A Scalable Method for Estimating Network Traffic Matrices from Link Counts

Jin Cao, Scott Vander Wiel, Bin Yu, Zhengyuan Zhu

Abstract— **Traffic matrices are extremely useful for network configuration, management, engineering, and pricing. Direct measurement is, however, expensive in general and impossible in some cases. This paper proposes a scalable algorithm for statistically estimating a traffic matrix from the readily available link counts. It relies on a divide-and-conquer strategy to lower the computational cost without losing estimation accuracy. The proposed algorithm is tested on a real network with 18 nodes. The estimates are comparable to the direct estimates but require dramatically less computation.**

Keywords—**divide-and-conquer, MLE, link counts, scalability, statistical estimation, traffic matrix**

I. Introduction

A traffic matrix measures the volume of traffic that a communications network carries between its source and destination nodes. Nodes may represent ingress and egress points of a backbone transit network, access or border routers at the edge of an ISP's regional POP, or subnets of an enterprise network. Traffic matrices are key to many network design, engineering, and management functions but they are often difficult to measure directly. Because networks are dynamic, analysis tools must be adaptive and computationally light weight.

Estimating a traffic matrix from aggregated traffic data such as link byte counts is an inverse problem that resembles tomography in medical imaging, so it was named *network tomography* by Vardi [1]. (This term has recently gained a broader interpretation and now includes many other inference problems on networks [2].) Cao, Davis, Vander Wiel, and Yu [3] develop a time-varying statistical model to estimate an evolving traffic matrix over time using link counts at router interfaces.

Their model is validated using data collected on a Lucent Technologies research network having only eight significant edge nodes. Their method does not scale to large networks, however. This paper proposes a computationally scalable divide-and-conquer methodology that allows traffic matrices to be estimated for networks with many more edge nodes. The main ideas of the paper can be understood by reading Sections II, III, VI, and VII and skimming introductory matter in the remaining sections. To facilitate reading, a partial list of notation appears in the Appendix.

II. Statistical Network Tomography

Computer networks consist of links connecting interfaces between routing devices. Figure 1 shows a network at Lucent Technologies that is analyzed in Section VI. The edge links in this example represent router interfaces in a local area network (LAN). If the figure represented an ISP with five local regions (the routers, *R1-4, G*) and a core transit network (the switch, *S*), the edge links would represent access points in the regional networks. The network has 18 edge nodes, $324 = 18^2$ origin-destination (OD) pairs, and 46 unidirectional links.

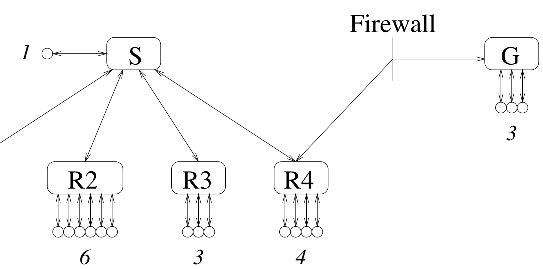

Fig. 1. A network at Lucent Technologies. Total edge nodes at each intermediate node are listed.

Cao *et al.* [3] represented link count measurements as summations of various OD counts that were modeled as independent random variables. Time-varying traffic matrices estimated from a sequence of link counts were validated on a small subnetwork of Figure 1 by comparing the estimates with actual OD counts that were collected by running *NetFlow* [4] software on the routers. Such direct point-to-point measurements are often not available because they require additional router CPU resources, can reduce packet forwarding efficiency, and involve a significant administrative burden when used on a large scale. Estimated traffic matrices matched the measured ones well, especially for moderate or large counts. Thus, for some purposes such as planning and provisioning activities estimates of OD traffic could be used as inexpensive substitutes for direct measurements.

Let $\mathbf{y} = (y_1, \ldots, y_J)'$ denote the *observed* column vector of incoming and outgoing byte counts measured on each router link interface during a given time interval and let $\mathbf{x} = (x_1, \ldots, x_I)'$ denote the *unobserved* vector of corresponding byte counts for all OD pairs in the network. Here $'$ indicates transpose and $\mathbf{x}$ is the 'traffic matrix' even though it is arranged as a column vector for convenience.

One element of $\mathbf{x}$, for example, corresponds to the number of bytes originating from a specified origin node to a specified destination node, whereas one element of $\mathbf{y}$ corresponds to bytes originating from the origin node regardless of their destination. Thus each element of $\mathbf{y}$ is a sum of selected elements of $\mathbf{x}$, so

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{1}$$

where $\mathbf{A}$ is a $J \times I$ *routing matrix* of 0's and 1's that is determined by the routing scheme of the network. In this paper, we only consider *fixed routing*, i.e. there is only one route from an origin to a destination.

As in [3] the unobserved OD byte counts are modeled as

$$x_i \sim \text{normal}(\lambda_i, \phi\lambda_i), \text{ independently} \tag{2}$$

and this implies

$$\mathbf{y} \sim \text{normal}(\mathbf{A}\boldsymbol{\lambda},\ \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}'), \tag{3}$$

where

$$\begin{aligned} \boldsymbol{\lambda} &= (\lambda_1, \ldots, \lambda_I)', \text{ and} \\ \boldsymbol{\Sigma} &= \phi \operatorname{diag}(\lambda_1, \ldots, \lambda_I). \end{aligned} \tag{4}$$

Here $\boldsymbol{\lambda} > \mathbf{0}$ s the vector of OD mean rates, and $\phi > 0$ is a scale parameter that relates the variance of the counts to their mean, since usually larger counts have larger variance. A more general power model relating the mean and variance is $\boldsymbol{\Sigma} = \phi \operatorname{diag}(\lambda_1{}^c, \ldots, \lambda_I{}^c)$, where c is a constant. Cao *et al.* [3] show that $c = 2$ fits the data for two different subnetworks of Figure 1 slightly better than $c = 1$. But only $c = 1$ provides a conceptually consistent representation in the sense that the individual and aggregated OD traffic have the same mean–variance relationship. This consistency is important to our approach of solving large network problems because we break the problem for the entire network into several subproblems that involve only a small subnetwork where many ODs are aggregated into a single entity. In this paper, $c = 1$ is assumed.

A sequence of N sets of link measurements $\mathbf{y}_1, \ldots, \mathbf{y}_N$ is assumed to be independent and identically distributed. Under model (3), the log-likelihood of the combined parameter $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \phi)$ is

$$\begin{aligned} l(\boldsymbol{\theta} \mid \mathbf{y}_1, \ldots, \mathbf{y}_N) = &-\frac{N}{2} \log |\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}'| \\ &-\frac{1}{2} \sum_{t=1}^{N} (\mathbf{y}_t - \mathbf{A}\boldsymbol{\lambda})'(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}')^{-1}(\mathbf{y}_t - \mathbf{A}\boldsymbol{\lambda}). \end{aligned} \tag{5}$$

Cao *et al.* [3] apply an expectation–maximization (EM, [5]) algorithm to produce an initial estimate of the maximum likelihood estimator (MLE) for $\boldsymbol{\theta}$, and then apply a second-order optimization method (as in [6]) to obtain the final MLE $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\lambda}}, \hat{\phi})$. Using equations (2-4), the point-to-point traffic for OD pair i at time t can be estimated by the conditional expectation

$$\hat{x}_{i,t} = \mathrm{E}[x_{i,t} | \hat{\boldsymbol{\theta}}, \mathbf{y} = \mathbf{A}\mathbf{x}, x_{i,t} > 0]. \tag{6}$$

An iterative proportional fitting (IPF) algorithm ([7],[8]) is then applied to $\hat{\hat{\mathbf{x}}}_t = (\hat{\mathbf{x}}_{1,t}, \ldots, \hat{\mathbf{x}}_{I,t})'$ to guarantee $\hat{\mathbf{x}}_t > \mathbf{0}$ and the constraints $\mathbf{y}_t = \mathbf{A}\hat{\mathbf{x}}_t$ are met. Cao *et al.* [3] also showed how the estimates can be improved by smoothing over successive blocks of time.

The method in [3] uses all available information to estimate rate parameters but it does not scale to networks with many nodes. In general, if there are N_e edge nodes, the number of floating point operations needed to compute the MLE is at least proportional to N_e^5.

III. Divide-and-conquer overview

Traffic matrix estimation for a large network can be conquered by dividing the problem into a number of smaller subproblems. The computational complexity of each subproblem can be made independent of the size of the complete network. This scalable method has three major steps.

1. Partition all OD pairs into a number of disjoint sets $S_1, S_2, \ldots, S_M$.
2. For each OD set S, select a corresponding set of link measurements to use for estimation. These link sets are not usually disjoint.
3. Estimate rate parameters for the ODs in S by using a reduced version of the traffic model that aggregates the large number of remaining rates into the fewest possible parameters.

The simplest version of this method is to treat each OD pair separately and estimate the mean for an OD pair using only measurements from the origin link and the destination link. This simple case illustrates the basic ideas in the general approach that are discussed in the sections that follow.

Suppose the mean traffic from origin o to destination d is to be estimated from link measurements $y[o]$ and $y[d]$ representing the byte counts originating from node o and destined to node d respectively. Obviously the only traffic included in both link measurements is $y[od]$, the target traffic from o to d. All other traffic originating from o is destined elsewhere and is denoted $y[o\bar{d}]$. All other traffic destined to d originates elsewhere and is denoted $y[\bar{o}d]$. These last three quantities are probabilistically independent because they are aggregates of disjoint pieces of the full OD traffic vector $\mathbf{x}$. These relationships are compactly written as

$$\begin{pmatrix} y[o] \\ y[d] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} y[od] \\ y[o\bar{d}] \\ y[\bar{o}d] \end{pmatrix} \tag{7}$$

where the components of the right-hand-side vector are independent with variances proportional to their means. Equation 7 has the same form as the original model $\mathbf{y} = \mathbf{A}\mathbf{x}$ but is much smaller and focuses on a single OD pair. Estimates of the three rate parameters in this reduced model are easy to compute and the first parameter, $\lambda[od] = \mathrm{E}y[od]$, is the one that we aim to estimate. The full vector $\boldsymbol{\lambda}$ can be estimated by solving one such problem for each OD pair. The biggest drawback of estimating each OD rate in this way is that too much relevant information is ignored. Sections that follow describe how to divide and conquer using somewhat larger subproblems and more link measurements.

Equation (7) has four properties that are important to preserve in developing a more for the general divide-and-conquer approach.

1. The linear system is small.
2. The right-side vector has independent elements, one of which is the target OD traffic.
3. The right-side matrix has only 0-1 elements.
4. A sample of left-side link measurements is sufficient to estimate the target rate parameter.

Independence and a 0-1 matrix keep the optimization problem uncomplicated because the constraints on the estimates are simple: each parameter has to be positive. If the right-side aggregates were not disjoint, they would be statistically dependent and their parameters would have general linear constraints that would be inherited from the positivity constraints on $\boldsymbol{\lambda}$. The property, termed *estimability*, ensures that $\lambda[od]$ can be estimated from the data. This can be established by noting that

$$\begin{aligned}\phi\lambda[od] &= \mathrm{Var}(y[od]) = \mathrm{Cov}(y[o], y[d]), \text{ and} \\ \phi &= \mathrm{Var}(y[o])/\mathrm{E}(y[o]).\end{aligned}$$

Thus, the moments of $y[o]$, and $y[d]$ completely determine $\lambda[od]$.

All three rate parameters can be estimated from the selected link measurements. This is not necessarily true for general partitions. The target parameters, however, are *always* estimable if their origin and destination links are included in the measurement set.

Some additional notation is needed for general partitions (also see the Appendix.)

An OD pair is written $i = (o, d)$ and an upper-case letter, such as S, denotes a set of OD pairs. A bold uppercase denotes an ordered collection of OD subsets, as in $\mathbf{S} = \{S_1, \ldots, S_m\}$. The complement of S is $\bar{S}$, and intersections are denoted either $S_1 S_2$ or $S_1 \cap S_2$.

Associated with each physical link l is the set L of OD pairs that utilize the link. For ease of exposition, both entities are referred to as the *link*. A bold $\mathbf{L}$ denotes an ordered collection of links as in $\mathbf{L} = \{L_1, \ldots, L_m\}$.

The total traffic count (at a given time) for any OD subset S is

$$y[S] = \sum_{i \in S} x_i$$

with corresponding mean $\lambda[S]$. In general, $y[S]$ is not observable unless S represents a link. Column vectors of these aggregates are written as

$$\mathbf{y}[\mathbf{S}] \quad = \quad (y[S_1], \ldots, y[S_m])'$$

and the same convention is used for $\boldsymbol{\lambda}[\mathbf{S}]$.

IV. Forming subproblems

At the heart of the divide-and-conquer methodology is a partition of the OD pairs into disjoint groups and selection of link sets from which to estimate each group's rate parameters. This section gives heuristic approaches to forming these subproblems. OD pairs are partitioned using distance scores and a clustering algorithm. Then links are selected for each cluster to balance information loss against computational cost. Often, the resulting subproblems are associated with natural subnetworks.

A. Grouping OD pairs

Routing topology usually comes into play when deciding how to group OD pairs. For example, *hierarchically routed* networks have various domains of edge-nodes. Traffic with origin and destination in the same domain stays within that domain, and traffic from one domain to another domain goes through a single set of intermediate gateways.

In general it is intuitive to group the OD pairs as follows.

1. Split all edge nodes into a number of domains.
2. Place all OD pairs into the same group that are originating from and destined to the same pair of domains.

Figure 2 shows the topology for a two router network. Routers $R1, R2$ each have two edge nodes and are interconnected by an intermediate link. In this case, the four edge nodes are naturally split into domain (1,2) and domain (3,4) according to their nearest router. Taking account of origin domain and destination domain produces four OD groups as shown in the top panel of Figure 3.

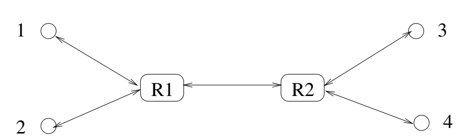

Fig. 2. A two router network. $R1$ and $R2$ are routers. $1, 2, 3,$ and 4 are end nodes.

A general version of this domain-based grouping considers the amount of overlap between routing paths. This

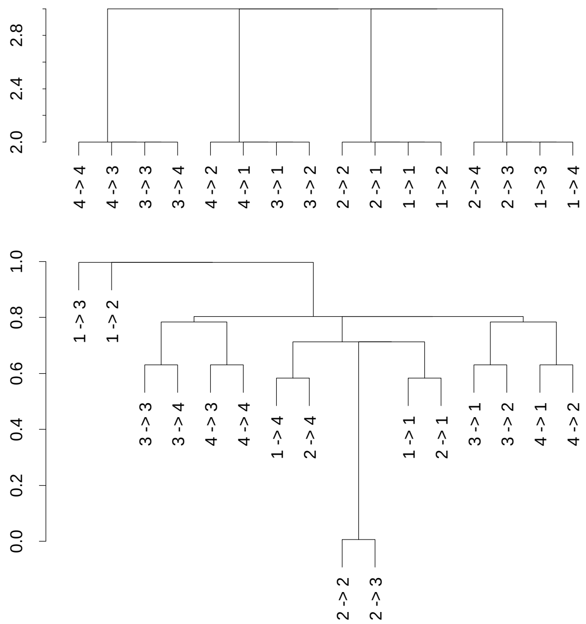

Fig. 3. Clustering tree of OD pairs for the two router network in Figure 2, using routing path distance (8) (top figure) and information distance (9) (bottom figure). The Y-axes are distance scores.

gives more flexible grouping that is especially helpful when edge node domains are not readily apparent. There are two steps to the process.

1. Define a distance between OD pairs based on routing.
2. Build a clustering tree using the distance metric.

For example, for any two OD pairs i and j, the distance could be defined as

$$d(i,j) = \text{number of links used by either } i \text{ or } j \text{ but not both.} \tag{8}$$

Clustering methods work from distance scores which, in this case, measure the separation between OD pairs in a network. Given the distance metric, hierarchical clustering (see [9]) can be used to partition the ODs. Distances between clusters can be measured using *single linkage*, which is the minimum distance between cluster elements. This is sensible because all closely related OD pairs are placed in a single cluster. ODs are partitioned by cutting the tree at a specified level. To control complexity of subproblems, it is important to restrict the size of each cluster. A single level-cut may work well for a tree that is well balanced. But for an ill-balanced tree, multi-level cuts or other separation methods must be used to determine the clusters.

The clustering tree using the distance (8) is shown in the top panel of Figure 3, where the Y-axis is the distance score. Cutting the tree just above the level 2 produces the four natural domain-based groups.

Other non-routing based distance metrics can be defined for OD pairs. For example, information about rate parameters comes from the probability distribution $f(\mathbf{y};\boldsymbol{\lambda},\phi)$ of link byte counts, so a distance can be based on aspects of this distribution. A natural candidate is the information matrix which is defined as

$$I_{ij}(\boldsymbol{\lambda}) = -\mathrm{E}[\partial^2 \log(f)/\partial\lambda_i\partial\lambda_j] = \langle\partial_i,\partial_j\rangle,$$

where $\langle\cdot,\cdot\rangle$ is an inner product and ∂_i, ∂_j are tangent vectors at $\boldsymbol{\lambda}$ in the coordinate directions λ_i and λ_j (See [10] and its references.) Here ∂_i can be interpreted as the information direction corresponding to λ_i in the sense that changing the distribution in the direction of ∂_i will change λ_i. The information matrix for the link measurement model (equations 1-3), is shown in equation (5) of [3]. The distance score between OD pairs i and j is then

$$d(i,j) = 1 - \frac{|I_{ij}|}{\sqrt{I_{ii}I_{jj}}}, \tag{9}$$

i.e. one minus cosine of the angle between the tangent vectors. There are two disadvantages to using this information-based distance. First, computation cost is of order N_e^4 for a network of N_e edge nodes. Second, the distance depends on the unknown rate parameters $\boldsymbol{\lambda}$, which are likely to change over time.

A clustering tree using the information distance (9) is shown in the lower panel of Figure 3. To demonstrate that the tree reflects information directions, the rate parameters were taken to be 1000 for OD pairs $(1 \to 3), (1 \to 2)$, and 1 for the rest. The information-based tree is similar to the one based on routing. (In fact, when all rate parameters have the same value, the trees are identical). So information and routing paths are somewhat related. However, the tree based on information is more complex and reveals more structure. For example, OD pairs $(1 \to 3), (1 \to 2)$ stand out and seem to have their own independent information direction. (Recall that their rate parameters are 1000, far larger than the others.)

B. Link selection according to network topology

For the target OD pairs S in each subproblem, a set of corresponding links must be specified that are informative for estimating the target rate parameters. There are two conflicting factors to consider: estimation accuracy and computational cost. The more links used for estimation, the better the accuracy but the higher the computing costs because additional parameters must be included in the optimization. A balance is needed between accuracy and cost.

In Section V below, the number of parameters to optimize over is shown to be at most $O(n_l^H)$ where n_l is the number of links and H is the maximum number of hops from an origin to a destination in the set of links $\mathbf{L}$ for the subproblem. As few as $O(n_l^2)$ parameters are needed for certain networks (e.g. networks in Results 5 and 6.) Given this relation between the number of links and the number of parameters, a limited number of links should be selected in each subproblem.

A choice of links. For a target set S of OD pairs, let $\mathbf{o} = \{o_i\}, \mathbf{d} = \{d_j\}$ be the n_1 origins and n_2 destinations respectively. We wish to construct a collection $\mathbf{L}$ of links to use in estimating the rate parameters of S. Section III showed that the origin-destination link counts are sufficient to estimate the corresponding OD traffic rate and this suggests that $\mathbf{L}$ should include at least these links. It remains to determine other links to include in $\mathbf{L}$. For the moment, all the OD pairs in S are assumed to share the same path except the origin and the destination link. Let $R_1, \ldots, R_h$ represent intermediate nodes along the route from $\mathbf{o}$ to $\mathbf{d}$. That is,

$$\mathbf{o} \Rightarrow R_1 \to R_2 \to \ldots \to R_{h-1} \to R_h \Rightarrow \mathbf{d}, \qquad (10)$$

where $\Rightarrow$ represents multiple links (because of multiple origins and destinations) and $\to$ represent a single link. Let $\mathbf{J}$ be the set of links in subnetwork (10), and $\bar{\mathbf{J}}$ be the remaining links in the network. The probability distribution of link byte counts $\mathbf{y}[\mathbf{J}]$ depends on the target rate parameters of S but that of $\mathbf{y}[\bar{\mathbf{J}}]$ does not. This can be interpreted as link measurements $\mathbf{y}[\mathbf{J}]$ contain primary information on the target rate parameters while $\mathbf{y}[\bar{\mathbf{J}}]$ provide secondary information. Therefore, it seems reasonable to also include links the intermediate $\mathbf{J}$ in $\mathbf{L}$ but not all the links of $\bar{\mathbf{J}}$.

Heuristics will now be used to select additional links for $\mathbf{L}$. In subnetwork (10), $R_1, \ldots, R_h$ are intermediate nodes. Any traffic arriving at R_i will be forwarded to another node. Adding all links that are connected to the intermediate nodes in subnetwork (10) gives

$$\begin{array}{ccccccc} \mathbf{o} & \Rightarrow & R_1 & \longleftrightarrow \ \ldots \ \longleftrightarrow & R_h & \Rightarrow & \mathbf{d} \\ & & \Uparrow\Downarrow & & \Uparrow\Downarrow & & \\ & & \mathbf{r}_1\tilde{\mathbf{r}}_1 & & \mathbf{r}_h\tilde{\mathbf{r}}_h & & \end{array} \qquad (11)$$

where $\mathbf{r}_i$ and $\tilde{\mathbf{r}}_i$ represent possibly multiple remaining incoming and outgoing links on R_i. For example, $\tilde{\mathbf{r}}_1$ includes the outbound links to $\mathbf{o}$. To limit the number of links in subnetwork (11), the topology is simplified. Each link set $\mathbf{r}_i$ and $\tilde{\mathbf{r}}_i$ is aggregated to form *superlinks* r_i and $\tilde{r}_i$. Such a aggregation of links is legitimate, because no traffic can pass through two of these incoming or outgoing links in loopless routing. After combining links, the subnetwork becomes

$$\begin{array}{ccccccc} \mathbf{o} & \Rightarrow & R_1 & \longleftrightarrow \ \ldots \ \longleftrightarrow & R_h & \Rightarrow & \mathbf{d} \\ & & \uparrow\downarrow & & \uparrow\downarrow & & \\ & & r_1\tilde{r}_1 & & r_h\tilde{r}_h & & \end{array} \qquad (12)$$

This subnetwork is *complete* in the sense that any traffic entering an internal node on one link will leave on another while the edge nodes are pure sources and sinks. In this subnetwork, $\mathbf{o}$ and the r_i are the only possible origins and $\mathbf{d}$ and the $\tilde{r}_i$ are the only possible destinations. Also, at each intermediate node R_i, the total incoming traffic balances the total outgoing traffic and therefore one redundant link can be removed per intermediate node. For example, the reverse links $R_i \leftarrow R_{i+1}$ can be removed without losing information. Thus, subnetwork (12) displays the final choice of link set $\mathbf{L}$ except the reverse links are removed because they are redundant.

Until now, the target OD pairs $\mathbf{S}$ are assumed to have the same routing path except the first and last hop. However, the link selection strategy can be easily generalized to an arbitrary set of target OD pairs.

Algorithm 1: Start with an empty link set $\mathbf{L}$.

1. Add the target origin links and destination links to $\mathbf{L}$.
2. Add to $\mathbf{L}$ intermediate links used by target ODs as well as their reversals (that is, the same links in the opposite direction.)
3. At each intermediate node R_i along the routing paths of target ODs, combine the remaining incoming links into a superlink and add it to $\mathbf{L}$. Similarly, add outgoing superlinks to $\mathbf{L}$.
4. Remove redundant links from $\mathbf{L}$.

Toward the center of the routing path (or the core of the network), more aggregation with interfering traffic is likely. Therefore, link traffic within the center may provide little additional information. To further reduce the link set, intermediate links and superlinks in the center can be excluded from in Step 2 and 3.

Information loss and recovery. From an information viewpoint, $\mathbf{L}$ should include the most informative set of links for estimating the rate parameters for S. Determining these links is a complicated optimization problem. Starting with the origin and destination links, a greedy version would sequentially select each new link to give the biggest increase in information on the target rate parameters. 'Information,' in this context, could be the inverse variance product for the target rate estimates. The information depends on the unknown rate parameters but our experience suggests that the estimation efficiency of links selected using Algorithm 1 is not overly sensitive to the rate parameters, (also see the application in Section VI.)

V. Grouping ODs to reduce parameters

In each subproblem of the simple method described in Section III, the two link measurements, $y[o]$ and $y[d]$, are re-expressed using the three statistically independent aggregated byte counts, $y[od], y[o\bar{d}]$, and $y[\bar{o}d]$ (equation 7). In this reduced form, each subproblem contains only three rate parameters and is inexpensive to solve. This can be done for an arbitrary set of links $\mathbf{L}$ used in each subproblem.

There are a few theorems in this section. To keep the paper focused and concise, proofs are omitted. An extended version of the paper is available from the authors.

For a given link set $\mathbf{L} = \{L_i\}$, group together all ODs that use exactly the same combination of links from $\mathbf{L}$. Let $\mathbf{P} = \{P_j\}$ denote this disjoint collection of OD sets. The aggregated byte counts $\mathbf{y}[\mathbf{P}]$ have the following properties.

1. The elements of $\mathbf{y}[\mathbf{P}]$ are independent.
2. Each link measurement in $\mathbf{y}[\mathbf{L}]$ is a sum of some elements from $\mathbf{y}[\mathbf{P}]$; i.e.

$$\mathbf{y}[\mathbf{L}] = \mathbf{A}^*\mathbf{y}[\mathbf{P}],$$

for some matrix $\mathbf{A}^*$ with elements 0 or 1.

Thus, link counts $\mathbf{y}[\mathbf{L}]$ are represented in the same form as in the full problem (equations 1-3), but in terms of reduced number of independent byte counts $\mathbf{y}[\mathbf{P}]$. This is the generalization of the simple strategy leading to equation (7). If $\mathbf{L}$ contains the origin and destination links for the target OD pairs, rate parameters for these OD pairs are estimable. Therefore methods developed in [3] (reviewed in Section II) can be directly applied to solve this subproblem. In addition, it can be shown that the grouping $\mathbf{P}$ is *optimal* among all groupings of OD pairs that satisfy the two properties above, in the sense that it has the smallest total elements. For this reason, this grouping will be referred as the *optimal grouping*.

Following are two iterative algorithms that efficiently construct the optimal grouping $\mathbf{P}$. Let $\mathbf{L} = \{L_i, 1 \le i \le n_l\}$, and let $\mathbf{C}$ be the set of all intersections of any number of elements in $\mathbf{L}$, i.e. $\mathbf{C} = \{L_{i_1} \cap \ldots \cap L_{i_m} \mid (i_1, \ldots, i_m)$ is an index set$\}$. Write $\mathbf{C} = \{c_j, 1 \le i \le n_c\}$, where c_j are ordered in cardinality from smallest to largest.

Algorithm 2: Set $\mathbf{P}$ be an empty set. For each $L_i \in \mathbf{L}, i = 1, \ldots, n_l$, let U be the union of all elements in $\mathbf{P}$, update $\mathbf{P}$ by

$$\mathbf{P} \longleftarrow \{L_i P, \bar{L}_i P, L_i \bar{U} \mid P \in \mathbf{P}\}.$$

Algorithm 3: Set $\mathbf{P}$ be an empty set. For each $c_j \in \mathbf{C}, j = 1, \ldots, n_c$, let U be the union of all elements in $\mathbf{P}$, update $\mathbf{P}$ by

$$\mathbf{P} \longleftarrow \mathbf{P} \cup \{c_i \bar{U}\}.$$

Algorithm 2 runs through the link set $\mathbf{L}$ to generate the grouping. Algorithm 3 uses all intersections of $\mathbf{L}$. For networks in which the set of all link intersections is the same as the set of all pairwise intersections (e.g. networks in Result 6), Algorithm 3 is efficient since the set of all pairwise intersections will usually be precomputed to speed up the optimization algorithm.

Algorithm 3 implies that the total number of elements n_p of the optimal grouping $\mathbf{P}$ is no larger than that of $\mathbf{C}$. This fact can be used to bound n_p. A tighter bound is obtained when links $\mathbf{L}$ are selected as in Algorithm 1. For this, a special case of fixed routing, *consistent routing*, is considered. Consistent routing means that there is only one route between any two nodes, including both intermediate and edge nodes. This routing property is usually a result of routing cost optimizations. Although this may not hold for the full network, it may hold for the subnetwork under study. A main conclusion from these bounds is that n_p only depends on the subnetwork size in the subproblem and is independent of the size of the full network. Therefore as long as the subnetwork size is limited, each subproblem is of limited computational complexity.

Result 4: For a network with at most H hops from an origin to a destination,

$$n_p \le \sum_{i=1}^{H} \binom{n_l}{i}.$$

In particular, $n_p \le 2^{n_l} - 1$.

Result 5: Suppose $\mathbf{L}$ is chosen as in Algorithm 1. Let n_1, n_2, h be the number of origination, destination and intermediate nodes respectively. Then

1. for fixed routing,

$$n_p \le (n_1 + n_2 + 1)2^{3h-1} + n_1 n_2 - 1.$$

2. for consistent routing,

$$n_p \le (n_1 + h)(n_2 + h).$$

Not all parameters $\boldsymbol{\lambda}[\mathbf{P}]$ of the optimal grouping $\mathbf{P}$ can be estimated. It is important to identify the estimable parameters so that their estimates can be extracted. This is not a problem for the target OD pairs in the subproblem, as long as their origin and destination links are included in the selected links. However when all rate parameters are estimable, there are no redundancies. The following result gives specification of networks for which this is true. In this case, Algorithm 3 implies the total number of parameters in $\boldsymbol{\lambda}[\mathbf{P}]$ is less than $n_l(n_l + 1)/2$.

Result 6: All rate parameters $\boldsymbol{\lambda}[\mathbf{P}]$ for the optimal grouping $\mathbf{P}$ are estimable,

1. if routing is consistent, and if traffic on any two links traverse in a unique order.
2. if routing is consistent and hierarchical, and if traffic on any two links in the same domain traverse in a unique order.
3. if routing is consistent and symmetric.

VI. Application to Multi-Router Network

We consider a real network using four different estimation methods to show that the divide-and-conquer methodology can perform well compared to full-information maximum likelihood. Byte counts measurements from the network depicted in Figure1 have been recorded every five minutes since August 1998. The network belongs to Lucent Technologies and serves many hundreds of business users. Measurements are gathered automatically through standard SNMP polling of link interfaces on routers. Although the actual network contains about 50 edge nodes, we pooled many of the low-usage links with higher-usage links to produce the skeleton version shown in Figure1. The primary reason for aggregating nodes was to reduce the network to a size that full-information maximum likelihood estimates could be computed within time and memory limitations.

The example network has six intermediate nodes, *R1, R2, R3, R4, S, and G*, 18 edge nodes (the small open circles), and 46 unidirectional links. Among the intermediate nodes, *R1* through *R4* are routers, S is a backbone switch and G is a gateway router to the Internet. The total traffic coming into each intermediate node matches the total going out. Removing these redundancies leaves 40 independent link measurements and $18^2 = 324$ OD pairs. Routing is hierarchical: traffic between any pair of routers *R1* through *R4* goes through the backbone switch *S*. Traffic from inside the corporate network to the Internet must traverse the *R4–G* link.

The best way to validate our traffic matrix estimates would be to compare them with direct measurements of point-to-point traffic. This is feasible on a small scale using, for example, Cisco's *NetFlow* collector in conjunction with CAIDA's *Cflowd* tool [11]. Our own experience ([3]) and that of others ([12]) shows that synchronization problems can make reliable data collection difficult. An alternative is to estimate the mean traffic for each OD pair from actual link measurements but then analyze simulated data from the estimated model. This has the advantage that direct verification is possible by comparing estimates to known values for both theoretical mean OD traffic, $\boldsymbol{\lambda}$ and the, typically unobservable, sequence of OD byte count vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

Figure 4 plots the 324 OD rate parameters (elements of $\boldsymbol{\lambda}$) that we used to generate simulated traffic. These come from an estimate of $\boldsymbol{\lambda}$ based on a sequence of $N = 25$ link measurements collected over two hours on a morning in October 1998. The vertical scaling is $\log(100 + \lambda)$ where the rates λ are measured in bytes/second. The rates vary by more than three orders of magnitude and have a highly skewed distribution. This is consistent with our expectations and our previous experience with directly measured traffic matrices.

Independent OD vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_N$, were then simulated with means $\boldsymbol{\lambda}$ and appropriate variances. The corresponding link measurements are easily obtained as $\mathbf{y}_t = \mathbf{A}\mathbf{x}_t$. The following results use the link data $\{\mathbf{y}_t\}$ to estimate $\boldsymbol{\lambda}$ and $\{\mathbf{x}_t\}$ with four methods.

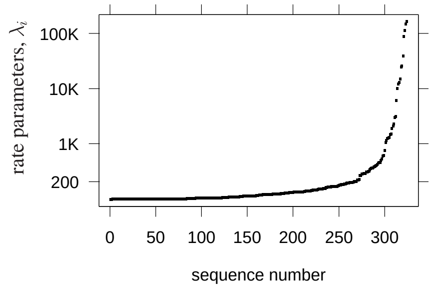

Fig. 4. Rate parameters $\boldsymbol{\lambda}$ (bytes/sec) used for simulation.

Two methods represent the divide and conquer extremes: *FULL* refers to the full-information method using all links to estimate all OD parameters as in [3]. *SIMPLE* represents the other extreme where each OD uses only the corresponding origin link and destination link.

The remaining two methods partition OD pairs based on grouping the routers into either three or five domains:

DIVIDE-3: $\{S, R1, R2\}, \{R3, R4\}, \{G\}$, and

DIVIDE-5: $\{S, R1\}, \{R2\}, \{R3\}, \{R4\}, \{G\}$.

In each case OD pairs are partitioned according to their origin and destination domains. For example, in *DIVIDE-5*, all pairs with origin node on $R2$ and destination node on $R3$ are grouped into the same subproblem. This is a special case of clustering nodes based on routing similarity as discussed in Section IV-A. Corresponding to each set of ODs, a set of links is then selected using the scheme of Section IV-B. This completes the *divide* portion of the method. Since routing is consistent and symmetric, there are no redundancies in the rate parameters of each subproblem (Result 6).

The complexity of the four approaches is summarized in top portion of Table I. The number of floating point operations (flops) required for maximizing the likelihood drops dramatically as the network is divided into more pieces. In this respect *DIVIDE-5* is a factor of 10 better than *FULL*. This is true even though more subproblems use more total links and estimate more total parameters because they overlap with one another. The flop ratios are based on theoretical scaling results for a large number of edge nodes.

TABLE I

Four estimation methods: complexity and estimation performance

Method	*FULL*	*DIVIDE-3*	*DIVIDE-5*	*SIMPLE*
flop ratio	1/1	1/3	1/10	1/3000
subproblems	1	9	25	324
total links	40	169	306	648
total parameters	324	658	822	972
$\boldsymbol{\lambda}$ error scaling	1	1.15	1.39	2.28
$\mathbf{x}$ error scaling	1	1.05	1.24	1.72

For a sample of size $N = 25$, Figure 5 compares estimates to true values of both $\boldsymbol{\lambda}$ (upper) and a representative $\mathbf{x}$ (lower) from the simulated sample. Large traffic rates (both λ and x) are estimated well by all of the divide and conquer methods. Small rates are more difficult to estimate and the upper display shows the accuracy degenerating as the network is divided into more pieces. Inaccuracy in estimating x (lower), however, does not degenerate as much. This is due to the fact that $\mathbf{x}$ estimates are constrained by $\mathbf{y} = \mathbf{Ax}$ which can be fairly restrictive when there are large variations among the link counts in $\mathbf{y}$. The last two rows of Table I report the increase in error sums of squares on the $log(100+x)$ scale. Additional simulations run with samples of size 50, 100, 200 and 400 produce similar results except that accuracy improves with sample size.

Although the primary focus has been on estimating $\boldsymbol{\lambda}$, there is an issue regarding the scale parameter ϕ because each subproblem produces a separate estimate. This has advantages and disadvantages. The advantage is that the method is robust to different scaling behavior across a large network because scale is only estimated locally to the subproblems. On the other-hand, if the scale is consistent across the network estimating it globally could produce statistically better estimates of $\boldsymbol{\lambda}$, especially if the subproblems are small.

A second 'local vs. global' issue involves estimating $\mathbf{x}$. There are three approaches: (i) use equation (6) and iterative proportional fitting (IPF) within each subproblem and assemble estimates for the target ODs to produce the complete estimate; (ii) as above except where subproblems overlap, average the overlapping pieces before assembling; (iii) assemble the complete estimate of $\boldsymbol{\lambda}$, and then use equation (6) and IPF globally. Figure 5 uses the second approach with straight averaging, although one could reasonably use weighted averaging with weights inversely proportional to uncertainty measured by the inverse information matrix.

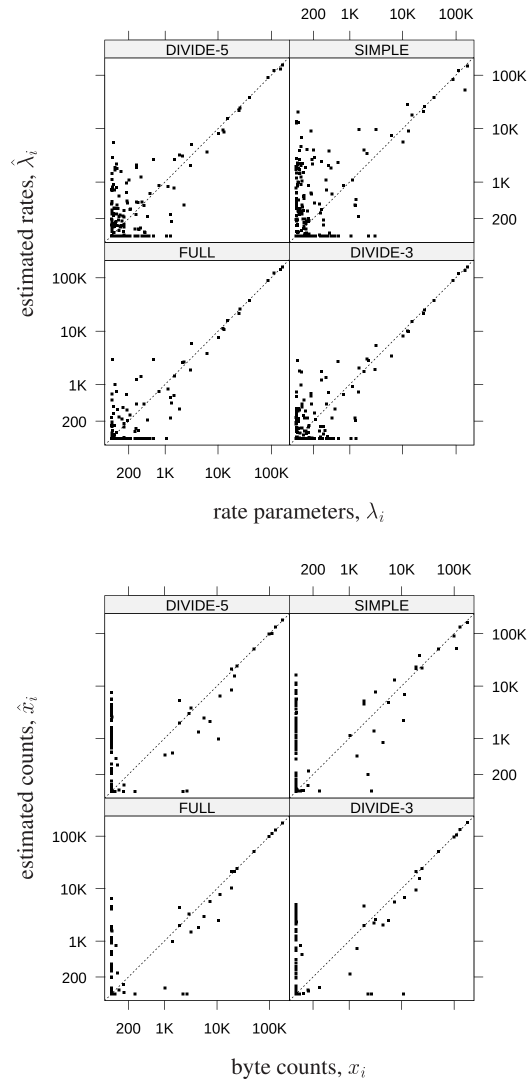

Fig. 5. Estimates of $\boldsymbol{\lambda}$ and $\mathbf{x}$ (bytes/sec) against true values for the four methods ($N = 25$).

VII. Discussion

A computationally scalable divide-and-conquer methodology has been developed to estimate traffic matrices for networks with a large number of edge nodes. The idea is to divide the estimation problem for a large network into several smaller subproblems and then solve each of these subproblems. For a network with N_e edge nodes, the computational complexity can be reduced from $O(N_e^5)$ to $O(N_e^2)$ through this approach.

There are several additional advantages of the proposed method. First, since we break up the full problem into *independent* subproblems, parallel computing could easily be applied with little need for communicating among processors or a central point of control. Second, since each subproblem involves only limited portion of the full network, the method is adaptive and more robust to possible misspecification of the full model as, for example, if scale parameters vary over different regions of the network. Third, the method can be easily adapted to deal with unreliable or missing link measurements. These links can be simply excluded from the sets of measurements used by subproblems. Fourth, a variety of related problems fit within this framework. For example, some routers are able to measure byte counts for at least some destination nodes. Often, however, other routers in the network do not have this capability. In this case, the traffic matrix is partially observed and this information can be combined with link measurements to estimate the complete traffic matrix. These situations are appealing because there is better control on the number of unknowns per measured quantity.

Several possibilities for future work fall into two broad categories—refinement and extension. Some possible refinements are: i) develop a theory for link selection based, for example, on information measures; ii) develop methods for combining estimates from overlapping subproblems;

iii) explore the possibility of using estimates obtained from one subproblem as supplemental information for another.

Regarding extensions, an immediate need is confidence and percentile estimation. A bound on the 95th percentile during times of peak network usage would be extremely useful for planning and provisioning. By using the inverse information matrix to measure uncertainty of the parameter estimates, combining this with the conditional distribution of $\mathbf{x}$, it may be straightforward to produce reasonably good quantile bounds. Another possibility is to extend the methods to accommodate dynamic routing. Some of the developed here can be fairly easily generalized if the link traffic can be expressed as weighted sums of the traffic between OD pairs. A final extension, is to include packet loss in the traffic model. Some per-link information of packet loss may be available from router interfaces, but this would need to be incorporated into the procedure for estimating traffic matrices.

Acknowledgments

We thank Drew Davis, Tom Limoncelli and Lookman Fazal setting up *MRTG*, *NetFlow*, and *Cflowd* data collection. We also thank Mark Hansen for helpful discussions and Diane Lambert for many good suggestions on improving the presentation of this paper.

Appendix: Notation

o, d, r	origin, destination, and generic nodes
n_l	number of links in a subnetwork
$i = (o, d)$	OD node pair
L, L_1	OD pairs with traffic on links l and l_1
S, S_1	sets of OD pairs
$\mathbf{S} = \{S_1, \ldots, S_m\}$	ordered collection of OD sets
$\mathbf{L} = \{L_1, \ldots, L_m\}$	ordered collection of link sets
$S_1S_2 = S_1 \cap S_2$	intersection
x_i	byte count for OD pair i
$\mathbf{x} = (x_1, \ldots, x_I)'$	all OD byte counts, unobserved
$\mathbf{y} = (y_1, \ldots, y_J)'$	all link byte counts, observed
$\boldsymbol{\lambda} = \mathrm{E}\mathbf{y}$	mean byte counts for all OD pairs
N_e	numbers of edge nodes in the whole network
$y[S] = \sum_{i \in S} x_i$	byte count total over OD pairs in S
$\lambda[S] = \mathrm{E}y[S]$	mean byte count total over S
$\mathbf{y}[\mathbf{S}] = (y[S_1], \ldots, y[S_m])'$	byte count totals over $S_1, \ldots, S_m$
$\lambda[\mathbf{S}] = (\lambda[S_1], \ldots, \lambda[S_m])'$	byte count means over $S_1, \ldots, S_m$
ϕ	scale parameter
$\mathbf{A}$	fixed $J \times I$ routing matrix

References

[1] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Association*, vol. 91, pp. 365–377, 1996.

[2] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Transanctions on Information Theory*, vol. 45, no. 7, pp. 2462–2480, 1999.

[3] J. Cao, D. Davis, S. Vander Wiel, and B. Yu, "Time-varying network tomography: router link data," *Journal of American Statistical Association*, 2000, in print.

[4] Cisco, *NetFlow FlowCollector 3.0*, Author, 2000, http://www.cisco.com/warp/public/732/netflow.

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

[6] D. M. Gay, "Algorithm 611: Subroutines for unconstrained minimization using a model/trust-region approach," *ACM Trans. Math. Software*, vol. 9, pp. 503–524, 1983.

[7] H. H. Ku and S. Kullback, "Interaction in multidimensional contingency tables: an information theoretic approach.," *J. Res. Nat. Bur. Standards*, vol. 72, pp. 159–199, 1968.

[8] C. T. Ireland and S. Kullback, "Contingency tables with given marginals," *Biometrika*, vol. 55, pp. 179–188, 1968.

[9] G.A.F. Seber, Ed., *Multivariate Observations*, John Wiley & Sons, Inc, 1984.

[10] R. Kass, "The geometry of asymptotic inference," *Statistical Science*, vol. 4, no. 3, pp. 188–234, 1999.

[11] The Cooperative Association for Internet Data Analysis, *Cflowd Flow Analysis Software (version 2.0)*, Author, 1999, www.caida.org/Tools/Cflowd.

[12] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational ip networks: Methodology and experience," *Proc. ACM SIGCOMM*, August/September 2000.