on graph connectivity," *Information Processing Letters*, vol. 76, no. 4-6, pp. 175-181, Dec. 2000.

[49] M. Brinkmeier, "Communities in graphs," Technical Report, Technical University Ilmenau, 2002. [Online]. Available: http://eiche.theoinf.tu-ilmenau.de/ mbrinkme/documents/communities.pdf

[50] R. Shamir, R. Sharan, D.Tsur, "Graph modification problems," *Lecture Notes in Computer Science*, vol. 2573, pp. 379-390, 2002.

[51] L.R. Ford and D.R. Fulkerson, "maximal flow through a network," *Canadian Journal of Mathematics*, pp. 99-404, 1956.

[52] D.W. Matula, "Determining edge connectivity in $O(nm)$," *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pp. 249-251, 1987.

[53] 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.

[54] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM Journal on Discrete Math*, pp. 54-66, 1992.

[55] D.W. Matula, "A linear time 2 + ϵ approximation algorithm for edge connectivity," *Proceedings of the 4th ACM-SIAM Symposium on Discrete Mathematics*, pp. 500-504, New York, 1993.

[56] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM*, vol. 44, no. 4, pp. 585-591, July 1997.

[57] D. Karger, "Minimum cuts in near-linear time," *J. of ACM*, vol. 47, no. 1, pp. 46-76, 2000.

[58] S. M. Selkow, "The tree-to-tree editing problem," *Inform. Process. Lett.*, vol. 6, no. 6, pp. 184-186, 1977.

[59] G. Valiente, "An efficient bottom-up distance between trees," *Proceedings of the 8th International Symposium of String Processing and Information Retrieval*, pp. 212-219, Laguna de San Rafael, Chile, 2001.

[60] K. Zhang, D. Shasha, and J. Wang, "Approximate tree matching in the presence of variable length don't cares," *Journal of Algorithms*, vol. 16, no. 1, pp. 33-66, 1994.

[61] K. Zhang, J. Wang, and D. Shasha, "On the editing distance between undirected acyclic graphs and related problems," *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching*, Helsinki, Finland, July, 1995.

[62] P.N. Klein, "Computing the edit-distance between unrooted ordered trees," *Proceedings of the 6th Annual Symposium on Algorithms (ESA) 1998*, pp. 91-102, Springer-Verlag, 1998.

[63] T. Jiang, L. Wang, and K. Zhang, "Alignment of trees - an alternative to tree edit," *Theor. Comput. Sci.*, vol. 143, no. 1, pp. 137-148, 1995.

[64] E. Tanaka and K. Tanaka, "The tree-to-tree editing problem," *J. ACM*, vol. 26, no. 3, pp.422-433, 1979.

[65] W. Yang, "Identifying syntactic differences between two programs," *Software - Practice and Experience*, vol. 21, no. 7, pp. 739-755, 1991.

[66] K. Zhang, R. Statman, and D. Shasha, "On the editing distance between unordered labeled trees," *Inform. Process. Letters*, vol. 42, no. 3, pp. 133-139, 1992.

[67] M. Rabbat, R. Nowak, M. Coates, "Network Tomography and the Identification of Shared Infrastructure", *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA., Nov. 2002.

[68] M. Coates, M. Rabbat, R. Nowak, "Merging logical topologies using end-to-end measurements," *ACM Internet Measurement Conference*, Miami, FL., Oct. 2003.

[69] M. Rabbat, R. Nowak, M. Coates, "Multiple Source, Multiple Destination Network Tomography," *IEEE Infocom*, Hong Kong, Mar. 2004.