UNICAST INTERNET TOMOGRAPHY

by

Meng-Fu Shih

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2005

Doctoral Committee:
Professor Alfred O. Hero III, Chair
Professor Jeffrey A. Fessler
Associate Professor George Michailidis
Assistant Professor Mingyan Liu

To my grandparents, my parents, my brother and his family.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Alfred O. Hero III, for his guidence, stimulus, and support throughout my graduate study in the University of Michigan. His brilliant insights and ideas have always enlighted me the appropriate ways to approach engineering problems. His patient and understanding personaltiy makes it an enjoyable experience to work with him. It would be impossible for me to finish this work without his support and motivation. I am sincerely grateful to other committee members, Professor Jeffrey A. Fessler, Professor Mingyan Liu, and Professor George Michailidis for their time and effort in reviewing my work and their valuable suggestions and comments on my dissertation.

I am grateful to the following funds which supported me financially throughout my graduate study: the MURI project on Low Energy Electronics Design for Mobile Platforms (ARO DOD-G-DAAH04-96-1-0377) and the NSF project on Modular Startegies for Internetwork Monitoring (NSF CCR-0325571).

I would like to thank my friends and collegues in the University of Michigan. I was benefited greatly from their knowledge and experience in research. I also enjoyed playing badminton and volleyball with them. I am as well grateful to the Ann Arbor Buddhist Society. It led me into a new spiritual world and brought me new meanings of life. The society members have become good friends of mine and helped me to relieve the tension by sharing their leisure time with me. I also received a lot of favor

from them in solving my daily life problems.

This work would not have been possible without the unconditional support and love from my family. I would like to express my deepest love and gratitude to my grandmother who passed away in the beginning of the year when I came to US. Without her caring and nourishing, I would not have grown up safe and sound. I am obliged a lot to my parents for their constant support and encouragement. I especially want to thank them for having a strong faith in me all the time. I am also grateful to my brother, who has always been generous and kind to me. I would also like to express my best wishes to his newly born baby, Yi-Ning. Finally, my sincere gratitude goes to my lovely girlfriend, Jing-I, for her comfort and caring which provides me the strength to finish my graduate study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure

LIST OF APPENDICES

APPENDIX

ABSTRACT

UNICAST INTERNET TOMOGRAPHY

by

Meng-Fu Shih

Chair: Alfred O. Hero III

Inference of network internal characteristics has become an increasingly important issue for communication network operation. Since it is impractical to directly monitor the network internal nodes, people use the end-to-end information collected by probe packets to estimate the statistics of interest. This new area of networking research is called *network tomography*. When specialized to the Internet it is called *Internet tomography*.

Our work focuses on unicast probing methods, which are supported by much of today's Internet. We first deal with the estimation of internal link delay distributions from end-to-end delay measurements. Unlike the discrete delay models used in previous work, we focus on continuous distributions of non-zero queueing delays. We send individual unicast packets throughout the network and develop an estimator for link delay cumulant generating functions (CGF) based on an over-determined system of equations. We propose a bias corrected estimator for the CGF which eliminates the

nonlinearity effect of the log function. When the network is modelled by a logical tree we use packet pair probes to collect end-to-end delay information. We propose a novel hybrid continuous/discrete finite mixture model for the link delay distributions. A penalized maximum-likelihood expectation-maximization (PML-EM) algorithm is developed to select the model and estimate its parameters. Since the complexity of the algorithm grows exponentially with the size of the network, we propose an accelerated algorithm to obtain a linear reduction in run-time.

The second problem we address is network topology discovery using end-to-end measurements. Topology estimation can be formulated as a hierarchical clustering problem of the leaf nodes based on pair-wise correlations as similarity metrics. Unlike previous work which first assumes the network topology being 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 hierarchical clustering algorithm for the leaf nodes. The performance of our algorithms are evaluated by `matlab` and `ns-2` simulations.

CHAPTER 1

Introduction

1.1 Background

Internet monitoring and diagnosis provides essential information for network operation, such as link delays, packet loss rates, and traffic intensities. Administrators with those statistics at hand are easily able to make operational decisions such as bandwidth allocations and upgrade plans. It is also possible to develop more sophisticated routing protocols or admission control schemes based on such information. End users and service providers can also use monitoring information to verify the quality-of-service (QoS) delivered across their administered domain. With the increasing threat of malicious activities in the Internet, it is essential to be able to promptly characterize network conditions for early detection of anomalies.

However, due to the large-scale, distributed, and heterogeneous structure of today's Internet, problems arise when people try to obtain the internal network information. Router-based direct monitoring software is usually disabled to avoid computation and communication overhead. Internal network information is usually considered confidential and is not shared freely with outsiders. Even when these

direct monitoring and confidentiality problems do not exist, it requires extravgant bandwidth and coordination efforts to relay data between every collection site and the processing centers. These factors have hindered efforts at real time optimization of network performance, e.g., finding the best traffic route, or determining a culprit for performance degradation such as a bottleneck link.

An emerging research field called *network tomography* was proposed by Vardi [1] to circumvent such difficulties. He investigated a type of network tomography problem in which end-to-end statistics are estimated from the observations inside the network. He used passive measurements of aggregated traffic rates at internal network devices to infer data flow intensities between directed source-destination pairs. The name *tomography* came from its strong resemblance of the problem formulation in positron emission tomography (PET) of medical imaging [2]. His work was followed by Tebaldi and West [3] who tried to solve the problem from a Bayesian perspective. Cao *et al.* developed Gaussian models for the link count data with power-law relationship between the means and the variances, and dealt with the non-stationary nature of the data by fitting their basic model locally using a moving data window [4].

To collect information from existing traffic flows in the network is called *passive probing*. It has the advantage of not generating much overhead and not significantly purturbing traffic flow. However, with passive probing there is no control over the quality of the collected statistics. For example, the traffic rate of the monitored data stream may be too low to capture the rapidly changing dynamics. An alternative is to send probe packets through the network and measure parameters such as packet loss rates or end-to-end delays; which is called *active probing*. The statistics collected by the probes are viewed as samples of true network statistics. To maintain minimal impact on the network performance one must carefully control the data rate of the probes.

Active probing schemes were used by Cáceres *et al.* [5] for a second type of network tomography, in a sense the dual of Vardi's application, where end-to-end observations are used to estimate internal statistics or parameters. They focused on sending probes over multicast networks and formed sample-average estimates for link packet loss probabilities. In a multicast network a packet can bear multiple destination addresses. The network automatically replicates the packet at the branching points of each path and every destination receives one of the copies. The correlation among end-to-end measurements can be used to draw inferences about the characteristics of the intersecting portion of the paths, without need for cooperation by intermediate network devices. Ziotopoulos *et al.* presented an estimator for link loss rates via chaining in multicast trees [6]. An efficient least-square estimator for link loss probabilities using end-to-end multicast measurements was proposed by Michailidis *et al.* [7, 8].

Although the performance of the multicast-based algorithms was impressive, the multicast protocol is not supported by most part of the Internet. Furthermore, however, the major component of the Internet traffic is unicast. The performance measured by multicast probes may differ considerably from that encountered by unicast traffic due to different processing by the routers [9]. Coates and Nowak proposed a unicast packet pair approach to mimic the behavior of multicast probes with two destinations [9]. A maximum likelihood estimator (MLE) for link loss rates was also provided.

Active unicast probing methods were also employed in *bottleneck bandwidth* estimation [10, 11, 12, 13, 14]. `Pathchar` [12] is a tool designed by Jacobson to infer the characteristics of individual links along an Internet path by measuring the round trip time of packets sent from a single host. Downey [13] improved its accuracy and applied adaptive data collection to reduce the required number of measurements. Lai

and Baker [14] presented a *packet-tailgating* technique to measure link bandwidths using packet delays.

Estimation of packet link delay statistics from end-to-end measurements is another research subject of recent interest. The causes of delays along a packet's path through the network can be separated as the sum of two types of delays: constant link transmission delays and time-varying link processing delays. Link transmission delays are due to the propagation delays through the physical medium, e.g., a wire, or optical fiber. Link processing delays are due to a combination of router queueing, buffering and servicing delays that depend on factors such as: the amount of cross-traffic at the router, the number of retransmits required over the link, and the integrity of router equipment and associated software. While transmission delays usually remain constant over a probing interval, processing delays are highly variable and commonly modelled as random variables. Thus it is generally impossible to recover the actual internal link delays that packets encounter along their end-to-end paths. However, the determination of the statistical distribution of the internal link delays from multiple end-to-end delay measurements can be formulated as a statistical inverse problem whose solution yields estimates of the internal delay distributions [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. These estimates can be used by an autonomous system (AS), e.g., an Internet service provider (ISP), to evaluate its average quality of service or to assess link performance of other, perhaps competing, AS's. When acquired over large portions of the network, link delay estimates can also be used for detecting network anomalies such as imminent link failures or coordinated denial of service (DoS) attacks.

Lo Presti *et al.* were the first to estimate link delay distributions from end-to-end measurements [16]. They uniformly discretized the delays and derived an algorithm based on empirical histogram estimation. Their method uses multicast probes, which

requires cooperation of the network to run a multicast session such as *Real-time Transport Protocol* (RTP) during the probing interval. To overcome this restriction, Coates and Nowak [17] developed an internal delay histogram estimator based on an alternative unicast probing scheme in which edge sites exchange a succession of closely spaced packet pairs. Their estimator is based on a statistical inverse problem formulation and used an iterative maximum likelihood via expectation maximization (ML-EM) approach. In related work these authors also developed a sequential Monte-Carlo method for tracking changes in non-stationary networks [20]. The principal restriction of the approaches in [16, 17, 20] is the requirement of discrete-valued link delays. Overly coarse discretization, or binning, of the link delays leads to excessive model approximation error and causes bias in derived estimates such as delay means and variances. At the opposite extreme, excessively fine discretization leads to high run-time complexity of these algorithms. Furthermore, the determination of the appropriate number and size of the bins requires tight bounds on link delay characteristics, such as maximum and minimum processing delays, which are usually unknown.

Several alternatives to the fixed and uniform binning scheme have been studied. Duffield *et al.* considered a variable bin size model, where smaller bins are used to describe probability mass concentrations for small delays [19]. Tseng, Coates, and Nowak proposed a nonparametric algorithm where the number of bins for internal link delays is adapted to the number of measurements [23]. They used a wavelet-based penalized maximum likelihood estimator to smooth the estimates.

Network tomography can also be applied to the estimation of the topology. The topology of the Internet is constantly changing due to devices going online and offline, and the corresponding routing table updates. Tools such as `traceroute` are usually used to obtain information about traffic paths in the network, and these

can be integrated to construct the network topology. However, such tools rely on the cooperation of internal routers, and have become less attractive as the traffic load of the Internet grows rapidly. Network tomography on topology discovery was initially investigated in [27, 28]. 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) [29, 30, 31] 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 [32]. The extension to a general tree is basically done by pruning the links with loss rates less than some heuristically selected threshold. The multicast algorithm of Lo Presti was also extended to use other metrics such as packet delays [30, 31].

The fundamental idea of topology inference falls in the framework of *metric-induced network topologies* (MINT) [33]. Every path is associated with a metric function, such as the product of successful transmission rates or the sum of delays on every link in 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 larger (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 packet loss is measured from unicast packet pair probes, packet loss rate on the shared path from root to a pair of leaf nodes can be estimated. The loss rate has non-decreasing mononicity because it never decreases as the number of links in the path increases.

Topology estimation in unicast networks was investigated by Castro *et al.* [32, 34, 35, 36, 37, 38, 39]. 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 another. Under a light load assumption, queueing delay for the second small packet behind the large one disappears once they depart from each other. The corresponding path metric is the delay difference between the two small packets. They 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) [32]. The special case of Gaussian distributed measurements was previously called the *likelihood-based binary tree algorithm* (LBT) [34]. 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 used a Monte-Carlo Markov Chain (MCMC) method to generate a sequence of tree candidates by birth (node insertion) and death (node deletion) transitions [32, 39]. The tree candidate which gives the highest likelihood is adopted as the estimate of the topology.

1.2 Contributions and Dissertation Outline

Our work focuses on the use of unicast probes for delay tomography and topology discovery. In Chapter 2 we propose a nonparametric estimation method for estimating internal link delay distributions using cumulant generating functions (CGF) [18]. The sum of link transmission and processing delays over a route can be measured by end-to-end delays of unicast probes sent over the route. After collecting a sufficient number of these probes over more different paths than internal links, an overdetermined system of equations is constructed for the delay CGF's. Based on a least-squares approximation, we propose a bias-corrected estimator for each link delay CGF. We evaluate the performance of the algorithm using the `ns-2` [40] network simulator. Several measures of performance are investigated, including overall mean-square goodness of fit of the estimated CGF to the empirical CGF, bottleneck localization, and bottleneck detection probability. Our contribution in this chapter is summarized as follows:

- We develop an estimator for link delay CGFs based on an overdetermined system of equations formulated by end-to-end measurements, without assuming the link delays to be discretely distributed as in previous work.
- We propose a method to correct the bias introduced by the non-linearity of the log function in the link delay CGF estimator and the result achieves closer match to true link delay CGFs than the original estimator.
- We illustrate an application of link delay CGFs in bottleneck link detection.

In Chapter 3 we move our attention to a more practical situation where the unicast probes are sent over fewer paths than internal links. The network is modelled as a logical tree with its root node being the probe source and its leaf nodes being

the receivers. In this case one can only form an underdetermined system of equations for the link delay CGF's. We take an alternative approach in which we first define an approriate link delay model and then estimate the model parameters based on the *incomplete* end-to-end observations.

The problem of empirically characterizing Internet link delay distributions has been investigated by several groups (see, for example, [41, 42, 43, 44]). A common observation is that when the link is lightly loaded, such as in the early morning, the link delay scatterplots appear stationary. Furthermore, while much of the scatter appears spread out over a continuum of delay values, a non-negligable proportion of the delays appears to concentrate at one or more discrete values, see for example [44, Figure 4]. This implies the existence of point masses in the time-averaged link delay distribution. The positions of these point masses vary according to factors such as: length of packet; incoming and outgoing queue sizes of routers on the link; router configuration; deployment of firewalls; and the physical distance between routers [44].

We propose to capture these empirically observed features by fitting hybrid continuous/discrete finite mixture models to the link delay distributions. While our algorithms are easily generalizable to multiple discrete point masses, for simplicity we focus on the case where the discrete component is a single point mass. Unlike purely continuous models the hybrid continuous/discrete model is identifiable and is justified under the lightly loaded scenario. In this scenario there is a non-zero probability that a packet will encounter an empty queue in which case the packet delay is non-random, being due to fixed propagation and processing delays. While this is unlikely in a congested network, the model is valid for a number of common monitoring situations such as service and performance verification and detection of onset congestion. Moreover, we would like to point out that the delay point mass is

implicit in canonical delay trees, used in discrete delay tomography, for which there is a non-zero probability that a packet traverses each link without any delay (see, e.g., [16]).

Herein we propose a method for estimation of internal delay distributions from unicast end-to-end measurements which is based on packet pair unicast probes and additive mixture models for the internal link delays. As the end-to-end delay measurement is a sum of the (assumed independent) internal link delays over the probe path, the densities of the measurements are convolutive mixtures of these additive mixture models. This makes our estimation problem more challenging than the standard mixture model estimation problem which has received much attention in both the statistical and engineering literature [45, 46, 47, 48]. Additional issues which we address are: 1) the additive mixture model orders are unknown in practice; and 2) the internal link delay distributions are composed of a combination of continuous and a discrete components. As the measurements are end-to-end delays, our estimation problem becomes the inference of a set of mixture models based on the realizations of their convoluted density functions, which are also mixture models by themselves. We handle this convolutive mixture complication by adopting an iterative ML-EM formulation of the estimation problem using an enlarged complete data space. We handle the problem of unknown model order by adapting the unsupervised minimum-message-length (MML) approach used in Figueiredo and Jain [48]. Specifically, we add an information theoretic order selection penalty to the log-likelihood to which a penalized ML-EM (PML-EM) algorithm is applied. We handle the presence of both discrete and continuous link delay components by the following simple additive mixture model: the delay density is a (unknown) convex combination of a point mass positioned at the transmission delay and a (unknown) number of mixture components with (unknown) means and variances. We adopted

Gaussian continuous components to simplify the implementation but heavy-tailed densities can also be easily accommodated in our framework. We illustrate the performance of the ML-EM and PML-EM algorithms on simulated data using `matlab` and `ns-2` simulators. One of the key observations is that while performance is good, the run-time is excessively long, due to slow convergence of the PML-EM algorithm. Our contribution in this chapter is summarized as follows:

- We propose a hybrid finite mixture model for link delay distrbutions with non-zero continuous mixture density and a point mass modelling empty link queue events. The proposed model avoids the drawbacks of discrete delay models used by previous work in network delay tomography.

- For the case when the mixture model order is given for each link in the network, we develop an ML-EM algorithm to estimate the parameters in link delay distributions based on end-to-end delay measurements.

- Without any prior information on mixture model orders, we develop an unsupervised PML-EM algorithm using MML-type model order penalty to estimate link delay distributions based on end-to-end delay measurements.

- Throughout model and `ns` simulations we demonstrate that our algorithms can produce accurate estimates for link delay distributions and link delay ranges without forcing quantization on delay values as in previous work.

In Chapter 4 we address the slow convergence problem in the algorithm based on the hybrid mixture models presented in Chapter 3. The mixture model order, i.e., the number of components in a mixture model, for end-to-end delays increases exponentially as the network size grows. Although it is impossible to diminish the complexity below the exponential rate because the number of links and probe paths

grows exponentially with tree depth, a feasible acceleration to reduce the exponential rate constant can be obtained. By representing the topology as the union of *probe trees*, the binary tree composed of the paths from the root to a pair of leaf nodes, the accelerated algorithm starts from the probe trees with the lowest branch splitting nodes and estimates the branch link delay distributions using the PML-EM algorithm. The path in a probe tree shared by both packets in a packet-pair probe is called the *shared path* of the probe tree. The key approximation is to consider the chain of links in the shared path as a single link and assign its delay distribution to a component number for a single link delay mixture model. This significantly reduces the end-to-end mixture model orders by sacrificing the number of allowed components on the shared paths. Our analysis shows that the exponential rate constant of the computational complexity approximately becomes independent of the link model orders for the shared paths. The approximation is justified under the circumstance that there are insufficient samples collected for resolving the distribution details, possibly due to the limited probing rates and/or the short time length of the probing session. These situations are common in practice because the probing rates need to be restricted to avoid congestion and the probing session length is upper-bounded by the network stationarity time, which is usually short due to the high variability of today's Internet environment. The accelerated algorithm is tested using the `ns-2` data generated for Chapter 3. The algorithm acceleration saves approximately 40% of the run-time without significant degradation in performance from that of the original algorithm. Our contribution in this chapter is summarized as follows:

- We propose an accelerated estimation algorithm for network link delay distributions based on end-to-end delay measurements. Our algorithm divides and conquers the problem in a bottom-up fashion which allows parallel processing

of the probe trees with branch-splitting nodes having the same depth.

- Through `ns` simulation we demonstrate the accelerated algorithm produces accurate estimates for link delay distributions and saves approximately 40% run-time compared to the original algorithm in Chapter 3.

In Chapter 5 we focus on the topology discovery problem using end-to-end measurements. The purpose of this work is to develop a likelihood-based deterministic algorithm which directly estimates a general logical tree topology. Estimating the topology is equivalent to hierarchical clustering of the leaf nodes using end-to-end data. Every internal node specifies a unique cluster of descendant leaf nodes which share the internal node as their common ancester. Each leaf node by itself is also considered as a *trivial cluster*. The clusters specified by sibling nodes, i.e., nodes having the same parent, define a partition for the set of leaf nodes specified by their parent node. Given a partition we call a pair of leaf nodes *intra-cluster* if they are in the same cluster, otherwise they are called *inter-cluster*. The similarities between every pair of leaf nodes can be described by the path metric function from the root node to their nearest ancestor in an MINT. In other words, the leaf nodes which have a deeper nearest common ancestor are always more similar than the shallower ones. For the binary tree depicted in Figure 1.1 the nearest common ancestors of the leaf node pairs (4,5) and (4,6) are node 2 and 1, respectively. As node 2 is deeper than node 1 in the tree, we conclude that node 4 is more similar to node 5 than node 6.

We describe three probing schemes to estimate the similarities: delay difference measured by the sandwich probes, delay variance measured by the packet pair [49], and loss rate measured by the packet pair. It is assumed that the estimates for every leaf node pair can be described by a pdf whose mean equals the true similarity value. Unlike the likelihood method proposed in previous work [30, 34], we add a

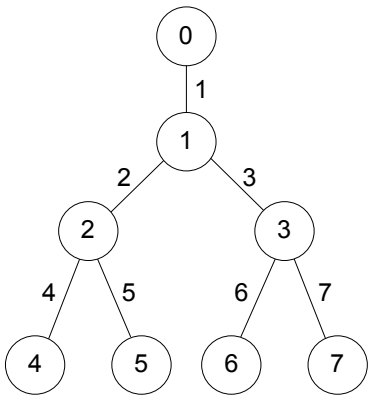


Figure 1.1: The four-leaf binary tree network.

prior on the nearest common ancester of the leaf node pairs to the model of the pair-wise similarities. This prior specifies for every internal node a probability mass proportional to the number of leaf node pairs who share the internal node as their nearest ancestor. It results in a finite mixture model in which every component corresponds to a distinct internal node. We propose an unsupervised PML-EM algorithm to estimate the finite mixture model with overfitting penalties derived from the MML criterion. Given the mixture model for the set of descendant leaf nodes of an internal node, the component with the smallest mean is exactly supported by the inter-cluster pairs of the leaf nodes for the partition derived from the children of the internal node. We call this component as *inter-cluster component*. It motivates a penalized partition likelihood of leaf nodes which is the product of individual cluster likelihoods times the inter-cluster likelihood. Each cluster likelihood itself obeys a finite mixture model and the inter-cluster likelihood is specified by the inter-cluster component. Then a novel *hierarchical topology likelihood* can be defined as the product of the conditional likelihoods for all the partitions of the leaf nodes resulting from the tree topology.

Our topology estimation algorithm adopts a greedy strategy to maximize the hierarchical topology likelihood with respect to the leaf node partitions at each level: it first finds the PML partition of the leaf nodes, then it finds the PML subpartition for every cluster, and so on, until no further subpartition is found. The estimation of each individual partition is accomplished by applying graph-theoretic clustering algorithms to a weighted complete graph derived from the set of leaf nodes to be partitioned. The vertices in the graph are the leaf nodes, and the edge weight connecting node i to j is 1 minus the probability that the similarity measurements for i and j are contributed by the inter-cluster component. Clustering problems based on graph connectivity has been widely investigated by the statistical and engineering communities. Many algorithms have been available in the literature [50, 51, 52, 53, 54]. Although basically any one of them would work in our algorithm, we choose to use a simple method called the *highly connected subgraphs* (HCS) algorithm proposed in [52]. We also develop pre-cluster and post-merge algorithms to improve the topology estimate. The performance of our topology discovery algorithm is compared with the DBT and LBT algorithms using `matlab` model simulations. Our algorithm is also applied to an `ns` simulated network and compared for sandwich and packet pair probing schemes. Our contribution in this chapter is summarized as follows:

- We suggest a finite mixture model for end-to-end similarity measurements of the leaf nodes in a logical tree network, and develop a PML-EM estimation algorithm using MML-type model order penalty.

- We propose a hierarchical topology likelihood for logical tree networks based on finite mixture models of end-to-end measurements.

- We develop a general unsupervised hierarchical estimation algorithm over logical tree network topologies using end-to-end measurements.

- We introduce packet pair probes in unicast network topology discovery and suggest two types of similarity metric measured by packet pairs. We compare the performance of the suggested packet pair probing schemes with that using sandwich probes through `ns` simulation under various network load conditions. Conclusions on the best performance scenario for each probing scheme is provided.

- Through model simulation we demonstrate that our algorithm outperforms the DBT and LBT algorithms because our algorithm adopts a less greedy approach to find the optimal topology.

- For Monte-Carlo experiments on network topology discovery, we define the median topology and the distribution of topology estimates using graph edit distance.

Chapter 6 concludes the thesis and discusses possible future directions. Some of the elements in this dissertation can be found in [18, 22, 24, 49].

CHAPTER 2

Unicast-Based Inference of Network Link Delay Distributions Using Cumulant Generating Functions

2.1 Introduction

Internal link delay statistics provide important information for network operation and security. The delay distributions can be reconstructed from their cumulant generating functions (CGF) since the CGF preserves the complete set of moments. Generally it is not possible to recover link delay CGF's from end-to-end accumulated delays. However, when there are more probe paths than internal links, we can form an over-determined system of equations for the CGF under the assumptions of a stationary environment and spatial and temporal independence. Although an estimate for the link delay CGF can be obtained by solving a linear inverse problem when it is full rank, the estimate is statistically biased due to the non-linearity of the log function in the CGF. We propose a bias correction using first and second order terms derived from the Taylor's expansion. We use `ns-2` simulations to demonstrate

the improvement achieved. An application to bottleneck link identification using the Chernoff bound is also illustrated.

Our contribution in this chapter is summarized as follows: (1) We develop an estimator for link delay CGFs based on an overdetermined system of equations formulated by end-to-end measurements, without assuming the link delays to be discretely distributed as in previous work; (2) We propose a method to correct the bias introduced by the non-linearity of the log function in the link delay CGF estimator and the result achieves closer match to true link delay CGFs than the original estimator; (3) We illustrate an application of link delay CGFs in bottleneck link detection.

The chapter is organized as follows. Section 2.2 introduces the network link and end-to-end delay model. Section 2.3 describes the proposed bias corrected estimator for the link delay CGF. Section 2.4 shows the computer experiment results. The bottleneck link identification using link delay CGF is illustrated in Section 2.5. Section 2.6 provides the conclusion and future work.

2.2 Network Delay Model

Let a communication network consist of m internal links. Identical probe packets are sent through n paths across the network. Suppose we know the routing of each of the probes which specifies the $n \times m$ probe routing matrix $\mathbf{A}$. $\mathbf{A}$ has elements a_{ij} equal to 1 when probe path i intersects link j, and equal to 0 otherwise. Let $\mathbf{M}_i$ denote the set of link indices which compose the ith probe path, $i = 1, \ldots, n$. Then $Y_i = \sum_{j \in \mathbf{M}_i} X_{ij}$ is the measured end-to-end delay of a probe transmitted along the ith path, where X_{ij} is the delay encountered by probe i across link j and $i = 1, \ldots, n$. Define the end-to-end probe delay cumulant generating function $K_{Y_i}(t) = \log E[e^{tY_i}]$ and the link delay CGF of the jth link $K_{X_{ij}}(t) = \log E[e^{tX_{ij}}]$,

$j \in \mathbf{M}_i$, with CGF parameter t, $t \in (-\infty, \infty)$. We make the following spatial independence and stationarity assumptions, respectively:

A1) The link delays X_{ij} are mutually independent, $j \in \mathbf{M}_i$, $i = 1, \ldots, n$.

A2) If paths of probe i and probe k both contain a common link j, then X_{ij} and X_{kj} are independently and identically distributed (i.i.d.) and have the same CGF denoted by K_{X_j}.

Under these assumptions the CGF of Y_i can be expressed as

$$\begin{aligned} K_{Y_i}(t) &= \log E\left[e^{tY_i}\right] \\ &= \log E\left[e^{t\left(\sum_{j\in\mathbf{M}_i} X_{ij}\right)}\right] \\ &= \log\left\{\prod_{j\in\mathbf{M}_i} E\left[e^{tX_{ij}}\right]\right\} \\ &= \sum_{j\in\mathbf{M}_i} \log E\left[e^{tX_{ij}}\right] \\ &= \sum_{j=1}^{m} a_{ij}\cdot K_{X_j}(t) \\ &= \mathbf{A}_{(i)}\cdot\mathbf{K}_X(t), \qquad i = 1, \ldots, n, \end{aligned} \tag{2.1}$$

where $\mathbf{A}_{(i)}$ denotes the ith row of the routing matrix $\mathbf{A}$ and $\mathbf{K}_X(t) = [K_{X_1}(t), \ldots, K_{X_m}(t)]^T$ (^{T} denotes transpose). Thus we can express the vector of end-to-end CGF's $\mathbf{K}_Y(t) = [K_{Y_1}(t), \ldots, K_{Y_n}(t)]^T$ by the linear relation

$$\mathbf{K}_Y(t) = \mathbf{A}\cdot\mathbf{K}_X(t). \tag{2.2}$$

When $\mathbf{A}$ is known, $n \geq m$ and $\mathbf{A}$ is full rank, the relation (2.2) is invertible and thus $\mathbf{K}_X(t)$ can be determined from $\mathbf{K}_Y(t)$ by the familiar formula $\mathbf{K}_X(t) =$

$(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}_Y(t)$. Let $\mathbf{B} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, then we have

$$K_{X_j}(t) = \sum_{i=1}^{n} b_{ji} \cdot K_{Y_i}(t) \tag{2.3}$$

where b_{ji} is the element of $\mathbf{B}$ in the jth row and ith column. A full rank matrix $\mathbf{A}$ can be ensured by making $n \geq m$, and selecting distinct probe paths which cover the network, i.e., every link is contained in some path. When $\mathbf{A}$ is not full rank, only linear combinations of link CGF's lying outside of the null space of $\mathbf{A}$ can be determined from (2.2).

2.3 Estimation of CGF

Let N_i be the number of probes collected for a given path i, $i = 1, \ldots, n$. Define

$$\hat{M}_{Y_i}(t) = \frac{1}{N_i} \sum_{k=1}^{N_i} e^{tY_{ik}}, \tag{2.4}$$

where Y_{ik} is the measured end-to-end delay of the kth received probe along path i. We obtain estimates of the vector $\mathbf{K}_X(t)$ from $\hat{\mathbf{M}}_Y(t) = [\hat{M}_{Y_1}(t), \ldots, \hat{M}_{Y_n}(t)]^T$ by the method of least-squares (LS). Note that as $\hat{M}_{Y_i}(t)$ is an unbiased estimate of the moment generating function $M_{Y_i}(t) = e^{K_{Y_i}(t)}$, a plausible estimator for $K_{X_j}(t)$ in (2.3) would be the method-of-moments estimate (MOM):

$$\hat{K}'_{X_j} = \sum_{i=1}^{n} b_{ji} \cdot \log \hat{M}_{Y_i}(t) \tag{2.5}$$

Unfortunately, this estimator is biased due to the non-linearity of the log function. In order to obtain a bias corrected estimator for $K_{X_j}(t)$, we apply a technique similar to that of Gibbens [55]. In [55] linearization was used to derive a bias corrected esti-

mator for *effective bandwidth*, which is of similar mathematical form to the cumulant generating function. Observe that as $\log(1+u) = u - \frac{u^2}{2} + H.O.T.$; hence we get

$$\begin{aligned}
\hat{K}'_{X_j}(t) &= \log\left\{\prod_{i=1}^{n}\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right\} \\
&= \log\left\{\prod_{i=1}^{n} E^{b_{ji}}\left[\hat{M}_{Y_i}(t)\right] - \left(\prod_{i=1}^{n} E^{b_{ji}}\left[\hat{M}_{Y_i}(t)\right] - \prod_{i=1}^{n}\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right)\right\} \\
&= \log\left\{\prod_{i=1}^{n} E^{b_{ji}}\left[\hat{M}_{Y_i}(t)\right]\left(1 - \left(1 - \frac{\prod_{i=1}^{n}\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}}{\prod_{i=1}^{n} E^{b_{ji}}\left[\hat{M}_{Y_i}(t)\right]}\right)\right)\right\} \\
&\approx K_{X_j}(t) - w_j - \frac{1}{2}w_j^2, \qquad (2.6)
\end{aligned}$$

where $w_j = 1 - \frac{\prod_{i=1}^{n}\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}}{\prod_{i=1}^{n}\left(E\left[\hat{M}_{Y_i}(t)\right]\right)^{b_{ji}}}$. This suggests that a reasonable way to correct the bias is to use (2.6) with an estimate of w_j:

$$\hat{K}_{X_j}(t) = \sum_{i=1}^{n} b_{ji}\log\left(\hat{M}_{Y_i}(t)\right) + \hat{E}[w_j] + \frac{1}{2}\hat{E}[w_j^2], \qquad (2.7)$$

where $\hat{E}[\cdot]$ denotes an empirical average for which we use MOM estimates

$$\hat{E}[w_j] = 1 - \frac{\prod_{i=1}^{n}\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right]}{\hat{M}_{X_j}(t)} \qquad (2.8)$$

$$\hat{E}[w_j^2] = 1 - \frac{2\prod_{i=1}^{n}\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right]}{\hat{M}_{X_j}(t)} + \frac{\prod_{i=1}^{n}\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{2b_{ji}}\right]}{\hat{M}^2_{X_j}(t)}. \qquad (2.9)$$

$\hat{M}_{X_j}(t)$ is an estimate of the moment generating function of link delay at link j, which can be obtained from

$$\hat{M}_{X_j}(t) = \prod_{i=1}^{n}\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}. \qquad (2.10)$$

In order to obtain the empirical average $\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right]$, we propose the following two methods:

1) **Sliding Window Method**: We define a sliding window with window size W and step size $S < W$. Define the number of window shifts as $N_w = \lfloor \frac{N_i - W}{S} \rfloor + 1$. Then

$$\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right] = \sum_{l=1}^{N_w} \frac{1}{N_w} \left(\frac{1}{W} \sum_{k=(l-1)S+1}^{(l-1)S+W} e^{t y_{ik}} \right)^{b_{ji}}. \tag{2.11}$$

2) **Monte Carlo Bootstrap Method**: The Monte Carlo bootstrap method [56] can be used to obtain statistical estimates from a set of i.i.d. observations, which is described as follows:

   (a) Fit the nonparametric maximum likelihood estimate (MLE) of the distribution of Y_i,

$$\hat{F}_i : \text{ mass } \frac{1}{N_i} \text{ at } Y_{ik}, \qquad k = 1, \ldots, N_i$$

   (b) Draw *bootstrap samples* from $\hat{F}_i$, $Y_{i1}^*, Y_{i2}^*, \cdots, Y_{iN_i}^* \overset{iid}{\sim} \hat{F}_i$ and compute

$$\hat{M}_{Y_i}^*(t) = \frac{1}{N_i} \sum_{k=1}^{N_i} e^{t Y_{ik}^*}$$

   (c) Independently repeat step (b) L times to obtain *bootstrap replicates* $\hat{M}_{Y_i}^*(t)_1, \hat{M}_{Y_i}^*(t)_2, \cdots, \hat{M}_{Y_i}^*(t)_L$, and calculate the empirical average

$$\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{b_{ji}}\right] = \frac{1}{L} \sum_{l=1}^{L} \left[\hat{M}_{Y_i}^*(t)_l\right]^{b_{ji}}. \tag{2.12}$$

The $\hat{E}\left[\left(\hat{M}_{Y_i}(t)\right)^{2b_{ji}}\right]$ can be obtained similarly using either method. The Monte-Carlo bootstrap method is generally more computationally expensive than the sliding window method. Therefore the latter is a preferred method when CPU resource is limited or real-time implementation is required.

2.4 Experimental Results

We use the `ns-2` [40] network simulator program to perform a TCP/UDP simulation of the network in Figure 2.1. Probes are sent through 5 different paths in order to estimate delay CGF for 4 links. The corresponding routing matrix $\mathbf{A}$ is

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

We set up a similar test environment to that reported in Lo Presti and Duffield [16]. All the links to be estimated have bandwidth 4Mb/sec with latency 50ms. Each link is modeled as a Drop-Tail queue (FIFO queue with finite buffer). The queue buffer sizes are 50 packets. We generate probes as 40 byte UDP packets. The probe transmissions are generated independently at each source node according to a Poisson process with mean interarrival time being 16ms and rate 20Kb/sec. The background traffic consists of both exponential on-off UDP traffic and FTP traffic.

N probes are collected for each path for a total of $5 \times N$ probes. We estimate each probe queueing delay by substracting the minimum probe delay over the N

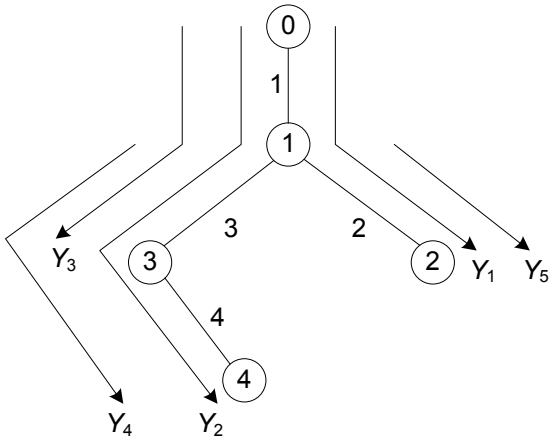


Figure 2.1: Network topology and probe routing paths for the `ns` experiment. Y_i denotes the end-to-end delay for path i, for $i = 1, \ldots, 5$.

Table 2.1: MSE of $\hat{K}_{X_j}$ (bias corrected) and $\hat{K}'_{X_j}$ (no bias correction).

Link	1	2	3	4
MSE of $\hat{K}_{X_j}$	0.0086	0.0247	0.0483	0.0096
MSE of $\hat{K}'_{X_j}$	0.0060	0.0326	0.0644	0.0325

trials. This provides a biased estimate of queueing delay across the probe path since the minimum probe delay is a biased estimator of transmission delay plus latency. However, the bias decreases as $1/N$. Here we use the moving window method to estimate the empirical averages in (2.8) and (2.9). We set the window size W to be $2/3$ of N, and the window shift step size S to be 10 probe delay samples.

We compare the proposed bias corrected estimator to the biased estimator (2.5) for $\mathbf{K}_X(t)$. We evaluate the CGF's over the range $t = -200$ to $t = 200$. Comparing the estimates of CGF of sampled link delays with and without bias correction in Table 2.1, we can see that the proposed estimator achieves lower MSE. The corresponding estimated CGF for link 2 and 4 are shown in Figure 2.2(a) and 2.2(b) respectively. These results are obtained by using $N = 1500$ probes per route.

Table 2.2: Chernoff bounds P_j and empirical estimates of $P(X_j \geq 0.02)$ for each link delay in the `ns` simulation.

Link	1	2	3	4
P_j	0.7517	0.4030	0.9620	0.9012
$\hat{P}(X_j \geq \delta)$	0.2504	0.1921	0.3447	0.2790

2.5 Applications and Extensions

Each link delay CGF preserves all the statistical information of the delay since it is the log of the Fourier transform of the link delay probability density function. We can accurately estimate many features of the delay distribution from the delay CGF. Here we give results for *bottleneck link detection*. We define a bottleneck as the event that the probability of a link delay exceeding some delay threshold $\delta > 0$ exceeds a prespecified threshold P. Using the Chernoff bound,

$$P(X_j \geq \delta) \leq e^{-t\delta} E\left[e^{tX_j}\right] = P_j \tag{2.13}$$

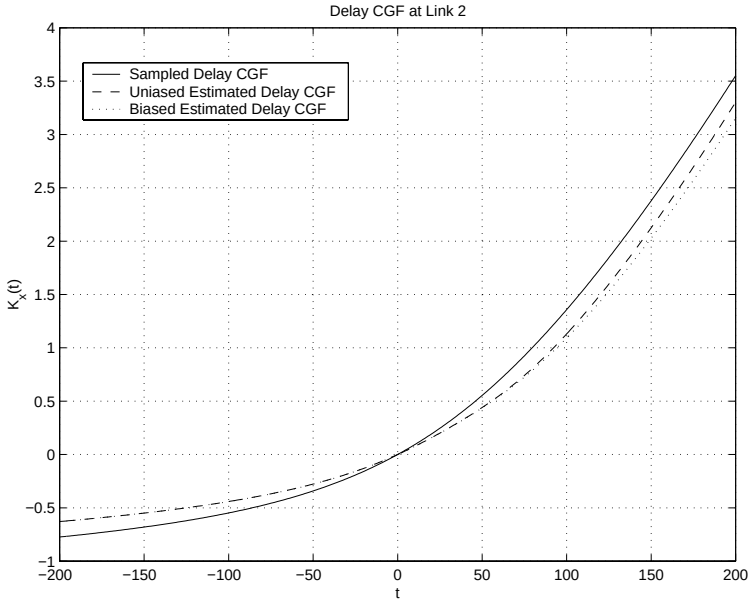
for $t > 0$. By appropriately selecting the threshold δ and a threshold P close to 1, we can detect a bottleneck link by testing whether $\max_{j=1,\ldots,m} P_j > P$. In Table 2.2, we show the Chernoff bounds for $P(X_j \geq \delta = 0.02s)$ which were estimated from the computer simulation in the previous section. By setting threshold P to be 0.95, we can identify link 3 as the bottleneck link.

2.6 Conclusion and Future Work

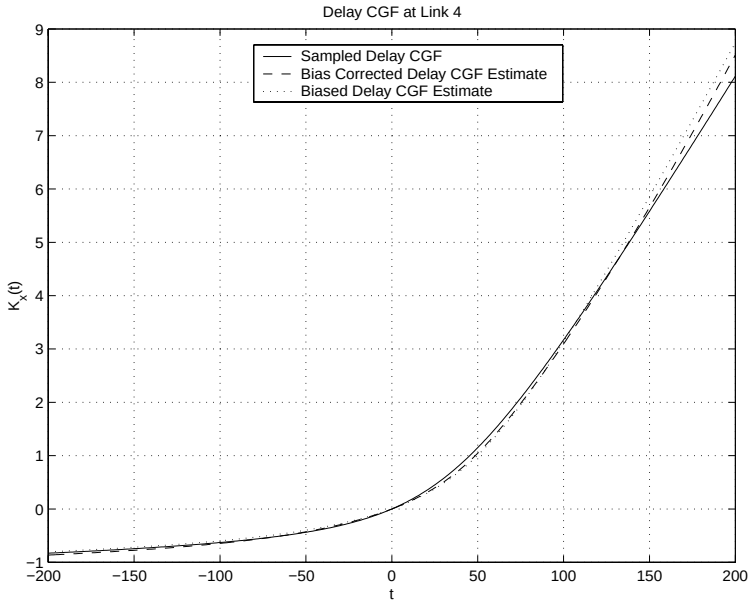
In this chapter we proposed a unicast method to perform inference on internal link delay characteristics. We derived a bias-corrected estimator for internal link delay cumulant generating functions based on LS approximation. The proposed estimator

was evaluated by `ns` simulations with TCP/UDP background traffic and FIFO finite buffer link queues. The MSE of the proposed estimator is lower than that of the direct biased sample mean estimator.

In the future, the following issues can be investigated into. First, our proposed estimator assumes stationarity of the network over the probing period (Assumption A2), which may be violated in real applications. Some adaptive estimation must be done in order to track the true link delay distributions. Furthermore, if the internal link delays are spatially dependent, a more sophisticated model needs to be used. Finally the estimator requires the probing matrix $\mathbf{A}$ to be known. The topology discovery methods introduced in Chapter 5 offer one way to estimate $\mathbf{A}$.



(a)



(b)

Figure 2.2: Biased and bias-corrected estimates of link delay CGF, compared with empirical delay CGF for (a) link 2 and (b) link 4.

CHAPTER 3

Unicast-Based Inference of Network Link Delay Distributions with Finite Mixture Models

3.1 Introduction

The estimation of internal link delay distributions from end-to-end measurements is explored from a different perspective in this chapter. Unlike the previous chapter, here we deal with a more practical situation in which there are fewer probe paths than internal links. We model the network as a logical tree and assume its topology is fixed and known. We send packet pairs from the root node to pairs of leaf nodes and collect their end-to-end delays. To circumvent the problems of using discrete delay models as explained in Chapter 1, we propose a hybrid finite mixture model for link delays which consists of a point mass located at the (fixed) minimum latency and a continuous mixture density. It more precisely captures the nature of queueing delays. We also show that identifiability holds when there is a positive probability for every link that a packet experiences the minimum delay. An ML-EM algorithm is developed to iteratively find the maximum likelihood estimates of the delay distributions when the model orders are known. In a real network, however, the delay model orders are

determined by various random factors such as background traffic rates and packet size distributions, hence they are unlikely to be unknown in advance. This problem is solved by applying model order penalties derived from the minimum message length (MML) criterion and use of a PML-EM algorithm to estimate the parameters. The ML-EM algorithm is evaluated by a model simulation using `MATLAB` to generate link delays which truly obey some hybrid mixture models, and the model orders are provided to the estimator. The PML-EM algorithm is applied to the data generated by `ns-2`, and the result is compared to the empirically estimated distributions.

Our contribution in this chapter is summarized as follows: (1) We propose a hybrid finite mixture model for link delay distrbutions with non-zero continuous mixture density and a point mass modelling empty link queue events. The proposed model avoids the drawbacks of discrete delay models used by previous work in network delay tomography; (2) For the case when the mixture model order is given for each link in the network, we develop an ML-EM algorithm to estimate the parameters in link delay distributions based on end-to-end delay measurements; (3) Without any prior information on mixture model orders, we develop an unsupervised PML-EM algorithm using MML-type model order penalty to estimate link delay distributions based on end-to-end delay measurements; (4) Throughout model and `ns` simulations we demonstrate that our algorithms can produce accurate estimates for link delay distributions and link delay ranges without forcing quantization on delay values as in previous work.

The chapter is organized as follows. Section 3.2 provides the network model and main assumptions. Section 3.3 discusses the discrete and continuous delay models, and points out the problems to use them in delay tomography. In Section 3.4 we propose a novel hybrid finite mixture delay model. The ML-EM algorithm for known model orders is illustrated. Then the MML penalized likelihood is derived for model

selection when the orders are unknown. We also discuss some details of the algorithm such as initialization and component-wise implementation. Computer simulations are illustrated in Section 3.5. Section 3.6 concludes the chapter and discusses some future directions of research.

3.2 Network Model and Main Assumptions

As in Coates and Nowak [17] we adopt the back-to-back packet pair probing framework and represent the network topology as a directed logical tree $T = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is the set of nodes, e.g., routers and terminals, and $\mathbf{E}$ is the set of links. The logical tree representation has a single root node 0, serving as a source, a set $\mathbf{V}_i$ of internal nodes having a degree at least 2, and a set $\mathbf{V}_r$ of leaf nodes, containing the receivers. Let there be a total of L nodes in $\mathbf{V}_i \cup \mathbf{V}_r$, numbered by $1, \ldots, L$, and label each link with its child end node number. If there are a total of R leaf nodes, then there are also R possible paths from the root to the receivers. To collect end-to-end information at the edge nodes, we send packet pairs from the source to every pair of the receivers. There are $S = \binom{R}{2}$ possible pairs of the receivers. For each of them the probe path forms a binary subtree, called probe trees. The identification of every link's delay distribution requires every pair of links to be seperated on different branches of some probe tree. This requires at least $\lceil \frac{deg(v)-1}{2} \rceil$ distinct probe trees branching at each internal node v and covering all its child nodes, where $deg(v)$ denotes the degree of node v and is defined by the number of links connected to it. Therefore the minimum number of probe trees required is $N_T^{min} = \sum_{v \in \mathbf{V}_i} \lceil \frac{deg(v)-1}{2} \rceil$. Throughout this chapter we assume a total of $N_T \geq N_T^{min}$ probe trees which satisfy the identifiability condition are used and we numbered them as $1, \ldots, N_T$. A network with ten links is depicted in Figure 3.1 showing a single root node, four internal nodes,

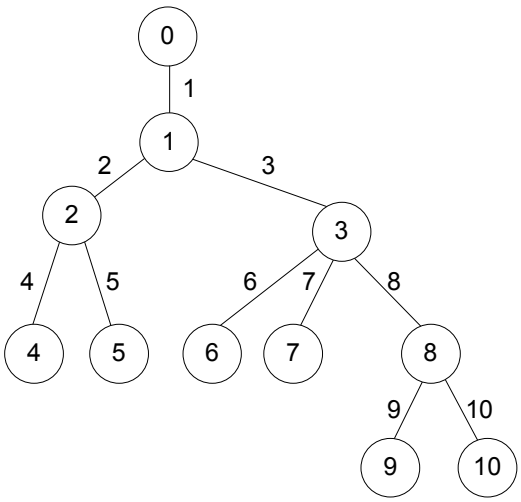


Figure 3.1: An example of logical tree network. Receivers are the leaf nodes while the source is the root node.

and six leaf nodes.

In a unicast probing session a pair of leaf nodes is (randomly) selected by the source and two time stamped packets, called a (unicast) probe pair, are sent to them respectively. The two packets are transmitted in rapid succession and assumed to encounter identical delays on the shared links of their paths. Each leaf node records the time that a packet is received. Subtracting this value from the packet's time stamp gives the end-to-end delay of the packet. End-to-end delays of the probe pairs on the same probe tree are random vectors due to the random ambient cross traffic through links along their paths. If any packet in a probe pair is dropped by the network, both packets are considered lost. Unicast probing is repeated until the session is over or enough packets are received by each leaf node to perform the next step: network delay tomography. The aim of network delay tomography is to identify parameters of the packet delay distribution for each individual internal link from the end-to-end delays observed by the receivers. Network tomography is possible since the end-to-end delay is a sum of the internal link delays encountered along the probe

path and any two paths in a probe tree must cross at common links.

Let X_l be the packet delay encountered by a probe at link l, $l = 1, \ldots, L$, and let Y_i be the end-to-end packet delay along the ith path, $i = 1, \ldots, R$. As in Chapter 2 we make the following independence and stationarity assumptions:

(A1) **Spatial Independence**: Packet delays at different links are statistically independent, i.e., X_i and X_j are independent for $i \neq j$.

(A2) **Temporal Independence and Stationarity**: For a given link, the delays encountered by packets in different probe pairs at that link are statistically independent and identically distributed.

For each probe pair we make an additional assumption:

(A3) **Consistency**: The delays encountered by both packets in a probe pair on the shared links of their paths are the same (with probability 1).

It is important to point out that while (A1) and (A2) are normally not satisfied in practice (see, e.g., [57]), these are commonly assumed in order to permit tractable analysis. An example where spatial independence (A1) is violated is when there is interaction among different data flows along the same path. As for (A2), temporal independence fails when Internet traffic is bursty or the network has a long latency time which correlates different packet pairs. Stationarity fails when the unicast probing session has a longer duration than the stationarity time of the network. However, experiments have shown that the performance of network tomography is remarkably insensitive to violations of (A1) and (A2) [5, 15, 16, 17, 20]. In (A3) the assumption of identical delays experienced by a probe pair on shared links does not hold when a small discrepancy between the two is observed from real network data (see, e.g., [23, 59]). Fortunately this random error has mean close to 0 and can be reduced by random ordering of the two packets [17].

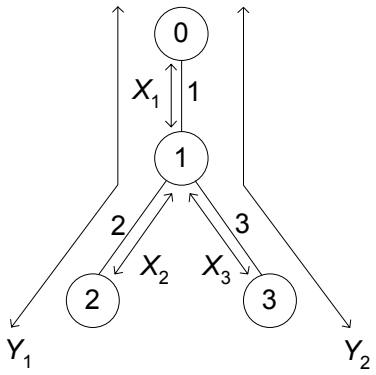


Figure 3.2: A two-leaf tree network.

Another issue in delay tomography is the synchronization of system clocks among the edge nodes. One common solution suggested to achieve synchronization is to deploy Global Positioning System (GPS) systems at the root and leaf nodes of the network (see, e.g., [16]). Pásztor and Veitch proposed a method using software clock to achieve precision timing for active network measurements without additional hardware deployment [58]. However, if the synchronization cannot be afforded but the clock shift remains fixed during the probing session between the root and each of the leaf nodes respectively, one can still perform accurate delay tomography as follows. For each end-to-end path from the root to one of the leaf nodes, we assume that at least one probe packet following that path encounters no link queueing delays at all. Then the end-to-end delay of any of those specific probe packets is the minimum over all the probes along that path, and this delay equals the sum of the non-random transmission and processing link delays over the path plus the clock shift between the root node and the leaf node. Since the end-to-end delay of each probe packet following that path also includes an offset introduced by the clock shift, substraction of the minimum end-to-end delay cancels the clock shift and results in an estimate of pure queueing delay over the path. In this case we can assume the minimum link delays are zero for empty link queues.

3.3 Unicast Network Delay Tomography

3.3.1 Discrete Delay Model

In the widely adopted discrete link delay model [16, 17, 20] a universal bin size q is used to discretize link delays X_l at each link $l = 1, \ldots, L$. The time intervals $(iq, (i+1)q]$, $i = 0, \ldots, D$, are called the delay bins. Here D is a positive integer and $D = \infty$ can be used to account for lost probe packets or large delays which are out of range. Discretization produces the discretized delay value i when X_l falls in the ith bin. A probability mass function (pmf), or histogram, $P_l = \{p_{l,d} : d = 0, \ldots, D\}$, is then associated with the discretized delays over link l, where the probability $p_{l,d} = P\left(X_l \in (dq, (d+1)q]\right)$ is an unknown to be estimated and $\sum_{d=0}^{D} p_{l,d} = 1$. For a probe path containing j links, the discretized end-to-end packet delay varies over the range $0, \ldots, jD \cdot q$.

Consider the two-leaf tree network shown in Figure 3.2, and the associated delay pmf's $P_l = \{p_{l,d} : d = 0, \ldots, D\}$ for $l = 1, 2, 3$. Probe pairs are sent from the source to receiver nodes 2 and 3. With assumption (A3) the identifiability of P_l's from end-to-end delays can be studied in a similar manner to multicast networks. More specifically, in multicast each packet is replicated by the network at the branching points of its paths and all the packets at the receivers again have common delays on shared links. Proof of identifiability in discrete network delay tomography with multicast probes is provided in [16] and the use of unicast probe pairs can be considered as a special case.

The discrete delay model adopted in [16, 17, 19, 20, 23] has two main drawbacks. First, the proper bin size needs to be carefully selected. Second, a universal bin size may not be suitable due to large variation of packet delay ranges over different links. Although it has been proposed to adopt different bin sizes for different links or delay

ranges [16, 19], those bin sizes still need to be chosen in advance.

3.3.2 Continuous Delay Model

One way to avoid the pitfalls of binning is to use a flexible continuous link delay model. For example, closed form expressions for the probability density function of queueing delay have been derived for simple queueing models such as M/M/1. These expressions could possibly be extended to a network of queues but it is well known that the M/M/1 model is an inadequate model for Internet traffic [60]. An alternative is to approximate each link delay density by a finite mixture which, with sufficiently large number of components, can describe a wide range of continuous density functions [61]. Let $f_l(x)$ be the link delay pdf at link l. A finite mixture model for this pdf is

$$f_l(x) = \sum_{m=1}^{k_l} \alpha_{l,m} \phi(x; \theta_{l,m}), \quad l = 1, \ldots, L \tag{3.1}$$

where k_l denotes the number of components, $\alpha_{l,m}$, $m = 1, \ldots, k_l$, denotes the mixing parameter for the mth component $\left(0 < \alpha_{l,m} < 1, \sum_{m=1}^{k_l} \alpha_{l,m} = 1\right)$, and $\phi(x; \theta_{l,m})$ is a density function over the x-domain parameterized by the parameter vector $\theta_{l,m}$. Many different choices for $\phi(x; \theta)$ are possible including: Gaussian; generalized Gaussian; exponential; or uniform. For the case of a Gaussian mixture $\theta_{l,m} = [\mu_{l,m}, \sigma^2_{l,m}]$ is the vector specifying the position (mean) and width (sqrt(variance)) of the mth mixture component at the lth link.

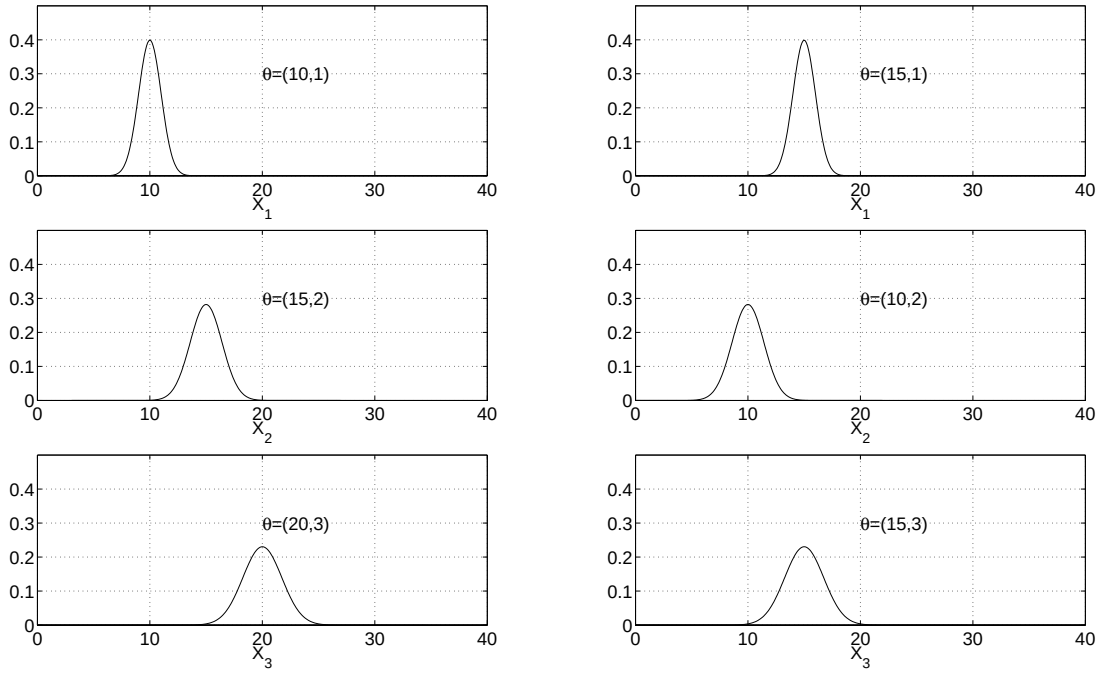
However, the use of pure continuous mixture density functions can cause serious identifiability problems. To illustrate consider again the simple two-leaf tree of Figure 3.2. Assume that all link delays are Gaussian, i.e., $k_1, k_2, k_3 = 1$ (single component mixtures), and $\phi(x; \theta) = \exp(-(x - \mu)^2/(2\sigma^2))/(\sqrt{2\pi}\sigma)$. The end-to-end delays

(Y_1, Y_2) has the following joint pdf:

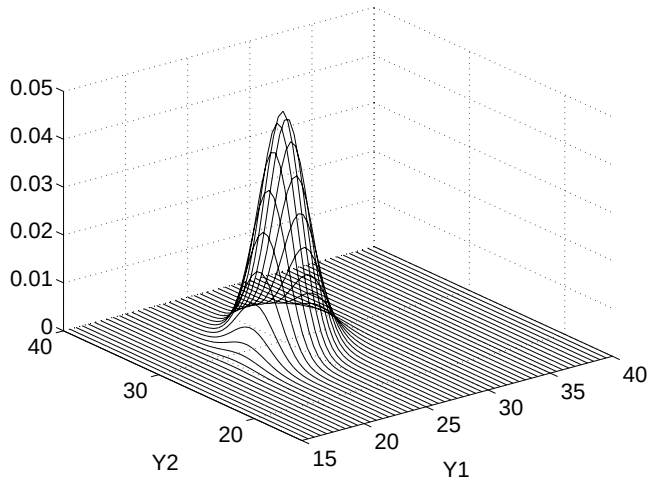
$$f(Y_1, Y_2) = \frac{1}{2\pi\sqrt{\sigma_1^2\sigma_2^2 + \sigma_1^2\sigma_3^2 + \sigma_2^2\sigma_3^2}} \cdot exp\left\{ -\frac{1}{2(\sigma_1^2\sigma_2^2 + \sigma_1^2\sigma_3^2 + \sigma_2^2\sigma_3^2)} \cdot \right.$$
$$\left. \left(\sigma_3^2(Y_1 - (\mu_1 + \mu_2))^2 + \sigma_2^2(Y_2 - (\mu_1 + \mu_3))^2 + \ \sigma_1^2((Y_1 - Y_2) - (\mu_2 - \mu_3))^2\right) \right\} \tag{3.2}$$

If we look at the mean parameters, they are completely described by the 3 parameters $\eta_1 = \mu_0 + \mu_1$, $\eta_2 = \mu_0 + \mu_2$, and $\eta_3 = \mu_1 - \mu_2$. This gives only two equations for the three unknowns parameters $[\mu_1, \mu_2, \mu_3]$ so the simple Gaussian model is not identifiable for any value of the mean parameters. An example is shown in Figure 3.3 where (a) and (b) are two different sets of internal link delay distributions for the network in Figure 3.2.

One can also consider the packet-stripe schemes suggested in [59], in which a *stripe* of several closely spaced unicast packets with distinct destinations are sent back-to-back from the root node. Similarly to packet pair probes, these packets are assumed to encounter virtually the same delays on shared links along their paths. As shown in [15], packet-stripe probing allows identification of higher order moments of internal link delays when the branching ratio is larger than two. However, under the Gaussian mixture link delay model, the link delay means still cannot be uniquely identified from end-to-end delays.



(a) (b)



(c)

Figure 3.3: Example of two sets (a) and (b) of Gaussian internal link delay densities along the two probe paths in the network in Figure 3.2. The two end-to-end delays of each received packet pair obeys a Gaussian bivariate density shown in (c). This bivariate density is parameterized by only two location parameters which is insufficient to recover the three location parameters in (a) and (b).

3.4 Hybrid Finite Mixture Approach

3.4.1 Hybrid Finite Mixture Model

In analysis of a queueing system, the utilization factor ρ is an important parameter for describing system behavior. The parameter ρ denotes the probability that the system is busy serving customers and, for a stable system, ρ must satisfy $0 \leq \rho < 1$ [62]. A lightly loaded link satisfies $\rho << 1$, i.e., there is a non-negligible probability that a packet encounters an empty queue, i.e., an idle router, and passes without delay. This suggests placing a point mass component with weight $1 - \rho$ in the link delay mixture model. If this point mass is included in addition to the continuous components the link delay pdf becomes a hybrid discrete/continuous finite mixture model. Hence, similar to (3.1), we obtain

$$f_l(x) = \alpha_{l,0}\delta(x - x_{l,0}) + \sum_{m=1}^{k_l} \alpha_{l,m}\phi(x; \theta_{l,m}) \quad l = 1, \ldots, L. \tag{3.3}$$

Here $\alpha_{l,0} = 1 - \rho_l$, $\delta(x)$ is a point mass (Dirac impulse) at zero and $x_{l,0}$ is the known pure (non-random) transmission delay experienced by the packet. All other parameters are defined as in (3.1), except now the α's must satisfy $\sum_{m=0}^{k_l} \alpha_{l,m} = 1$, $\alpha_{l,m} \geq 0$. The discrete mass component $\delta(x)$ not only makes the delay distribution more precisely model the behavior of a link queue, but as shown below also buys us identifiability of all the link delay distribution parameters.

For any probe pair the distributions of the end-to-end probe delay densities will be the convolution of the link distributions, which are also hybrid mixtures. Now, similarly to the previous section, let's assume that the continuous mixture component is a single Gaussian pdf. Let the point masses $\alpha_{l,0} = \alpha_l$ and assume that they are all concentrated at zero delay, i.e., $x_{l,0} = 0$. Figure 3.4 shows the end-to-end joint delay

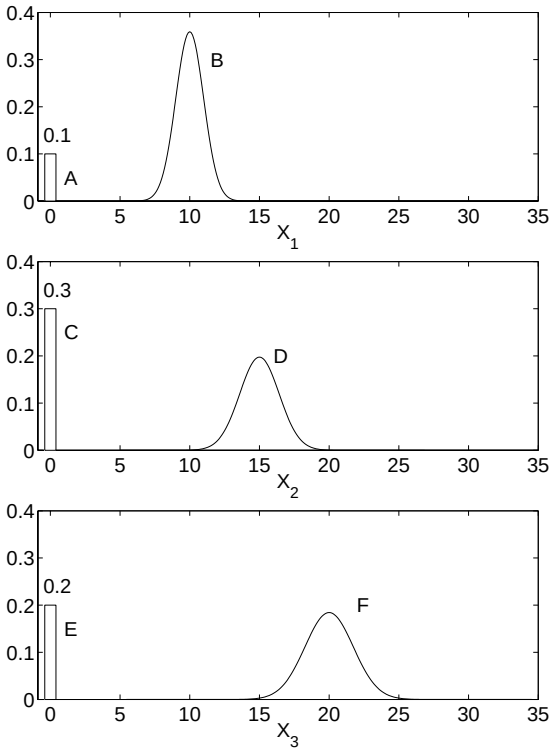
distribution in the two-leaf tree network of Figure 3.2, whose mathematical form is

$$\begin{aligned} f(Y_1, Y_2) &= \beta_1\beta_2\beta_3\tilde{f}(Y_1, Y_2) + \alpha_1\alpha_2\alpha_3\delta(Y_1)\delta(Y_2) + \alpha_1\alpha_2\beta_3\delta(Y_1)\phi(Y_2; \mu_3, \sigma_3^2) + \\ & \alpha_1\beta_2\alpha_3\delta(Y_2)\phi(Y_1; \mu_2, \sigma_2^2) + \beta_1\alpha_2\alpha_3\delta(Y_1 - Y_2)\phi(Y_1; \mu_1, \sigma_1^2) + \\ & \alpha_1\beta_2\beta_3\phi(Y_1; \mu_2, \sigma_2^2)\phi(Y_2; \mu_3, \sigma_3^2) + \\ & \beta_1\beta_2\alpha_3\phi(Y_2; \mu_1, \sigma_1^2)\phi(Y_1 - Y_2; \mu_2, \sigma_2^2) + \\ & \beta_1\alpha_2\beta_3\phi(Y_1; \mu_1, \sigma_1^2)\phi(Y_2 - Y_1; \mu_3, \sigma_3^2), \end{aligned} \tag{3.4}$$

where $\beta_l = 1 - \alpha_l$ for $l = 1, 2, 3$ and $\tilde{f}(Y_1, Y_2)$ is the joint distribution shown in (3.2). Due to the point mass in (3.3), (3.4) has additional isolated Gaussian components which appear with discrete masses at locations in the (Y_1, Y_2) plane specified by : $\{Y_1 = 0\}, \{Y_2 = 0\}$, and $\{Y_1 = Y_2\}$. It is obvious that identifiability can be achieved as long as $\alpha_l \neq 0$. It might seem strange to the readers that the addition of a point mass allows one to uniquely identify the set of parameters of the internal link components from a single probe tree. However, one still needs multiple probe trees to assign these parameters to specific links.

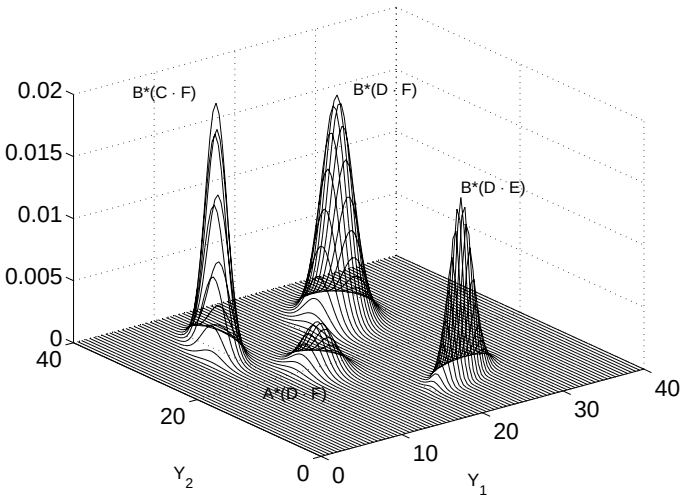
3.4.2 ML-EM Algorithm

Here we present an ML-EM algorithm for approximating the maximum likelihood estimates of the internal link mixture model parameters from end-to-end packet pair measurements. Let U be a finite mixture random variable with k components and pdf of the form $f(U) = \sum_{m=1}^{k} \alpha_m \phi_m(U)$ where $\sum_{m=1}^{k} \alpha_m = 1, \ \alpha_m \geq 0$. An example of a Gaussian mixture with three components is given in Figure 3.5. The solid line depicts the density function and the dashed line shows each component. There are two different interpretations of finite mixture models which will be useful in the

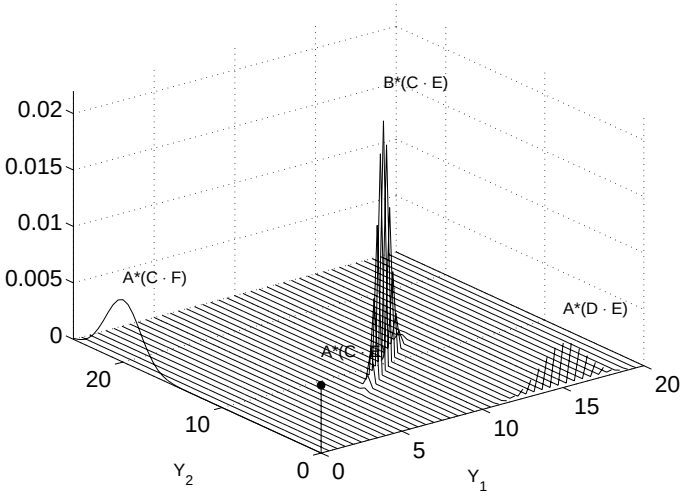


(a)

Figure 3.4: Example of internal link delay hybrid mixture densities (a) for links 1,2,3, over the two-leaf tree of Figure 3.2. The non-random minimum delays for all the links are set to 0. The end-to-end packet pair delay distribution is also a hybrid mixture whose purely continuous components are shown in (b), and components associated with discrete masses are in (c). All link parameters can be identified from this two-dimensional distribution. Here $B * (C \cdot F)$ denotes a function of (y_1, y_2) which is the convolution of the internal link components labelled B, C, and F, in the form of $\int B(x)C(y_1 - x)F(y_2 - x)dx$.



(b)



(c)

Figure 3.4 (continued)

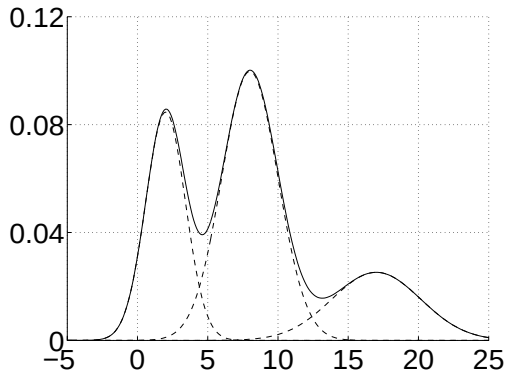


Figure 3.5: Gaussian mixture example. $f_U(u) = 0.3N(u;2,2) + 0.5N(u;8,4) + 0.2N(u;17,10)$.

sequel. The first one is simply that $f(U)$ is a multi-component pdf for U. The second interpretation is that U is selected at random from a pool of simpler hidden random variables $U_1, \ldots, U_k$ with selection probabilities $\alpha_1, \ldots, \alpha_k$, respectively. Define the binary random selection vector $\mathbf{Z} = [Z_1, \ldots, Z_k]^T$ where $Z_m = 1$ if and only if the mth variable U_m is selected and assign to this event probability α_m. U can be expressed as $U = \sum_{m=1}^{k} Z_m U_m$. Thus, if U_m has pdf $\phi_m(U_m)$ then this is identically the conditional pdf $f(U|Z_m = 1)$. Then $f(U) = \sum_{m=1}^{k} \alpha_m \phi_m(U)$, which is the mixture model for U that we started out with. The second interpretation is critical for development of the ML-EM algorithm which we address below.

Assume that we have prior knowledge of all the link mixture orders $\{k_l\}_{l=1}^{L}$. We will relax this assumption in the next section. Let N_i be the number of packet pairs sent from the source to the receivers of probe tree i and let $\mathbf{M}_i$ be the set of links along that tree. Define $X_l^{(i,n)}$ the delay at link l encountered by the nth packet pair sent through probe i. Let $\mathbf{Z}_l^{(i,n)} = \left[Z_{l,0}^{(i,n)}, \ldots, Z_{l,k_l}^{(i,n)}\right]$ be the selection vector for $X_l^{(i,n)}$.

With these definitions, maximum likelihood (ML) estimation of the set of internal link mixture densities can be formulated as a missing data problem. The Expectation Maximization (EM) algorithm [63] has been extensively applied to approximate ML

and penalized ML (PML) estimates for mixture models [45, 46, 48]. Let $\mathbf{X} = \{X_l^{(i,n)}\}$ and $\mathbf{Z} = \{\mathbf{Z}_l^{(i,n)}\}$ for all l, i, n. $\{\mathbf{X}, \mathbf{Z}\}$ is called *missing data* or *hidden data*. Define $\mathbf{Y}^{(i,n)} = (Y_1^{(i,n)}, Y_2^{(i,n)})$ as the pair of end-to-end delays of the nth packet pair received by two receivers in the ith probe tree. The observables $\mathbf{Y} = \{\mathbf{Y}^{(i,n)}\}_{i,n}$ are called the *incomplete data* and the set $\{\mathbf{X}, \mathbf{Z}, \mathbf{Y}\}$ is said to be the *complete data*. The EM algorithm generates a sequence of estimates of the unknown parameters $\mathbf{\Theta}$ which have the property that the likelihood sequence $\mathcal{L}(\mathbf{\Theta}) = f(\mathbf{Y}|\mathbf{\Theta})$ is nondecreasing.

It is easily shown that the likelihood of the *complete data* can be factorized as

$$\mathcal{L}_c(\mathbf{\Theta}) \stackrel{\text{def}}{=} f(\mathbf{X}, \mathbf{Y}, \mathbf{Z}|\mathbf{\Theta}) = f(\mathbf{Y}|\mathbf{X}) f(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta}), \tag{3.5}$$

and thus maximization of $\mathcal{L}_c(\mathbf{\Theta})$ is equivalent to maximization of the likelihood function $\mathcal{L}(\mathbf{\Theta}) \stackrel{\text{def}}{=} f(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta})$. For a specific link l, X_l is a mixed random variable with density function f_l given by (3.3) and therefore, up to a constant, the complete data log-likelihood function is:

$$\log \mathcal{L}(\mathbf{\Theta}) = \sum_{l=1}^{L} \sum_{i:l \in \mathbf{M}_i} \sum_{n=1}^{N_i} \left\{ Z_{l,0}^{(i,n)} \log \alpha_{l,0} + \sum_{m=1}^{k_l} Z_{l,m}^{(i,n)} \left(\log \alpha_{l,m} + \log \phi(X_l^{(i,n)}; \theta_{l,m}) \right) \right\}. \tag{3.6}$$

The EM algorithm updates parameter estimates by applying two steps at each iteration. At the tth iteration, the E-step computes conditional expectation of complete data log-likelihood given observations $\mathbf{Y}$ and current parameter estimates $\hat{\mathbf{\Theta}}^{(t)}$

$$Q(\mathbf{\Theta}, \hat{\mathbf{\Theta}}^{(t)}) = E\left[\log \mathcal{L}(\mathbf{\Theta}) | \mathbf{Y}; \hat{\mathbf{\Theta}}^{(t)} \right]. \tag{3.7}$$

The M-step maximizes the Q function computed in the E step with respect to $\mathbf{\Theta}$ to

produce

$$\hat{\mathbf{\Theta}}^{(t+1)} = \arg\max_{\mathbf{\Theta}} Q(\mathbf{\Theta}, \hat{\mathbf{\Theta}}^{(t)}). \tag{3.8}$$

The E and M steps for the hybrid mixture model is similar to that for a pure Gaussian mixture model [45] and is illustrated in Appendix A.

3.4.3 PML-EM Algorithm with MML Penalty

When the number of link components k_l's is unknown the ML-EM algorithm is not guaranteed to converge. This is due to a fundamental ambiguity of unknown model order. To illustrate, consider the estimation of a k-component mixture having the form of (3.1) with parameters $\mathbf{\Theta} = \{\alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k\}$. These parameters have the same likelihood as the $k+1$ component mixture $\mathbf{\Theta}' = \{\alpha_1, \ldots, \alpha_{k-1}, (1-\beta)\alpha_k, \beta\alpha_k, \theta_1, \ldots, \theta_k, \theta_k\}$ for any $0 < \beta < 1$. One of the most effective ways to eliminate this ambiguity is to add a penalty to the log-likelihood function which penalizes the unecessary addition of more components to the mixture.

The Minimum Message Length Criterion

Many model order penalties have been proposed including: Akaike Information Criterion (AIC) [64], Minimum Description Length (MDL) [65] and Minimum Message Length (MML) [66]. Figueiredo and Jain [48] applied the MML penalty specifically to finite mixture models by introducing a prior to the parameters and an information theoretic penalty depending on quantization of parameter space. They developed an unsupervised method for simultaneously selecting model order and estimating parameters and the performance of their algorithm was impressive. As our approach in delay tomography focuses on the estimation of finite mixture models in an enlarged parameter space, a good choice is to generalize the algorithm of

Figueiredo and Jain to fit our case. The incomplete data penalized log-likelihood is expressed as

$$\tilde{\mathcal{L}}(\boldsymbol{\Theta}) \stackrel{\text{def}}{=} \log f(\boldsymbol{\Theta}) + \log f(\mathbf{Y}|\boldsymbol{\Theta}) - \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\Theta})| - \frac{c}{2}(1+\log\kappa_c), \tag{3.9}$$

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

To apply the MML algorithm [48] of Figueiredo and Jain to our network delay tomography problem their method has to be extended to another layer of hidden data. More specifically, while in [48] the realizations from the mixture model were observed directly, in our application only sums of these realizations (along probe paths) are observed. In other words, the end-to-end delays are themselves convolutive mixtures of the additive mixtures describing the link delays.

The standard incomplete data Fisher Information matrix $\mathbf{I}(\boldsymbol{\Theta})$ is not closed form, even for a directly observed finite mixture [47]. Therefore, similar to [48] we replace it by the complete data Fisher information matrix which in the network tomography setting is

$$\tilde{\mathbf{I}}(\boldsymbol{\Theta}) = -E\left[\nabla^2_{\boldsymbol{\Theta}} \log f(\mathbf{X},\mathbf{Z}|\boldsymbol{\Theta})\right] = \text{block-diag}\left\{n_l \mathbf{I}_1(\boldsymbol{\Theta}_l)\right\}_{l=1}^{L}, \tag{3.10}$$

where $\mathbf{I}_1(\boldsymbol{\Theta}_l)$ is the Fisher information matrix associated with the complete data at link l, $\boldsymbol{\Theta}_l$ denotes the parameter set of the lth link, and $n_l = \sum_{i:l\in\mathbf{M}_i} N_i$ is the total number of packet pairs passing through the lth link. $\mathbf{I}_1$ itself has block-diagonal structure

$$\mathbf{I}_1(\boldsymbol{\Theta}_l) = \text{block-diag}\left\{\mathbf{A}_l, \alpha_{l,1}\mathbf{I}_2(\theta_{l,1}), \cdots, \alpha_{l,k_l}\mathbf{I}_2(\theta_{l,k_l})\right\}, \tag{3.11}$$

where $\mathbf{I}_2(\theta_{l,m})$ is the Fisher information matrix associated with the hidden mth component delay variable $X_{l,m}$ on link l, and $\mathbf{A}_l = \text{diag}\left\{\alpha_{l,m}^{-1}\right\}_{m=0}^{k_l}$. If any one of the $\alpha_{l,m}$'s is zero, it is removed from $\mathbf{A}_l$ and k_l is decreased by 1.

The prior on the parameter set was taken as

$$f(\mathbf{\Theta}) = \prod_{l=1}^{L} \left\{ f(\alpha_{l,0}, \ldots, \alpha_{l,k_l}) \prod_{m=1}^{k_l} f(\theta_{l,m}) \right\}, \tag{3.12}$$

where $f(\alpha_{l,0}, \ldots, \alpha_{l,k_l})$ and $f(\theta_{l,m})$ are the non-informative Jeffreys' priors [67]:

$$f(\alpha_{l,0}, \ldots, \alpha_{l,k_l}) \propto \sqrt{|\mathbf{A}|} \quad = \quad (\alpha_{l,0}\alpha_{l,1} \ldots \alpha_{l,k_l})^{-1/2} \tag{3.13}$$

$$f(\theta_{l,m}) \quad \propto \quad \sqrt{|\mathbf{I}_2(\theta_{l,m})|}, \tag{3.14}$$

for $\sum_{m=0}^{k_l} \alpha_{l,m} = 1$ and $0 < \alpha_{l,m} < 1$. In addition, as in [48], we make the approximation $\kappa_c = 1/12$. This yields the MML penalized likelihood function

$$\tilde{\mathcal{L}}(\mathbf{\Theta}) = \log f(\mathbf{Y}|\mathbf{\Theta}) - \frac{d}{2} \sum_{l=1}^{L} \sum_{m=1}^{k_l} \log \alpha_{l,m} - \sum_{l=1}^{L} \frac{k_l(d+1)+1}{2} \left(\log \frac{n_l}{12} + 1 \right), \tag{3.15}$$

where d is the dimension of $\theta_{l,m}$, e.g., $d = 2$ for a Gaussian component mixture.

The Complete Algorithm

To derive the E-step of the PML-EM algorithm applied to maximizing (3.15) we adopt the same complete data as in the previous section. With this it is easy to see that the E step is a modification of (3.7) where $Q(\mathbf{\Theta}, \hat{\mathbf{\Theta}}^{(t)})$ now has an additional penalty given by the second and third additive terms on the RHS of (3.15). The modified M-step gives the updates for the mixing parameters in (3.16) (see Appendix

A).

$$\hat{\alpha}_{l,m}^{(t+1)} = \frac{\max\left\{\left(\sum_{i:l\in\mathbf{M}_i}\sum_{n=1}^{N_i}\omega_{l,m}^{(i,n)}\right) - \frac{d}{2}, 0\right\}}{\sum_{j=1}^{k_l}\max\left\{\left(\sum_{i:l\in\mathbf{M}_i}\sum_{n=1}^{N_i}\omega_{l,j}^{(i,n)}\right) - \frac{d}{2}, 0\right\} + \sum_{i:l\in\mathbf{M}_i}\sum_{n=1}^{N_i}\omega_{l,0}^{(i,n)}} \tag{3.16}$$

for $m = 1, \ldots, k_l$, where

$$\omega_{l,m}^{(i,n)} = E\left[Z_{l,m}^{(i,n)} | \mathbf{Y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}\right], \quad m = 0, \ldots, k_l. \tag{3.17}$$

The M-step for the remaining parameters depends on the specific form of the mixture density components.

Component Annihilation The algorithm uses the following strategy to select the number k_l of components at the lth link. It starts by setting each of $k_l, l = 1, \ldots, L$ to some user-specified upper bound and annihilates components as follows. If $\alpha_{l,m}^{(t+1)} = 0$, component m is removed from f_l and its probability mass is redistributed over the other non-zero-probability components at the next iteration. After convergence of the algorithm, however, the estimate is not guaranteed to be the MML estimate since additional increase of $\tilde{\mathcal{L}}(\mathbf{\Theta})$ may be achieved by further decreases of the k_l's. Consequently, it is necessary to manually remove the surviving components and recompute $\tilde{\mathcal{L}}(\mathbf{\Theta})$. The effort would be exorbitant if we go through all possible combinations of the k_l's. Therefore we propose a myopic approach as follows. We simply annihilate the component with the least mixing parameter $\hat{\alpha}_{l,m}$ for $m = 1, \ldots, k_l,\ l = 1, \ldots, L$ and rerun the PML-EM algorithm until convergence. This procedure is repeated until all the k_l's reach some pre-defined lower bound k_{min}.

Estimation of $\{\alpha_{l,0}\}$ Although one can treat the $\{\alpha_{l,0}\}$ as parameters to be estimated by the PML-EM algorithm, we find it more reliable to estimate them using sample averages. Consider again the two-leaf tree network in Figure 3.2. The probabilities of the following events have the expression

$$\begin{aligned} P_1 &= P\left((Y_1 = x_{1,0} + x_{2,0}) \vee (Y_2 = x_{1,0} + x_{3,0})\right) &= \alpha_{1,0}\left(\alpha_{2,0} + \alpha_{3,0} - \alpha_{2,0}\alpha_{3,0}\right) \\ P_2 &= P\left(Y_1 = x_{1,0} + x_{2,0}\right) &= \alpha_{1,0}\alpha_{2,0} \\ P_3 &= P\left(Y_2 = x_{1,0} + x_{3,0}\right) &= \alpha_{1,0}\alpha_{3,0} \end{aligned} \tag{3.18}$$

Let $\hat{p}_1, \hat{p}_2, \hat{p}_3$ be the sample average estimates for P_1, P_2, P_3, respectively, and let $\gamma = \hat{p}_2 + \hat{p}_3 - \hat{p}_1$. When $\hat{p}_2 \neq 0$ and $\hat{p}_3 \neq 0$ (hence $\hat{p}_1 \neq 0$) we obtain the estimates of $\{\alpha_{l,0}\}$ by

$$\begin{cases} \hat{\alpha}_{1,0} &= \hat{p}_2\hat{p}_3/\gamma \\ \hat{\alpha}_{2,0} &= \gamma/\hat{p}_3 \\ \hat{\alpha}_{3,0} &= \gamma/\hat{p}_2 \end{cases} . \tag{3.19}$$

For a larger tree network with more nodes and links, similar estimates can be calculated for each binary probe tree where $\hat{\alpha}_{i,0},\ i = 1, 2, 3$ in (3.19) may be products of link $\alpha_{l,0}$'s when the branches consist of multiple links. The $\alpha_{l,0}$ for each link are estimated either directly by (3.19) or by the quotient between the minimum-delay probabilities for the two probe tree branches which differ by that link only. Subsequently, we can remove $\{\alpha_{l,0}\}_{l=1}^{L}$ from $\boldsymbol{\Theta}$ and replace them with sample average estimates $\{\hat{\alpha}_{l,0}\}$ in the PML-EM algorithm. The reduced parameter set is denoted by $\boldsymbol{\Theta}_p$.

To explain why (3.19) is more reliable than the PML-EM estimates it is helpful to investigate the underlying delay model. The sample average estimate in (3.19) assumes a simple binary model for link delays in which there is only one parameter

$\alpha_{l,0}$, having pmf

$$p_l(x) = \begin{cases} \alpha_{l,0} & x = x_{l,0} \\ 1 - \alpha_{l,0} & \text{otherwise} \end{cases}. \tag{3.20}$$

On the other hand, the original PML-EM algorithm uses a much more complicated model in which most of the parameters in $\boldsymbol{\Theta}$ are associated with the continuous mixture model. Therefore the EM algorithm is dominated by the mixture model parameters, and consequently sacrifices the accuracy of $\{\hat{\alpha}_{l,0}\}$. The improvement made by the sample average estimates is remarkable especially when the amount of data is sufficient for the binary delay model but not enough for the hybrid mixture model. The M-step of the mixing parameters in (3.16) should be modified correspondingly as

$$\hat{\alpha}_{l,m}^{(t+1)} = (1 - \hat{\alpha}_{l,0}) \frac{\max\left\{\left(\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,m}^{(i,n)}\right) - \frac{d}{2}, 0\right\}}{\sum_{j=1}^{k_l} \max\left\{\left(\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,j}^{(i,n)}\right) - \frac{d}{2}, 0\right\}}, \quad m = 1, \ldots, k_l. \tag{3.21}$$

Initialization An initialization procedure is suggested that uses a PML-EM algorithm to first estimate a mixture model for the end-to-end delays independently of each root-to-leaf path and independently of topology. Then a minimum distance criterion is used to select common components for each shared link. These common components are used to initialize the proposed PML-EM algorithm for delay tomography that accounts for topology. For example, the common components selected for link 3 in Figure 3.1 to initialize the proposed PML-EM algorithm are obtained from the most significant components (according to a minimum distance criterion) in the estimated mixture models for end-to-end delays over the paths from the root to leaf nodes 6,7,9, and 10, respectively, which share link 3 as a common link.

The end-to-end delay between the root node and one of the leaf nodes is the sum

of the independent link delays over the path. The convolution of the link delay pdf's gives the end-to-end delay distribution which is also a hybrid finite mixture model. Its continuous mixture part is contributed by the delays that are greater than the minimum path latency. PML estimates of the continuous part can be obtained by EM algorithms such as that of Figueiredo and Jain [48]. It is generally impossible to resolve the link delay distributions directly from the path delay distributions without using an iterative algorithm. However, a simple direct estimate of the link delay distribution can provide useful information to initialize the PML-EM algorithm. The reason to initialize our PML-EM algorithm using PML estimates from another EM algorithm is the following. Firstly we find the computational complexity of the PML-EM algorithm increases exponentially with respect to the model order k_l of each link delay distribution. This is because the number of components in an end-to-end delay distribution equals the product of model orders for all the links on the path. Remember our algorithm starts with an upper bound of k_l for each link. When the bound is loose the PML-EM algorithm converges very slowly because of the high computational complexity for large k_l's. This situation can be mitigated if a tight upper-bound can be estimated. Secondly the algorithms used to estimate one-dimensional finite mixture models directly from the realizations are generally much faster than the PML-EM algorithm for delay tomography. Therefore it is worth to obtain a tight bound for each k_l from the estimates of end-to-end delay distributions at the expense of running another EM algorithm for every end-to-end path. A strategy to accelerate the estimation of internal link delay distributions with the proposed hybrid finite mixture model is illustrated in the next chapter.

Let $\mathbf{R}_l$ be the set of descendant leaf nodes of the internal node l. Recall that node l is connected to its parent by link l according to our definition. Then the set of routes from the root to the nodes in $\mathbf{R}_l$ includes exactly every end-to-end path

intersecting link l. Let the continuous mixture model in the end-to-end path delay distribution estimated from the measurements at leaf node r be

$$\chi_r(x) = \sum_{m=1}^{\kappa_r} \rho_{r,m}\phi(x;\theta_{r,m}) \quad r \in \mathbf{R}_l, \tag{3.22}$$

where $0 < \rho_{l,m} < 1$ and $\sum_{m=1}^{\kappa_r} \rho_{r,m} = 1$. We propose an initialization scheme for the parameters in f_l which selects a subset of $\mathbf{\Theta}(\mathbf{R}_l) = \{(\theta_{r,m}, \rho_{r,m})\}_{r,m}$ as its initial component parameters. The idea is motivated by the following obeservation. In a lightly-loaded network the probability mass for zero queueing delays is the dominant contribution for each link. Thus the copy of each link delay pdf, obtained from the convolution with all other link's point masses, constitutes a dominant part in $\chi_r(x)$. Each copy is shifted by an offset equal to the sum of all other link's minimum delays, but the distance between a mixture component and the point mass is preserved in $\chi_r(x)$. Since the delay distribution for link l appears in every $\chi_r(x)$ for $r \in \mathbf{R}_l$, a simple strategy is to choose the most similar sets in $\mathbf{\Theta}(\mathbf{R}_l)$ as initialization parameters. Although others may suggest more sophisticated methods for initialization, we find this simple method works well in our simulation.

We set the initial model order of f_l to be $k_l^{init} = \max_{r\in\mathbf{R}_l} \kappa_r$. Define x_r as the non-random minimum path delay from the root node to r. Let $\tau_{r,m}$ be the mean of $\phi(x;\theta_{r,m})$ minus x_r, which is the preserved relative distance between $\phi(x;\theta_{r,m})$ and the point mass. The initial parameters of the k_l^{init} components are selected from $\mathbf{\Theta}(\mathbf{R}_l)$ according to the distance

$$d_{r,m} = \min_{\substack{n=1,\ldots,\kappa_s \\ s\in\mathbf{R}_l\setminus\{r\}}} |\tau_{r,m} - \tau_{s,n}| \quad \forall m = 1,\ldots,\kappa_r, \quad r \in \mathbf{R}_l. \tag{3.23}$$

$d_{r,m}$ denotes the minimum distance of $\phi(x;\theta_{r,m})$ to a component estimated at the

other leaf nodes in $\mathbf{R}_l$. The parameters $(\theta_{r,m}, \rho_{r,m})$ in $\mathbf{\Theta}(\mathbf{R}_l)$ corresponding to the smallest k_l^{init} distances are selected for the initialization, and the $\rho_{r,m}$'s are normalized to have a sum equal to $(1-\hat{\alpha}_{l,0})$. Let's denote the initial distribution by $f_l^{init}(x)$. For the links directly connected to the leaf nodes, $f_l^{init}(x) = \hat{\alpha}_{l,0}\delta(x-x_{l,0})+(1-\hat{\alpha}_{l,0})\chi_l(x)$.

Component-wise Implementation We use the *component-wise EM algorithm for mixtures* (CEM)^2 [46] to accelerate the PML-EM algorithm. Similar to the SAGE algorithm of Fessler and Hero [68], the CEM^2 algorithm updates the parameters sequentially for each mixture component. It is especially helpful for the component annihilation algorithm because when a component is annihilated the CEM^2 immediately distributes its probability mass to the remaining components before their parameters are updated, hence increases their chance of survival [48]. The monotonicity property of the CEM^2 is not affected by the order of updating. We adopted a cyclic updating procedure as follows: update $\alpha_{1,1}$ and $\theta_{1,1}$, recompute Q, update $\alpha_{1,2}$ and $\theta_{1,2}$, recompute Q, and so on, until all the parameters for link 1 are updated; then proceed in the same way for link 2, 3, and so on, until all the link parameters are updated.

In CEM^2 there are additional Lagrangian penalty terms for the constraints: $\sum_{m=1}^{k_l} \alpha_{l,m} = 1 - \hat{\alpha}_{l,0}, \;\; l = 1, \ldots, L$ [46]. The penalized likelihood function becomes

$$\mathcal{L}_{CEM}(\mathbf{\Theta}_p) = \tilde{\mathcal{L}}(\mathbf{\Theta}_p) - \sum_{l=1}^{L} \lambda_l \left(\sum_{m=1}^{k_l} \alpha_{l,m} - (1 - \hat{\alpha}_{l,0}) \right) \tag{3.24}$$

with Lagrange multipliers $\lambda_1, \ldots, \lambda_L$, which can be computed as follows. Lets redefine

$$\omega_{l,m}^{(i,n)} = E\left[Z_{l,m}^{(i,n)} | \mathbf{Y}^{(i,n)}; \hat{\mathbf{\Theta}}_p^{(t)} \right], \quad m = 0, \ldots, k_l, \tag{3.25}$$

$$Q_{l,m}^{(i,n)}(\theta_{l,m}) = E\left[Z_{l,m}^{(i,n)} \log \phi(X_l^{(i,n)}; \theta_{l,m}) | \mathbf{Y}^{(i,n)}; \hat{\mathbf{\Theta}}_p^{(t)} \right], \quad m = 1, \ldots, k_l, \tag{3.26}$$

for $l = 1, \ldots, L$, where $\hat{\boldsymbol{\Theta}}_p^{(t)}$ denotes the estimate of $\boldsymbol{\Theta}_p$ in the tth iteration. The conditional expectation of $\mathcal{L}_{CEM}(\boldsymbol{\Theta}_p)$ in the E-step is

$$\begin{aligned} Q(\boldsymbol{\Theta}_p, \hat{\boldsymbol{\Theta}}_p^{(t)}) &= \sum_{l=1}^{L} \sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \left\{ \omega_{l,0}^{(i,n)} \log \hat{\alpha}_{l,0} + \sum_{m=1}^{k_l} \left[\omega_{l,m}^{(i,n)} \log \alpha_{l,m} + Q_{l,m}^{(i,n)}(\theta_{l,m}) \right] \right\} - \\ & \frac{d}{2} \sum_{l=1}^{L} \sum_{m=1}^{k_l} \log \alpha_{l,m} - \sum_{l=1}^{L} \frac{k_l(d+1)+1}{2} \left(\log \frac{n_l}{12} + 1 \right) - \\ & \sum_{l=1}^{L} \lambda_l \left(\sum_{m=1}^{k_l} \alpha_{l,m} - (1 - \hat{\alpha}_{l,0}) \right). \end{aligned} \tag{3.27}$$

We obtain, at the optimum of (3.27),

$$\alpha_{l,m}^* = \frac{\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,m}^{(i,n)}}{\lambda_l} \quad m = 1, \ldots, k_l, \;\; l = 1, \ldots, L. \tag{3.28}$$

Inserting $\alpha_{l,m}^*$ into (3.27) results in

$$\begin{aligned} Q(\boldsymbol{\Theta}_p, \hat{\boldsymbol{\Theta}}_p^{(t)}) &= \sum_{l=1}^{L} \sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \sum_{m=1}^{k_l} \omega_{l,m}^{(i,n)} \log \frac{\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,m}^{(i,n)}}{\lambda_l} - \\ & \sum_{l=1}^{L} \lambda_l \left(\sum_{m=1}^{k_l} \frac{\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,m}^{(i,n)}}{\lambda_l} - (1 - \hat{\alpha}_{l,0}) \right) - \\ & \frac{d}{2} \sum_{l=1}^{L} \sum_{m=1}^{k_l} \log \frac{\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,m}^{(i,n)}}{\lambda_l} + \mathcal{R}, \end{aligned} \tag{3.29}$$

where the terms independent of λ_l are represented by the remainder $\mathcal{R}$. The value of λ_l maximizing (3.29) in the M-step is

$$\lambda_l^* = (1 - \hat{\alpha}_{l,0})^{-1} \left\{ \sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \sum_{m=1}^{k_l} \omega_{l,m}^{(i,n)} - \frac{dk_l}{2} \right\} \quad l = 1, \ldots, L. \tag{3.30}$$

The complete PML-CEM^2 algorithm is summarized in Table 3.1 below.

Table 3.1: Summary of the complete PML-CEM^2 algorithm for network delay tomography.

Inputs: T, $\mathbf{Y}$, k_{min}, ϵ, $\{\hat{\alpha}_{l,0}\}_{l=1}^{L}$, $\{k_l^{init}\}_{l=1}^{L}$,
initial parameters $\hat{\mathbf{\Theta}}_p^{(0)} = \left\{\hat{\alpha}_{l,1}^{(0)}, \ldots, \hat{\alpha}_{l,k_l^{init}}^{(0)}, \hat{\theta}_{l,1}^{(0)}, \ldots, \hat{\theta}_{l,k_l^{init}}^{(0)}\right\}_{l=1}^{L}$
Outputs: $\{f_l(x)\}_{l=1}^{L}$ with parameters $\hat{\mathbf{\Theta}}_p^{(opt)}$

$t \leftarrow 0$, $\mathcal{L}_{opt} \leftarrow -\infty$, Finish $\leftarrow 0$, $k_l \leftarrow k_l^{init}$ for $l = 1, \ldots, L$.
while ($\sim$ Finish) do
  Converge $\leftarrow 0$
  while ($\sim$ Converge) do
    $t \leftarrow t + 1$
    for $l = 1$ to L do
      $u_{l,m}^{(i,n)} \leftarrow P_{\hat{\theta}_{l,m}^{(t-1)}}\left(\mathbf{y}^{(i,n)} | Z_{l,m}^{(i,n)} = 1\right)$, $n = 1, \ldots, N_i$, $i : l \in \mathbf{M}_i$,
      $m = 0, \ldots, k_l$, $l = 1, \ldots, L$
      $m \leftarrow 1$
      while ($m \leq k_l$) do
        $\omega_{l,m}^{(i,n)} \leftarrow \hat{\alpha}_{l,m}^{(t-1)} u_{l,m}^{(i,n)} / \left(\hat{\alpha}_{l,0} u_{l,0}^{(i,n)} + \sum_{j=1}^{k_l} \hat{\alpha}_{l,j}^{(t-1)} u_{l,j}^{(i,n)}\right)$,
        $m = 1, \ldots, k_l$
        $\hat{\alpha}_{l,m}^{(t)} \leftarrow (1 - \hat{\alpha}_{l,0}) \max\left[\left(\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,m}^{(i,n)}\right) - \frac{d}{2},\ 0\right] \times$
        $\left\{\sum_{j=1}^{k_l} \max\left[\left(\sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} \omega_{l,j}^{(i,n)}\right) - \frac{d}{2},\ 0\right]\right\}^{-1}$
        $\hat{\theta}_{l,m}^{(t)} \leftarrow \arg\max_\theta \sum_{i:l\in\mathbf{M}_i} \sum_{n=1}^{N_i} Q_{l,m}^{(i,n)}(\theta)$
        if ($\hat{\alpha}_{l,m}^{(t)} > 0$) then
          $u_{l,m}^{(i,n)} \leftarrow P_{\hat{\theta}_{l,m}^{(t)}}\left(\mathbf{y}^{(i,n)} | Z_{l,m}^{(i,n)} = 1\right)$
          $m \leftarrow m + 1$
        else

remove the mth component from f_l by
$\hat{\alpha}_{l,j}^{(t)} \leftarrow \hat{\alpha}_{l,j}^{(t)} + \hat{\alpha}_{l,m}^{(t-1)}/(k_l - 1), \quad j = 1, \ldots, m-1$
$\hat{\alpha}_{l,j}^{(t-1)} \leftarrow \hat{\alpha}_{l,j+1}^{(t-1)} + \hat{\alpha}_{l,m}^{(t-1)}/(k_l - 1), \quad j = m, \ldots, k_l - 1$
$\hat{\theta}_{l,j}^{(t-1)} \leftarrow \hat{\theta}_{l,j+1}^{(t-1)}, \quad j = m, \ldots, k_l - 1$
$k_l \leftarrow k_l - 1$
end if
end while
$\omega_{l,m}^{(i,n)} \leftarrow \hat{\alpha}_{l,m}^{(t-1)} u_{l,m}^{(i,n)} / \left(\hat{\alpha}_{l,0} u_{l,0}^{(i,n)} + \sum_{j=1}^{k_l} \hat{\alpha}_{l,j}^{(t-1)} u_{l,j}^{(i,n)} \right),$
$m = 1, \ldots, k_l$
end for
$\hat{\mathbf{\Theta}}_p^{(t)} \leftarrow \left\{ \hat{\alpha}_{l,1}^{(t)}, \ldots, \hat{\alpha}_{l,k_l}^{(t)}, \hat{\theta}_{l,1}^{(t)}, \ldots, \hat{\theta}_{l,k_l}^{(t)} \right\}_{l=1}^{L}$
$\mathcal{L}_{CEM}(\hat{\mathbf{\Theta}}_p^{(t)}) \leftarrow \log f(\mathbf{Y}|\mathbf{\Theta}_p) - \frac{d}{2} \sum_{l=1}^{L} \sum_{m=1}^{k_l} \log \hat{\alpha}_{l,m}^{(t)} -$
$\sum_{l=1}^{L} \frac{k_l(d+1)+1}{2} \left(\log \frac{n_l}{12} + 1 \right) -$
$\sum_{l=1}^{L} \left\{ \sum_{i:l \in \mathbf{M}_i} \sum_{n=1}^{N_i} \sum_{m=1}^{k_l} \omega_{l,m}^{(i,n)} - \frac{dk_l}{2} \right\} \left((1 - \hat{\alpha}_{l,0})^{-1} \sum_{m=1}^{k_l} \hat{\alpha}_{l,m}^{(t)} - 1 \right)$
if $\left(\left| \mathcal{L}_{CEM}(\hat{\mathbf{\Theta}}_p^{(t)}) - \mathcal{L}_{CEM}(\hat{\mathbf{\Theta}}_p^{(t-1)}) \right| < \epsilon \right)$ then
Converge $\leftarrow 1$
end if
end while
if $\left(\mathcal{L}_{CEM}(\hat{\mathbf{\Theta}}_p^{(t)}) > \mathcal{L}_{opt} \right)$ then
$\mathcal{L}_{opt} \leftarrow \mathcal{L}_{CEM}(\hat{\mathbf{\Theta}}_p^{(t)})$
$\hat{\mathbf{\Theta}}_p^{(opt)} \leftarrow \hat{\mathbf{\Theta}}_p^{(t)}$
end if
if $(\exists l \in \{1, \ldots, L\} \;\; k_l > k_{min})$ then
$\mathbf{E}_s \leftarrow \{ l \in \{1, \ldots, L\} : \; k_l > k_{min} \}$
$(l^*, m^*) \leftarrow \arg\min_{m=1,\ldots,k_l, \; l \in \mathbf{E}_s} \hat{\alpha}_{l,m}^{(t)}$
remove the m^*th component from f_{l^*} by
$\hat{\alpha}_{l^*,j}^{(t)} \leftarrow \hat{\alpha}_{l^*,j}^{(t)} + \hat{\alpha}_{l^*,m^*}^{(t)}/(k_{l^*} - 1), \quad j = 1, \ldots, m^* - 1$
$\hat{\alpha}_{l^*,j}^{(t)} \leftarrow \hat{\alpha}_{l^*,j+1}^{(t)} + \hat{\alpha}_{l^*,m^*}^{(t)}/(k_{l^*} - 1), \quad j = m^*, \ldots, k_{l^*} - 1$

$\hat{\theta}^{(t)}_{l^*,j} \leftarrow \hat{\theta}^{(t)}_{l^*,j+1}, \quad j = m^*, \ldots, k_{l^*} - 1$
$k_{l^*} \leftarrow k_{l^*} - 1$
else
$Finish \leftarrow 1$
end if
end while

3.5 Experimental Results

3.5.1 Model Simulation: ML-EM Algorithm for Known Model Orders

We simulated a small network with the simple virtual tree topology shown in Figure 3.6. Throughout this first experiment the numbers of components $\{k_l\}_{l=1}^{L}$ are assumed known and used in the mixture model estimator. We specialized the EM algorithm to a Gaussian continuous component mixture plus a point mass (see Appendix A). From two to four Gaussian components were assigned to each link in addition to a point mass at 0. These simulations were implemented in `matlab` and we generated 2000 i.i.d. end-to-end probe pair delays for each of the six probe tree paths, i.e., $N_T = 6$. The ML-EM algorithm was applied to estimate the Gaussian components, their mixing parameters and the weight of the point mass at zero. Convergence was achieved after 955 iterations, or approximately 15 iterations per parameter. Figure 3.7 compares the estimated Gaussian mixture components to the true Gaussian mixture components. It also lists the number of mixture components for each link and the true/estimated probabilities $\alpha_{l,0}$ of the probe encountering empty queue on link l. The convergence curve of the log-likelihood is shown in Figure 3.7(h). These results illustrate high accuracy for the case where there is no

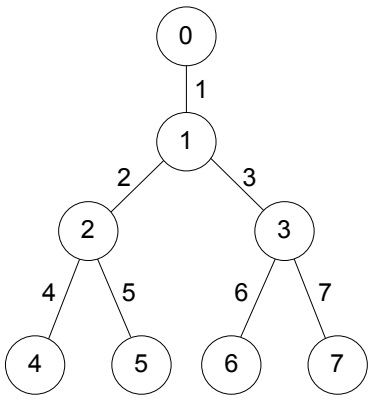


Figure 3.6: The four-leaf tree network used in computer experiments.

model error and the number of components is known.

3.5.2 NS Simulation: MML for Unknown Model Order

For a more realistic simulation we used `ns-2` [40] to simulate the network shown in Figure 3.6 with a variety of cross traffic types and router configurations. The links were assigned bandwidths and latencies listed in Table 3.2. The `ns` parameters for each link were set to a Drop-Tail queue (FIFO queue with finite buffer). The queue buffer sizes were 50 packets long. Each packet in a probe pair was defined as a 40 byte UDP packet. Probe pairs were generated independently and sent along each of the three probe trees in Figure 3.8 according to a Poisson process with mean interarrival time 2ms.

Cross traffic was also generated in each link by `ns` and consisted of 41 Pareto On-Off TCP flows and 25 constant-bit-rate UDP streams with random noise introduced in the scheduled packet departure times. The design of background traffic reflects today's IP network environment in which the UDP traffic is mainly video/audio data streams and TCP comprises the major fraction of the Internet traffic [69, 70]. The backgroud traffic approximately occupied 40links. We evenly divided the bandwidth

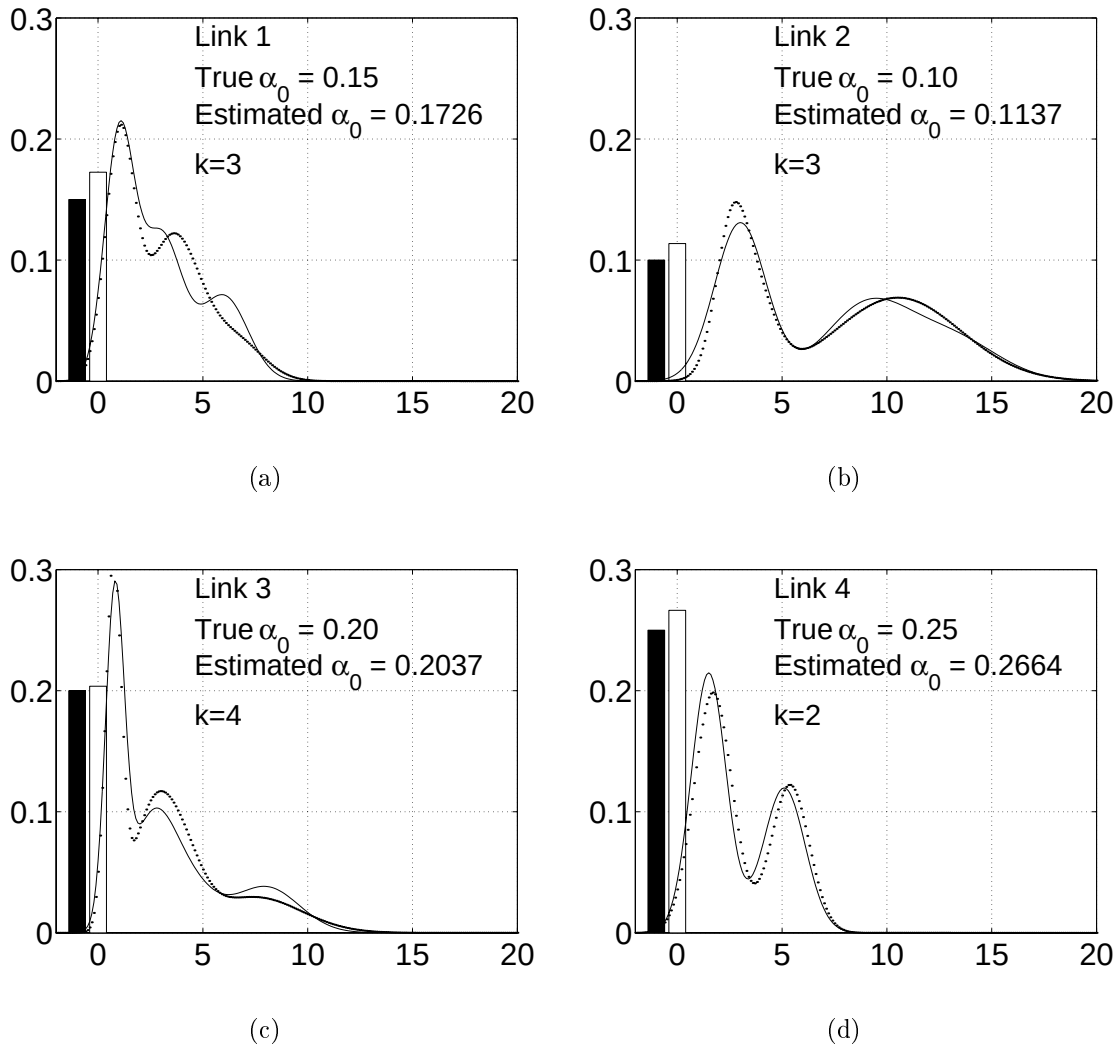
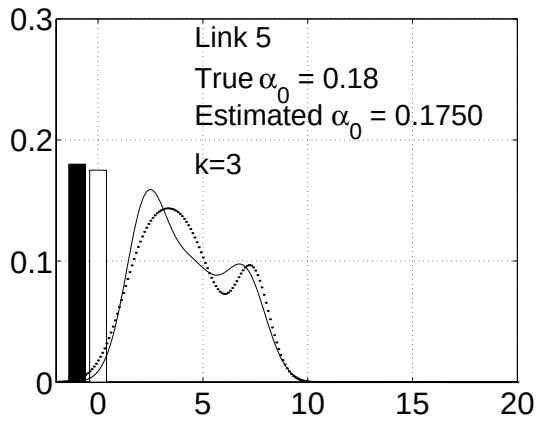
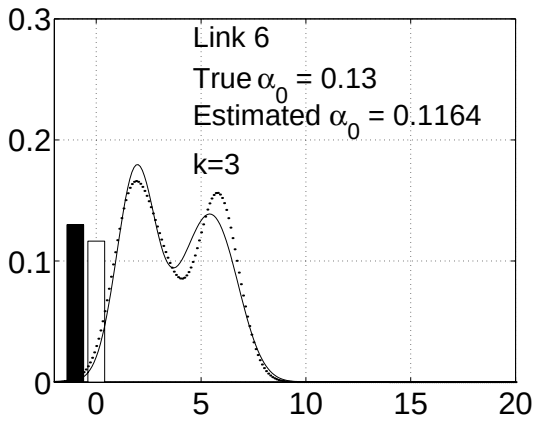


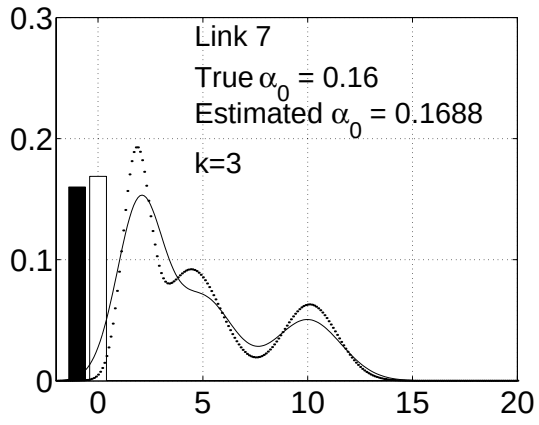
Figure 3.7: (a) - (g)True(solid curve) and estimated(dotted curve) Gaussian mixture components along with the true(black bar) and estimated(white bar) empty queue probabilities $\{\alpha_{l,0}\}$ for model simulation. The horizontal axes denote link packet delays in ms. Here the EM algorithm is used to estimate the hybrid Gaussian mixture parameters for simulated measurements obeying a true hybrid Gaussian mixture with known numbers of components which are listed along with the link delay pdf's. 2000 packet pairs are generated for each of the six probe tree paths in Figure 3.6. (h) shows the convergence curve of the log-likelihood function.



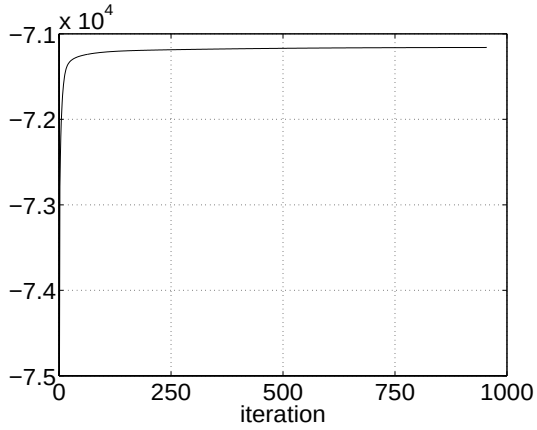
(e)



(f)



(g)



(h)

Figure 3.7 (continued)

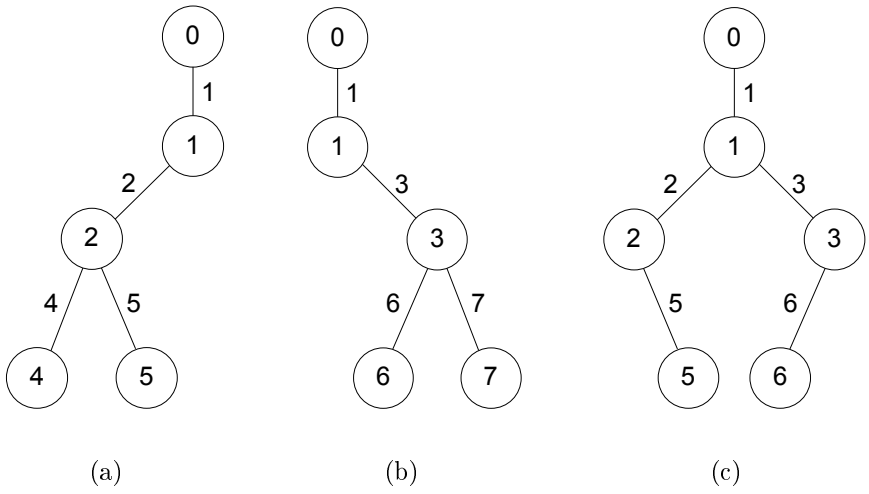
Table 3.2: Link bandwidth and latency parameters used in `ns-2` simulation.

Link	1	2	3	4	5	6	7
Bandwidth (Mbps)	20	20	20	10	15	15	12
Latency (ms)	100	100	100	30	50	50	40

utilization of the background traffic into each data flow. It means that the packet admission rate in a traffic flow with a large packet size was always less than that in a flow with a small packet size. As the link buffer sizes were fixed in number of packets, it implies that most queueing delays encountered by the probes were caused by small packets. Therefore we could expect concentration at small delays in the link delay distributions.

A total of $N = 1500$ packet pairs for each probe tree were collected at the receiver nodes. We estimated each probe queueing delay by subtracting the minimum probe delay over the total samples for the same path and set the empty-queue delays $x_{l,0}$ to 0 for $l = 1, \ldots, 7$. The PML-CEM^2 algorithm was implemented with Gaussian continuous mixtures. To initialize the parameters we estimated the continuous mixture models for delays over the four end-to-end paths using Figueiredo and Jain's algorithm in [48], which itself was initialized with 15 components. Figure 3.9 shows the estimated Gaussian mixtures and the convergence curves of the penalized log-likelihood. The results were used to obtain f_l^{init}, $l = 1, \ldots, 7$ for the PML-CEM^2 algorithm to start with, as described in Section 3.4.3. These initial distributions are shown in Figure 3.10.

We set $k_{min} = 2$ as the lower bound of mixture model orders. The convergence threshold ϵ was set to 10^{-3}. When model error exists it is possible for a Gaussian component to converge to some single delay value with the variance approaching zero. When it happens the likelihood goes to infinity and the algorithm becomes unstable. We solve this problem by adding another threshold ϵ_{σ^2} in the algorithm



(a) (b) (c)

Figure 3.8: Probe Trees used in `ns` simulation.

to simply kill a component whose variance goes below it. The reason is when a component converges to a single delay it could imply a point mass at the specific delay and the component is no longer suitable for describing the distribution. It may be necessary to include more point delay masses in the model to avoid such component annihilation. Here we let $\epsilon_{\sigma^2} = 10^{-15}$. Convergence was achieved after 6966 iterations and the link delay pdf estimates are shown in Figure 3.11. To obtain ground truth the true internal link delay distributions were estimated empirically from the `ns` simulated data. The mass of the atom, which is denoted as "True $\alpha_{l,0}$" in the figure, is the empirically estimated probability of an empty queue at link l calculated from sample averages. The continuous portion of the true distribution was estimated by the histogram of non-zero link delay samples and normalized to have mass $1 - (\text{True } \alpha_{l,0})$.

The estimated Gaussian mixtures are shown along with the normalized histogram for comparison. Since link delays are strictly no less than zero, we redefine the continuous portion of the estimate by distributing the mixture probability mass in

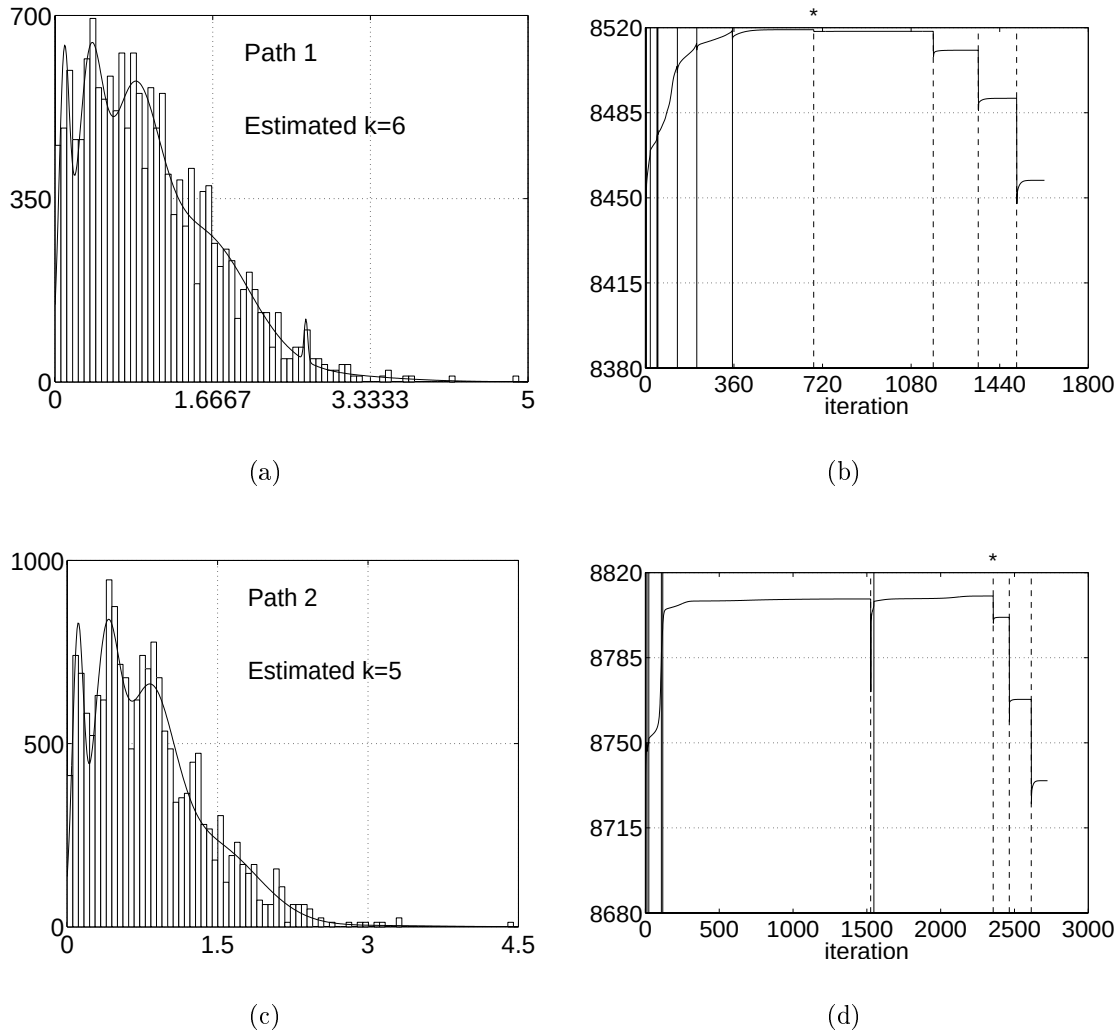
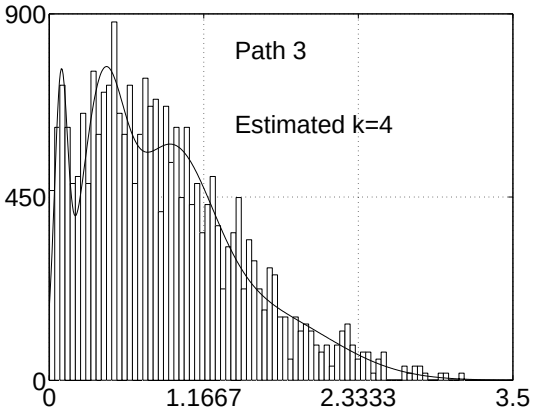
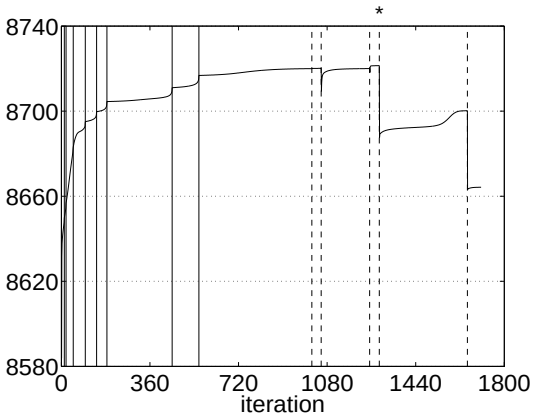


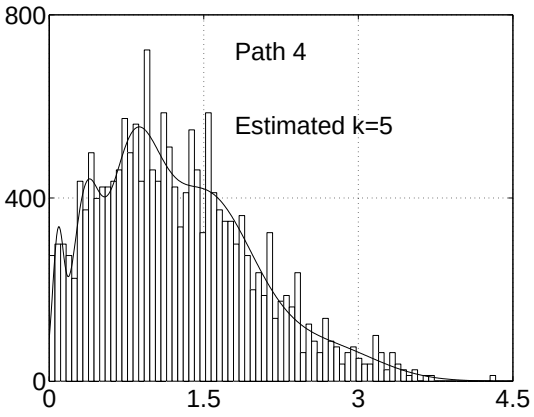
Figure 3.9: The continuous mixture estimates in the `ns` simulation of non-zero end-to-end delays for the paths from the root to leaf nodes 4,5,6, and 7, denoted by path 1,2,3, and 4, respectively. (a),(c),(e),(g) show the estimates (solid curve) and the empirical non-zero path delay histograms (white bars) which were normalized to have a total probability mass equal to 1. The horizontal axes denote link packet delays in milliseconds. The estimates were obtained using the algorithm in [48]. The corresponding convergence curves of the penalized log-likelihood are shown to the right of the estimates in (b),(d),(f), and (h), respectively. The solid vertical lines show the iterations in which a mixture component is removed by component annihilation in the algorithm. The dashed lines show the convergent iterations where the least-probable component is removed and the algorithm is restarted with the remained components. The asterisk indicates the iteration converging to the highest penalized likelihood.



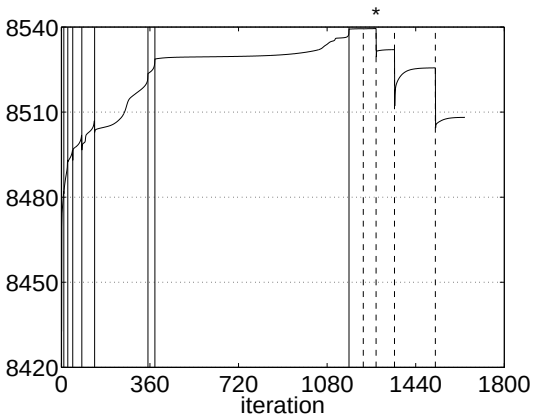
(e)



(f)



(g)



(h)

Figure 3.9 (continued)

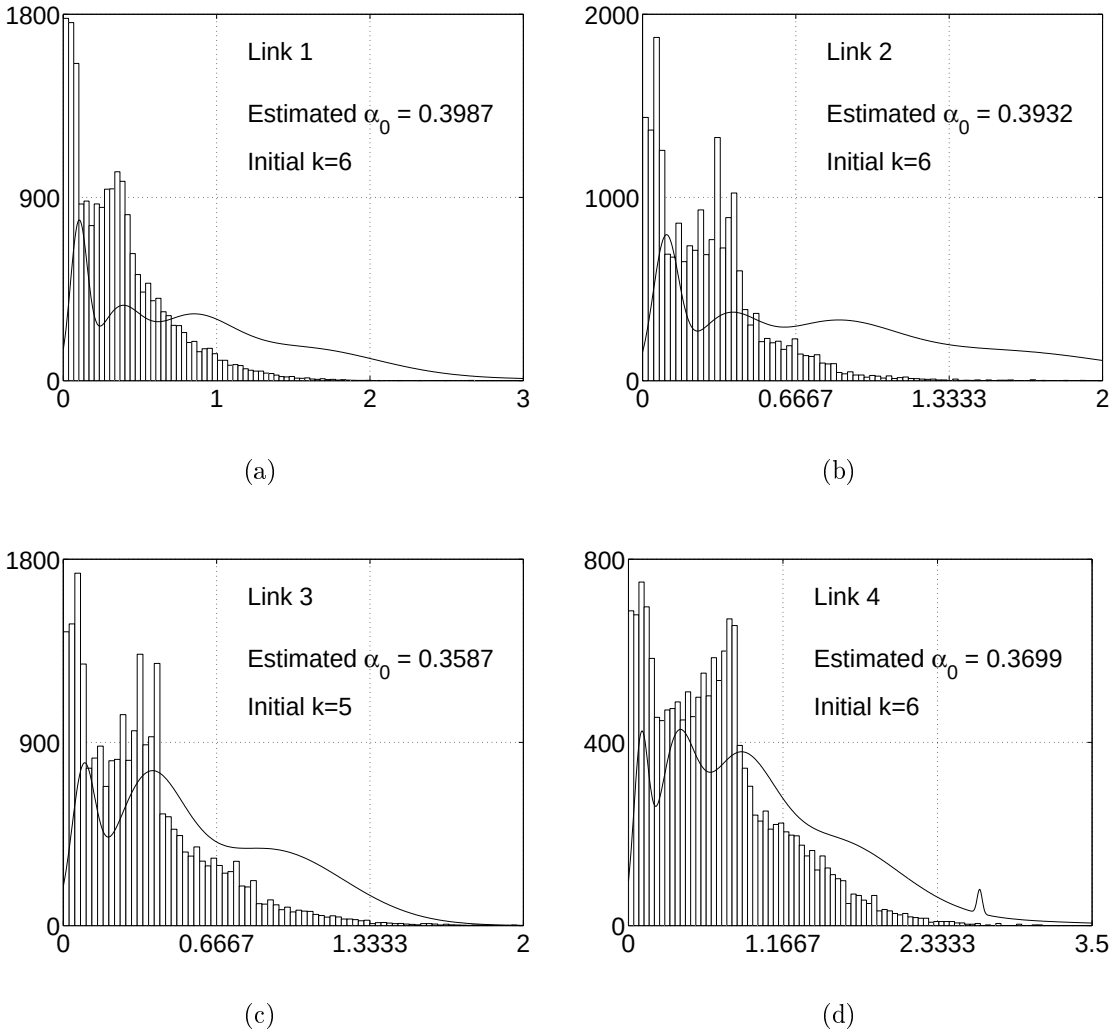
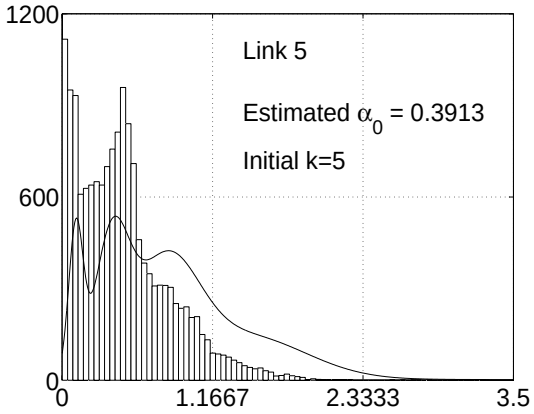
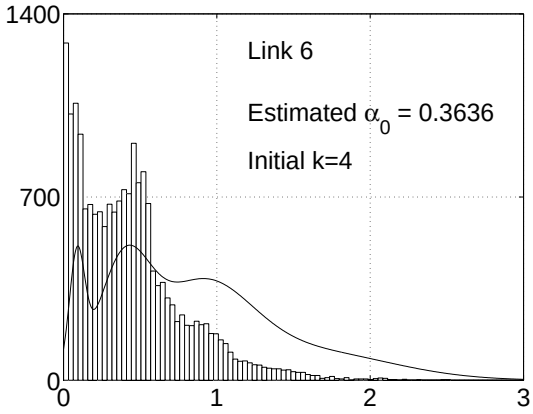


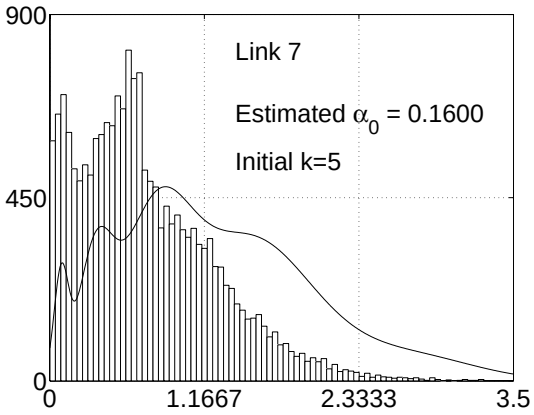
Figure 3.10: Initialization of the link delay distributions (solid curve) using path delay mixture extimates in the `ns` simulation. Here only the continuous mixture components are shown for each f_l^{init}. The horizontal axes denote link packet delays in milliseconds. Empirical non-zero link delay histograms (white bars) were normalized to have a total probability mass equal to $(1 - \hat{\alpha}_{l,0})$ for comparison.



(e)



(f)



(g)

Figure 3.10 (continued)

$(-\infty, 0)$ to the region $(0, \infty)$. The total probability mass over the positive delay region is $1 - \hat{\alpha}_{l,0}$. The convergence curve of the penalized log-likelihood function is shown in Figure 3.11(h). The vertical lines indicate the iterations when at least one component is annihilated automatically. Note that the annihilations in iteration 159 and 3060 were due to excessively small variances in components of link 7 and 1, respectively. Likelihood spikes can be seen at those iterations, but the corresponding parameter sets were rejected as PML estimates because the algorithm did not converge.

The hybrid delay mixture models were also empirically estimated from the internal link delays generated by `ns`. With the mixture components being Gaussian, they were similarly normalized to the support region $(0, \infty)$ by distributing the probability mass over $(-\infty, 0)$. The results were compared to the PML-CEM^2 estimates, as shown in Figure 3.12. The L_1 error norms between the empirical and PML-CEM^2 hybrid mixture estimates are depicted in Figure 3.12(h). L_1 error norm is widely used in density estimation to assess the goodness of inference [71, 72]. In our case it is simply the sum of the absolute difference between the minimum delay probabilities and the integral of the absolute difference between the continuous mixtures over $(0, \infty)$.

As shown in Figure 3.11 and 3.12, the Gaussian mixture components capture the profile of the empirical continuous portion of the density for most of the links. They also provide accurate estimates to all the queueing delay ranges. Some modal mismatches occur in the estimates at, for example, link 1. This error is probably due to the limitation of the $k_1^{init} = 6$ Gaussian + 1 point mass component model. For a better fit to the internal delay histograms it may be necessary to assign more point masses and include other density models which are flatter or more heavy-tailed than Gaussian. Other sources of error might include: violation of the spatial or

temporal independence assumptions; insufficient number of probe samples to resolve link densities; existence of local maxima in the likelihood function; and burstiness (non-stationarity) of the traffic. These are topics worthy of additional investigation.

3.6 Conclusion and Future Work

This chapter focuses on the estimation of internal link delay distributions from end-to-end unicast packet pair delay measurements when there is a positive probability of zero queueing delay, i.e., lightly to moderately loaded networks. We proposed a new hybrid discrete-continuous finite mixture model which circumvents the difficulties of link delay discretization. For the case that mixture model orders are known, we derived an EM algorithm for approximating the ML estimates. Model simulation showed that when all model assumptions hold the EM algorithm can very accurately estimate the delay distributions for each internal link. When the model orders are unknown, we implemented an MML order selection penalty and derived an unsupervised algorithm for estimating both the number of mixture components and the continuous density parameters. Results of `ns-2` simulation showed that reasonably accurate estimates of internal link delay distributions are possible. The estimated link delay distribution yield a statistic which could be used to extract information such as mean link delay, tail probability of large link delays, and multi-modality in link delay distribution.

Future work could include finding ways to accelerate convergence of the ML-EM and PML-EM algorithms for real-time implementation. EM algorithms are generally slow and the improvement made by CEM^2 is still limited. This makes it difficult to perform extensive comparisons. In the `ns` simulation we generated traffic flows that travelled across two or more consecutive links. These data flows contributed

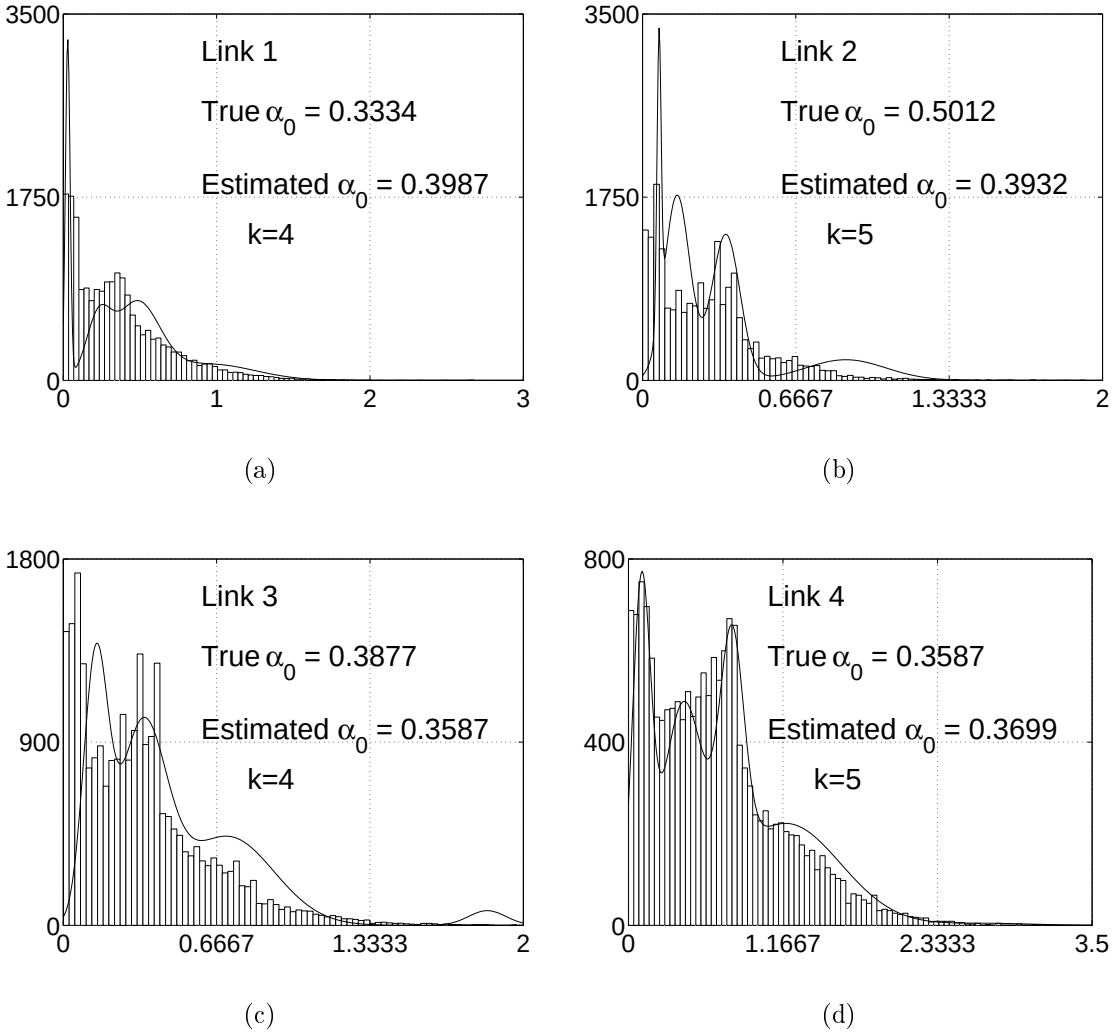
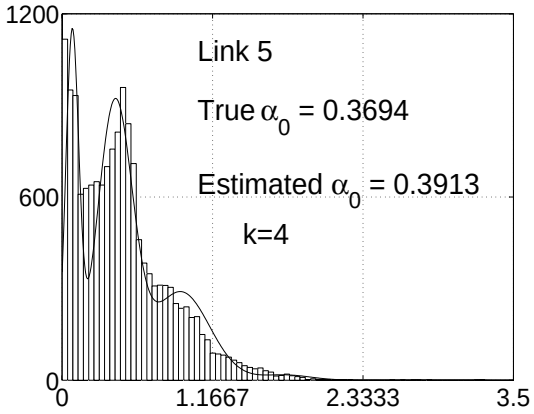
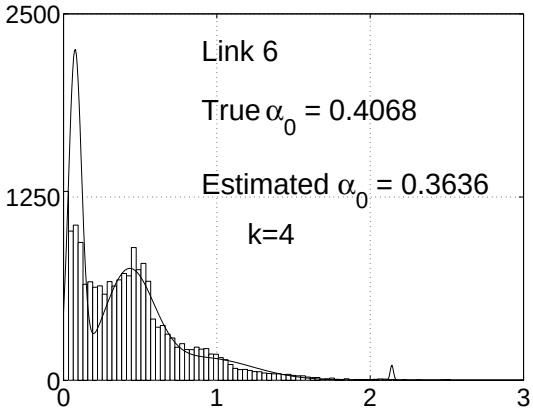


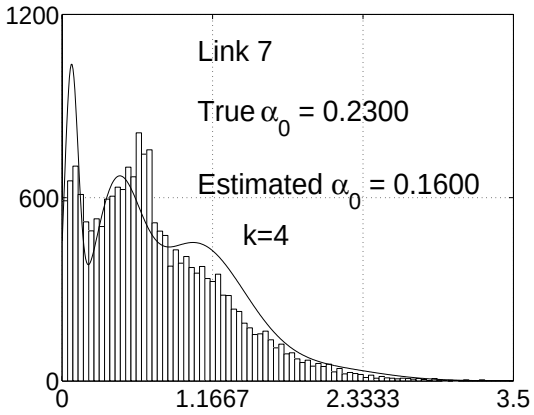
Figure 3.11: (a) - (g) Normalized ns-derived histograms for non-zero link delays and estimated Gaussian mixture density for indicated links. The horizontal axes denote link packet delays in milliseconds. (h) shows the convergence curve of the MML penalized likelihood function. The solid vertical lines denote the iteration numbers where the number of Gaussian mixture components is reduced by component annihilation in the algorithm. The dashed vertical lines show the convergent iterations where the component with the least mixture probability over all the links is removed and the algorithm is restarted with the remained components. The links affected by component removal in the increasing order of the iterations are $\{4, 5, 7, 3, 1, 2, 1, 2, 6, 4, 5, 3, 2, 1, 7, 3, 2, 6, 4, 7, 1, 4, 5\}$. The asterisk indicates the iteration which converges to the highest penalized likelihood.



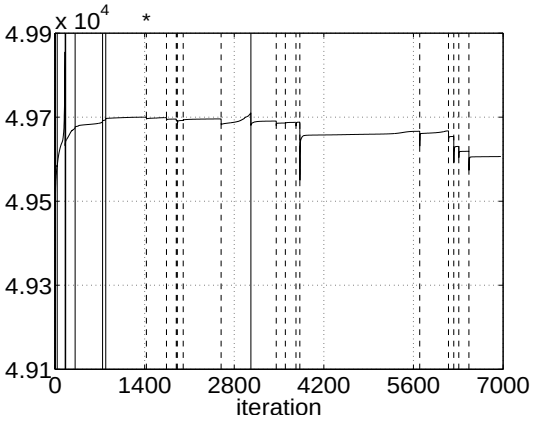
(e)



(f)



(g)



(h)

Figure 3.11 (continued)

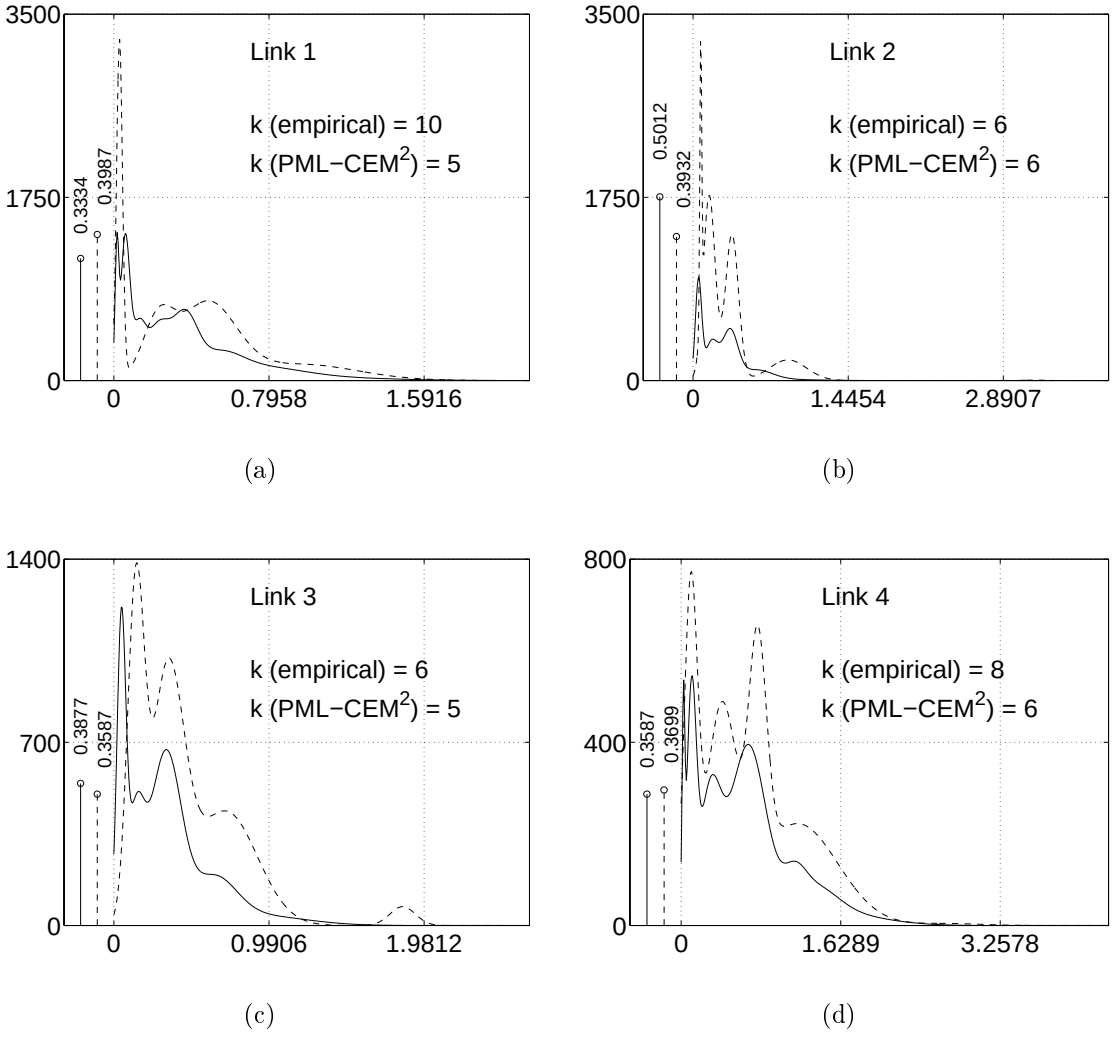


Figure 3.12: (a) - (g) Estimated hybrid mixture models by empirical link delays (solid) and end-to-end delays using the PML-CEM^2 algorithm (dashed) for indicated links in the `ns` simulation. k(empirical) and k(PML-CEM^2) denote the model orders for the two estimates, respectively. The estimated empty queue probabilities are shown by stems left to the mixture model estimates. The horizontal axes denote link packet delays in milliseconds. L_1-norm errors between the two estimates are shown in (h).

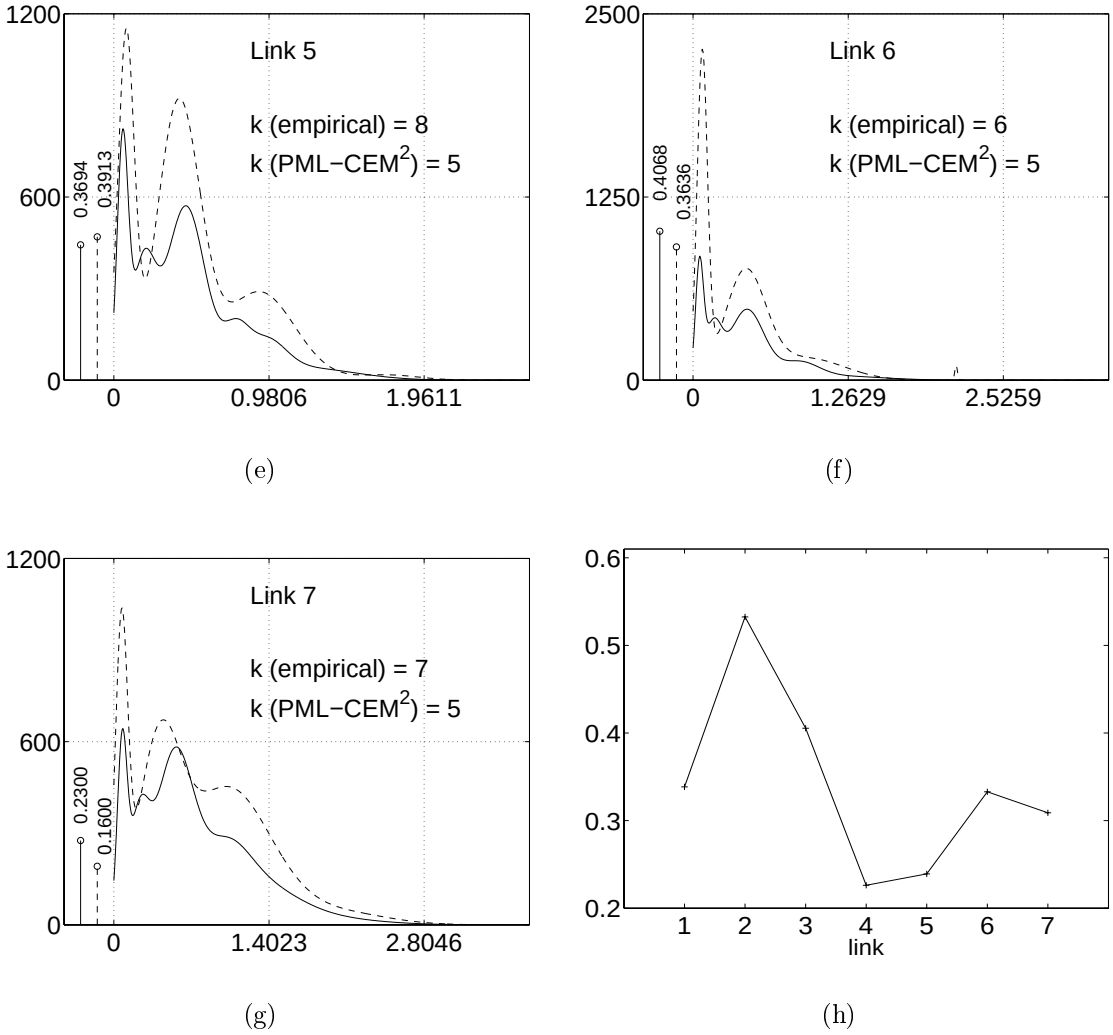


Figure 3.12 (continued)

approximately 10% of the background traffic and introduced spatial dependenecy on link delays over their traffic path. The experiment result showed that our algorithm produced accurate estimates for link delay distributions in spite of the violation of spatial independence assumption. However, it is necessary to evaluate the effect on our algorithm when the spatial dependency on delays among different links becomes more severe.

Another possible direction is extension of our model to include spatial dependencies of link delays among different links, especially the links along the same path. To better fit link delay distributions in a real network, one could extend our hybrid finite mixture model to include more point masses and combinations of different families of probability densities which are flatter or more heavy-tailed than Gaussian. For example, exponential densities are more efficient to describe exponential tails of the distributions which correspond to rare large delay events in a lightly or moderately loaded network. For heavily loaded situations some heavy tailed functions, such as Pareto densities, might be used. This could also reduce the model orders in the finite mixture models and diminish the computational complexity for the ML-EM and PML-EM algorithm. For time-varying scenarios adaptive schemes need to be developed in order to capture possible changes in the traffic statistics and the network environment. It could be achieved by estimating only the tail components of the mixture models after each consecutive probing session if the change on large packet delays is the focus. It may also be viable to apply these methods to detecting abnormal changes in link delay distributions, which could be helpful to early detection of possible failures and/or malicious activities in the network.

CHAPTER 4

Accelerated Estimation of Hybrid Mixture Delay Models

4.1 Introduction

When hybrid mixture models are used in network delay tomography, the number of components in a path delay distribution is the product of link delay model orders throughout that path. The computational complexitiy of the PML-EM algorithm is extremely high due to the exponential growth of end-to-end delay model orders with the network size. This makes the algorithm of chapter 3 extremely slow even with moderate-sized networks. In this chapter we propose a divide-and-conquer strategy to sequentially estimate link delay distributions from the bottom level to the top of the network. Although it is impossible to reduce the complexity lower than exponential rate since the number of links and probe trees increases exponentially with the depth of the network tree, we can decrease the exponential rate of increase for the shared paths of the probe trees. The accelerated algorithm is tested with the `ns-2` data generated in Section 3.5.2 and the results are compared with those obtained from the original algorithm. These results show that the accelerated algorithm at-

tains close L_1 error norm performance to the original while saving approximately 40% of runtime.

Our contribution in this chapter is summarized as follows: (1) We propose an accelerated estimation algorithm for network link delay distributions based on end-to-end delay measurements. Our algorithm divides and conquers the problem in a bottom-up fashion which allows parallel processing of the probe trees with branch-splitting nodes having the same depth; (2) Through `ns` simulation we demonstrate the accelerated algorithm produces accurate estimates for link delay distributions and saves approximately 40% run-time compared to the original algorithm in Chapter 3.

The chapter is organized as follows. In Section 4.2 we analyze the computational complexity of the original algorithm and discuss the motivation to reduce its exponential base. In Section 4.3 we provide notations and definitions to be used in the accelerated algorithm. Then the algorithm is summarized step by step and illustrated by an example. In Section 4.4 we apply the accelerated algorithm to `ns-2` generated data and compare the results with those from the original algorithm introduced in Chapter 3. Section 4.5 concludes the chapter and discusses the future work.

4.2 Motivation

The complexity of the PML-EM algorithm in the previous chapter has an exponential order of magnitude with respect to the depth of the logical network tree. In addition to the notations and assumptions used in Chapter 3, we define the level of a node v by its hop distance to the root node, denoted by $level(v)$. Consider a symmetric binary tree with depth D in which all leaf nodes have levels D. For example, the network used for simulations in Chapter 3, as shown in Figure 3.6, is a symmetric binary tree with depth 3. In the network there are a total of $L = 1+\cdots+2^{D-1} = (2^D-1)$

links and 2^{d-1} level-d internal nodes for $d = 1, \ldots, D$. (Remember we assume the root node always has a single child node.) The total number of internal nodes is $1 + \cdots + 2^{D-2} = (2^{D-1} - 1)$, which is also the minimum required number of probe trees N_T^{min} in this case. Here we use only N_T^{min} probe trees to sufficiently cover the network. Assume the number of mixture components in every link delay model of the same magnitude order and that this number has an average equal to $\bar{k} = \frac{1}{L}\sum_{l=1}^{L} k_l$, where k_l is the number of components used in link i. The average computation needed to perform one iteration for each term of $\{\omega_{l,m}^{(i,n)}, Q_{l,m}^{(i,n)}(\theta_{l,m})\}$ (see Appendix A) in the conditional expectation (3.7) is lower bounded by $\mathcal{O}(\bar{k}+1)^{D-1}$. This bound follows by Jensen's inequality $\frac{1}{L}\sum_{l=1}^{L}(k_l+1)^{D-1} \geq (\bar{k}+1)^{D-1}$. Let $N_t = N$ be the number of probe pairs collected at probe tree t. From Eq. (3.6) an lower-bound for the average complexity of the PML-EM algorithm is

$$\begin{aligned} &\sum_{l=1}^{L}\sum_{t=1}^{N_T^{min}}\sum_{n=1}^{N}(2\bar{k}+1)(\bar{k}+1)^{D-1} \\ &= \quad N(2^D-1)(2^{D-1}-1)(2\bar{k}+1)(\bar{k}+1)^{D-1} \sim \mathcal{O}\left((4\bar{k}+4)^D\right), \end{aligned} \tag{4.1}$$

where $L = (2^D - 1)$, $N_T^{min} = (2^{D-1} - 1)$, and $2\bar{k} + 1$ is the average number of the terms $\{\omega_{l,m}^{(i,n)}$ and $Q_{l,m}^{(i,n)}(\theta_{l,m})$ to be computed for one packet pair delay sample. To be specific, one needs to compute $\bar{k}+1$ $\{\omega_{l,m}^{(i,n)}$ and $\bar{k}$ $Q_{l,m}^{(i,n)}(\theta_{l,m})$ for each sample (see Appendix A for details).

From the `ns-2` simulation in chapter 3 we found there was an inconsistency between the model order of the path delay distribution obtained directly from the end-to-end delays and the product of the model orders for individual links along that path. Although theoretically the former should be in the order of $(\bar{k}+1)^D$, we observed it to be an order of $\bar{k}$ instead. Possible reasons are that we did not collect enough data samples and/or the interprobing time sampling was too coarse

to yield low variance estimates of the distributions. However, these are also practical restrictions in real networks. The Internet is a highly variable environment that has insufficient stationarity for the collection of large amount of independent identically distributed samples. The interprobing must not be too large in order to avoid flooding the network with probes. Under such circumstances one simply has to accommodate moderately high variance in aggregated delays along chains of links and hope this does not cause much loss in accuracy for the estimates of individual delay distributions. This is possible when there are many probe paths intersecting the same links.

For each probe tree we propose to approximate the delay distribution of the shared path with a low model order, e.g., similar to that for a single link, regardless of the actual length of the path. Let $b(t)$ be the branching node of probe tree t. The depth of t is determined by the level of its branching node, i.e., $depth(t) = level(b(t))$. In a symmetric binary tree with depth D there are 2^{d-1} probe trees with depth d for $d = 1, \ldots, D-1$, and they all have d links in their shared paths. The original algorithm of Chapter 3 requires an average not less than $(\bar{k}+1)^d$ components to describe the delays along these shared paths, when, as above, $\bar{k}$ is the average number of components per link. Therefore the computational complexity of the EM iteration over these shared paths is lower-bounded by

$$\begin{aligned}
&\sum_{l=1}^{L} \sum_{t=1}^{N_T^{min}} \sum_{n=1}^{N} (2\bar{k}+1)(\bar{k}+1)^{level(t)} \\
&= N(2^D-1)(2\bar{k}+1) \sum_{d=1}^{D-1} 2^{d-1}(\bar{k}+1)^d \\
&= N(2^D-1)(2\bar{k}+1) \frac{(2\bar{k}+2)^D - (2\bar{k}+2)}{2(2\bar{k}+1)} \\
&\sim \mathcal{O}\left((4\bar{k}+4)^D\right) = \mathcal{O}\left(e^{\rho_0 D}\right), \qquad (4.2)
\end{aligned}$$

where $\rho_0 = \log(4\bar{k}+4)$. (4.2) is the same as the bound on average complexity over all paths in (4.1). On the other hand, if we use only $\bar{k}$ components on the shared paths the average per-iteration complexity becomes

$$\begin{aligned}
&\sum_{l=1}^{L}\sum_{t=1}^{N_T^{min}}\sum_{n=1}^{N}(2\bar{k}+1)(\bar{k}+1) \\
&= \quad N(2^D-1)(2^{D-1}-1)(2k+1)(k+1) \\
&\sim \quad \mathcal{O}\left(4^D\right) = \mathcal{O}\left(e^{\rho_1 D}\right), \qquad (4.3)
\end{aligned}$$

where $\rho_1 = \log(4)$. Thus we have a complexity exponential rate constant which is significantly reduced (from ρ_0 to ρ_1) and that is independent of $\bar{k}$. More specifically, we have reduced dependency in $\bar{k}$ from D-depth power to quadratic.

If the model order reduction approximation is directly applied to the PML-EM algorithm to update all the parameters simutaneously, the algorithm may not converge. This is because there is no longer guaranteed monotonicity in the likelihood sequence since the algorithm is not fully consistent with its complete data model. To counteract this deficiency, we use a divide-and-conquer strategy instead of simutaneous updating. We estimate link parameters level by level, from the deepest internal nodes to the root's child node. For each internal node in a level of this "bottom-up approach", we apply the PML-EM algorithm to a subset of probe trees whose shared path includes the route from root to that node. The purpose is that every time when we use the PML-EM algorithm the path from root to the specific internal node is treated as a single link, called a superlink, whose delay distribution is described by a single hybrid mixture model in the algorithm. In this way, the algorithm corresponds to an identical complete data model used in Chapter 3 and we maintain the property of monotonic likelihood convergence. Details will be provided in the next section.

4.3 Accelerated PML-EM Algorithm

4.3.1 Notations and Definitions

Consider a logical tree network $T = (\mathbf{V}, \mathbf{E})$. Probe pairs are generated from the root node 0 and sent through $N_T (\geq N_T^{min})$ different binary probe trees which satisfy the sufficient probing condition. The probe trees are labelled from $t = 1$ to N_T and the set of probe trees is called $\mathbf{T}_p = \{1, \ldots, N_T\}$. Let $c_1(t)$ and $c_2(t)$ be the left and right child node of the branching node $b(t)$ in probe tree t, respectively. We say that probe tree t_1 is a descendant of t_2 if $b(t_1) = c_1(t_2)$ or $b(t_1) = c_2(t_2)$, denoted by $t_1 \prec t_2$, for $t_1, t_2 \in \mathbf{T}_p$. By convention we say $t \prec t$. Denote the set of all descendant probe trees of t by $\mathbf{T}_d(t) = \{t' \in \mathbf{T}_p : t' \prec t\}$, for $t \in \mathbf{T}_p$. We call a superlink any chain of links in the tree and define the superlink from the root to $b(t)$ of a probe tree t by $l_1(t)$. We also represent the directed links in t which connect $b(t)$ to $c_1(t)$ and $c_2(t)$ by $l_2(t)$ and $l_3(t)$ by $l_2(t) = (b(t), c_1(t))$ and $l_3(t) = (b(t), c_2(t))$, respectively. A pseudo two-leaf tree obtained from probe tree t is defined by $\tilde{t} = \left\{\tilde{\mathbf{V}}(t) = \{0, b(t), c_1(t), c_2(t)\}, \tilde{\mathbf{E}}(t) = \{l_1(t), l_2(t), l_3(t)\}\right\}$.

Consider the example of the logical tree network in Figure 4.1(a). A minimum number $N_T^{min} = 5$ of probe trees are used to send probe pairs and they are shown in Figure 4.1(b)-(f). Here $\mathbf{T}_p = \{1, 2, 3, 4, 5\}$. For example, the branching nodes and levels of probe tree 1,2, and 5 are $b(1) = 6$, $b(2) = 3$, $b(5) = 1$, and $depth(1) = 3$, $depth(2) = 2$, $depth(5) = 1$, respectively. Two examples for sets of descendant trees are $\mathbf{T}_d(2) = \{1, 2\}$ and $\mathbf{T}_d(5) = \{2, 3, 4, 5\}$. For probe tree 2, $c_1(2) = 6$ and $c_2(2) = 7$, $l_1(2)$ is the superlink from node 0 to 3, $l_2(2) = 6$ and $l_3(2) = 7$. It's pseudo two-leaf tree $\tilde{2}$ is shown in Figure 4.1(g).

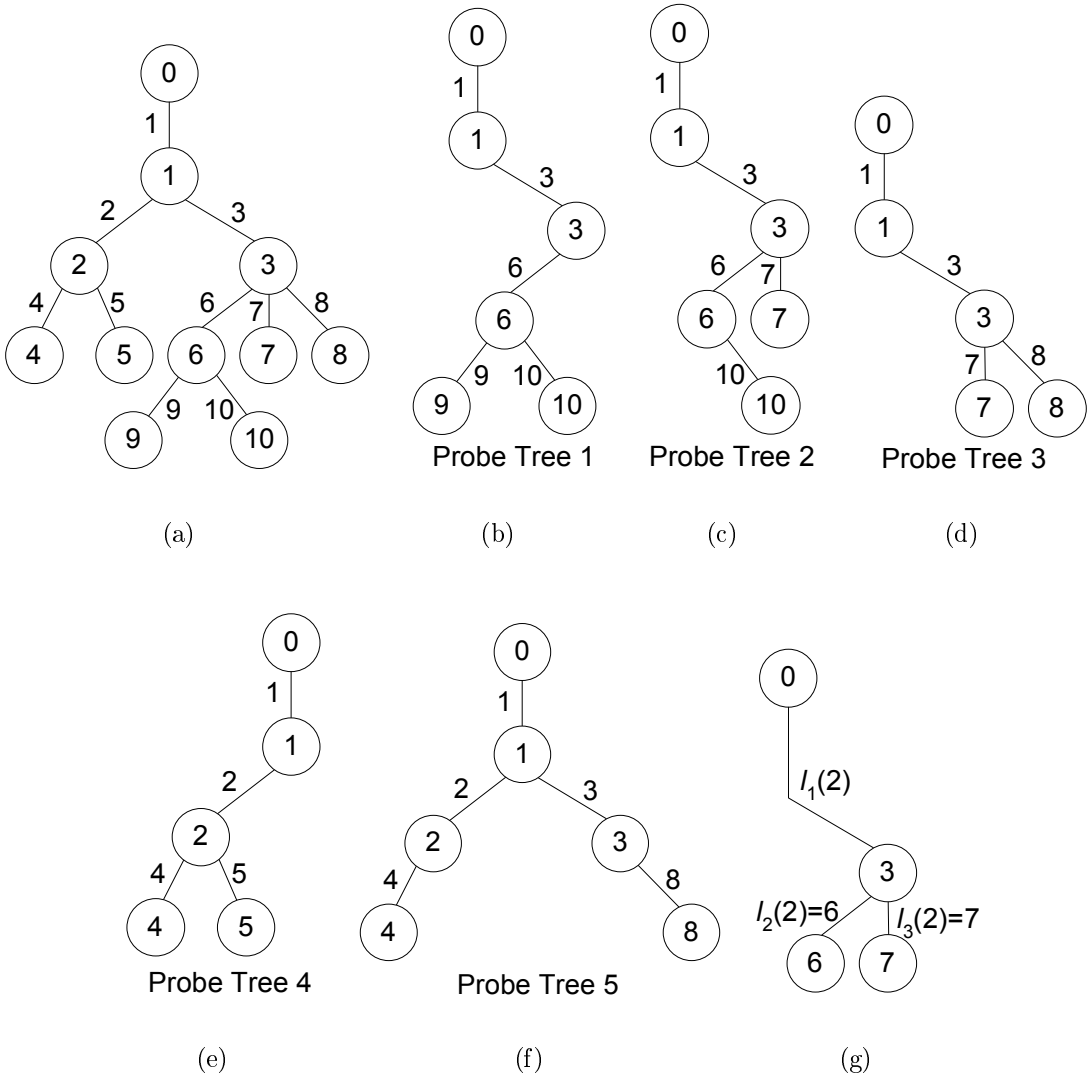


Figure 4.1: Example of a logical tree network (a) and its 5 probe trees (b) - (f).(g) shows the pseudo two-leaf tree $\tilde{2}$.

4.3.2 Initialization

We form sample average estimates for the point masses in the delay distributions as in Chapter 3. For the mixture models we adopt the same idea of initialization for a two-leaf tree network from Section 3.4.3. Consider the pseudo two-leaf tree network $\tilde{t}$ in Figure 4.2 for one of the probe trees t. Define X_1, X_2, and X_3 the delays on links $l_1(t)$, $l_2(t)$, and $l_3(t)$, respectively. The end-to-end delays from root to leaf of the pseudo probe tree are $Y_1 = X_1 + X_2$ and $Y_2 = X_1 + X_3$. Note that $\tilde{t}$ may be entire probe tree t or just a portion of it. The continuous non-zero delay mixture model of the aggregated delays Y_1, for example, can be obtained from the previous level of the algorithm if node $c_1(t)$ is not a leaf node of T, or is otherwise estimated by applying the mixture EM algorithm to the end-to-end probe delays collected at $c_1(t)$ which are greater than the minimum path transmission latency. Suppose the mixture models for Y_1 and Y_2 are

$$\chi_r^{(t)}(y) = \sum_{m=1}^{\kappa_r} \rho_{r,m}\phi(y;\theta_{r,m}) \quad r = 1, 2, \tag{4.4}$$

respectively. Let $\eta = \sum_{l \in l_1(t)} x_{l,0}$ be the minimum transmission delay on $l_1(t)$, and $\hat{\beta}$ be the sample average estimate of its probability point mass. Define $\kappa = max(\kappa_1, \kappa_2)$. κ mixture components $\left\{\rho_{r_j,m_j}\phi(y;\theta_{r_j,m_j})\right\}_{j=1}^{\kappa}$ are selected from $\chi_1^{(t)}(y)$ and $\chi_2^{(t)}(y)$ according to the least distance criterion described in Section 3.4.3. The initialization of delay distribution estimates $h_1^{(t)}(x)$, $h_2^{(t)}(x)$, and $h_3^{(t)}(x)$ for X_1, X_2, and X_3, respectively, are selected as before:

$$\begin{aligned} h_1^{(t)init}(x) &= \hat{\beta}\delta(x-\eta) + \xi \sum_{j=1}^{\kappa} \rho_{r_j,m_j}\phi(x;\theta_{r_j,m_j}) \\ h_i^{(t)init}(x) &= \hat{\alpha}_{l_i(t),0}\delta(x - x_{l_i(t),0}) + (1 - \hat{\alpha}_{l_i(t),0})\chi_{i-1}(x) \quad i = 2, 3, \end{aligned} \tag{4.5}$$

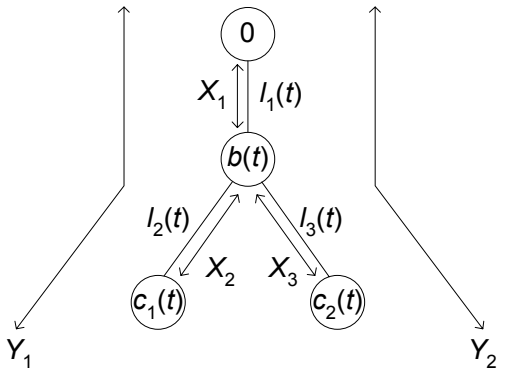


Figure 4.2: A pseudo two-leaf tree network denoted $\tilde{t}$.

where ξ is a factor which normalizes the sum of $\{\rho_{r_j,m_j}\}_{j=1}^{\kappa}$ to $1-\hat{\beta}$. In the sequal we will assume that the deterministic zero-queueing delays η and $\{x_{l_i(t),0}\}_i$ are known. Without loss of generality, we will assume that $\eta = x_{l_2(t),0} = x_{l_3(t),0} = 0$ for x being pure queueing delay.

4.3.3 The Accelerated Algorithm

The accelerated algorithm proceeds as follows. It starts with the set of deepest probe trees. For each t in the set we first form its pseudo tree $\tilde{t}$ and the set of descendent probe trees $\mathbf{T}_d(t)$. When we apply the PML-EM algorithm to $\mathbf{T}_d(t)$ we only estimate the 3 (super)links in pseude probe tree $\tilde{t}$. For a lower depth of probe trees it is possible that there exist other links in $\mathbf{T}_d(t)$ which do not belong to $\tilde{t}$. In that case those additional links are all below $\tilde{t}$ and their delay distribution estimates can be obtained from previous levels. In that case the delay distributions of those links are replaced by their estimates and remain fixed in the algorithm. If either path delay mixture models of $\chi_1^{(t)}(y)$ and $\chi_2^{(t)}(y)$ is unknown for $\tilde{t}$, which occurs when the path ends at a leaf node of the logical tree of the network, we run the unsupervised learning algorithm of Figueiredo and Jain [48] with respect to the end-

to-end non-zero probe delays collected for that path. Then we initialize the delay distributions with $h_i^{(t)init}(x), \;\; i = 1, 2, 3$ in (4.5). To take advantage of component-wise implementation as in Section 3.5.2 we apply the PML-CEM^2 algorithm to all the data samples collected at $\mathbf{T}_d(t)$. The results are passed to the next higher level and used for initialization or as fixed parameters in the PML-CEM^2 algorithm for this level. The process is repeated until all levels are explored and all the link delay distributions are estimated. Note that this approach allows parallel processing of all the probe trees with the same depth. The complete accelerated algorithm is summarized in Table 4.1.

Table 4.1: Summary of the accelerated PML-CEM^2 algorithm for network delay tomography.

Inputs: T, $\mathbf{Y}$, k_{min}, ϵ
Outputs: $\{f_l(x)\}_{l=1}^{L}$ in $\hat{\mathbf{\Theta}}_p^{(opt)}$

$d \leftarrow \max_{t \in \mathbf{T}_p} depth(t)$
while $(d > 0)$ do
    for each $\{t \in \mathbf{T}_p : depth(t) = d\}$ do
        form $\tilde{t}$
        if ($\chi_1^{(t)}(y)$ or $\chi_2^{(t)}(y)$ is unknown) then
            run unsupervised mixture learning algorithm
            with respect to the unknown path(s).
        end if
        compute $h_i^{(t)init}(x), \;\; i = 1, 2, 3$
        run PML-CEM^2 algorithm :
            Inputs: $\mathbf{T}_d(t)$, $\left\{\mathbf{Y}^{(i,n)} : i \in \mathbf{T}_d(t)\right\}$, k_{min}, ϵ, $\left\{h_i^{(t)init}(x), \;\; i = 1, 2, 3\right\}$,
                $\{f_l(x) : l \in \mathbf{T}_d(t), \;\; l \neq l_i(t), i = 1, 2, 3\}$
            Outputs: $h_1^{(t)}(x)$, $\left\{f_{l_i(t)}(x) = h_i^{(t)}(x)\right\}_{i=2,3}$
    end for
    $d \leftarrow d - 1$
end while

4.3.4 An Example

Consider the network in Figure 4.1(a) and its 5 probe trees in (b)-(f). The deepest probe tree is probe tree 1 with depth 3, so we begin the algorithm with $d = 3$. Figure 4.3 shows the pseudo tree $\tilde{1}$ in (a). (b) and (c) are the two paths used to obtain $\chi_1^{(1)}(y)$ and $\chi_2^{(1)}(y)$, respectively. Here $\mathbf{T}_d(1) = \{1\}$ so we run the PML-CEM^2 algorithm with respect to probe tree 1 only and we get results $h_1^{(1)}(x)$ for superlink $l_1(1)$, $\hat{f}_9(x) = h_2^{(1)}(x)$ and $\hat{f}_{10}(x) = h_3^{(1)}(x)$.

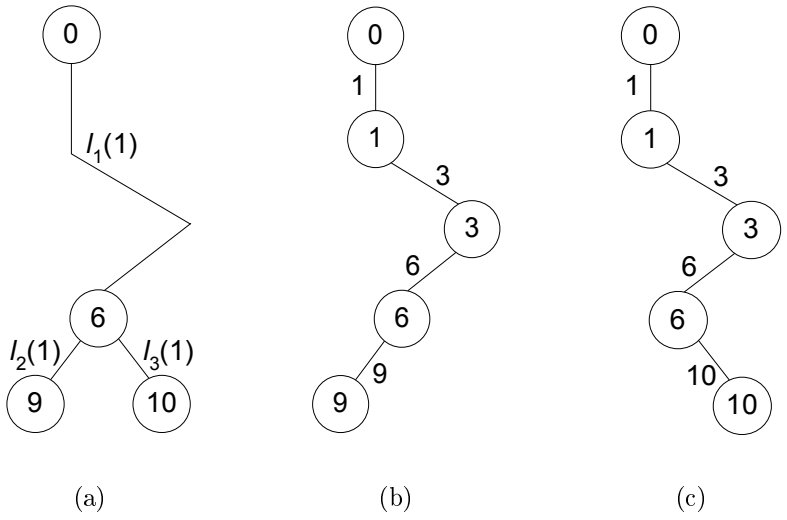
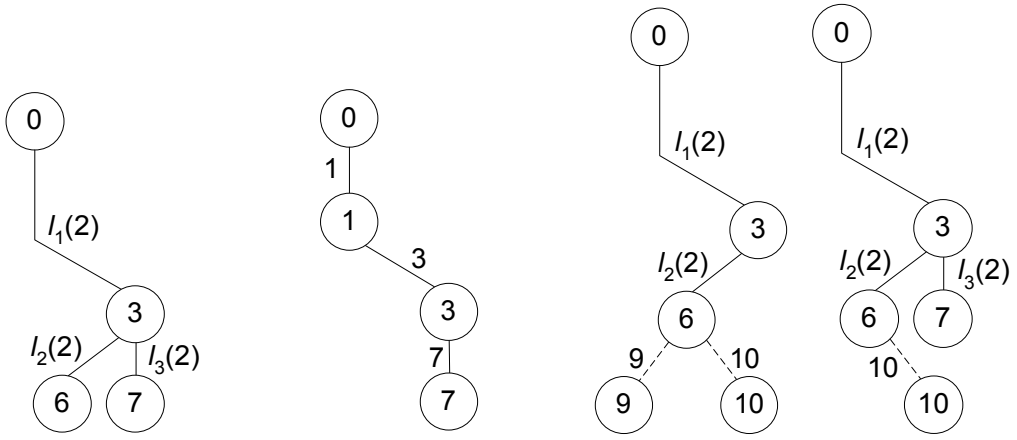


Figure 4.3: Topologies used in the example in Section 4.3.4 for level $d = 3$ in the accelerated algorithm. (a) The pseudo tree $\tilde{1}$. (b) The path for $\chi_1^{(1)}(y)$. (c) The path for $\chi_2^{(1)}(y)$.

The level $d = 2$ contains three probe trees 2, 3 and 4. We process them in parallel as follows. For probe tree 2 we form the pseudo probe tree $\tilde{2}$ as shown in Figure 4.4(a). The mixture model $\chi_1^{(2)}(y)$ for $\tilde{2}$ is $h_1^{(1)}(y)$, obtained from the previous level, and $\chi_2^{(2)}(y)$ is estimated using data collected for the path depicted in (b). The PML-CEM^2 algorithm is applied to $\mathbf{T}_d(2) = \{1, 2\}$ as shown in (c), where $\hat{f}_9(x)$ and $\hat{f}_{10}(x)$ (denoted by dashed lines) are known and fixed in the algorithm. The results



(a) (b) (c)

Figure 4.4: Topologies used in the example in Section 4.3.4. for level $d = 2$ in the accelerated algorithm. (a) The pseudo tree $\tilde{2}$. (b) The path for $\chi_2^{(2)}(y)$. (c) Probe trees 1 and 2, from left to right, in $\mathbf{T}_d(2)$ used in the PML-CEM^2 algorithm. The estimates for the dashed links are obtained from the previous level and fixed in the algorithm.

are $h_1^{(2)}(x)$ for superlink $l_1(2)$, $\hat{f}_6(x) = h_2^{(2)}(x)$ and $\hat{f}_7(x) = h_3^{(2)}(x)$. Similarly, for probe tree 3 the pseudo tree $\tilde{3}$ and the two paths for $\chi_1^{(3)}(y)$ and $\chi_2^{(3)}(y)$ are depicted in Figure 4.5(a)-(c), respectively. $\mathbf{T}_d(3) = \{3\}$ and the results of the PML-CEM^2 algorithm are $h_1^{(3)}(x)$ for superlink $l_1(3)$, $\hat{f}_7(x) = h_2^{(3)}(x)$ and $\hat{f}_8(x) = h_3^{(3)}(x)$. Using probe tree 4 we get $h_1^{(4)}(x)$ for $l_1(4)$, $\hat{f}_4(x) = h_2^{(4)}(x)$ and $\hat{f}_5(x) = h_3^{(4)}(x)$. Here $\mathbf{T}_d(4) = \{4\}$ and the topologies used for probe tree 4 are shown in Figure 4.6.

In the last level $d = 1$ the only probe tree having depth equal to 1 is probe tree 5. Its pseudo probe tree $\tilde{5}$ does not have any superlinks, as shown in Figure 4.7(a). In this case $\chi_1^{(5)}(y) = h_1^{(4)}(y)$ and $\chi_2^{(5)}(y) = h_1^{(2)}(y)$ or $h_1^{(3)}(y)$. The set of descendant trees of probe tree 5 is $\mathbf{T}_d(5) = \{2, 3, 4, 5\}$. They are depicted in Figure 4.7(b) where the links $4, 5, 6, 7, 8, 10$ are drawn in dashed lines because their delay distribution estimates are known and fixed in the PML-CEM^2 algorithm.

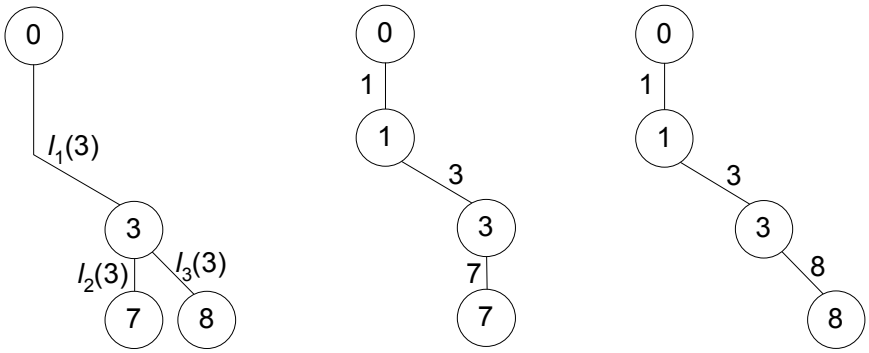


Figure 4.5: Topologies used in the example in Section 4.3.4 for level $d = 2$ in the accelerated algorithm. (a) The pseudo tree $\tilde{3}$. (b) The path for $\chi_1^{(3)}(y)$. (c) The path for $\chi_2^{(3)}(y)$.

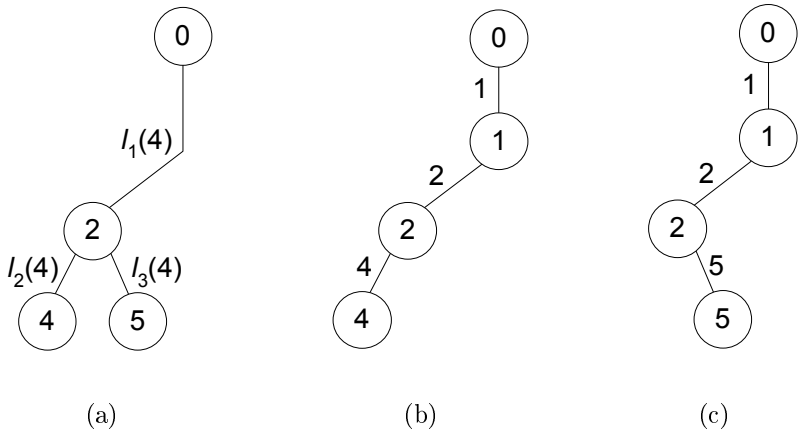
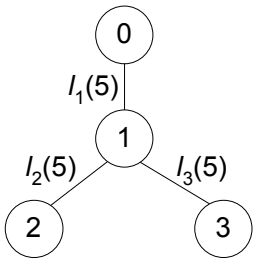
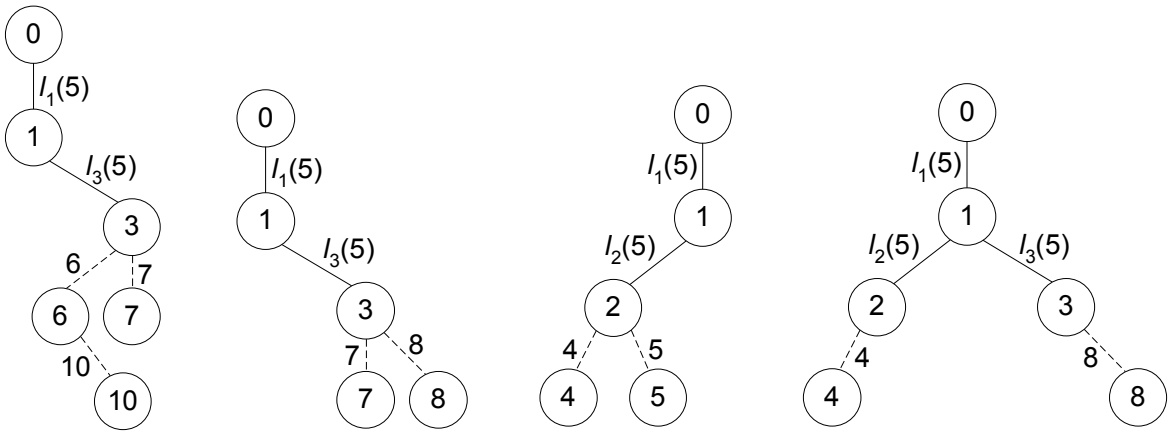


Figure 4.6: Topologies used in the example in Section 4.3.4 for level $d = 3$ in the accelerated algorithm. (a) The pseudo tree $\tilde{4}$. (b) The path for $\chi_1^{(4)}(y)$. (c) The path for $\chi_2^{(4)}(y)$.



(a)



(b)

Figure 4.7: Topologies used in the example in Section 4.3.4 for level $d = 3$ in the accelerated algorithm. (a) The pseudo tree $\tilde{5}$. (b) Probe trees 2,3,4,5, from left to right, in $\mathbf{T}_d(5)$ used in the PML-CEM^2 algorithm. The estimates for the dashed links are obtained from previous levels and fixed in the algorithm.

4.4 Experimental Results

We use the accelerated algorithm on the `ns-2` data simulated in the previous chapter. The network and probe tree diagrams are redisplayed in Figure 4.8 for convenience, where (b), (c), and (d) depict probe tree 1, 2, and 3, respectively. Here we adopt Gaussian mixture components with a single point mass as before. The fixed empty queue delays $\{x_{l,0}\}_{l=1}^{7}$ are assumed equal to 0 and their probabilities are estimated from sample averages. There are two levels in the accelerated algorithm.

In the first level $d = 2$, probe trees 1 and 2 are used to estimate links 4 - 7 and superlinks $l_1(1)$ and $l_1(2)$. Figure 4.9 and 4.10 show the estimated distributions with their mixture orders and zero-delay point masses for probe tree 1 and 2, respectively. Normalized "ground truth" empirical delay histograms are also shown superimposed for comparison of fitting error. These were estimated directly from `ns-2` simulated link level delays that were not available to the tomographic reconstruction algorithm. The histogram for superlink $l_1(2)$ from node 0 to 2, for example, is derived from convolution of those in link 1 and 2, and it is displayed with twice as many number of bins. Estimated $h_1^{(1)}(x)$ and $h_1^{(2)}(x)$ are drawn in Figure 4.9(a) and Figure 4.10(a) with titles "Link 1 and 2" and "Link 1 and 3", respectively. Other estimates give link delay distributions: $\hat{f}_4(x) = h_2^{(1)}(x)$, $\hat{f}_5(x) = h_3^{(1)}(x)$, $\hat{f}_6(x) = h_2^{(2)}(x)$, and $\hat{f}_7(x) = h_3^{(2)}(x)$. Note that here we also implement the component annihilation criterion for excessively small variances. Such annihilations result in spikes in likelihood curves. Figure 4.11 gives the results of the second level $d = 1$ in which the only probe tree having depth equal to 1 is probe tree 3 and all the three probe trees are included in $\mathbf{T}_d(3)$ Here $h_1^{(1)}(x)$ and $h_1^{(2)}(x)$ obtained from the previous level are used for initialization. Since this is the shallowest level, there is no superlink in the pseudo probe tree $\tilde{3}$. The results $h_1^{(3)}(x)$, $h_2^{(3)}(x)$, and $h_3^{(3)}(x)$ are estimates for delay distributions $f_1(x)$, $f_2(x)$, and $f_3(x)$, respectively.

The results are compared with those obtained using the original algorithm as introduced in Chapter 3. Figure 4.12(a)-(g) show the estimates obtained from both algorithms superimposed with empirical hybrid mixture model estimates derived from the simulated link delays. The mixture models in the empirical estimates were obtained by applying the algorithm of Figueiredo and Jain to non-zero link delays. The estimates from both algorithms for link 4, 5, 6, and 7 are close to each other, but the original algorithm achieves somewhat better modal match to the modes of

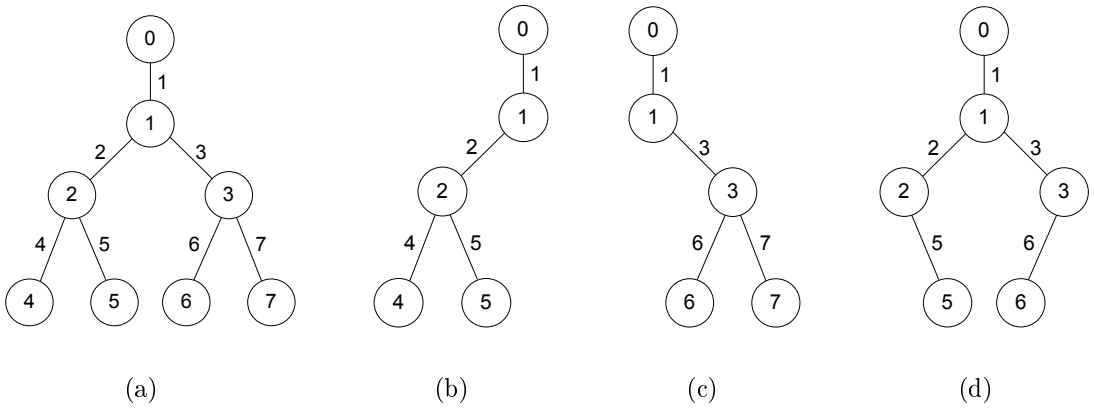


Figure 4.8: (a) The four-leaf tree network used in `ns` simulation. (b)-(d) The probe trees.

the true density for link 1 and 3. However, the L_1 error norms in Figure 4.12(h) show there is not a significant difference between the two and the accelerated algorithm even achieves slightly lower L_1 error for most of the links. This is probably because the original algorithm does not have sufficient data samples to take advantage of its finer resolution for the shared path distributions. When data collection is insufficient for some of the links the estimation error for those links propagates through the whole parameter space. However, in the accelerated algorithm this propagation is partially mitigated due to its divide-and-conquer structure. In a practical situation where the number of observations is limited, this demonstrates the advantage of the accelerated algorithm to expedite the estimation process with little loss in accuracy. The total run time savings obtained by the accelerated algorithm is approximately 40%.

4.5 Conclusion and Future Work

In this chapter we proposed a solution to the slow convergence problem of the PML-EM algorithm for network delay tomography developed in Chapter 3. It adopts

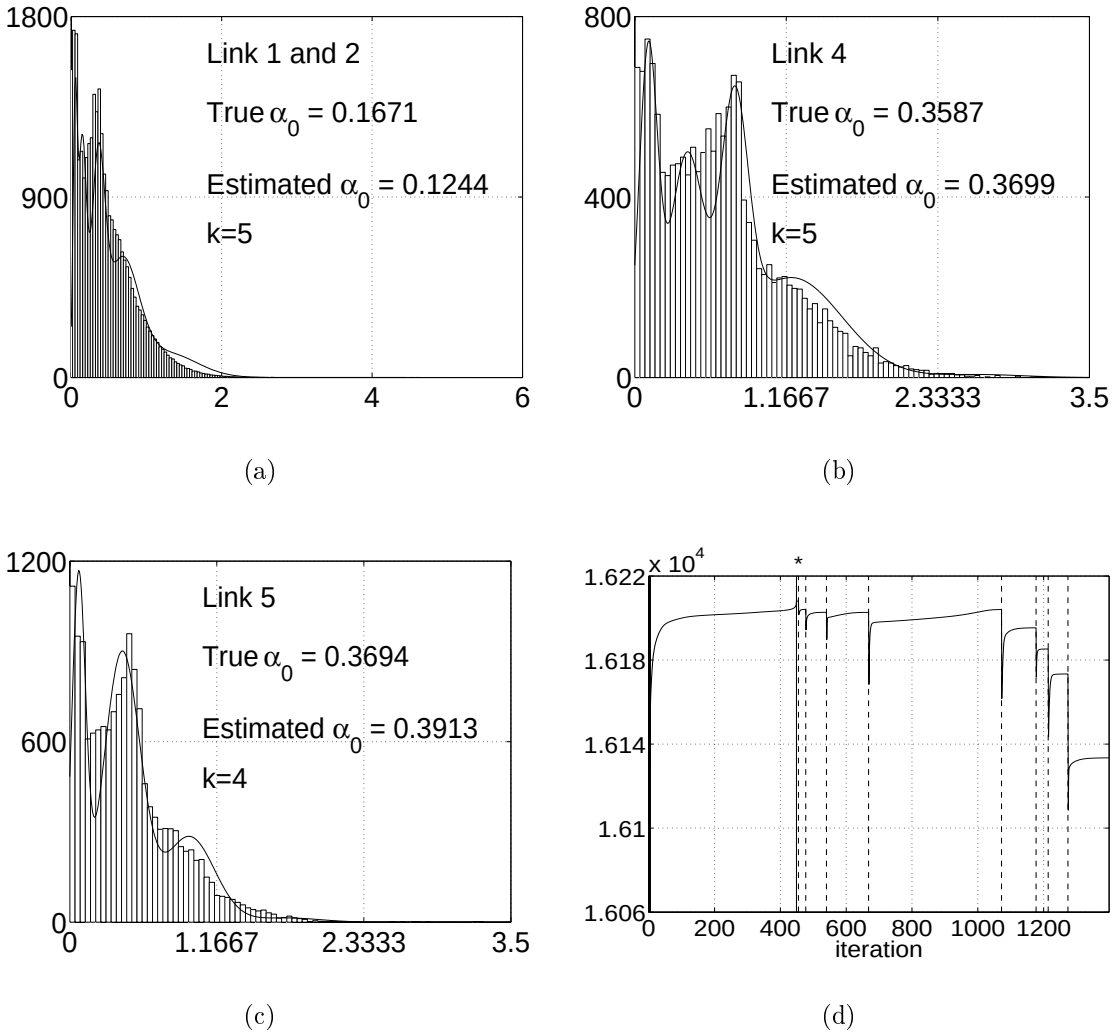
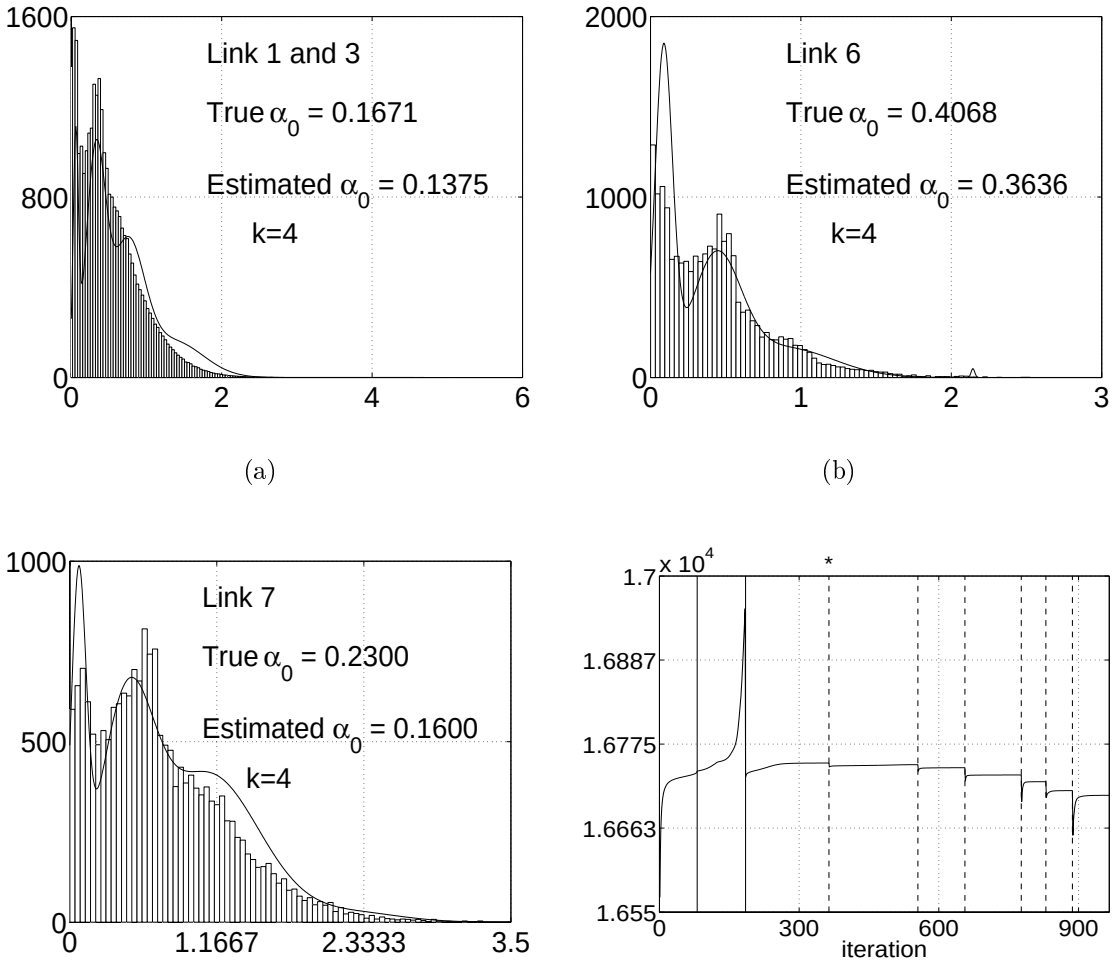


Figure 4.9: Results from the first level $d = 2$ of the accelerated algorithm using probe tree 1. (a) - (c) Normalized ns-derived histograms for non-zero link delays and estimated Gaussian mixture density for indicated (super)links. The horizontal axes denote link packet delays in milliseconds. (d) shows the convergence curve of the MML penalized likelihood function. The solid vertical lines denote the iteration numbers where the number of Gaussian mixture components is reduced automatically by component annihilation in the EM algorithm. The dashed vertical lines show the convergent iterations where the component with the least mixture probability over all the links is removed and the algorithm is restarted with the remained components. The links $\{l_i(1)\}$ affected by this reduction are indexed by {2,3,1,2,3,1,2,1,2,3,1} in iterations {2,5,449,455,477,540,668,1072,1177,1214,1274}, respectively. The asterisk indicates the iteration which converges to the highest penalized likelihood.



(a) (b)

(c) (d)

Figure 4.10: Results from the first level $d = 2$ of the accelerated algorithm using probe tree 2. (a) - (c) Normalized ns-derived histograms for non-zero link delays and estimated Gaussian mixture density for indicated (super)links. The horizontal axes denote link packet delays in milliseconds. (d) shows the convergence curve of the MML penalized likelihood function. The solid vertical lines denote the iteration numbers where the number of Gaussian mixture components is reduced by component annihilation in the EM algorithm. The dashed vertical lines show the convergent iterations where the component with the least mixture probability over all the links is removed and the algorithm is restarted with the remained components. The links $\{l_i(2)\}$ affected by this reduction are indexed by {1,3,2,3,1,2,3,1} in iterations {81,185,364,555,656,777,830,887}, respectively. The asterisk indicates the iteration which converges to the highest penalized likelihood.

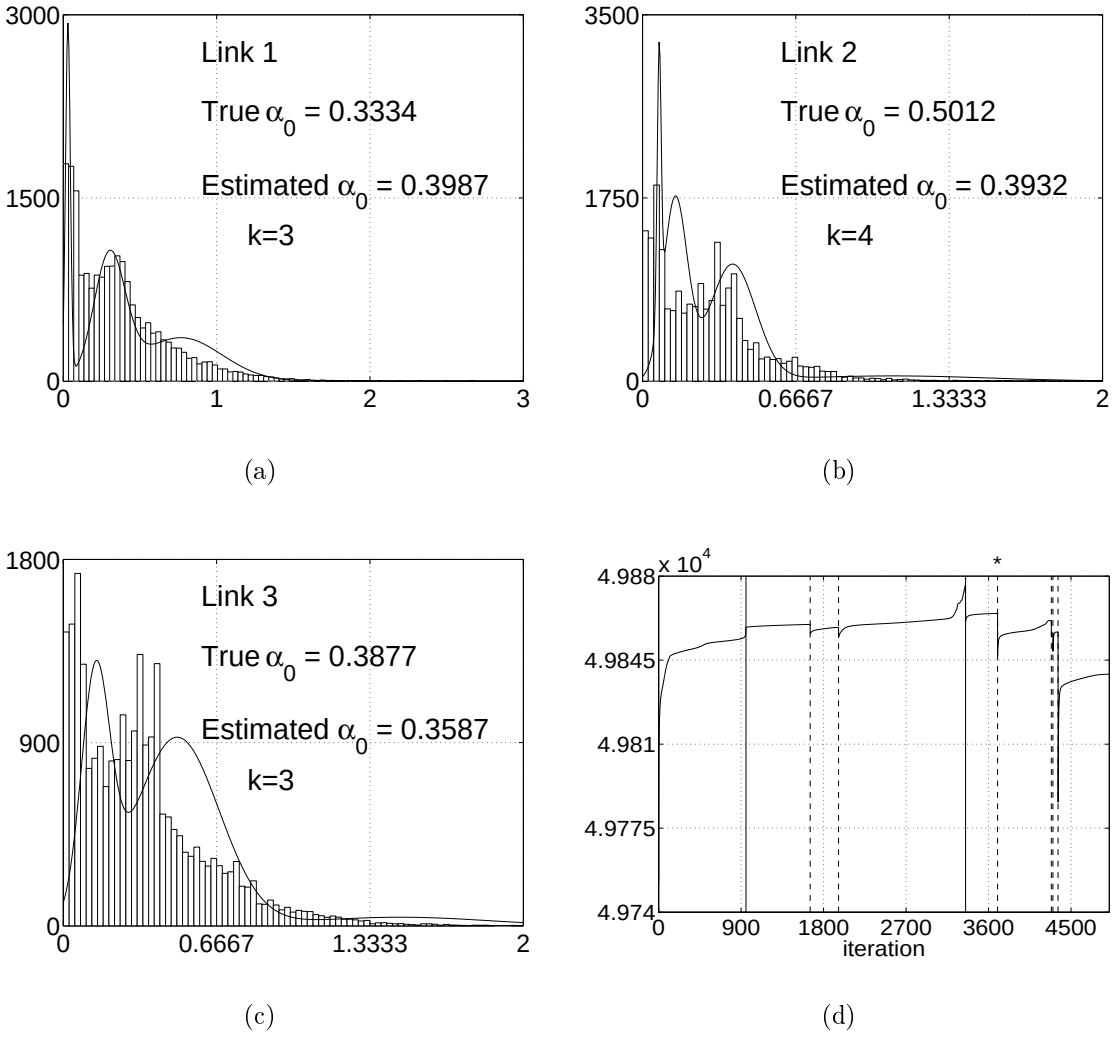


Figure 4.11: Results from the second level $d = 1$ of the accelerated algorithm using probe tree 3. (a) - (c) Normalized ns-derived histograms for non-zero link delays and estimated Gaussian mixture density for indicated (super)links. The horizontal axes denote link packet delays in milliseconds. (d) shows the convergence curve of the MML penalized likelihood function. The solid vertical lines denote the iteration numbers where the number of Gaussian mixture components is reduced by component annihilation in the EM algorithm. The dashed vertical lines show the convergent iterations where the component with the least mixture probability over all the links is removed and the algorithm is restarted with the remained components. The links $\{l_i(3)\}$ affected by this reduction are indexed by {1,2,3,1,3,2,2,1} in iterations {953,1654,1963,3349,3698,4285,4302,4358}, respectively. The asterisk indicates the iteration which converges to the highest penalized likelihood.

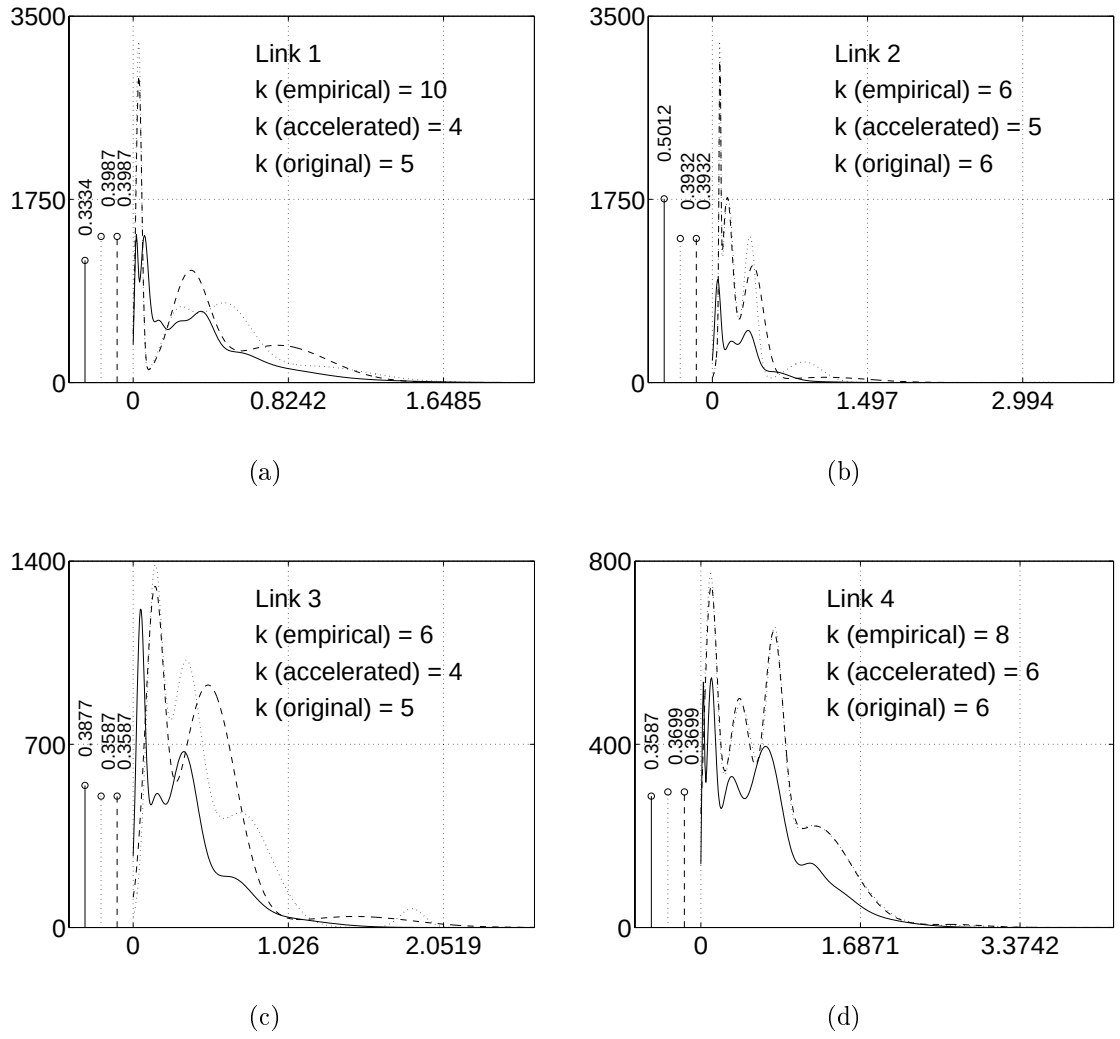
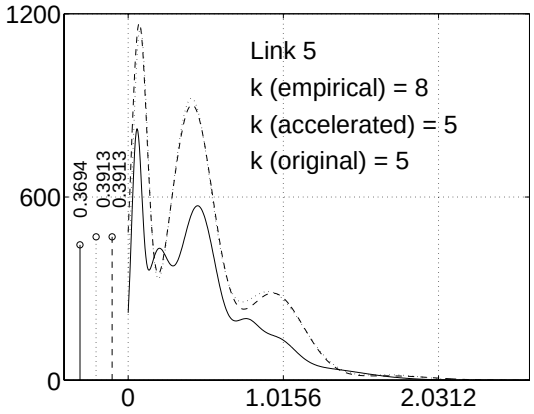
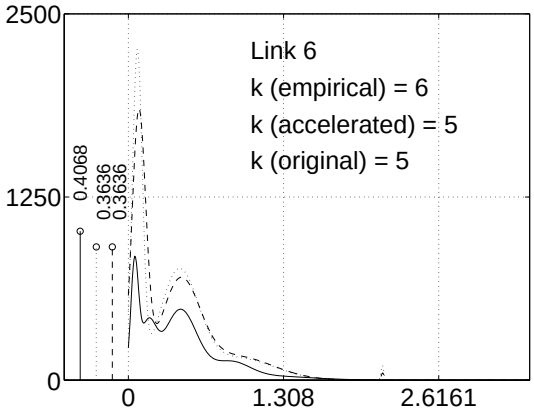


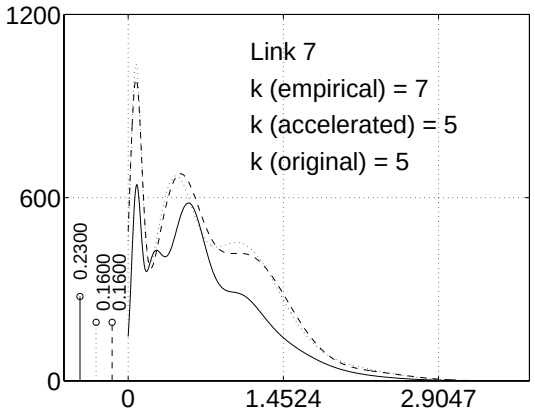
Figure 4.12: (a) - (g) Comparison of empirically estimated hybrid mixture link delay distributions with k(empirical) components (solid) to the estimates with k(accelerated) components obtained from the accelerated algorithm (dashed) and the estimates with k(original) components obtained from the original algorithm (dotted). The horizontal axes denote link packet delays in milliseconds. L_1 error norms between the tomographic and empirical estimates are shown in (h) for the accelerated (solid) and original (dashed) algorithms.



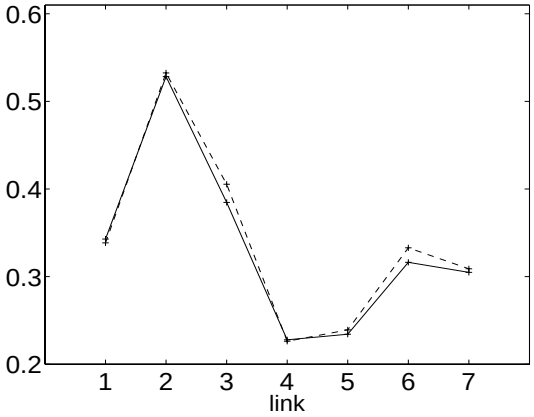
(e)



(f)



(g)



(h)

Figure 4.12 (continued)

a bottom-up approach to divide the estimation problem into subproblems where shared probe path delays have model sizes in the order of that for a single link. The analysis proves that the complexity of the algorithm can be reduced to a less exponential base with respect to the network size. The `ns-2` simulation results show that the accelerated algorithm achieves the same level of estimation accuracy as the original algorithm and reduces by approximately 40% the runtime complexity.

A possible future direction is to reduce the delay model orders by using more sophisticated mixtures which use combinations of different families of probability densities. For example, exponential densities are more efficient to describe exponential tails of the distributions which correspond to rare large delay events in a lightly or moderately loaded network. For heavily loaded situations some heavy tailed functions, such as Pareto densities, might be used. The reduction in model orders directly diminish the exponential rate constant in the computational complexity for the algorithm and further reduction would attain further acceleration at the possible expense of additional fitting error. Another issue, which we have not addressed in either Chapter 3 or 4 is the estimation of the minimum packet delay, which has been assumed known. This would not be difficult, indeed it is a simple modification of the degrees of freedom of the point mass estimated via sample averages to initialize the EM algorithm. One could include the additional unknown parameters into the EM iterations but this unknown parameter could affect the convergence rate of the EM algorithm.

CHAPTER 5

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

5.1 Introduction

The infrastructure of a packet network is composed of switching devices (as nodes) and communication channels (as links). Packet routes across the network provide topology information. Tools such as `traceroute` can 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 diffcult 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. Several method of solution have been investigated using *network tomography* approaches [28, 29, 30, 31, 32, 49].

In this chapter we propose a general method for estimation of unicast network topologies. As in [30] and [39] we focus on estimation of the logical tree structure of the network. Unlike in 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 considered as a single-node cluster. The clusters for sibling nodes, i.e., nodes having the same parent, define a partition into cluster of the set of leaf nodes specified by their parent. Given a partition we call a pair of leaf nodes an *intra-cluster* if each node in the pair is in the same cluster, otherwise they are an called *inter-cluster*.

The similarity of a pair of leaf nodes can be represented by a metric function associated with the path from the root node to the nearest ancestor of the two leaf nodes. Our algorithm can be applied to any probing scheme that produces accurate estimates of a discriminating similarity metric. We apply our clustering algorithm to three different types of similarity estimates obtained from end-to-end measurements: delay difference using sandwich probes, delay variance using packet pairs, loss rate also using packet pairs. We modify the likelihood model for the pair-wise similarities proposed in [34] to include a prior distribution on the nearest common ancestor node of each pair of the leaf nodes. This results in a finite mixture model with every 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. The PML-EM algorithm developed here is similar to the mixture model PML-EM algorithm developed for delay tomography in Chapter 3. However, while the algorithm of Figueiredo and Jain was applicable to delay tomography, the MML penalty of [48] becomes degenerate for our formulation of the topology estimation problem. Therefore a different type of penalty, introduced by Oliver *et al.* in [89], is adopted here.

Given the similarity mixture model for the set of leaf nodes specified by an internal node, the component with the smallest mean is exactly supported by the inter-cluster pairs of leaf nodes for the partition derived from the children of the internal node. 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 specified 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 the leaf nodes into clusters based on a simple rule. For each node we use the inter-cluster component to determine the set of the remaining nodes which are in a different cluster from that node. Then we simply cluster the leaf nodes which correspond to identical sets. We propose 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 other ones 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 previous algorithms proposed in [30, 32]: (1) our use of hierarchical topology likelihood with

finite mixture models and MML model order penalty; (2) our top-down recursive partitioning of the leaf nodes using graph-based clustering and unsupervised learning of the finite mixture models; (3) our intelligent search of the partition likelihood surface using the graph clustering procedures. Our algorithm has the advantage of direct estimation of the general logical tree structure without the need for any heuristic thresholds and it yields unbiased estimates of topology. It also avoids the complexity introduced by Monte-Carlo methods as in [32]. The performance of our algorithm is compared with the previous ones using `matlab` model simulation. 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 [101, 102, 103, 104]. These results are obtained under a wide range of conditions on the magnitudes and variances of the similarity estimates.

The three candidate probing schemes are evaluated on a `ns` simulated network using our algorithm. The Monte-Carlo simulation shows the delay difference measured by the sandwich probes have the best performance when the network load is light. For a moderate load situation the delay variance using packet pair probes provides the most reliable estimates for the leaf node similarities. When the network is congested with heavy traffic the loss rate measured by packet pair probes generates topology estimates with the lowest error as measured by graph edit distance. We also use graph edit distance to define the distribution of topology estimates and illustrate it with a network simulated in `ns`.

Our contribution in this chapter is summarized as follows: (1) We suggest a finite mixture model for end-to-end similarity measurements of the leaf nodes in a logical tree network, and develop a PML-EM estimation algorithm using MML-type model order penalty; (2) We propose a hierarchical topology likelihood for logical

tree networks based on finite mixture models of end-to-end measurements; (3) We develop a general unsupervised hierarchical estimation algorithm over logical tree network topologies using end-to-end measurements; (4) We introduce packet pair probes in network topology discovery and suggest two types of similarity metric measured by packet pairs. We compare the performance of the suggested packet pair probing schemes with that using sandwich probes through `ns` simulation under various network load conditions. Conclusions on the best performance scenario for each probing scheme is provided; (5) Through model simulation we demonstrate that our algorithm outperforms the DBT and LBT algorithms because our algorithm adopts a less greedy approach to find the optimal topology; (6) For Monte-Carlo experiments on network topology discovery, we define the median topology and the distribution of topology estimates using graph edit distance.

This chapter is organized as follows. In Section 5.2 we set up the logical tree network model. The probing methods and their associated similarity metrics are also introduced. In Section 5.3 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 5.4 we formulate the PML-EM algorithm to estimate the proposed finite mixture model. A *hierarchical topology estimation algorithm* (HTE) is then developed and the partition schemes are illustrated. In Section 5.5 we conduct comprehensive simulations in `matlab` and `ns-2` to evaluate the performance of the proposed algorithm with different probing methods and network environments. We also compare our algorithm with the algorithms in [30, 34] based on binary trees. Section 5.6 provides the conclusion and discusses future work.

5.2 Background

5.2.1 Problem Formulation

Our work focuses on the problem of estimating the *logical tree* network structure 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 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 a virtual link. Such devices are not identifiable from the end-to-end measurements. 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 its parent. The topology estimation problem is illustrated in Figure 5.1, where the right part shows an example of a logical tree.

The topology estimation problem can be formulated as *hierarchical clustering* of the leaf nodes in which each group of nodes may be recursively partitioned into subgroups [32, 49]. The corresponding tree is known as a *dendrogram*. Each internal

node in a dendrogram specifies a cluster of leaf nodes which share the internal node as their common ancester. The clusters for sibling nodes, i.e., nodes having the same parent, define a partition of the set of leaf nodes specified by their parent node. Each leaf node itself is also a cluster, called a *trivial cluster*. For example, the children of node 7 in Figure 5.1 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} corresponding to node 9.

Hierarchical clustering has been widely studied in many areas such as database classification and machine learning [73, 74, 75, 76, 77]. 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 other (sub)clusters. Suppose the similarity between each pair of leaf nodes can be quantified 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 $\gamma_{i,i} = \infty$, indicating 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. Figure 5.3 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 similarities. Define $\mathbf{C}$ as a hierarchical clustering of the leaf nodes, i.e., a set of groups of these 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 $T_s(\mathbf{C})$

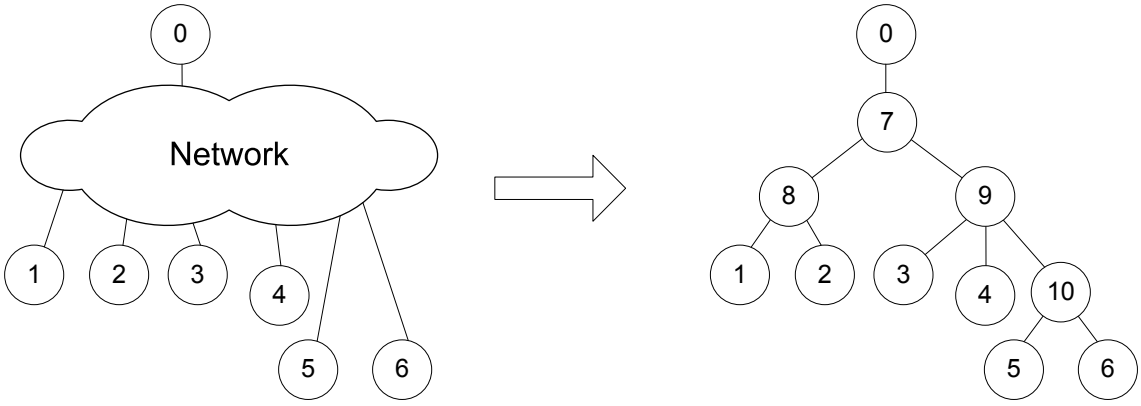


Figure 5.1: Illustration of the topology estimation problem. The logical tree (right) has $\mathbf{V}_r = \{1, 2, 3, 4, 5, 6\}$, $\mathbf{V}_i = \{7, 8, 9, 10\}$, and $\mathbf{E} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, where the links are numbered after their child end nodes.

corresponds to the top-level partition in $\mathbf{C}$, and is associated with the set of all inter-cluster similarities of 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 that set are between two trivial clusters, otherwise it is called *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 the parent node of $T_s(\mathbf{C})$. Figure 5.2 shows the hierarchical clustering $\mathbf{C}$ for the leaf nodes in Figure 5.1 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 a hierarchical clustering to a tree graph, which means that finding the similarity grouping is sufficient to determine the hierarchical clustering of the leaf nodes. This property will be the key to the development of our algorithm.

In topology estimation, the concept of *Metric-Induced Network Topology* (MINT)

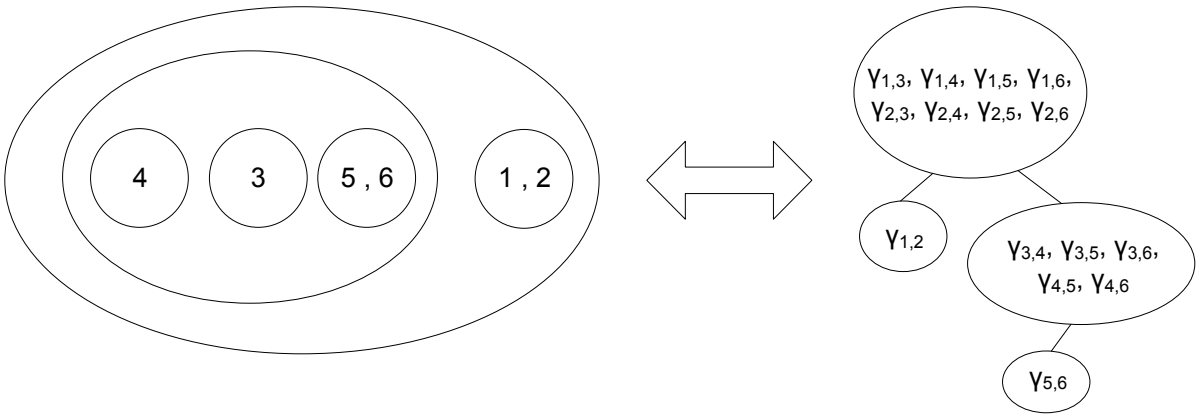
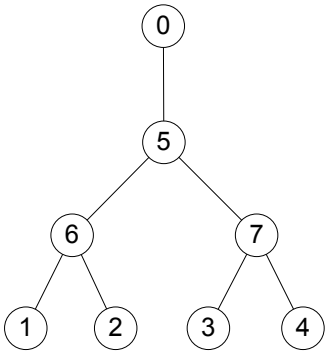


Figure 5.2: The hierarchical clustering $\mathbf{C}$ of the leaf nodes in Figure 5.1 (left) and the corresponding similarity clustering tree $T_s(\mathbf{C})$ (right).

introduced by Bestavros *et al.* [33] 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 being a descendant of i, which is expressed as the union of the links intersecting the path. To simplify the notation we let $p_i = p_{0,i}$ for $i \in \mathbf{V} \setminus \{0\}$. Let $a(i,j)$ be the nearest common ancestor of leaf node i and j. We know that 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 framework [34, 39]. 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 [33, 39]:

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



i / j	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 5.3: 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.

$\mathbf{V}_r$ 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.

5.2.2 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. We also assume the following statistical properties on the network environment:

A1) **Spatial Independence**. The delays of a packet over different links of its path are independent. The delays of any two packets on different links are also independent.

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

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 tree formed by the union of path p_i and p_j from root to nodes i and j, respectively, 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 5.4 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. [16, 23, 24, 58].

Sandwich Probes

Sandwich probes were invented by Castro *et al.* in [34] 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 5.4 the small packets A and C are sent to node 5 and the large packet B is sent to node 3. According to [34] 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 queue behind the large packet until they seperate from each other at node $a(i,j)$, 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 5.4, 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 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

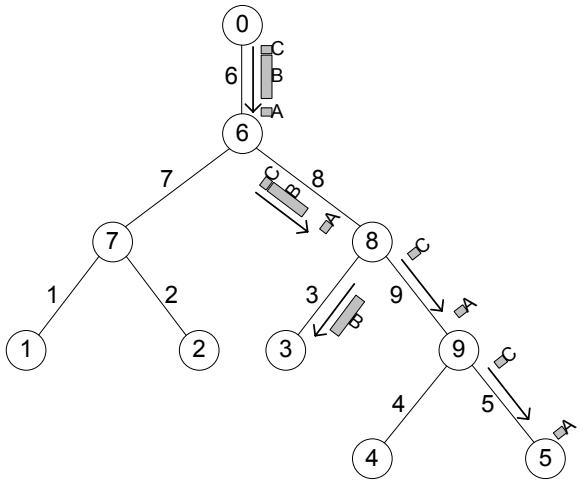


Figure 5.4: A sandwich probe example. The small packets A and C are sent to node 5 and the large packet B is sent to node 3. The probe tree $t_{3,5}$ is defined by the routes of the probe packets, which consists of links 3, 5, 6, 8, and 9. The metric $\gamma_{3,5}$ is the end-to-end delay difference between A and C.

in a saturated network where the link queues remain nearly full all the time (when it is assumed that 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.

Packet Pair Probes

Packet pair techniques have been previously applied to flow control and bottleneck bandwidth estimation in networks [14, 78, 79]. Recently packet pairs were also used to perform in network loss and delay tomography [9, 23, 24]. 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 5.5

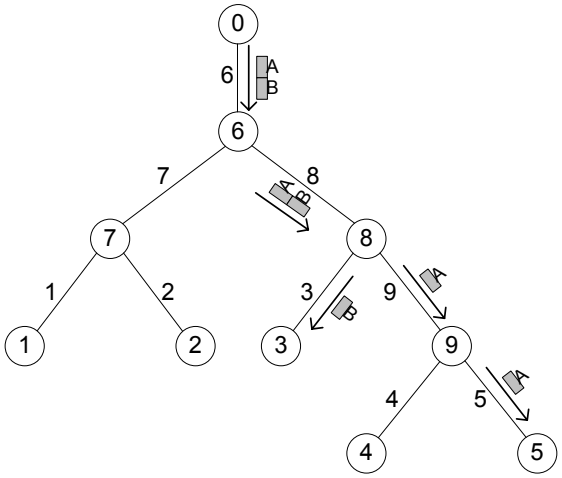


Figure 5.5: A packet pair probe example. Packet A and B are sent back-to-back from the root node to node 5 and 3, respectively. We assume they stay together and encounter identical delays on shared path which includes link 6 and 8.

depicts an example of packet pair probe. The main assumption justifying the use of packet pair probes is the following:

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 [17]. Suppose the packet pair probes are sent through the probe tree $t_{i,j}$. Let the sum of the queueing delay of the packet sent from root 0 to node i on shared path $p_{a(i,j)}$ be $d_{0,i} = d_{0,a(i,j)} + d_{a(i,j),i}$, and the packet sent to node j be $d_{0,j} = d_{0,a(i,j)} + d_{a(i,j),j}$. The random ordering of the two packets makes $d_{0,i}$ and $d_{0,j}$ have approximately the same mean, i.e., the mean of $d_{0,i} - d_{0,j} = 0$. Under the stated statistical assumptions the difference $d_{0,i} - d_{0,j}$ 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 non-decreasing with respect to the number of links in the path since variances cannot be negative. The consistency property can be achieved if the delay variance over $p_{a(i,j)}$ can be calculated from the end-to-end delays of the 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 5.6.

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

$$\hat{s}_n^{(i,j)} = \frac{\sum_{n_0=(n-1)N_{cov}+1}^{nN_{cov}} \left(y_{1,n_0}^{(i,j)} - \bar{y}_{1,n}^{(i,j)}\right)\left(y_{2,n_0}^{(i,j)} - \bar{y}_{2,n}^{(i,j)}\right)}{N_{cov} - 1}, \tag{5.1}$$

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 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

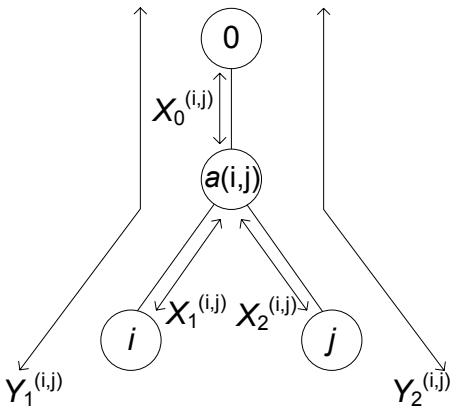


Figure 5.6: A *logical probe tree* $t_{i,j}$ is shown to illustrate the computation of the delay variance over $p_{a(i,j)}$ from end-to-end delays. Due to the independence of $X_0^{(i,j)}$, $X_1^{(i,j)}$, and $X_2^{(i,j)}$, $Var(X_0^{(i,j)}) = Cov(Y_1^{(i,j)}, Y_2^{(i,j)})$.

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 size of the queueing delay variances 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 $p_{a(i,j)}$. 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 and our methods can be extended to this scenario once a suitable similarity metric has been identified.

Figure 5.7 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:

$$\begin{aligned} u_0^{(i,j)} &= P(\{Y_{1,n}^{(i,j)} < \infty\} \wedge \{Y_{2,n}^{(i,j)} < \infty\}) \\ &= (1-q_0^{(i,j)})(1-q_1^{(i,j)})(1-q_2^{(i,j)}) \\ u_1^{(i,j)} &= P(\{Y_{1,n}^{(i,j)} = \infty\} \wedge \{Y_{2,n}^{(i,j)} < \infty\}) \\ &= (1-q_0^{(i,j)})q_1^{(i,j)}(1-q_2^{(i,j)}) \\ u_2^{(i,j)} &= P(\{Y_{1,n}^{(i,j)} < \infty\} \wedge \{Y_{2,n}^{(i,j)} = \infty\}) \\ &= (1-q_0^{(i,j)})(1-q_1^{(i,j)})q_2^{(i,j)}. \end{aligned} \tag{5.2}$$

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 $p_{i,j}$ can be estimated by

$$\hat{q}_0^{(i,j)} = 1 - \frac{(\hat{u}_0^{(i,j)} + \hat{u}_1^{(i,j)})(\hat{u}_0^{(i,j)} + \hat{u}_2^{(i,j)})}{\hat{u}_0^{(i,j)}}, \qquad \text{for } \hat{u}_0^{(i,j)} \neq 0. \tag{5.3}$$

Eqn. (5.3) implies the consistency property for the loss metric, and the monotonicity property holds because the loss rate never decreases as the number of links in the path increases. Similar to the covariance estimator in (5.1), one needs to collect

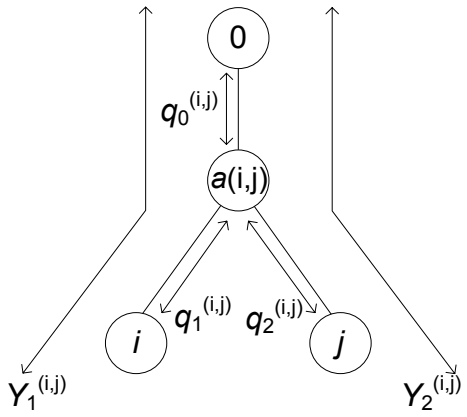


Figure 5.7: A *logical probe tree* $t_{i,j}$ is shown to illustrate the estimation of the packet loss rate over $p_{a(i,j)}$ from end-to-end loss rates.

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.

5.3 Hierarchical Topology Likelihood Using Finite Mixture Models

5.3.1 Finite Mixture Model for Similarity Estimation

Modelling the behavior and dynamics of today's network has been an active research area for decades [80, 81, 82, 83, 84]. 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., [85, 86, 87]). 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 [5, 6, 9].

To establish a simple and unified framework for both delay and packet loss metrics, 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 (5.1), or packet loss rate in (5.3), are i.i.d according to A1) and A2). If we average every N_{norm} estimates $\hat{\gamma}_n^{(i,j)}$, $n = 1, \ldots, N_{norm}$, the result will be approximately Gaussian distributed when N_{norm} is large, according to the Central Limit Theorem (C.L.T.). We call such sample *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 5.1. 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 5.1 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 $T(v) = \{t_{i,j} : a(i,j) = v, i < j, \quad i,j \in \mathbf{V}_r\}$ as the set of probe trees whose branches split at node 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 5.1 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

$$f_S(\mathbf{\Gamma}) = \prod_{v \in \mathbf{V}_i} \phi(\mathbf{\Gamma}(v); \theta_v), \tag{5.4}$$

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 5.1, $f_S(\mathbf{\Gamma}) = \phi(\mathbf{\Gamma}(7); \theta_7)\phi(\mathbf{\Gamma}(8); \theta_8)\phi(\mathbf{\Gamma}(9); \theta_9)\phi(\mathbf{\Gamma}(10); \theta_{10})$, 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 $\mathbf{\Gamma}(10) = \{\mathbf{\Gamma}_{5,6}\}$. (5.4) is similar to the model used in [32]. 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 (5.4) 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 [32].

Herein we provide an alternative model which will be used later to develop an unsupervised estimation algorithm. Suppose there is no knowledge of the common parent node $a(i,j)$ for any pair of leaf nodes (i,j), but the number of probe trees in $T(v)$ is known. Let $N_T(v) = |T(v)|$. 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., [47]). 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

$$f_{FM}(\mathbf{\Gamma}) = \prod_{\substack{i,j \in \mathbf{V}_r \\ i<j}} \sum_{v \in \mathbf{V}_i} \alpha(v)\phi(\mathbf{\Gamma}_{i,j}; \theta_v). \tag{5.5}$$

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 example, in Figure 5.1, using the mixture model gives $f_{FM}(\mathbf{\Gamma}) = \prod_{i,j \in \{1,\ldots,6\}, i<j} \left[\frac{8}{15}\phi(\mathbf{\Gamma}_{i,j}; \theta_7) + \frac{1}{15}\phi(\mathbf{\Gamma}_{i,j}; \theta_8) + \frac{5}{15}\phi(\mathbf{\Gamma}_{i,j}; \theta_9) + \frac{1}{15}\phi(\mathbf{\Gamma}_{i,j}; \theta_{10})\right]$. The key difference between the models in (5.4) and (5.6) 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 [48, 89]. To discover the topology, the similarity sets in the $T_s(\mathbf{C})$ are determined by associating each $a(i,j)$ with the component that the probability of $\mathbf{\Gamma}_{i,j}$ being contributed by the component is the maximum over all the components in f_{FM}.

5.3.2 The MML Penalized Likelihood for The Mixture Model

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

$$f_{FM}(\mathbf{\Gamma};\mathbf{\Theta}) = \prod_{\substack{i,j\in\mathbf{V}_r \\ i<j}} \sum_{m=1}^{k} \alpha_m \phi(\mathbf{\Gamma}_{i,j};\theta_m), \tag{5.6}$$

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 $0 \leq \alpha_m \leq 1, \sum_{m=1}^{k} \alpha_m = 1$. For a given model order k which also denotes the number of internal nodes, the *unobserved data* in our case is the $\{a(i,j)\}$, which indicates the contributing component for $\mathbf{\Gamma}_{i,j}$. 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 [48, 63] 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 [91]. We adopt a criterion called Minimum Message Length (MML) [66] to derive the penalty function. MML has been widely used in unsupervised learning of mixture models [24, 48, 89, 90] and was adopted in Chapter 3 of this thesis. The incomplete data

penalized log-likelihood is generally expressed as [48]

$$\tilde{\mathcal{L}}(\mathbf{Y};\boldsymbol{\Theta}) \stackrel{\text{def}}{=} \log f(\boldsymbol{\Theta}) + \log f(\mathbf{Y}|\boldsymbol{\Theta}) - \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\Theta})| - \frac{c}{2}(1+\log\kappa_c), \tag{5.7}$$

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

As one may suggest the unsupervised algorithm of Figueiredo and Jain in [48] for the estimation of finite mixture models as in Chapter 3 and 4, the MML penalty could become degenerate in our mixture model formulation for the topology. For a component in a finite mixture model, we define its support from the realizations as the sum of the conditional probability for each sample being contributed by the component given the value of the sample. Equation (17) in [48] indicates that a component in a Gaussian mixture model survives in the M-step of the PML-EM algorithm if and only if its support from the realizations is at least 1. Compared with our model in (5.6) it implies that for an internal node v if there exists only one pair of leaf nodes (i,j) such that $a(i,j)=v$, the mixture component corresponding to v could be easily annihilated by the PML-EM algorithm and node v could become unidentifiable from the mixture model estimate. Therefore we adopt a different MML-type approach for the PML-EM algorithm introduced by Oliver *et al.* in [89], which is predated to [48].

Our choice for the prior distributions of the parameters follows the approach of [89]. The mixing probabilities in $\boldsymbol{\alpha}$ have a uniform prior:

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

The prior for σ_m is uniform between 0 and σ_p, where σ_p is the standard deviation of the entire population $\mathbf{\Gamma}$:

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

We also take a uniform prior for μ_m distributed within one standard deviation σ_p of μ_p, where μ_p is the mean of the population $\mathbf{\Gamma}$:

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

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

$$f(\mathbf{\Theta}) = \frac{(k-1)!}{2^k \sigma_p^{2k}}. \tag{5.8}$$

In general it is difficult to derive a closed form for the FIM of finite mixture models with more than one component. The aughors of [89] suggested to replace the determinant of the FIM by the product of the the determinant of the FIM for each component times the FIM determinant for the mixing probabilities. Hence

$$|\mathbf{I}_{FM}(\mathbf{\Theta})| \approx |\mathbf{I}_0(\boldsymbol{\alpha})| \times \prod_{m=1}^{k} |\mathbf{I}_m(\theta_m)|, \tag{5.9}$$

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 $\theta_m = (\mu_m, \sigma_m^2)$. $\mathbf{I}_m(\theta_m)$ can be expressed as

$$\mathbf{I}_m(\theta_m) = \sum_{\substack{i,j \in \mathbf{V}_r \\ i<j}} \mathbf{I}_m^{(i,j)}(\theta_m), \tag{5.10}$$

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

$$\mathbf{I}_m^{(i,j)}(\theta_m) = \begin{bmatrix} \frac{\alpha_m N_{i,j}}{\sigma_m^2} & 0 \\ 0 & \frac{2\alpha_m N_{i,j}}{\sigma_m^2} \end{bmatrix}. \tag{5.11}$$

Therefore

$$|\mathbf{I}_m(\theta_m)| = \frac{2\alpha_m^2 N^2}{\sigma_m^4}. \tag{5.12}$$

To determine the FIM for $\boldsymbol{\alpha}$ one can view the $\boldsymbol{\alpha}$ as being the parameters of a multinomial distribution having N_T i.i.d. realizations (selecting N_T $a(i,j)$'s from k internal nodes), and hence

$$|\mathbf{I}_0(\boldsymbol{\alpha})| = \frac{N_T}{\prod_{m=1}^k \alpha_m}. \tag{5.13}$$

We do not require a particular ordering on the components, so the corresponding factorial term in the MML expression (5.7) 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 [89]. Substituting (5.8)-(5.13) into (5.7) we have

$$\begin{aligned} \mathcal{L}_p(\mathbf{\Gamma}; \mathbf{\Theta}) &= \log f_{FM}(\mathbf{\Gamma}; \mathbf{\Theta}) + \log \frac{(k-1)!}{2^k \sigma_p^{2k}} + \log(k!) - \frac{1}{2}\log T - k\left(\log\sqrt{2} + \log N\right) \\ &\quad -\frac{1}{2}\sum_{m=1}^k \log \alpha_m + \sum_{m=1}^k \log \sigma_m^2 - \frac{3k}{2}(1 - \log 12). \end{aligned} \tag{5.14}$$

The details of the maximum likelihood estimation algorithm maximizing (5.14) over

$\boldsymbol{\Theta}$ will be provided in the next section.

5.3.3 The Hierarchical Topology Likelihood

Eqn. (5.14) is difficult to use directly for topology estimation due to identifiability problems even with the overmodelling 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 5.1 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 5.1. Hence we are not able to find the correct set of the internal nodes because the two mixture components merge to a single component. 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 $\boldsymbol{\gamma}(\mathbf{G}) = \{\gamma_{i,j} : i < j, \quad i,j \in \mathbf{G}\}$ be the set of pair-wise similarity metrics for $\mathbf{G}$, and $\boldsymbol{\Gamma}(\mathbf{G}) = \{\boldsymbol{\Gamma}_{i,j} : i < j, \quad i,j \in \mathbf{G}\}$ be the normalized samples of $\boldsymbol{\gamma}(\mathbf{G})$. Let $\mathbf{K} = \{K_1, \ldots, K_D\}$ be a partition 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 $\boldsymbol{\gamma}$. Hence the set of all the 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, $l \in \{1, \ldots, D'\}$. The set of intra-cluster $\mathbf{\Gamma}_{i,j}$'s in K_l, denoted by $\mathbf{\Gamma}_l(\mathbf{K})$, itself follows a finite mixture model. Let $\mathbf{\Theta}_l(\mathbf{K})$ be the mixture parameter set for $f_{FM}(\mathbf{\Gamma}_l(\mathbf{K}))$. If K_l has exactly two leaf nodes or all the subclusters in K_l are trivial, the finite mixture model degenerates to a single component. We define the penalized partition likelihood as:

$$\mathcal{L}_k(\mathbf{\Gamma}(\mathbf{G}); \mathbf{K}, \mathbf{\Theta}(\mathbf{K})) \stackrel{\text{def}}{=} \mathcal{L}_p(\mathbf{\Gamma}_0(\mathbf{K}); \mathbf{\Theta}_0(\mathbf{K})) + \sum_{l=1}^{D'} \mathcal{L}_p(\mathbf{\Gamma}_l(\mathbf{K}); \mathbf{\Theta}_l(\mathbf{K})), \tag{5.15}$$

where $\mathbf{\Theta}(\mathbf{K}) = (\mathbf{\Theta}_0(\mathbf{K}), \ldots, \mathbf{\Theta}_{D'}(\mathbf{K}))$. For example, the partition specified by the children of node 7 in Figure 5.1 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 $\mathbf{\Gamma}(\mathbf{G}_1) = \mathbf{\Gamma}$, $\mathbf{\Gamma}_0(\mathbf{K}_1) = \mathbf{\Gamma}(7)$, $\mathbf{\Gamma}_1(\mathbf{K}_1) = \mathbf{\Gamma}(8)$, and $\mathbf{\Gamma}_2(\mathbf{K}_1) = \mathbf{\Gamma}(9) \cup \mathbf{\Gamma}(10)$. Hence

$$\begin{aligned} \mathcal{L}_k(\mathbf{\Gamma}(\mathbf{G}_1); \mathbf{K}_1, \mathbf{\Theta}(\mathbf{K}_1)) \;\; = \;\; & \mathcal{L}_p(\mathbf{\Gamma}_0(\mathbf{K}_1); \mathbf{\Theta}_0(\mathbf{K}_1)) + \mathcal{L}_p(\mathbf{\Gamma}_1(\mathbf{K}_1); \mathbf{\Theta}_1(\mathbf{K}_1)) + \\ & \mathcal{L}_p(\mathbf{\Gamma}_2(\mathbf{K}_1); \mathbf{\Theta}_2(\mathbf{K}_1)). \end{aligned}$$

Similarly, for the cluster $K_{1,2}$ in $\mathbf{K}_1$ 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

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

where $\mathbf{G}_2 = \{3, 4, 5, 6\}$, $\mathbf{\Gamma}(\mathbf{G}_2) = \mathbf{\Gamma}_2(\mathbf{K}_1)$, $\mathbf{\Gamma}_0(\mathbf{K}_2) = \mathbf{\Gamma}(9)$, and $\mathbf{\Gamma}_1(\mathbf{K}_2) = \mathbf{\Gamma}(10)$. Recall that the similarity sets associated with a similarity clustering tree $T_s(\mathbf{C})$ are sets of inter-(sub)cluster similarities. One can easily find that the normalized samples for non-trival similarity sets $\mathbf{\Gamma}_0(\mathbf{K}_1)$, $\mathbf{\Gamma}_0(\mathbf{K}_2)$ and for trivial similarity sets

$\mathbf{\Gamma}_1(\mathbf{K}_1)$, $\mathbf{\Gamma}_1(\mathbf{K}_2)$ have a one-to-one mapping from the similarity sets associated with the $T_s(\mathbf{C})$ in Figure 5.2. This implies the topology of the logical tree can be obtained by hierarchical estimation of inter-cluster similarity set using inter-cluster mixture components.

This example also motivates the following *hierarchical topology likelihood* for the logical tree in Figure 5.1:

$$\mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T)) = \mathcal{L}_k(\mathbf{\Gamma}(\mathbf{G}_1); \mathbf{K}_1, \mathbf{\Theta}(\mathbf{K}_1)) + \mathcal{L}_k(\mathbf{\Gamma}(\mathbf{G}_2); \mathbf{K}_2, \mathbf{\Theta}(\mathbf{K}_2)|\mathbf{K}_1), \qquad (5.16)$$

where $\mathbf{\Theta}(T) = \{\mathbf{\Theta}(\mathbf{K}_1), \mathbf{\Theta}(\mathbf{K}_2)\}$. Comparing (5.16) with the similarity clustering tree $T_s(\mathbf{C})$ in Figure 5.2 we find $\mathcal{L}_T$ recursively sums up the partition likelihoods corresponding to the non-trival similarity sets. For a node v_s in $T_s(\mathbf{C})$ define $\boldsymbol{\gamma}_s(v_s)$ as the similarity set associated with v_s, $\mathbf{G}_s(v_s)$ as the set of leaf nodes involved in $\boldsymbol{\gamma}_s(v_s)$, $\mathbf{K}_s(v_s)$ as the partition of $\mathbf{G}_s(v_s)$ derived by v_s, and $\mathbf{\Gamma}_s(v_s)$ as the corresponding set of $\mathbf{\Gamma}_{i,j}$'s in $\mathbf{G}_s(v_s)$. Also let $v_1 \succ v_2$ denote "v_1 is a child node of v_2". The construction of the hierarchical topology likelihood $\mathcal{L}_T$ is summarized by the procedure in Table 5.1.

The construction of $\mathcal{L}_T$ mimics exactly the construction of the similarity clustering tree. 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 $\mathbf{K}_s(v_s)$ is uniquely determined by the inter-cluster similarity set $\boldsymbol{\gamma}_s(v_s)$, which can be inferred by the inter-clustering component in $f_{FM}(\mathbf{\Gamma}(\mathbf{G}_s(v_s)))$ because that component is supported by the estimates of $\boldsymbol{\gamma}_s(v_s)$. The details of the clustering algorithm will be provided in the next section. Before

Table 5.1: Definition procedure for the hierarchical topology likelihood.

Input: $T = \{\mathbf{V}, \mathbf{E}\}$, $\mathbf{\Theta}(T)$, $\mathbf{\Gamma}$
Output: $\mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T))$

$\mathbf{C} \leftarrow$ the hierarchical clustering of $\mathbf{V}_r$ according to T
$T_s(\mathbf{C}) \leftarrow$ the similarity clustering tree for $\mathbf{C}$
$v_r \leftarrow$ the root node of $T_s(\mathbf{C})$
$\mathbf{S}' \leftarrow \{v_r\}$
$\mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T)) \leftarrow 0$
while $(\mathbf{S} \neq \phi)$
    $\mathbf{S} \leftarrow \mathbf{S}'$
    $\mathbf{S}' \leftarrow \phi$
    for all $v_s \in \mathbf{S}$ do
        $\mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T)) \leftarrow \mathcal{L}_T(\mathbf{\Gamma}; T, \mathbf{\Theta}(T)) +$
            $\mathcal{L}_k(\mathbf{\Gamma}(\mathbf{G}_s(v_s)); \mathbf{K}_s(v_s), \mathbf{\Theta}(\mathbf{K}_s(v_s)))$
        for all $\tilde{v}_s \succ v_s$ do
            if $(|\boldsymbol{\gamma}_s(\tilde{v}_s)| > 1)$
                $\mathbf{S}' \leftarrow \mathbf{S}' \cup \{\tilde{v}_s\}$
            end if
        end for
    end for
end while

explaining the algorithm we would like to make a comment about our model. So far the model is restricted to Gaussian components because of the approximation using C.L.T. In fact, (5.5) also holds for non-Gaussian ϕ_v's as long as $\bar{\gamma}^{(i,j)}$ and $\bar{\gamma}^{(k,l)}$ are approximately equal in distribution when $a(i,j) = a(k,l)$. In that case our model may be extended accordingly, which may be useful when new probing and/or approximation methods are proposed in the future.

5.4 Topology Estimation Algorithm

5.4.1 Estimation of The Finite Mixture Model

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

$$\mathcal{L}_c(\mathbf{U};\mathbf{\Theta}) = \sum_{\substack{i,j\in\mathbf{G}\\ i<j}} \sum_{m=1}^{k} Z_m^{(i,j)} \left[\log\alpha_m + \sum_{l=1}^{N_{i,j}} \log\phi(\bar{\gamma}_l^{(i,j)};\theta_m) \right]. \tag{5.17}$$

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

$$\begin{aligned} Q(\mathbf{\Theta};\hat{\mathbf{\Theta}}^{(t)}) &= E\left[\mathcal{L}_c(\mathbf{U};\mathbf{\Theta})|\mathbf{\Gamma}=\mathbf{g};\hat{\mathbf{\Theta}}^{(t)}\right] \\ &= \sum_{\substack{i,j\in\mathbf{G}\\ i<j}} \sum_{m=1}^{k} \left[\omega_m^{(i,j)} \log\alpha_m + \sum_{l=1}^{N_{i,j}} Q_{m,l}^{(i,j)}(\theta_m) \right], \end{aligned} \tag{5.18}$$

where

$$\begin{aligned}
\omega_m^{(i,j)} &= E\left[Z_m^{(i,j)} | \mathbf{\Gamma} = \mathbf{g}; \hat{\mathbf{\Theta}}^{(t)}\right] \\
&= \frac{P(Z_m^{(i,j)} = 1, \mathbf{\Gamma}_{i,j} = \mathbf{g}_{i,j}; \hat{\mathbf{\Theta}}^{(t)})}{P(\mathbf{\Gamma}_{i,j} = \mathbf{g}_{i,j}; \hat{\mathbf{\Theta}}^{(t)})} \\
&= \frac{\hat{\alpha}_m^{(t)} \prod_{l=1}^{N_{i,j}} \phi(g_l^{(i,j)}; \hat{\theta}_m^{(t)})}{\sum_{m'=1}^{k} \hat{\alpha}_{m'}^{(t)} \prod_{l=1}^{N_{i,j}} \phi(g_l^{(i,j)}; \hat{\theta}_{m'}^{(t)})} \qquad (5.19) \\
Q_{m,l}^{(i,j)}(\theta_m) &= E\left[Z_m^{(i,j)} \log \phi(\bar{\gamma}_l^{(i,j)}; \theta_m) | \bar{\gamma}_l^{(i,j)} = g_l^{(i,j)}; \hat{\mathbf{\Theta}}^{(t)}\right]. \qquad (5.20)
\end{aligned}$$

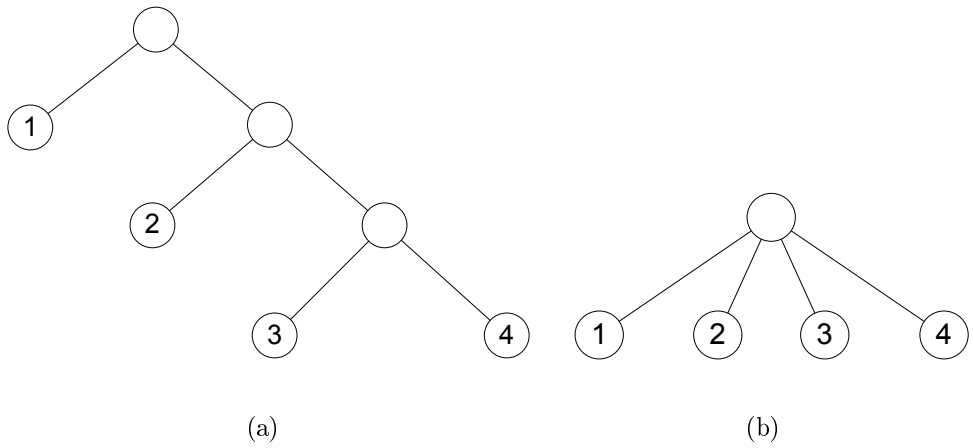
The M-step maximizes (5.18) with respect to $\mathbf{\Theta}$:

$$\begin{aligned}
\alpha_m^* &= \frac{\sum_{i,j \in \mathbf{G}, i<j} \omega_m^{(i,j)}}{T} \qquad (5.21) \\
\theta_m^* &= \arg\max_{\theta_m} \sum_{\substack{i,j \in \mathbf{G} \\ i<j}} \sum_{l=1}^{N_{i,j}} Q_{m,l}^{(i,j)}(\theta_m). \qquad (5.22)
\end{aligned}$$

When ϕ is Gaussian the M-step becomes

$$\begin{aligned}
\mu_m^* &= \frac{\sum_{i,j \in \mathbf{G}, i<j} \left[\sum_{l=1}^{N_{i,j}} g_l^{(i,j)}\right] \omega_m^{(i,j)}}{\sum_{i,j \in \mathbf{G}, i<j} N_{i,j} \cdot \omega_m^{(i,j)}} \qquad (5.23) \\
\sigma_m^* &= \frac{\sum_{i,j \in \mathbf{G}, i<j} \left[\sum_{l=1}^{N_{i,j}} \left(g_l^{(i,j)} - \mu_m^*\right)^2\right] \omega_m^{(i,j)}}{\sum_{i,j \in \mathbf{G}, i<j} N_{i,j} \cdot \omega_m^{(i,j)}}. \qquad (5.24)
\end{aligned}$$

The best model order k can be determined as follows. We assume k_{min} and k_{max} are the upper and lower bounds for k, respectively, which are determined according to some a priori information. If such information is not available, for $|\mathbf{G}| = n$ the maximum value for k is $n - 1$, which corresponds to a binary (sub)tree with dangling leaf nodes (see Figure 5.8(a)), and the minimum value for k is 1, which corresponds to the case where all the leaf nodes are siblings (see Figure 5.8(b)). For



(a) (b)

Figure 5.8: Illustration of the corresponding topologies to mixture models with model order equal $k = 3$ (a) and $k = 1$ (b) when there are 4 leaf nodes in $\mathbf{G}$.

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 (5.14). The best model is then chosen as the one which achieves the maximum penalized likelihood.

5.4.2 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 induced by 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 5.9 shows an example which illustrates how the algorithm works. Figure 5.9(a) is the true topology of the network. The estimator first identify 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 5.9(b). The two shaded nodes are the internal nodes found by this initial clustering. In the next iteration the estimator determines

there is no partition existing in the second cluster, but the third cluster includes 4 subclusters, as depicted in Figure 5.9(c), which defines two more shaded internal nodes in the topology. This iterative procedure is a greedy algorithm because it doesn't take into account any possible partition in the subsequent iterations during the estimation at current iteration.

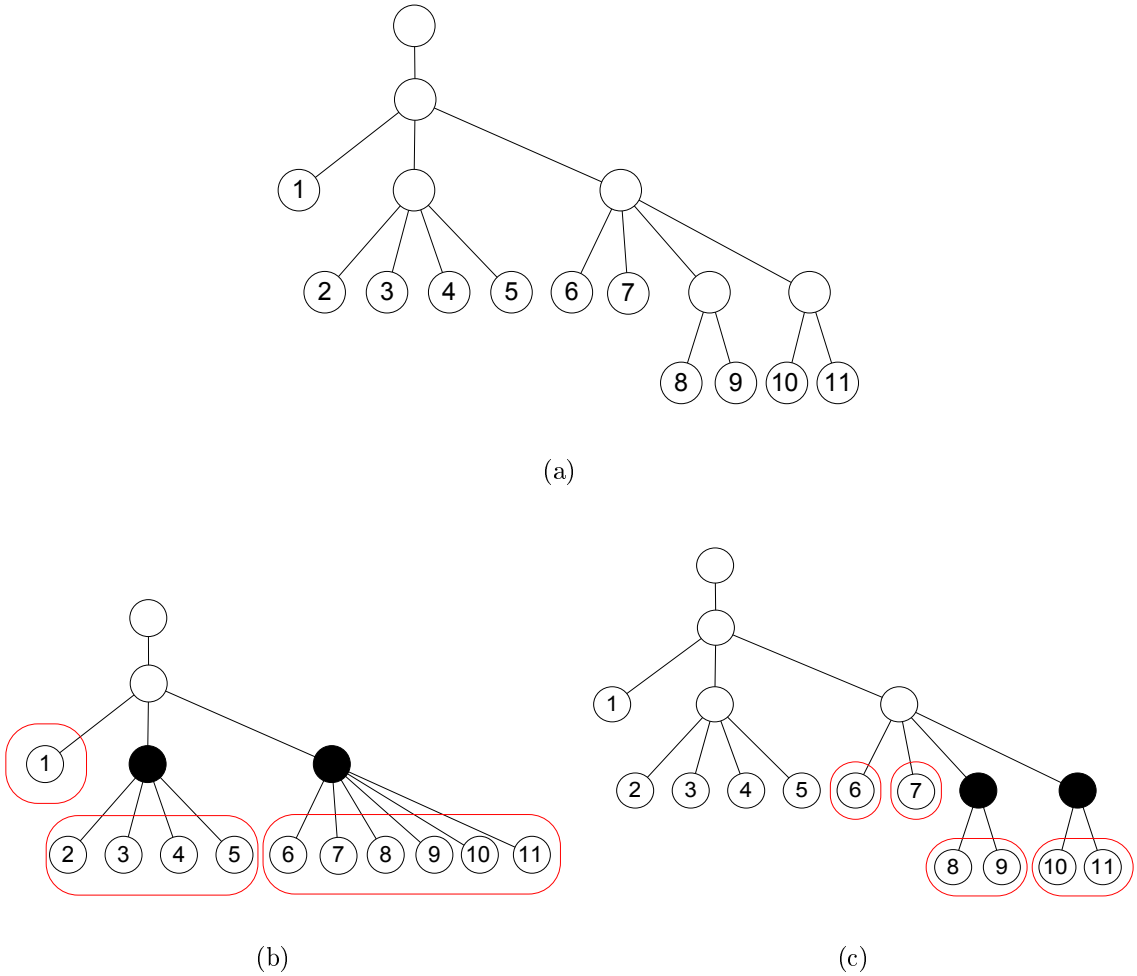


Figure 5.9: Illustration of the hierarchical topology estimation. (a) depicts the true topology. (b) and (c) are the first and second steps in the estimation process, respectively. The encircled groups of leaf nodes are the clusters found by the estimator, which are used to identify the shaded internal nodes.

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. Label the component in $f_{FM}(\mathbf{\Gamma}(\mathbf{G}))$ having the smallest mean by component 1. 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 $\omega_1^{(i,j)}$ in (5.20) is approximately 1 for any inter-cluster pair (i,j) 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 $\mathbf{G}$ such that there exists an edge between every pair of the vertices. If we specify a weight

$$w_{i,j} = w(e_{i,j}) = 1 - \omega_1^{(i,j)} \tag{5.25}$$

to every edge $e_{i,j}$, one can easily find that a vertex in H strongly connects only to its peers in the same cluster. This implies that the partition of the leaf nodes can be estimated based on the connectivity of the graph H. The partition in Figure 5.9(c) using graph connectivity is depicted in Figure 5.10.

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 [50, 51, 52, 53, 54]. Basically any graph-based clustering algorithm for weighted graphs could work for our purpose. Here we describe a simple algorithm proposed in [52], 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 $H = (V_H, E_H)$ be a graph both undirected and weighted, where V_H is the set of vertices and E_H is the set of edges. Every edge e in E_H has a nonnegative real weight $w(e)$. The degree of a

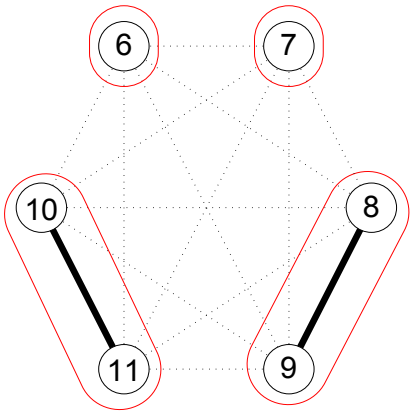


Figure 5.10: The partition of leaf nodes $\{6, 7, 8, 9, 10, 11\}$ in Figure 5.9(c) based on the graph connectivity. The solid edges have weights ≈ 1, and the dotted edges have weights ≈ 0. The vertices in a circle denote a cluster found in the graph.

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 [52], which is provided below:

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

Table 5.2: 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
```

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

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

which contradicts the fact that H is highly connected.

The HCS algorithm requires a subroutine MINCUT(H) which, for a given graph H, returns S, H', and $\bar{H}'$, 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 [92, 93, 94, 95, 96, 97, 98]. The HCS algorithm is summarized in Table 5.2. [52].

Before applying the HCS algorithm to the topology estimation problem, we would like to discuss several detailed improvements which can be made to the EM 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 nodes $\mathbf{G}$. 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 $\mathbf{B} \subset \{1, \ldots, k\}$ 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. 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 $Z_{\mathbf{B}}^{(i,j,n)}$ 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 $Z_{\mathbf{B}}^{(i,j,n)} = 1$ if $\bar{\gamma}_n^{(i,j)}$ is contributed by the composite component $f_{\mathbf{B}}$, and $Z_{\mathbf{B}}^{(i,j,n)} = 0$ otherwise. $\{Z_{\mathbf{B}}^{(i,j,n)}\}_{i,j,n}$ is i.i.d. and assumed to have mean $E[Z_{\mathbf{B}}^{(i,j,n)}] = \eta_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)}$.

5.4.3 Robust Edge Weights in the Complete Graph

The probability $\omega_1^{(i,j)}$ in (5.20) used in the edge weights $w_{i,j}$ (5.25) is not a robust estimator of $Z_1^{(i,j)}$. 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 (5.20) 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 $Z_1^{(i,j,n)}$ using only one $\bar{\gamma}_n^{(i,j)}$ in $\mathbf{\Gamma}_{i,j}$. As contrasted to the original definition in (5.20), 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

$$\begin{aligned} w_{\mathbf{A}_1,\mathbf{A}_2}(\mathbf{B}) &= 1 - \frac{1}{N_{\mathbf{A}_1,\mathbf{A}_2}} \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} \sum_{n=1}^{N_{i,j}} E\left[Z_{\mathbf{B}}^{(i,j,n)} | \bar{\gamma}_n^{(i,j)}; \mathbf{\Theta}^* \right] \\ &= 1 - \frac{1}{N_{\mathbf{A}_1,\mathbf{A}_2}} \sum_{i \in \mathbf{A}_1} \sum_{j \in \mathbf{A}_2} \sum_{n=1}^{N_{i,j}} \frac{\sum_{m \in \mathbf{B}} \hat{\alpha}_m^* \phi(\bar{\gamma}_n^{(i,j)}; \hat{\theta}_m^*)}{\sum_{m'=1}^{k} \hat{\alpha}_{m'}^* \phi(\bar{\gamma}_n^{(i,j)}; \hat{\theta}_{m'}^*)}, \end{aligned} \tag{5.26}$$

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

$$\lim_{N_{\mathbf{A}_1,\mathbf{A}_2}\to\infty} P\left(\left|w_{\mathbf{A}_1,\mathbf{A}_2}(\mathbf{B}) - (1-\eta_{\mathbf{B}}^{(\mathbf{A}_1,\mathbf{A}_2)})\right| < \epsilon\right) = 1, \tag{5.27}$$

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

5.4.4 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 [97] has an overall run time complexity of $\mathcal{O}(|V_H||E_H| + |V_H|^2\log|V_H|)$ for any graph H. One way to reduce the number of vertices is to pre-cluster H into groups of leaf nodes which are obviously in the same cluster.

Let $\mathbf{B}$ denote the inter-cluster component for a partition of the set of leaf nodes $\mathbf{G}$. If there is no estimation error in the finite mixture model, $\mathbf{B}$ is 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 $w_{\{i\},\{j\}}(\mathbf{B})$ is less than $\frac{1}{2}$. Define the set of *foreign leaf nodes* of node i with respect to $\mathbf{B}$ as

$$\mathbf{F}_{\mathbf{B}}(i) = \left\{j : w_{\{i\},\{j\}}(\mathbf{B}) < \frac{1}{2},\ j \in \mathbf{G}\backslash\{i\}\right\}, \quad \forall i \in \mathbf{G}. \tag{5.28}$$

$\mathbf{F}_{\mathbf{B}}(i)$ 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 $\mathbf{F}_{\mathbf{B}}(i_1) = \mathbf{F}_{\mathbf{B}}(i_2)$.

However, when there exists significant error in the finite mixture model estimates, it is possible that the 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, in which the component 1 is only one of the subcomponents, or, an estimate with too coarse resolution could merge the inter-cluster component with other intra-cluster components, which results in an overly fine clustering (too many clusters) of the leaf nodes.

5.4.5 Progressive Search Algorithm

We deal with the first situation by a *progressive search* method. Let the estimated components 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 letting $\mathbf{B}_1 = \{1\}$ be the inter-cluster component and forming a pre-clustering $\mathbf{K}_p(\mathbf{B}_1)$. Then expand the inter-cluster component to $\mathbf{B}_2 = \{1, 2\}$ and form another pre-clustering $\mathbf{K}_p(\mathbf{B}_2)$. Repeat this procedure until $\mathbf{B}_k = \{1, \ldots, k\}$. Then we select the pre-clusterings with the least number of node clusters as the *pre-clustering estimates*:

$$\hat{\mathbf{K}}_p = \{\mathbf{K}_p(\mathbf{B}_i) : |\mathbf{K}_p(\mathbf{B}_i)| \leq |\mathbf{K}_p(\mathbf{B}_j)|, i, j \in \{1, \ldots, k\}, i \neq j\}. \tag{5.29}$$

A complete graph can be drawn from each pre-clustering estimate with its vertices representing the clusters which may contain two or more leaf nodes in one cluster. The modification of the edge weights in (5.26) can also be applied here. Let $H(\mathbf{K})$ be the complete graph induced by the clustering $\mathbf{K}$. Then the graphs $H(\mathbf{K}_p(\mathbf{B}_i))$ for $\mathbf{K}_p(\mathbf{B}_i) \in \hat{\mathbf{K}}_p$ are passed as the inputs of the HCS algorithm and the output with the highest $\mathcal{L}_k$ is the clustering estimate of $\mathbf{G}$, denoted by $\hat{\mathbf{K}}(\mathbf{G})$. We also denote its corresponding inter-cluster component $\mathbf{B}_i$ by $\hat{\mathbf{B}}$.

5.4.6 Post-merge Algorithm

To address the second situation described at the end of Section 5.4.4 we propose a post-merge algorithm to deal with overly fine clusters. Given $\hat{\mathbf{K}}(\mathbf{G})$ we form the complete graph $H(\hat{\mathbf{K}}(\mathbf{G}))$ using $\hat{\mathbf{B}}$ as the inter-cluster component. 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.,

$$c(\mathbf{A}) = \arg\max_{\mathbf{A}' \in \hat{\mathbf{K}}(\mathbf{G})} w_{\mathbf{A},\mathbf{A}'}(\hat{\mathbf{B}}). \tag{5.30}$$

Then for each $\mathbf{A} \in \hat{\mathbf{K}}(\mathbf{G})$ 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 $\hat{\mathbf{K}}(\mathbf{G})$, then we update $\hat{\mathbf{K}}(\mathbf{G})$ by the corresponding new partition and repeat the process, until no improvement is made by any pair-wise merge. Note that some of the merges may result in redundant partition when a pair of clusters are the closest to each other. The number of merges needed to be tested when there are M clusters is lower bounded by $\lceil \frac{M}{2} \rceil$ and upper bounded by $M - 1$.

5.5 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.

5.5.1 MATLAB Model Simulation

First we simulated a small network with the simple non-binary virtual topology shown in Figure 5.11. 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 generated according to a Gaussian distribution with a randomly generated mean γ_l and a standard deviation σ_l proportional to the mean. Note that the true link metric was specified by γ_l. γ_l was generated according to a uniform distribution over a region centered at η with width equal to β, i.e., $\gamma_l \sim [\eta - \frac{\beta}{2}, \eta + \frac{\beta}{2}]$. 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 [34, 32] and the DBT algorithm [29, 31]. 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 a 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 [31] 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

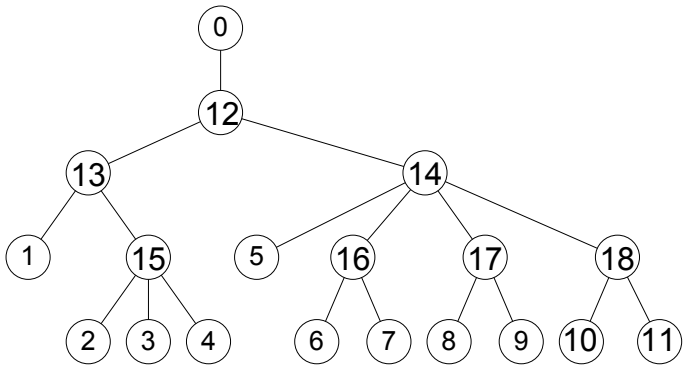


Figure 5.11: The logical tree topology for the network used in computer simulations in Section 5.5.1 and 5.5.2.

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 [100]. Several distance metrics between a pair of trees had been proposed in the past decades, such as the edit distance [101, 102, 103, 104, 105, 106, 107], the alignment distance [108], the isolated-subtree distance [109], the top-down distance [110] and the bottom-up distance [100]. 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 5.12 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 [102, 103]. The graph edit distance problem for unordered trees has been shown to be NP-complete [111].

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 5.1 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.

We conducted two experiments using the proposed HTE algorithm, along with

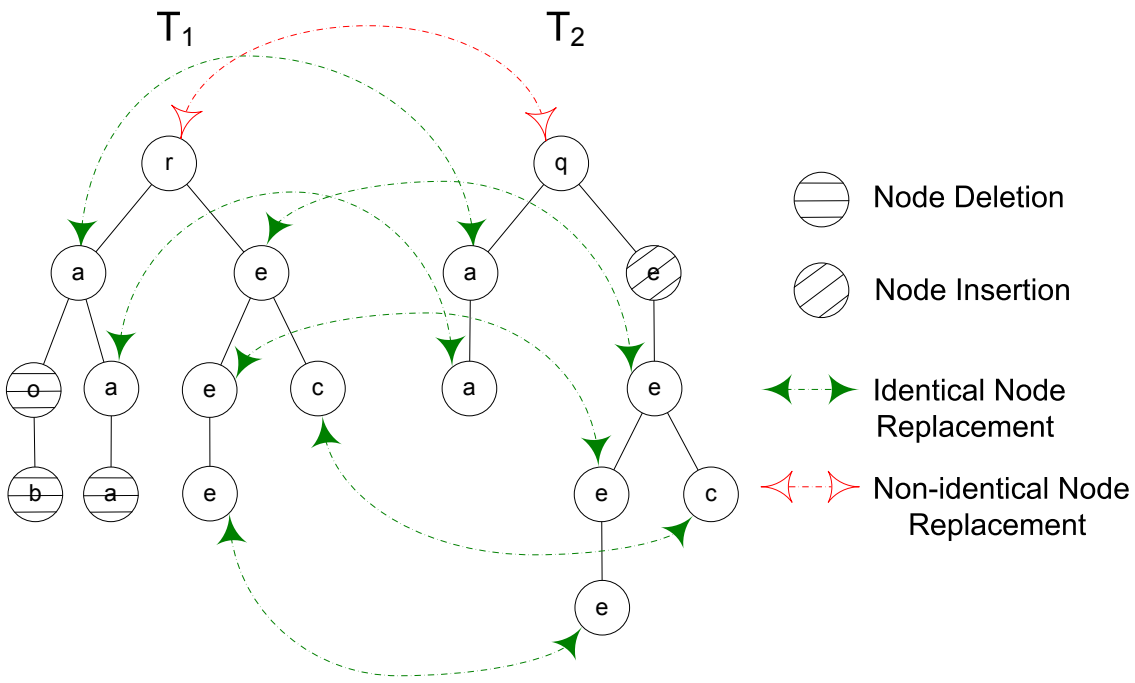
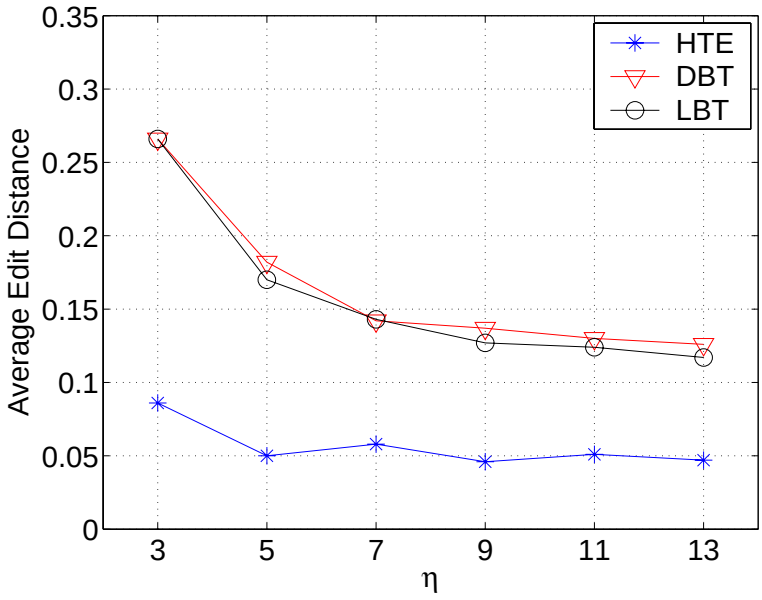


Figure 5.12: Illustration of the editing operations between two ordered trees.

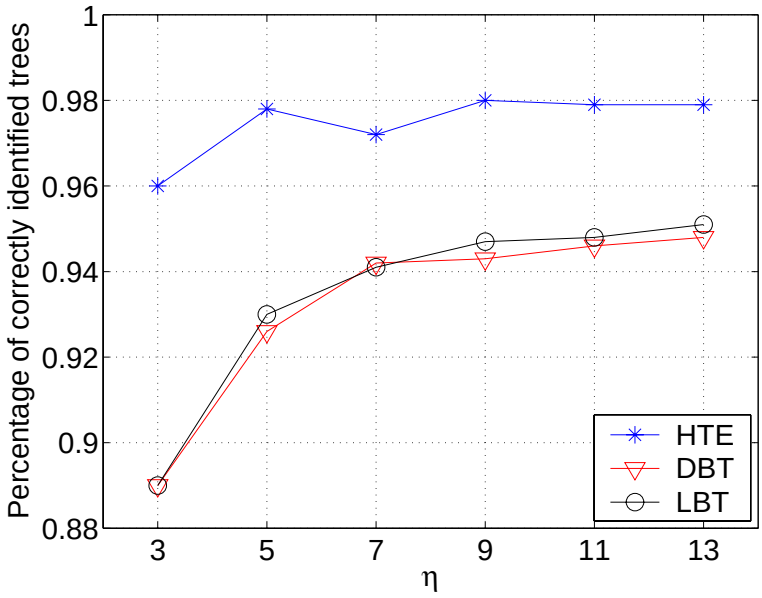
the DBT and LBT. These experiments were intended to clarify the advantages and drawbacks of each method and should not be considered as a thorough comparison of performance. In the first experiment we let each γ_l obey the same uniform distribution with a fixed width β equal 2 and a mean η varied from 3 to 13. We also fixed the noise variance scale factor $\rho = \frac{1}{\sqrt{2}}$. The result is illustrated in Figure 5.13. 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 editing 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 remain small.

In the second experiment we fixed the range of the uniform distribution for each link metric to $[2, 6]$. The proportionality factor ρ for sample standard deviations were 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 5.14. Each data point was averaged from the outcomes of 1000 independent simulations. As the accuracy of topology estimation decreases 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 Chapter 1 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.

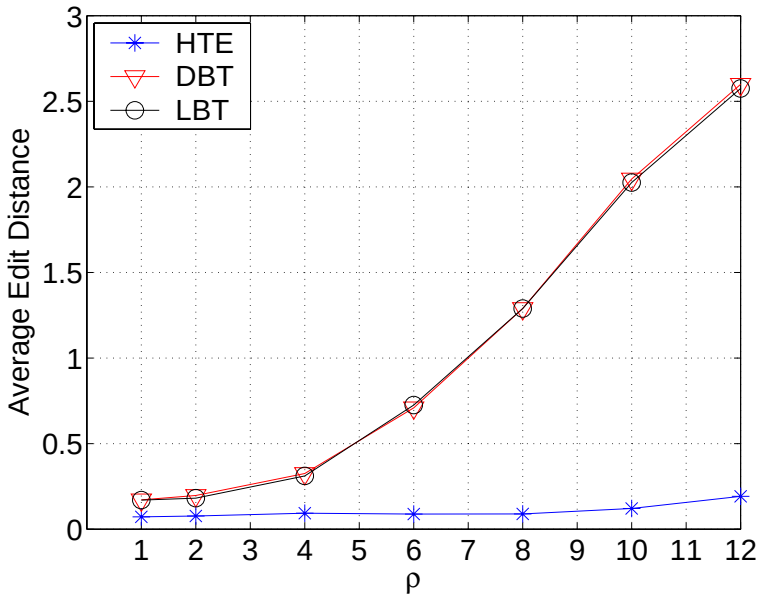


(a)

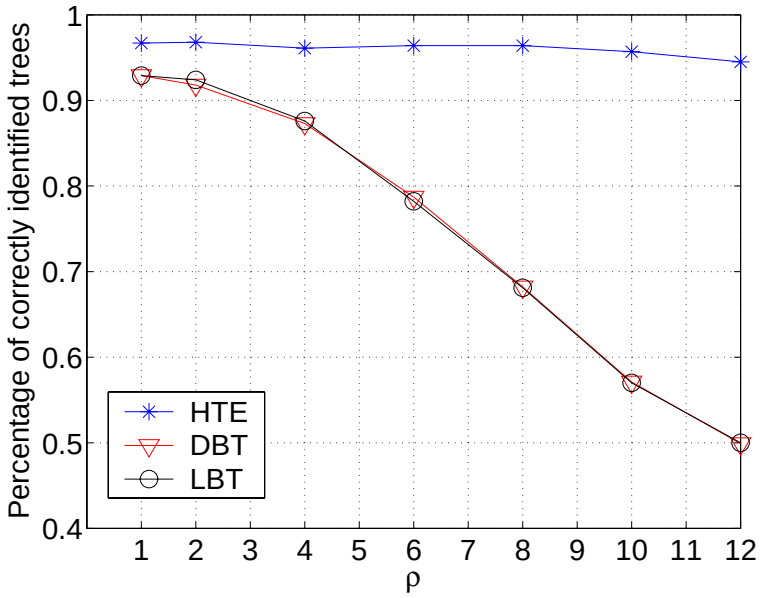


(b)

Figure 5.13: 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 5.14: 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.

5.5.2 NS Simulation

For a more pratical environment we used `ns-2` [40] to simulate the network in Figure 5.11. 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` to simulate various network condition. 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 5.2: delay difference 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. We let $N_2 = N_{cov} = N_{loss}$ in the `ns` simulation. The parameters specifying the number of probes used in `ns` simulation are listed in Table 5.3. Each column gives a set of parameters for a simulation. Figure 5.15 shows how to collect N_1 normalized similarity samples $\bar{\gamma}_n^{(i,j)}$ for each probe tree. Each normalized sample is the average of N_{norm} similarity

Table 5.3: 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

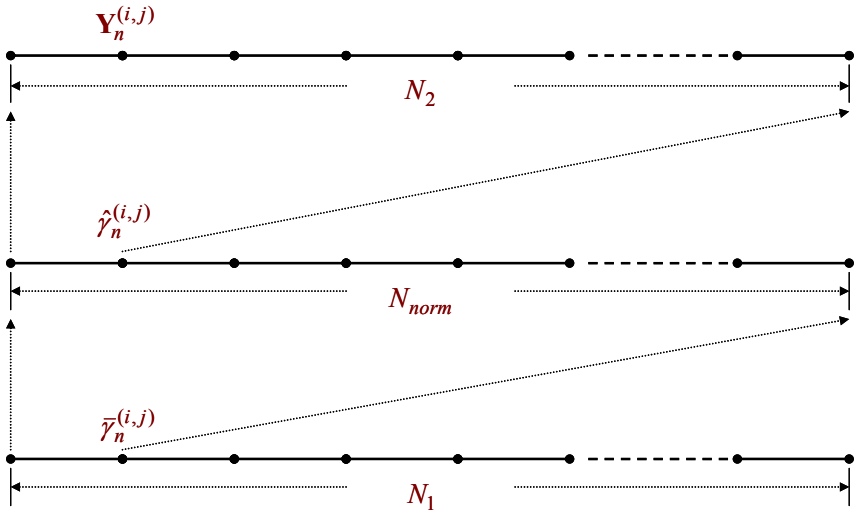


Figure 5.15: Illustration of the computation of N_1 normalized similarity samples. Each normalized sample is the average of N_{norm} similarity estimates. For packet pair probes each similarity estimate is obtained using N_2 measurements. In this figure it shows the delay measurement $\mathbf{Y}_n^{(i,j)}$ of packet pair probes.

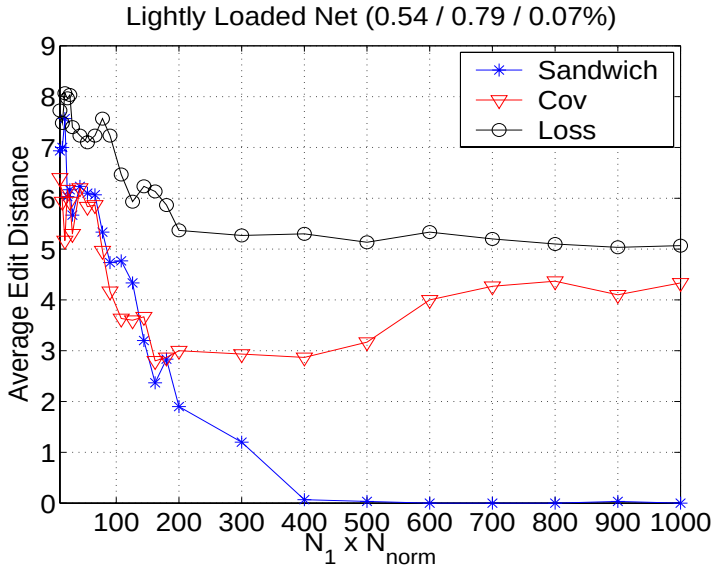
estimates $\bar{\gamma}_n^{(i,j)}$, which are delay differences, delay covariances, or packet drop rates. For packet pair probes each similarity estimate of delay covariance or packet drop rate is obtained using N_2 measurements. If any probe was dropped for a set of N_2 packet pairs, the similarity 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 a normalized approximately 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. The edit distance between the estimated tree and the true topology was averaged over 30 independent simulations for each case. In each case we used the proposed hierarchical topology estimator. Figure 5.16(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 delay (ms), delay standard deviation (ms), and packet drop rate for each link in each load condition is plotted in Figure 5.16(b),(d),(f). The notation $(a/b/c)$ in the titles of Figure 5.16(a),(c),(e) denotes the average condition for the whole network in which a, b, and c are the average delay, delay variance, and packet drop rate over all the links, respectively. Throughout the experiments in `ns` we used this notation to indicate the load condition of the network. The legends for delay difference, delay covariance, and loss rate similarities were marked by 'Sandwich', 'Cov', and 'Loss', respectively. As predicted in Section 5.2 the sandwich probes provided the most reliable topology estimate in a lightly-loaded network. From Figure 5.16(b) we found some of links had very small delay variances and hence could not be identified using delay covariances. 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 5.16(a). In a moderately-loaded network each link queue provided enough delay variation to perform topology estimation using packet pair delay covariance. Figure 5.16(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

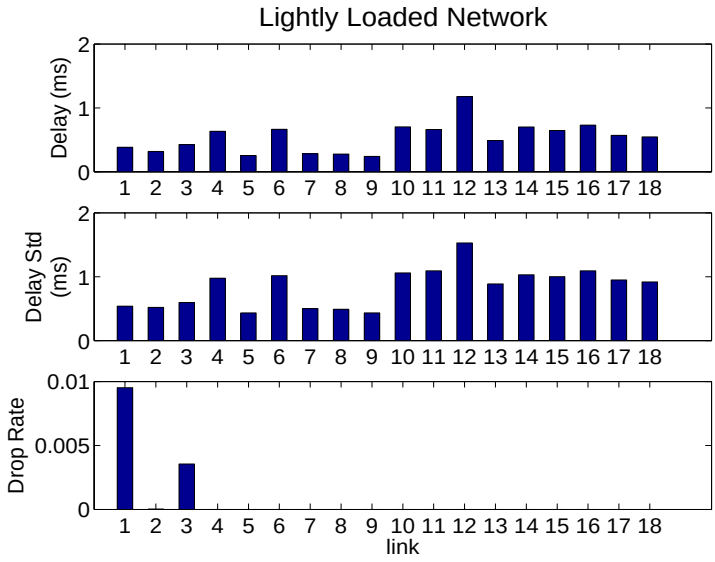
which did not experience any dropped packets. For a heavily-loaded network, each link had a substantial packet drop probability which made the loss rate similarities contain the most reliable information about the network structure. Although the delay variance on each link increased in the heavily-loaded 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 5.17.

The edit distance provides us a metric to describe the distribution of the topology estimates. To illustrate, we simulated a larger network in `ns`, whose topology is shown as the top tree in Figure 5.18. 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 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 normal-

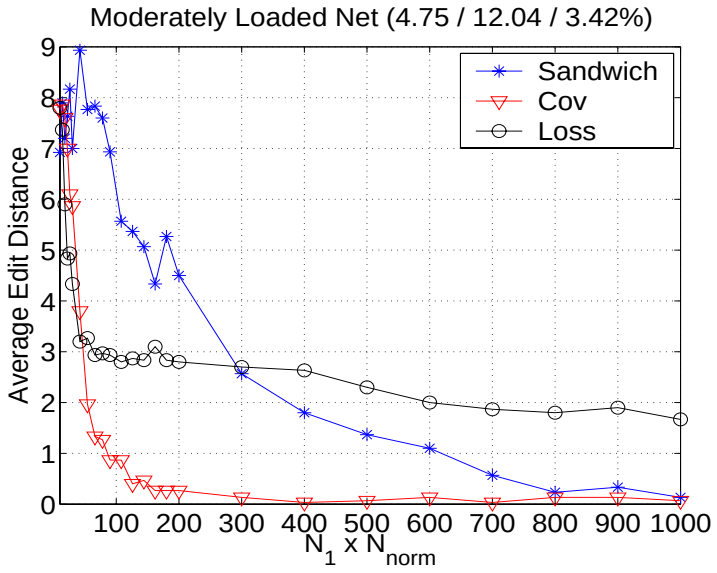


(a)

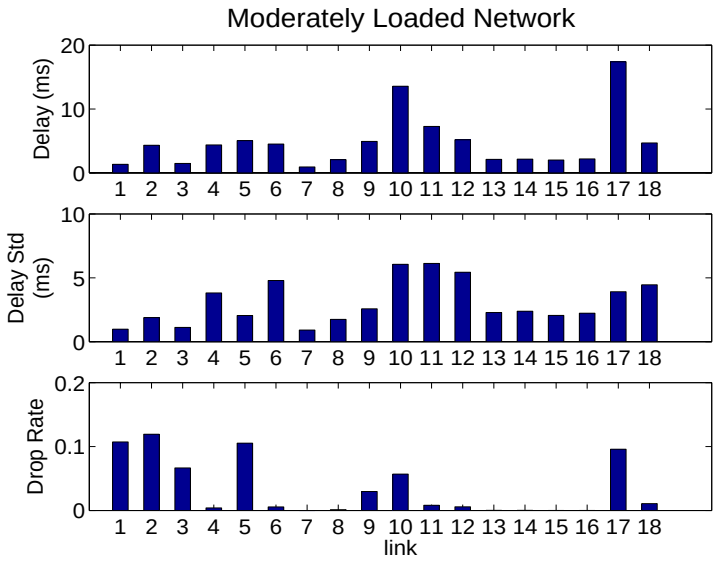


(b)

Figure 5.16: (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 5.2. (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.

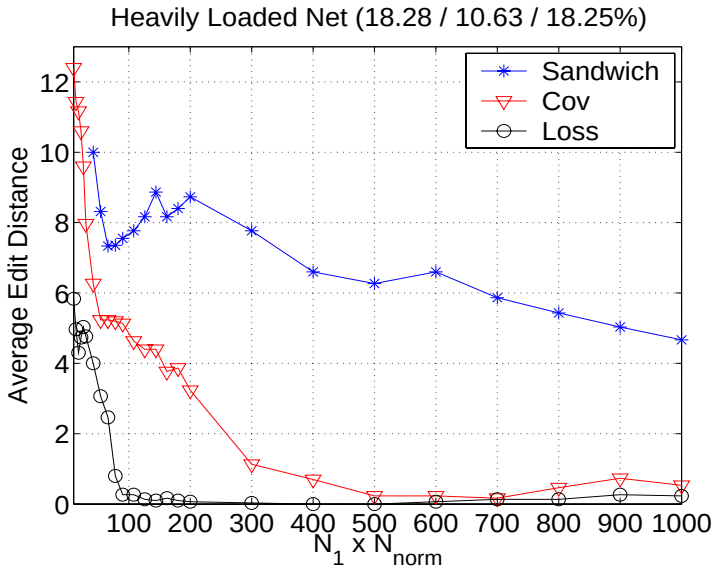


(c)

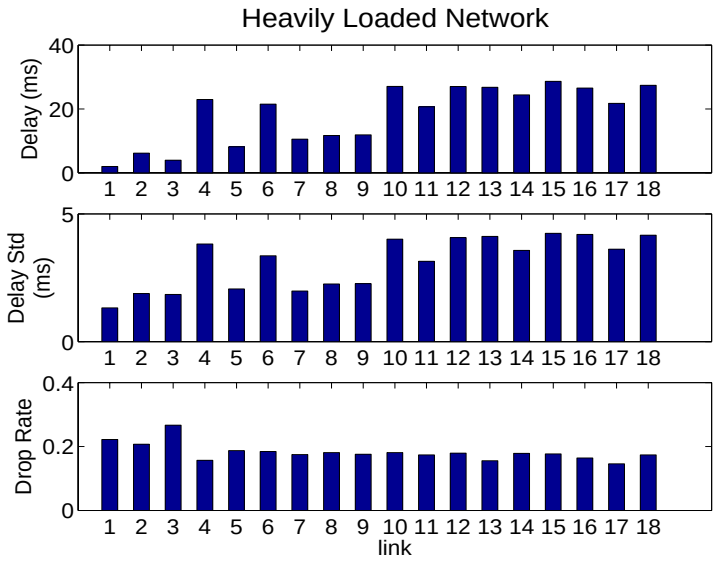


(d)

Figure 5.16 (continued)

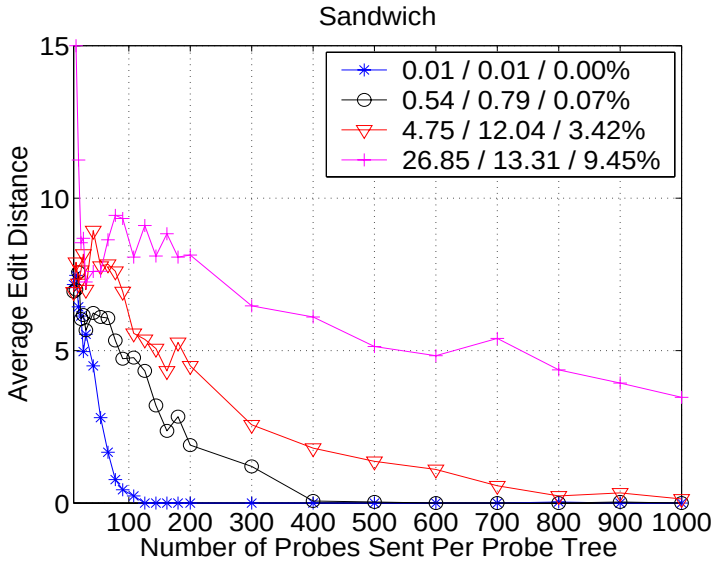


(e)

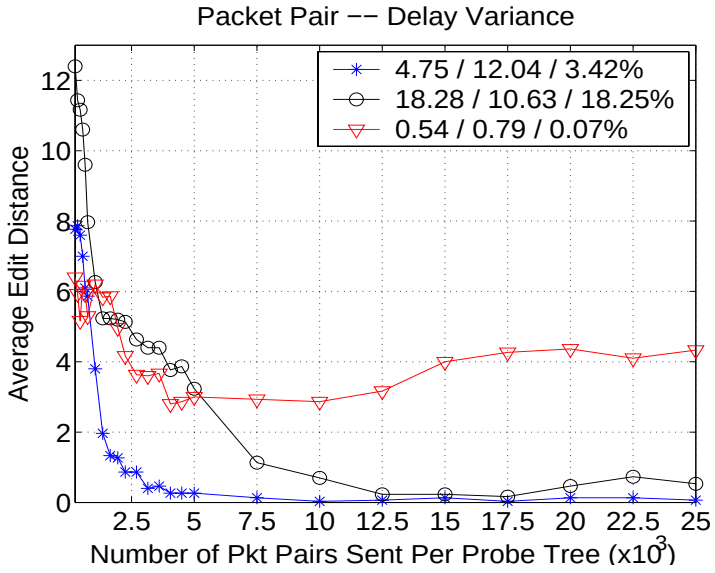


(f)

Figure 5.16 (continued)

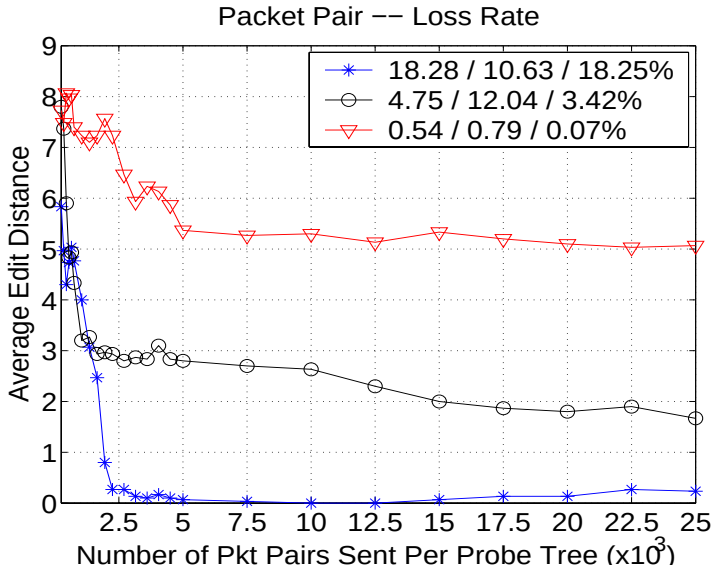


(a)



(b)

Figure 5.17: 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)

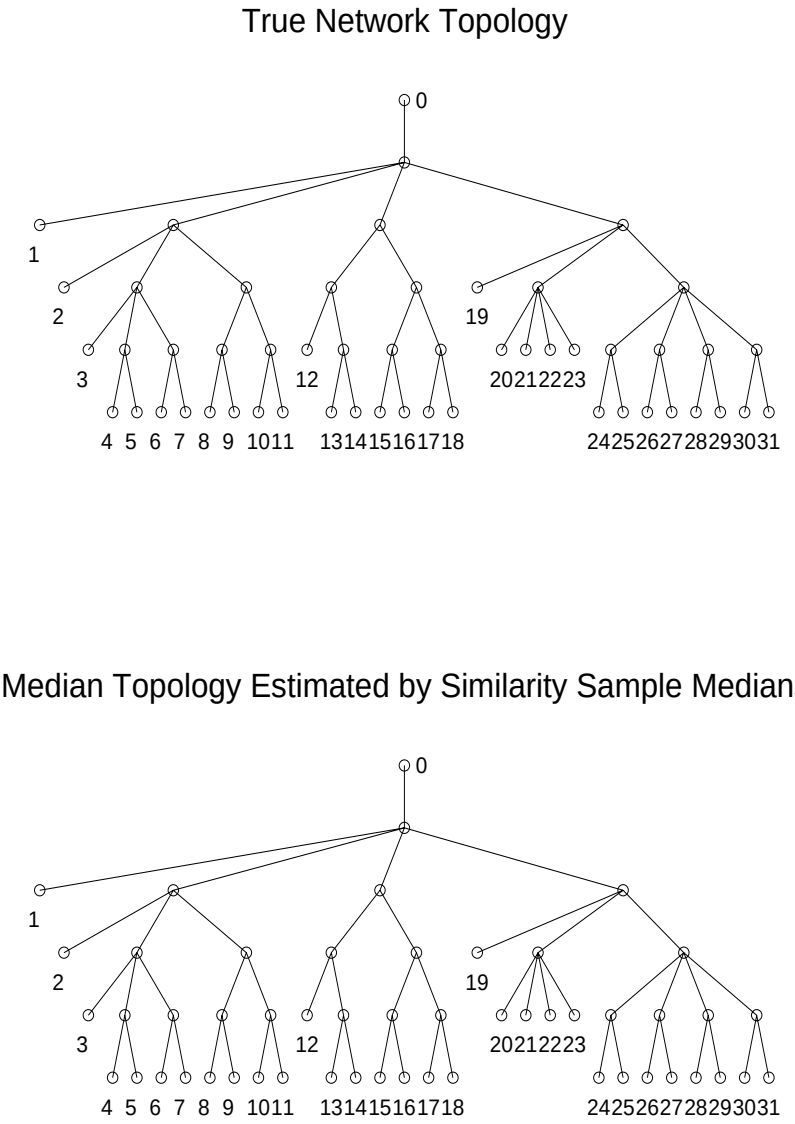
Figure 5.17 (continued)

ized 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 5.18 is shown in the bottom of the same figure, and the corresponding distribution of the topology estimates is shown in Figure 5.19 as pmf of 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 5.20. 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.

5.6 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 delay difference; 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 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



(a)

Figure 5.18: 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*.

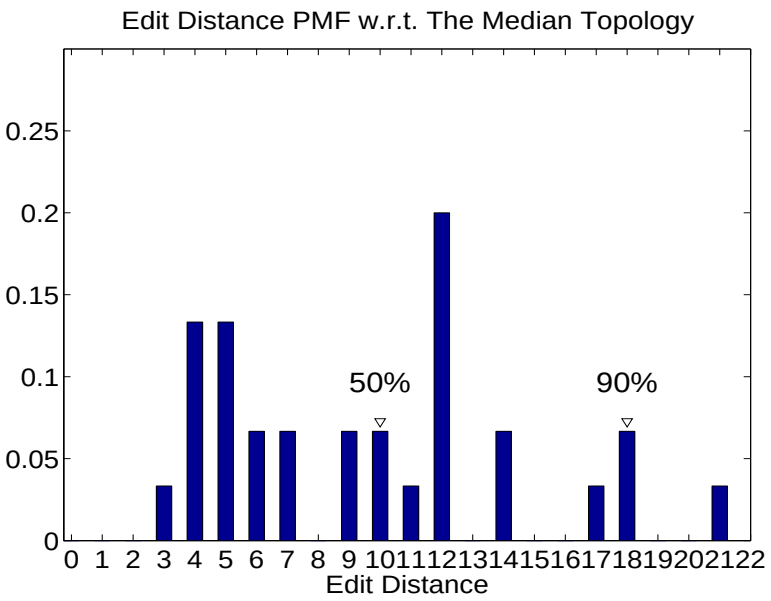
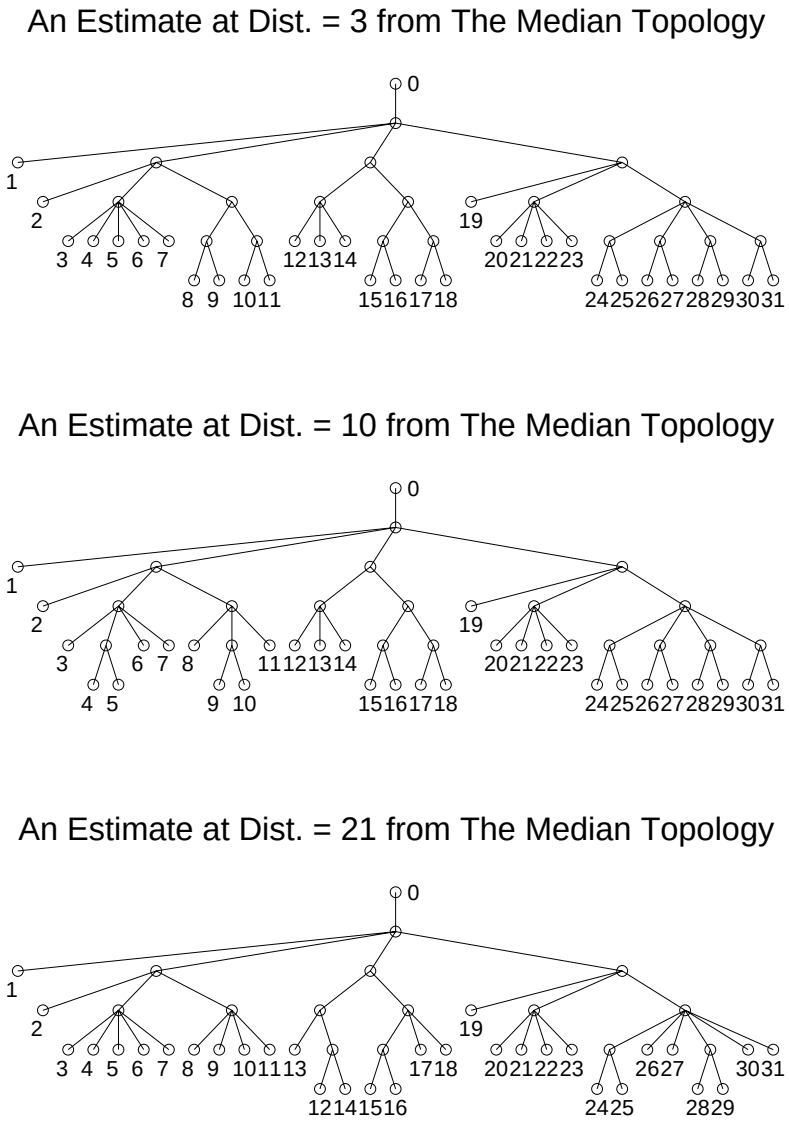


Figure 5.19: The pmf of the graph edit distance with respect to the median topology for the estimates from the `ns` simulation using the network in Figure 5.18. The locations of the 50th and 90th percentiles are also indicated.

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. 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.



(a)

Figure 5.20: Examples of the estimated topology along with their tree edit distance to the median topology for the `ns` simulated network in Figure 5.18.

The pre-clustering algorithm progressively includes components, starting with the one having the smallest mean. The post-merging algorithm 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` 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 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 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` simulations on a large 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 [36, 37, 38]. Extensive real network experiments should be implemented in the future to compare to ground truth real network topologies.

CHAPTER 6

Conclusion and Future Work

In this thesis we have studied the inference problems for packet switched networks such as the Internet. The focus was the estimation of internal link characteristics using unicast end-to-end delay measurements, supported by almost every network currently in operation. Specifically speaking we focused on network delay tomography and topology discovery.

We started with the estimation of link delay cumulant generating functions (CGF) which specify a linear system of equations for the internal link delay distribution given the end-to-end delay distributions. Under the assumption that link delays are spatially and temporally independent in a stationary network, we proposed a bias corrected estimator for the internal link delay CGF. Through simulation we showed that the proposed estimator attains a level of mean squared error comparable to link delay CGF estimates obtained from directly measured link delay statistics. These CGF estimates were used to estimate level exceedance probabilities of delays for each link and identify the bottleneck in the network.

Subsequently we presented a new method for estimation of internal link delay distributions using end-to-end packet pair delay statistics gathered by back-to-back

packet-pair unicast probes. The network was modelled by a logical tree with packet-pair probes sent from the root node to pairs to pairs of leaf nodes. We proposed a hybrid mixture delay model in which a point mass is used to represent the occurrence of an empty link queue and a continuous finite mixture density is included to describe the multi-modal non-empty queueing delay distribution. For the case that the model orders are known exactly, we used a maximum likelihood expectation-maximization (ML-EM) algorithm applied to the hybrid mixture model for the link delay probability density functions. Since the density model orders are generally unknown in practice, we suggested a method based on a variant of the penalized ML-EM (PML-EM) algorithm. We used a minimum message length (MML) penalty for selection of model orders. We presented results of `matlab` and `ns-2` simulations to illustrate the premise of our delay tomography algorithm for moderate cross-traffic scenarios.

The hybrid mixture model based algorithm has an exponential complexity with respect to the network size. Although it is impossible to reduce the complexity below exponential rate because the number of links and probe trees increases exponentially with the depth of the network tree, we proposed a bottom-up divide-and-conquer strategy to estimate link delay distributions. The key approximation is to replace the delay distributions for shared probe tree paths by those with model orders for single links. Analysis showed that the exponential rate of the runtime complexity is reduced significantly. The `ns-2` simulation results showed that the accelerated algorithm achieved the same level of estimation accuracy as the original algorithm and reduced by approximately 40% the runtime complexity.

The estimation of the logical tree topology from end-to-end unicast measurements of the network was also investigated here. We established a general framework for this problem in which no assumptions on the logical tree structure need is required. The topology estimation problem was represented as a hierarchical clustering prob-

lem 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 delay difference; 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 using MML-type penality 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 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. 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 pre-clustering algorithm progressively includes components, starting with the one having the smallest mean. The post-merging algorithm 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` 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 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 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` simulations on a large network using

sandwich probes.

Future research directions are numerous. For network delay tomography future work includes extension of our CGF and hybrid finite mixture model to include spatial dependencies of link delays among different links, especially the links along the same path. For time-varying scenarios adaptive schemes need to be developed in order to capture possible changes in traffic statistics and network environment. The proposed ML-EM and PML-EM algorithms could be extended to include the additional unknown minimum packet delay into the EM iterations but this unknown parameter could affect the convergence rate of the EM algorithm. To better fit link delay distributions in a real network, one could extend our hybrid finite mixture model to include more point masses and combinations of different families of probability densities which are flatter or more heavy-tailed than Gaussian. For example, exponential densities are more efficient to describe exponential tails of the distributions which correspond to rare large delay events in a lightly or moderately loaded network. For heavily loaded situations some heavy tailed functions, such as Pareto densities, might be used. This could also reduce the model orders in the finite mixture models and directly diminish the exponential rate constant in the computational complexity for the ML-EM and PML-EM algorithm. Another possible direction is finding ways to further accelerate convergence of the ML-EM and PML-EM algorithms for real-time implementation. EM algorithms are generally slow and the improvement made by CEM^2 is still limited. It may also be viable to apply these methods to detecting abnormal changes in link delay distributions, which is helpful to early detection of possible failures and/or malicious activities in the network. It could be achieved by estimating only the tail components of the mixture models after each consecutive probing session if the change on large packet delays is the focus.

Future work in topology discovery could focus on the use of hybrid probing

schemes, which could 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 [36, 37, 38]. Extensive real network experiments should be implemented in the future to compare to ground truth real network topologies.

APPENDICES

APPENDIX A

EM Algorithm for Network Delay Tomography Using Finite Mixture Models

A.1 General Formulation

Here we sketch the derivation of the E step and M step quantities required for ML, PML, and CEM^2 algorithms in Chapter 3 and 4. We assume the point mass is located at zero delay for all the links. Define the following notation:

1. $g_i(\mathbf{y}^{(i,n)}; \mathbf{\Theta})$ denotes the joint p.d.f. of the nth end-to-end packet pair delay vector $\mathbf{y}^{(i,n)} = \{y_1^{(i,n)}, y_2^{(i,n)}\}$ for the ith probe tree, parameterized by $\mathbf{\Theta}$. It admits a hybrid mixture model which is the convolution of all the $f_l(x)$'s along the tree path, according to the spatial independence assumption.

2. $g_{i,(l,m)}(\mathbf{y}^{(i,n)}; \mathbf{\Theta})$ is defined similarly to $g_i(\mathbf{y}; \mathbf{\Theta})$, except in the convolution $f_l(x)$ is replaced by its mth component, which is $\alpha_{l,0}$ when $m = 0$ or $\alpha_{l,m}\phi(x; \theta_{l,m})$ when $m \neq 0$. This is the likelihood of $\mathbf{y}^{(i,n)}$ given the delay at link l is contributed by the mth hidden component.

3. $h_{i,l}(\mathbf{y}^{(i,n)}; \mathbf{\Theta})$ is defined similarly to $g_i(\mathbf{y}; \mathbf{\Theta})$, except the lth link is excluded in

the convolution.

E-step: E-Step computes the conditional expectation of the complete data log-likelihood in (3.7). Let

$$\begin{aligned}
\omega_{l,m}^{(i,n)} &= E\left[Z_{l,m}^{(i,n)} | \mathbf{Y} = \mathbf{y}; \hat{\mathbf{\Theta}}^{(t)}\right] \\
&= \frac{P(Z_{l,m}^{(i,n)} = 1, \mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})}{P(\mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})} \\
&= \frac{g_{i,(l,m)}(\mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})}{g_i(\mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})}
\end{aligned} \tag{A.1}$$

for $m = 0, \ldots, k_l$ and $l = 1, \ldots, L$, where $\hat{\mathbf{\Theta}}^{(t)}$ denotes the parameter estimates obtained from the tth iteration. Define

$$Q_{l,m}^{(i,n)}(\theta_{l,m}) = E\left[Z_{l,m}^{(i,n)} \log \phi(X_l^{(i,n)}; \theta_{l,m}) | \mathbf{Y} = \mathbf{y}; \hat{\mathbf{\Theta}}^{(t)}\right] \tag{A.2}$$

for $m = 1, \ldots, k_l$ and $l = 1, \ldots, L$. $Q_{l,m}^{(i,n)}(\theta_{l,m}) = 0$ in the following two cases: (a) $y_1^{(i,n)} = 0$ or $y_2^{(i,n)} = 0$ when link l is shared by the unicast paths of probe tree i, and (b) $y_a^{(i,n)} = 0$ or $y_a^{(i,n)} = y_b^{(i,n)} \neq 0$ when l is in the splitting branch of the path for $y_a^{(i,n)}$, where $a \in \{1, 2\}$ and $b \in \{1, 2\} \backslash \{a\}$. It is because in these cases $Z_{l,m}^{(i,n)} = 0$ for

$m = 1, \ldots, k_l$. Otherwise,

$$
\begin{aligned}
Q_{l,m}^{(i,n)}(\theta_{l,m}) &= E\left[\log\phi(X_l^{(i,n)};\theta_{l,m})E\left[Z_{l,m}^{(i,n)}|X_l^{(i,n)}, \mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}\right]\right| \\
&\quad \left.\mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}\right] \\
&= E\left[\log\phi(X_l^{(i,n)};\theta_{l,m})E\left[Z_{l,m}^{(i,n)}|X_l^{(i,n)}; \hat{\mathbf{\Theta}}_l^{(t)}\right]|\mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}\right] \\
&= E\left[\log\phi(X_l^{(i,n)};\theta_{l,m})P\left(Z_{l,m}^{(i,n)} = 1|X_l^{(i,n)}; \hat{\mathbf{\Theta}}_l^{(t)}\right)|\mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}\right] \\
&= E\left[\log\phi(X_l^{(i,n)};\theta_{l,m})\frac{\hat{\alpha}_{l,m}^{(t)}\phi(X_l^{(i,n)};\hat{\theta}_{l,m}^{(t)})}{f_l(X_l^{(i,n)};\hat{\mathbf{\Theta}}_l^{(t)})}|\mathbf{Y}^{(i,n)} = \mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}\right] \\
&= \int \log\phi(x;\theta_{l,m})\frac{\hat{\alpha}_{l,m}^{(t)}\phi(x;\hat{\theta}_{l,m}^{(t)})}{f_l(x;\hat{\mathbf{\Theta}}_l^{(t)})} \cdot \frac{f_l(x;\hat{\mathbf{\Theta}}_l^{(t)})h_{i,l}((\mathbf{y}-\mathbf{x})^{(i,n)};\hat{\mathbf{\Theta}}^{(t)})}{g_i(\mathbf{y}^{(i,n)};\hat{\mathbf{\Theta}}^{(t)})}dx \\
&= \int \frac{\hat{\alpha}_{l,m}^{(t)}\phi(x;\hat{\theta}_{l,m}^{(t)})h_{i,l}((\mathbf{y}-\mathbf{x})^{(i,n)};\hat{\mathbf{\Theta}}^{(t)})\log\phi(x;\theta_{l,m})}{g_i(\mathbf{y}^{(i,n)};\hat{\mathbf{\Theta}}^{(t)})}dx, \qquad \text{(A.3)}
\end{aligned}
$$

where $(\mathbf{y}-\mathbf{x})^{(i,n)} = (y_1^{(i,n)} - x, y_2^{(i,n)} - x)$ if link l is shared by the unicast paths of probe tree i, or $(\mathbf{y}-\mathbf{x})^{(i,n)} = (y_1^{(i,n)} - x, y_2^{(i,n)})$, for example, if l is in the splitting branch of the path for $y_1^{(i,n)}$. The integral limits in (A.3) depends on the support region of function ϕ. The conditional expectation in (3.7) becomes

$$
Q(\mathbf{\Theta}, \hat{\mathbf{\Theta}}^{(t)}) = \sum_{l=1}^{L}\sum_{i:l\in\mathbf{M}_i}\sum_{n=1}^{N_i}\left\{\sum_{m=0}^{k_l}\omega_{l,m}^{(i,n)}\log\alpha_{l,m} + \sum_{m=1}^{k_l}Q_{l,m}^{(i,n)}(\theta_{l,m})\right\}. \qquad \text{(A.4)}
$$

M-step: M-Step updates the parameter estimates by maximizing $Q(\mathbf{\Theta}, \hat{\mathbf{\Theta}}^{(t)})$ over $\mathbf{\Theta}$:

$$
\hat{\alpha}_{l,m}^{(t+1)} = \frac{\sum_{i:l\in\mathbf{M}_i}\sum_{n=1}^{N_i}\omega_{l,m}^{(i,n)}}{\sum_{i:l\in\mathbf{M}_i}N_i} \quad m = 0, \ldots, k_l \qquad \text{(A.5)}
$$

$$
\hat{\theta}_{l,m}^{(t+1)} = \arg\max_{\theta}\sum_{i:l\in\mathbf{M}_i}\sum_{n=1}^{N_i}Q_{l,m}^{(i,n)}(\theta) \quad m = 1, \ldots, k_l. \qquad \text{(A.6)}
$$

Note that in this general EM formulation ϕ can be an arbitrary probability density function.

A.2 EM algorithm Using Gaussian Mixture Components

A.2.1 End-to-End Delay Likelihoods

Here we present the mathematical details of g_i, $g_{i,(l,m)}$, and $h_{i,l}$ using Gaussian mixture components. The framework also holds for other choices of densities with support region $(-\infty, \infty)$. To simplify the notations we drop the superscript $^{(i,n)}$ from the measurements $\mathbf{y}$ and $\hat{}^{(t)}$ from the parameters.

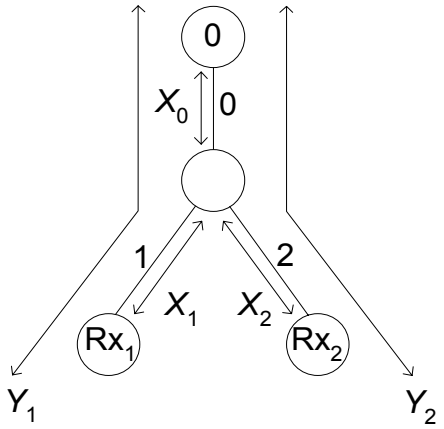


Figure A.1: Binary probe tree i with internal delays X_0, X_1, X_2 for branches 0, 1, 2, respectively, and end-to-end delays $Y_1 = X_0 + X_1$ and $Y_2 = X_0 + X_2$.

I. $g_i(y_1, y_2)$

Consider the binary probe tree i in Fig. A.1. Y_1 and Y_2 represent end-to-end delays from node 0 to receiver Rx_1 and Rx_2, respectively. X_0, X_1, and X_2 denote

the internal delays on the three branches 0, 1, and 2, respectively. Note that each branch in the probe tree may be a chain of links in the network. X_j has a hybrid mixture distribution as a result of convolution,

$$F_j(x) = \beta_j \delta(x) + GM_j(x), \quad j = 0, 1, 2, \tag{A.7}$$

where GM_j is the pure Gaussian mixture part of F_j. Let $A*(B \cdot C)$ denote a function of (y_1, y_2) which is the convolution of A, B, and C, in the form of $\int_{-\infty}^{\infty} A(x)B(y_1 - x)C(y_2 - x)dx$. Then the end-to-end delay distribution g_i is

$$
\begin{aligned}
g_i(y_1, y_2) &= F_0 * (F_1 \cdot F_2) \\
&= \beta_0\beta_1\beta_2\delta(y_1)\delta(y_2) + \beta_0\beta_1 GM_2(y_2)\delta(y_1) + \beta_0\beta_2 GM_1(y_1)\delta(y_2) + \\
&\quad \beta_1\beta_2 GM_0(y_1)\delta(y_1 - y_2) + \beta_0 GM_1(y_1)GM_2(y_2) + \\
&\quad \beta_1 GM_0(y_1)GM_2(y_2 - y_1) + \beta_2 GM_0(y_2)GM_1(y_1 - y_2) + \\
&\quad GM_0 * (GM_1 \cdot GM_2), \qquad \text{(A.8)}
\end{aligned}
$$

where $GM_0 * (GM_1 \cdot GM_2)$ is a linear combination of joing p.d.f.'s in the form of (3.2).

II. $g_{i,(l,m)}(y_1, y_2)$

Assume l is a link in probe tree i. $g_{i,(l,m)}$ includes only the mth component of f_l in the convolution. We define the likelihood function

$$F_{j,(l,m)}(x) = \beta_{j,(l,m)}\delta(x) + GM_{j,(l,m)}(x) \qquad \text{(A.9)}$$

for X_j if l is on branch $j \in \{0, 1, 2\}$, given that the delay at l is contributed by the mth component. Consider the following cases:

Case 1. l is on branch 0.

a) $m = 0$

1. l is the only link on the branch. In this case $GM_{0,(l,0)}(x) = 0$ and $\beta_{0,(l,0)} = \alpha_{l,0}$.

$$
\begin{aligned}
g_{i,(l,0)}(y_1, y_2) &= F_{0,(l,0)} * (F_1 \cdot F_2) \\
&= \begin{cases}
0 & y_1 = y_2 \neq 0 \\
\alpha_{l,0} \left(\beta_1\beta_2\delta(y_1)\delta(y_2) + \right. & \\
\beta_1 GM_2(y_2)\delta(y_1) + & \text{otherwise} \\
\beta_2 GM_1(y_1)\delta(y_2) + & \\
\left. GM_1(y_1)GM_2(y_2)\right) &
\end{cases}
\end{aligned}
\tag{A.10}
$$

2. l is not the only link on the branch.

$$
\begin{aligned}
g_{i,(l,0)}(y_1, y_2) &= F_{0,(l,0)} * (F_1 \cdot F_2) \\
&= \beta_{0,(l,0)}\beta_1\beta_2\delta(y_1)\delta(y_2) + \beta_{0,(l,0)}\beta_1 GM_2(y_2)\delta(y_1) + \\
&\quad \beta_{0,(l,0)}\beta_2 GM_1(y_1)\delta(y_2) + \beta_1\beta_2 GM_{0,(l,0)}(y_1)\delta(y_1 - y_2) + \\
&\quad \beta_{0,(l,0)} GM_1(y_1)GM_2(y_2) + \beta_1 GM_{0,(l,0)}(y_1)GM_2(y_2 - y_1) + \\
&\quad \beta_2 GM_{0,(l,0)}(y_2)GM_1(y_1 - y_2) + GM_{0,(l,0)} * (GM_1 \cdot GM_2)
\end{aligned}
\tag{A.11}
$$

b) $m \neq 0$. In this case $\beta_{0,(l,m)} = 0$.

$$
\begin{aligned}
g_{i,(l,m)}(y_1, y_2) &= F_{0,(l,m)} * (F_1 \cdot F_2) \\
&= \begin{cases}
0 & y_1 = 0 \quad \text{or} \quad y_2 = 0 \\
\beta_1\beta_2 GM_{0,(l,m)}(y_1)\delta(y_1 - y_2) + & \\
\beta_1 GM_{0,(l,m)}(y_1)GM_2(y_2 - y_1) + & \text{otherwise} \\
\beta_2 GM_{0,(l,m)}(y_2)GM_1(y_1 - y_2) + & \\
GM_{0,(l,m)} * (GM_1 \cdot GM_2) &
\end{cases}
\end{aligned}
\tag{A.12}
$$

Case 2. l is on branch $a \in \{1,2\}$. Let b be the branch connected to the other receiver node, i.e., $(a,b) = (1,2)$ or $(2,1)$.

a) $m = 0$

1. l is the only link on the branch. In this case $GM_{a,(l,0)}(x) = 0$ and $\beta_{a,(l,0)} = \alpha_{l,0}$.

$$\begin{aligned} g_{i,(l,0)}(y_1,y_2) &= F_0 * (F_{a,(l,0)} \cdot F_b) \\ &= \begin{cases} 0 & y_a \neq 0, \text{ and } y_b = 0 \\ \alpha_{l,0}\,(\beta_0\beta_b\delta(y_1)\delta(y_2)+ & \\ \beta_0 GM_b(y_b)\delta(y_a)+ & \text{otherwise} \\ \beta_b GM_0(y_a)\delta(y_a-y_b)+ & \\ GM_0(y_a)GM_b(y_b-y_a)) & \end{cases} \end{aligned} \tag{A.13}$$

2. l is not the only link on the branch.

$$\begin{aligned} g_{i,(l,0)}(y_1,y_2) &= F_0 * (F_{a,(l,0)} \cdot F_b) \\ &= \beta_0\beta_{a,(l,0)}\beta_b\delta(y_a)\delta(y_b) + \beta_0\beta_{a,(l,0)}GM_b(y_b)\delta(y_a) + \\ &\quad \beta_0\beta_b GM_{a,(l,0)}(y_a)\delta(y_b) + \beta_{a,(l,0)}\beta_b GM_0(y_a)\delta(y_a-y_b) + \\ &\quad \beta_0 GM_{a,(l,0)}(y_a)GM_b(y_b) + \beta_{a,(l,0)}GM_0(y_a)GM_b(y_b-y_a) + \\ &\quad \beta_b GM_0(y_b)GM_{a,(l,0)}(y_a-y_b) + GM_0 * (GM_{a,(l,0)} \cdot GM_b) \end{aligned} \tag{A.14}$$

b) $m \neq 0$. In this case $\beta_{a,(l,m)} = 0$.

$$
\begin{aligned}
g_{i,(l,m)}(y_1, y_2) &= F_0 * (F_{a,(l,m)} \cdot F_b) \\
&= \begin{cases}
0 & y_a = 0 \ \text{ or } \ y_a = y_b \neq 0 \\
\beta_0 \beta_b G M_{a,(l,m)}(y_a)\delta(y_b) + & \\
\beta_0 G M_{a,(l,m)}(y_a) G M_b(y_b) + & \text{otherwise} \\
\beta_b G M_0(y_b) G M_{a,(l,m)}(y_a - y_b) + & \\
G M_0 * (G M_{a,(l,m)} \cdot G M_b) &
\end{cases}
\end{aligned}
\tag{A.15}
$$

III. $h_{i,l}(y_1, y_2)$

Similarly, since $h_{i,l}$ doesn't include link l in the convolution we define

$$F_{j,l}(x) = \beta_{j,l}\delta(x) + G M_{j,l}(x) \tag{A.16}$$

for X_j if l is on branch $j \in \{0, 1, 2\}$, given that the delay at l is zero with probability 1. Note that if l is the only link on branch j, $F_{j,l}(x) = \delta(x)$. Because $h_{i,l}$ is used only in (A.3) we formulate $h_{i,l}(\mathbf{y} - \mathbf{x})$ instead.

Case 1. l is on branch 0.

1. l is the only link on the branch.

$$
\begin{aligned}
h_{i,l}(\mathbf{y} - \mathbf{x}) &= \left[F_{0,l} * (F_1 \cdot F_2)\right](y_1 - x, y_2 - x) \\
&= \beta_1\beta_2\delta(y_1 - x)\delta(y_1 - y_2) + \beta_1\delta(y_1 - x) G M_2(y_2 - y_1) \\
&\quad + \beta_2\delta(y_2 - x) G M_1(y_1 - y_2) + G M_1(y_1 - x) G M_2(y_2 - x)
\end{aligned}
\tag{A.17}
$$

2. l is not the only link on the branch.

$$\begin{aligned} h_{i,l}(\mathbf{y}-\mathbf{x}) &= [F_{0,l} * (F_1 \cdot F_2)]\,(y_1 - x, y_2 - x) \\ &= \beta_1\beta_2\delta(y_1 - y_2)\,(\beta_{0,l}\delta(y_1 - x) + GM_{0,l}(y_1 - x)) \\ &\quad +\beta_1 GM_2(y_2 - y_1)\,(\beta_{0,l}\delta(y_1 - x) + GM_{0,l}(y_1 - x)) \\ &\quad +\beta_2 GM_1(y_1 - y_2)\,(\beta_{0,l}\delta(y_2 - x) + GM_{0,l}(y_2 - x)) \\ &\quad +\beta_{0,l} GM_1(y_1 - x) GM_2(y_2 - x) \\ &\quad +GM_{0,l} * (GM_1 \cdot GM_2)(y_1 - x, y_2 - x) \end{aligned} \tag{A.18}$$

Case 2. l is on branch $a \in \{1, 2\}$. Let b be the branch connected to the other receiver, i.e., $(a, b) = (1, 2)$ or $(2, 1)$.

1. l is the only link on the branch.

$$\begin{aligned} h_{i,l}(\mathbf{y}-\mathbf{x}) &= [F_0 * (F_{a,l} \cdot F_b)]\,(y_a - x, y_b) \\ &= \beta_0\beta_b\delta(y_a - x)\delta(y_b) + \beta_0\delta(y_a - x) GM_b(y_b) \\ &\quad +\beta_b\delta(y_a - y_b - x) GM_0(y_b) \\ &\quad +GM_0(y_a - x) GM_b(y_b - y_a + x) \end{aligned} \tag{A.19}$$

2. l is not the only link on the branch.

$$\begin{aligned} h_{i,l}(\mathbf{y}-\mathbf{x}) &= [F_0 * (F_{a,l} \cdot F_b)](y_a - x, y_b) \\ &= \beta_0 (\beta_b \delta(y_b) + GM_b(y_b)) (\beta_{a,l}\delta(y_a - x) + GM_{a,l}(y_a - x)) \\ &\quad + \beta_b GM_0(y_b) (\beta_{a,l}\delta(y_a - y_b - x) + GM_{a,l}(y_a - y_b - x)) \\ &\quad + \beta_{a,l} GM_0(y_a - x) GM_b(y_b - y_a + x) \\ &\quad + GM_0 * (GM_{a,l} \cdot GM_b)(y_a - x, y_b) \end{aligned} \tag{A.20}$$

A.2.2 The EM Algorithm

The E-Step

It is straight forward to obtain $\omega_{l,m}^{(i,n)}$ using (A.8) for $g_i(\mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})$ and (A.10) - (A.15) for $g_{i,(l,m)}(\mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})$.

Let $\mathbf{N}_{l,m}^i = \left\{ j \in \{1, \ldots, N_i\} : Q_{l,m}^{(i,n)}(\theta_{l,m}) \neq 0 \right\}$ be the subset of support indices for probe tree i with respect to the mth component at link l. When ϕ is Gaussian, (A.3) becomes

$$\begin{aligned} Q_{l,m}^{(i,n)}(\theta_{l,m}) &= -\omega_{l,m}^{(i,n)} \left(\frac{\log 2\pi}{2} + \log \sigma_{l,m} + \frac{\mu_{l,m}^2}{2\sigma_{l,m}^2} \right) - \frac{1}{2\sigma_{l,m}^2 g_i(\mathbf{y}^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)})} \cdot \\ &\quad \int_{-\infty}^{\infty} (x^2 - 2\mu_{l,m}x) \hat{\alpha}_{l,m}^{(t)} \phi(x; \hat{\theta}_{l,m}^{(t)}) h_{i,l}((\mathbf{y}-\mathbf{x})^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}) dx. \end{aligned} \tag{A.21}$$

for $n \in \mathbf{N}_{l,m}^i$. The constant term $\frac{\log 2\pi}{2}$ can be ignored since it does not affect the M step. Define

$$A_{l,m}^{(i,n)} = \int_{-\infty}^{\infty} x \cdot \hat{\alpha}_{l,m}^{(t)} \phi(x; \hat{\theta}_{l,m}^{(t)}) h_{i,l}((\mathbf{y}-\mathbf{x})^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}) dx \tag{A.22}$$

$$B_{l,m}^{(i,n)} = \int_{-\infty}^{\infty} x^2 \cdot \hat{\alpha}_{l,m}^{(t)} \phi(x; \hat{\theta}_{l,m}^{(t)}) h_{i,l}((\mathbf{y}-\mathbf{x})^{(i,n)}; \hat{\mathbf{\Theta}}^{(t)}) dx \tag{A.23}$$

for $n \in \mathbf{N}^i_{l,m}$. From (A.17) - (A.20) the integrals $A^{(i,n)}_{l,m}$ and $B^{(i,n)}_{l,m}$ are sums of basic terms

$$a = \int_{-\infty}^{\infty} x \cdot \hat{\alpha}_{l,m} \phi(x; \hat{\theta}_{l,m}) h(\mathbf{y} - \mathbf{x}) dx = \rho \cdot \Phi \cdot \eta \tag{A.24}$$

and

$$b = \int_{-\infty}^{\infty} x^2 \cdot \hat{\alpha}_{l,m} \phi(x; \hat{\theta}_{l,m}) h(\mathbf{y} - \mathbf{x}) dx = \rho \cdot \Phi \cdot (\eta^2 + \xi), \tag{A.25}$$

respectively. Let $\tilde{f}(y_1, y_2)$ be the joint density function in (3.2) with parameters $\{\mu_i, \sigma_i^2\}_{i=1,2,3}$ and c, c_1, c_2 be constant coefficients. We list close forms of ρ, Φ, η, and ξ for the following 5 types of functions h which are used in $h_{i,l}$:

Type 1. $h(\mathbf{y} - \mathbf{x}) = c\phi(y - x; \mu, \sigma^2)$.

Let $\gamma = \sigma^2 + \hat{\sigma}^2_{l,m}$. Then

$$\begin{aligned} \rho &= \frac{c\hat{\alpha}_{l,m}}{\sqrt{2\pi\gamma}} \\ \Phi &= exp\left(-\frac{(y - (\mu + \hat{\mu}_{l,m}))^2}{2\gamma}\right) \\ \eta &= \frac{\sigma^2 \hat{\mu}_{l,m} + \hat{\sigma}^2_{l,m}(y - \mu)}{\gamma} \\ \xi &= \frac{\sigma^2 \hat{\sigma}^2_{l,m}}{\gamma}. \end{aligned}$$

Type 2. $h(\mathbf{y} - \mathbf{x}) = c_1\phi(y_1 - x; \mu_1, \sigma_1^2) \cdot c_2\phi(y_2 - x; \mu_2, \sigma_2^2)$.

Let $\gamma = \sigma_1^2\sigma_2^2 + \sigma_1^2\hat{\sigma}_{l,m}^2 + \sigma_2^2\hat{\sigma}_{l,m}^2$. Then

$$\begin{aligned}
\rho &= \frac{c_1 c_2 \hat{\alpha}_{l,m}}{2\pi\sqrt{\gamma}} \\
\Phi &= exp\left\{-\frac{1}{2\gamma}\left[\sigma_2^2(y_1 - (\mu_1 + \hat{\mu}_{l,m}))^2 + \sigma_1^2(y_2 - (\mu_2 + \hat{\mu}_{l,m}))^2 + \right.\right. \\
&\qquad \left.\left.\hat{\sigma}_{l,m}^2((y_1 - y_2) - (\mu_1 - \mu_2))^2\right]\right\} \\
\eta &= \frac{\sigma_1^2\sigma_2^2\hat{\mu}_{l,m} + \sigma_2^2\hat{\sigma}_{l,m}^2(y_1 - \mu_1) + \sigma_1^2\hat{\sigma}_{l,m}^2(y_2 - \mu_2)}{\gamma} \\
\xi &= \frac{\sigma_1^2\sigma_2^2\hat{\sigma}_{l,m}^2}{\gamma}.
\end{aligned}$$

Type 3. $h(\mathbf{y} - \mathbf{x}) = c\tilde{f}(y_1 - x, y_2 - x)$.

Let $\gamma_1 = \sigma_1^2\sigma_2^2 + \sigma_2^2\sigma_3^2 + \sigma_1^2\sigma_3^2$ and $\gamma_2 = (\sigma_1^2 + \hat{\sigma}_{l,m}^2)\sigma_2^2 + \sigma_2^2\sigma_3^2 + (\sigma_1^2 + \hat{\sigma}_{l,m}^2)\sigma_3^2$.

Then

$$\begin{aligned}
\rho &= \frac{c\hat{\alpha}_{l,m}}{2\pi\sqrt{\gamma_2}} \\
\Phi &= exp\left\{-\frac{1}{2\gamma_2}\left[\sigma_3^2(y_1 - (\mu_1 + \mu_2 + \hat{\mu}_{l,m}))^2 + \sigma_2^2(y_2 - (\mu_1 + \mu_3 + \hat{\mu}_{l,m}))^2\right]\right. \\
&\qquad \left.-\frac{\sigma_1^2 + \hat{\sigma}_{l,m}^2}{2\gamma_2}\left((y_1 - y_2) - (\mu_2 - \mu_3)\right)^2\right\} \\
\eta &= \frac{\gamma_1\hat{\mu}_{l,m} + \sigma_3^2\hat{\sigma}_{l,m}^2(y_1 - (\mu_1 + \mu_2)) + \sigma_2^2\hat{\sigma}_{l,m}^2(y_2 - (\mu_1 + \mu_3))}{\gamma_2} \\
\xi &= \frac{\gamma_1\hat{\sigma}_{l,m}^2}{\gamma_2}.
\end{aligned}$$

Type 4. $h(\mathbf{y} - \mathbf{x}) = c_1\phi(y_1 - x; \mu_1, \sigma_1^2) \cdot c_2\phi(y_2 + x; \mu_2, \sigma_2^2)$.

Let $\gamma = \sigma_1^2\sigma_2^2 + \sigma_1^2\hat{\sigma}_{l,m}^2 + \sigma_2^2\hat{\sigma}_{l,m}^2$. Then

$$\begin{aligned}
\rho &= \frac{c_1 c_2 \hat{\alpha}_{l,m}}{2\pi\sqrt{\gamma}} \\
\Phi &= exp\left\{-\frac{1}{2\gamma}\left[\sigma_2^2(y_1 - (\mu_1 + \hat{\mu}_{l,m}))^2 + \sigma_1^2(y_2 - (\mu_2 - \hat{\mu}_{l,m}))^2 + \right.\right. \\
&\qquad \left.\left.\hat{\sigma}_{l,m}^2((y_1 + y_2) - (\mu_1 + \mu_2))^2\right]\right\} \\
\eta &= \frac{\sigma_1^2\sigma_2^2\hat{\mu}_{l,m} + \sigma_2^2\hat{\sigma}_{l,m}^2(y_1 - \mu_1) + \sigma_1^2\hat{\sigma}_{l,m}^2(\mu_2 - y_2)}{\gamma} \\
\xi &= \frac{\sigma_1^2\sigma_2^2\hat{\sigma}_{l,m}^2}{\gamma}.
\end{aligned}$$

Type 5. $h(\mathbf{y} - \mathbf{x}) = c\tilde{f}(y_1 - x, y_2)$.

Let $\gamma_1 = \sigma_1^2\sigma_2^2 + \sigma_2^2\sigma_3^2 + \sigma_1^2\sigma_3^2$ and $\gamma_2 = \sigma_1^2(\sigma_2^2 + \hat{\sigma}_{l,m}^2) + (\sigma_2^2 + \hat{\sigma}_{l,m}^2)\sigma_3^2 + \sigma_1^2\sigma_3^2$.

Then

$$\begin{aligned}
\rho &= \frac{c\hat{\alpha}_{l,m}}{2\pi\sqrt{\gamma_2}} \\
\Phi &= exp\left\{-\frac{1}{2\gamma_2}\left[\sigma_3^2(y_1 - (\mu_1 + \mu_2 + \hat{\mu}_{l,m}))^2 + \right.\right. \\
&\qquad \left.\left.\sigma_1^2((y_1 - y_2) - (\mu_2 + \hat{\mu}_{l,m} - \mu_3))^2\right] - \frac{\sigma_2^2 + \hat{\sigma}_{l,m}^2}{2\gamma_2}(y_2 - (\mu_1 + \mu_3))^2\right\} \\
\eta &= \frac{\gamma_1\hat{\mu}_{l,m} + \sigma_3^2\hat{\sigma}_{l,m}^2(y_1 - (\mu_1 + \mu_2)) + \sigma_1^2\hat{\sigma}_{l,m}^2((y_1 - y_2) - (\mu_2 - \mu_3))}{\gamma_2} \\
\xi &= \frac{\gamma_1\hat{\sigma}_{l,m}^2}{\gamma_2}.
\end{aligned}$$

The M-Step

The estimate updates for mixing probabilities $\hat{\alpha}_{l,m}^{(t+1)}$ can be obtained directly from (A.5). The updates for means and variances are

$$\hat{\mu}_{l,m}^{(t+1)} = \frac{1}{\sum_{i:l\in M_i}\sum_{n\in\mathbf{N}_{l,m}^i}\omega_{l,m}^{(i,n)}}\cdot\sum_{i:l\in M_i}\sum_{n\in\mathbf{N}_{l,m}^i}\frac{A_{l,m}^{(i,n)}}{g_i(\mathbf{y}^{(i,n)};\hat{\mathbf{\Theta}}^{(t)})} \tag{A.26}$$

$$\hat{\sigma}_{l,m}^{2\ (t+1)} = \frac{1}{\sum_{i:l\in M_i}\sum_{n\in\mathbf{N}_{l,m}^i}\omega_{l,m}^{(i,n)}}\cdot \sum_{i:l\in M_i}\sum_{n\in\mathbf{N}_{l,m}^i}\left\{\frac{B_{l,m}^{(i,n)}-2\hat{\mu}_{l,m}^{(t+1)}A_{l,m}^{(i,n)}}{g_i(\mathbf{y}^{(i,n)};\hat{\mathbf{\Theta}}^{(t)})}+(\hat{\mu}_{l,m}^{(t+1)})^2\omega_{l,m}^{(i,n)}\right\}. \tag{A.27}$$

BIBLIOGRAPHY

BIBLIOGRAPHY

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

[2] Y. Vardi, L.A. Shepp, L. Kaufman, "A statistical model for positron emission tomography," *J. Am. Stat. Assoc.*, vol.80, pp.8-37, 1985.

[3] C. Tebaldi, M. West, "Bayesian inference on network traffic using link count data," *J. Am. Stat. Assoc.*, vol.93, pp.557-573, 1998.

[4] J. Cao, D. Davis, S.V. Wiel, B. Yu, "Time-varing network tomography: router link data," *J. Am. Stat. Assoc.* ,Vol.95, pp.1063-1075, 2000.

[5] 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.

[6] 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.

[7] G. Michailidis, V.N. Nair, B. Xi, "Fast least-squares based algorithms for estimating and monitoring network losses with active network tomography," submitted, 2003.

[8] B. Xi, G. Michailidis, V.N. Nair, "Least-Squares estimates of network link loss probabilities using end-to-end multicast measurements," Technical Report, Department of Statistics, University of Michigan, Mar. 2003

[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] J. C. Bolot, "Characterizing end-to-end packet delay and loss in the internet," *J. High-Speed Networks*, vol.2, pp.305-323, Sep. 1993.

[11] V. Paxson, "End-to-end internet packet dynamics," *ACM Sigcomm*, Cannes, France, Sep. 1997.

[12] V. Jacobson, "Pathchar - a tool to infer characteristics of internet paths," presented at the *Mathematical Sciences Research Institute*(MSRI), Apr. 1997.

[13] A. Downey, "Using pathchar to estimate internet link characteristics," *ACM SIGCOMM*, Cambridge, MA., Aug. 1999.

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

[15] N.G. Duffield, F. Lo Presti, "Multicast inference of packet delay variance at interior network links," *IEEE Infocom 2000*, Tel-Aviv, Israel, Mar. 2000.

[16] 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.

[17] 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.

[18] M. Shih and A.O. Hero III, "Unicast inference of network link delay distributions from edge measurements," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP) 2001*, Salt Lake City, Utah, May 2001.

[19] N.G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley, "Network delay tomography from end-to-end unicast measurements," *Int'l Workshop on Digital Communications 2001- Evolutionary Trends of the Internet*, Taormina, Italy, Sep. 2001.

[20] M. Coates and R. Nowak, "Sequential monte carlo inference of internal delays in nonstationary communication networks," *IEEE Trans. on Signal Processing*, vol. 50, no. 2, pp. 366 -376, Feb. 2002.

[21] M. Coates, A.O. Hero III, R. Nowak, B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47-65, May 2002.

[22] M. Shih and A.O. Hero III, "Unicast-based inference of network link delay distributions using mixed finite mixture models," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP) 2002*, Orlando, Florida, May 2002.

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

[24] 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.

[25] E. Lawrence, G. Michailidis, V.N. Nair, "Inference of network delay distributions using the EM algorithm," Technical Report, Department of Statistics, University of Michigan, 2003.

[26] E. Lawrence, G. Michailidis, V.N. Nair, "Maximum likelihood estimation of internal network link delay distributions using multicast measurements," Technical Report, Department of Statistics, University of Michigan, Mar. 2003.

[27] R. Cáceres, N.G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley, "Loss-based inference of multicast network topology," *IEEE Conf. Decision and Control*, Phoenix, Arizona, 1999.

[28] 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.

[29] 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.

[30] 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.

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

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

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

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

[35] M. Coates, M. Rabbat, R. Nowak, "Discovering general logical network topologies", Technical Report TREE0202, Department of Electrical and Computer Engineering, Rice University, Nov. 2002.

[36] 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.

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

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

[39] 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

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

[41] National Laboratory for Applied Network Research (NLNAR). [Online]. Homepage: http://moat.nlanr.net/index.html.

[42] The Surveyor Project. [Online].
Homepage: http://www.advanced.org/surveyor/.

[43] W. Jiang and H. Schulzrinne, "QoS measurement of internet real-time multimedia services," *Proceedings NOSSDAV*, Chapel Hill, NC., Jun. 2000.

[44] K. Mochalski, J. Micheel, S. Donnelly, "Packet delay and loss at the Auckland internet access path," *Passive & Active Measurement Workshop (PAM) 2002*, Fort Collins, Colorado, Mar. 2002.

[45] L. Xu and M. Jordon, "On convergence properties of the EM algorithm for Gaussian mixtures," *Neural Computation*, vol. 8, pp. 129-151, 1996.

[46] G. Celeux, S. Chrètien, F. Forbes, A. Mkhadri, "A component-wise EM algorithm for mixtures," *Techical Report 3746*, INRIA Rhône-Alpes, France, 1999.

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

[48] 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.

[49] 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.

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

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

[52] 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.

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

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

[55] R.J. Gibbens, "Traffic characterization and effective bandwidths for broadband network traces," *Stochastic Networks, Theory and Applications*, pp. 169-179, Oxford Science Pub., 1996.

[56] B. Efron, "The jackknife, the bootstrap and other resampling plans," *Society for Industrial and Applied Math*, 1982.

[57] Y. Zhang, N.G. Duffield, V. Paxson, S. Shenker, "On the constancy of internet path properties," *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov. 2001.

[58] A Pásztor and D. Veitch, "PC based precision timing without GPS," *ACM SIGMETRICS*, A, Marina Del Rey, Los Angeles, June 2002.

[59] N.G. Duffield, F. Lo Presti, V. Paxson, D. Towsley, "Inferring link loss using striped unicast probes," *IEEE Infocom 2001*, Anchorage, Alaska, Apr. 2001.

[60] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Trans. on Networking*, vol. 3, pp. 226-244, Jun. 1994.

[61] J.S. Marron and M.P. Wand, "Exact mean integrated squared error," *Annals of Statistics*, vol. 20, pp. 712-736, 1992.

[62] D. Bertsekas and R. Gallager, "Data networks," Prentice-Hall, 1992.

[63] 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.

[64] M. Whindham and A. Cutler, "Information ratios for validating mixture analysis," *Journal of the American Statistical Association*, vol. 87, pp. 1188-1192, 1992.

[65] J. Rissanen, "Stochastic complexity in statistical inquiry," Singapore: World Scientific, 1989.

[66] 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.

[67] J. Bernardo and A. Smith, *Bayesian Theory*. Chichester, UK: J. Wiley & Sons, 1994.

[68] J. Fessler and A. Hero, "Space-alternating generalized expectation-maximization algorithm," *IEEE Trans. on Signal Processing*, vol. 42, no. 10, pp.2664-77, Oct. 1994.

[69] K. Claffy, G. Miller, K. Thompson, "The nature of the beast: recent traffic measurements from an internet backbone," *INET'98*, Geneva, Switzerland, July 1998.

[70] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns: a view from Ames internet exchange," *ITC Specialist Seminar on IP Traffic Measurement, Modeling, and Management*, Monterey, California, September 14, 2000.

[71] B.W. Silverman, '*Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York, 1986.

[72] D.W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*, Willey, New York, 1992.

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

[74] 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.

[75] 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.

[76] 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.

[77] 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.

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

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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

[85] 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.

[86] 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.

[87] 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.

[88] C. Tang and P. McKinley, "Modeling multicast packet losses in wireless LANs," *Proceedings of ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, San Diego, CA, Sep. 2003.

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

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

[91] 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.

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

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

[94] 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.

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

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

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

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

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

[100] 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.

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

[102] 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.

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

[104] 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.

[105] 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.

[106] 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.

[107] 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.

[108] 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.

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

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

[111] 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.