Robust Traffic Matrix Estimation with Imperfect Information: Making Use of Multiple Data Sources [*]

Qi (George) Zhao
College of Computing
Georgia Institute of Technology

Zihui Ge   Jia Wang
AT&T Labs–Research

Jun (Jim) Xu
College of Computing
Georgia Institute of Technology

ABSTRACT

Estimation of traffic matrices, which provide critical input for network capacity planning and traffic engineering, has recently been recognized as an important research problem. Most of the previous approaches infer traffic matrix from either SNMP link loads or sampled NetFlow records. In this work, we design novel inference techniques that, by statistically correlating SNMP link loads and sampled NetFlow records, allow for much more accurate estimation of traffic matrices than obtainable from either information source alone, even when sampled NetFlow records are available at only a subset of ingress. Our techniques are practically important and useful since both SNMP and NetFlow are now widely supported by vendors and deployed in most of the operational IP networks. More importantly, this research leads us to a new insight that SNMP link loads and sampled NetFlow records can serve as "error correction codes" to each other. This insight helps us to solve a challenging open problem in traffic matrix estimation, "How to deal with dirty data (SNMP and NetFlow measurement errors due to hardware/software/transmission problems)?" We design techniques that, by comparing notes between the above two information sources, identify and remove dirty data, and therefore allow for accurate estimation of the traffic matrices with the cleaned data.

We conducted experiments on real measurement data obtained from a large tier-1 ISP backbone network. We show that, when full deployment of NetFlow is not available, our algorithm can improve estimation accuracy significantly even with a small fraction of NetFlow data. More importantly, we show that dirty data can contaminate a traffic matrix, and identifying and removing them can reduce errors in traffic matrix estimation by up to an order of magnitude. Routing changes is another a key factor that affects estimation accuracy. We show that using them as the a priori, the traffic matrices can be estimated much more accurately than those omitting the routing change. To the best of our knowledge, this work is the first to offer a comprehensive solution which fully takes advantage of using multiple readily available data sources. Our results provide valuable insights on the effectiveness of combining flow measurement and link load measurement.

[*]The work of Qi Zhao and Jun Xu is supported in part by NSF grant NETS-NBD 0519745 and NSF CAREER Award ANI 0238315.

Categories and Subject Descriptors

C.2.3 [COMPUTER-COMMUNICATION NETWORKS]: Network Operations - Network Monitoring

General Terms

Algorithms, Measurement, Theory

Keywords

Network Measurement, Traffic Matrix, Statistical Inference

1. INTRODUCTION

A traffic matrix captures the aggregate traffic volume traversing from every ingress point to every egress point in a network over a given time interval. Obtaining accurate traffic matrix is essential in a number of network management tasks in operational IP networks such as capacity planning and traffic engineering, etc. Realtime traffic matrix also enables online diagnosis and mitigation of network events such as network device failures, routing changes, and traffic congestion. For example, when a link fails, the network operators need to determine whether such an event will cause congestion based on the current traffic matrix and routing configurations, and re-optimize OSPF link weights to alleviate the traffic congestion. Also, without an accurate traffic matrix, network operators are not able to diagnose the severity of network events and evaluate the effectiveness of possible solutions.

Estimating traffic matrix has been recognized as a challenging research problem. Direct precise measurement of traffic matrix on large IP networks is not feasible due to the large number of Origin-Destination (OD) pairs, the high volume of traffic at each link/interface, and the lack of measurement infrastructure. Most of the prior approaches attempted to infer traffic matrix from related information that can be readily measured using existing network measurement infrastructure such as SNMP link loads and sampled NetFlow data, and periodic snapshots of network routing configuration. They can be roughly classified into two categories based on the input data they use for inference. The first type of approaches, including [16, 2, 19, 20, 11, 15], infers traffic matrix indirectly from observed SNMP link loads. This inference problem however is severely underconstrained as the number of SNMP link loads is usually far less than that of the traffic matrix elements. Consequently, the accuracy of these techniques is limited to around 10% error, which may not be good enough for certain network management tasks such as fault diagnosis[1]. The second type of approaches,

[1]Recent work [11, 15] propose changing the Interior Gateway Protocol (IGP) link weights to obtain more information and hence more accurate traffic matrix estimations (i.e., around 5%). But it

including [5, 12, 22], estimate traffic matrices directly from flow measurements data gathered on routers through (sampled) Cisco NetFlow.

In this work, we propose and answer the following question —

If both SNMP link loads and sampled NetFlow records are available, can we combine them to obtain more accurate estimation of traffic matrices than those obtained from any of the data sources? and how?

We extend our solution to accommodate scenarios in which NetFlow records are available on only a subset of ingress and egress nodes. These techniques are practically important and useful since both SNMP and NetFlow are now widely supported by vendors and deployed in most of the operational IP networks. While a few prior work such as [9, 20] briefly mentioned this problem, this work is the first to offer a comprehensive solution which fully takes advantage of using multiple readily available data sources to achieve better estimation accuracy[2].

More importantly, our work leads to a new insight that SNMP link counts and sampled NetFlow records can serve as "error correction codes" to each other. This insight helps us to solve another challenging open problem in traffic measurement —

How to deal with dirty data (i.e., measurement errors in SNMP and NetFlow due to hardware, software or transmission problems)?

We design techniques that, by comparing notes between the above two information sources, identify and remove dirty data. This capability of removing dirty data not only improves the accuracy of traffic matrix estimation, but may also benefit a number of other applications that depend on these data.

In this work, we make four important contributions. First, we provide a comprehensive formulation and design an algorithm for estimating traffic matrix during a given time interval by using both SNMP link loads and sampled NetFlow data. The algorithm simply uses the traffic matrix estimated solely based on NetFlow data as a prior and further calibrates it using the SNMP link loads in a well-designed weighted manner. We find that under the existing configuration of sampled NetFlow the prior from it is pretty accurate already and our algorithm by combining SNMP link loads improve the estimation accuracy slightly (5% improvement of the weighted errors), specially when the measurement noise of SNMP link loads is high.

Second, we enhance the above algorithm to handle the case that the NetFlow data is not complete due to partial deployment or data loss in transit. In this case the prior is generated by combining the traffic matrix elements directly estimated by the existing NetFlow data and those produced by the generalized gravity model [19]. One of the contributions we make here is to discover the probability model in the gravity model using the Equivalent Ghost Observation (EGO) method. Then it will be further calibrated using the SNMP link loads after carefully setting up the relative weight between NetFlow data and the generalized gravity model. The experimental results in Section 6 shows that only with a small portion of NetFlow data the estimation accuracy can be significantly improved. We also study the problem on where to turn on the NetFlow to collect flow measurement data in order to achieve the best performance on estimating traffic matrices given a fixed percentage of deployment of NetFlow.

is not clear that network operators are willing to do this because it would introduce negative impact to users.

[2]The work [14] studies the similar problem but uses different approach to take advantage of the flow measurement data. It uses 24 hour NetFlow data to estimate the parameters in their model.

Third, we propose novel algorithms to identify and remove dirty data in the traffic measurement. This will not only help in traffic matrix estimation but also in a number of other important network management tasks such as anomaly detection. We assume that, in practice, there are only a small number of OD pairs/links which produce dirty data at a given time and cause inconsistency in the measurement data. We identify dirty data by finding the simplest explanation of the inconsistency.

Finally, we also develop the algorithm to estimate traffic matrices upon topology and routing changes such as link failure, hot potato routing, and BGP session reset events. This is very helpful in evaluating the impact of such network events and mitigating undesirable events. The routing changes can promptly be reported by monitoring OSPF and BGP routing updates [13, 17]. Then we can obtain the corresponding NetFlow data before and after the routing change respectively. Using them as the apriori, the traffic matrices can be estimated much more accurately than that omitting the routing change.

The rest of the paper is organized as follows. Section 2 formally defines the traffic matrix estimation problem. We describe our base model for traffic matrix estimation based on both SNMP link load data and sampled NetFlow data in Section 3 and evaluate the proposed methodology using empirical data collected from a tier-1 ISP network in Section 4. Section 5 discusses a number of factors that affect traffic matrix estimation and proposes enhanced methods and Section 6 evaluates the proposed enhancements. We discuss the related work in Section 7 and conclude the paper in Section 8.

2. PROBLEM STATEMENT

In this section, we state precisely our problem of estimating traffic matrix from imperfect (noisy and possibly "dirty") SNMP link counts and NetFlow measurement data. After a brief introduction of terminologies, we pinpoint the source of noise and dirty data in both types of data, and formulate the problem we would like to solve in this work.

2.1 Terminologies

The topology of an IP network can be viewed as a set of routers and links. The routers and links that are internal to the network are *backbone* routers and links, while others are *edge* routers and links. The routers inside the network run Interior Gateway Protocol (IGP) to learn how to reach each other. The two most common IGPs are OSPF and IS-IS, which compute shortest paths based on configurable link weights. The edge routers at the periphery of the network learn how to reach external destinations through Border Gateway Protocol (BGP). Both IGP and BGP together determine how traffic flow through the network.

Traffic matrices are often computed at interface, router, or PoP level. We use the term "node" to denote the entity at which level the traffic matrices are computed. Given edge nodes i and j, the traffic between i and j is defined as the total traffic volume that enters the network at node i and exits the network at node j. We refer to node i as the *ingress node* and node j as the *egress node*. The pair of ingress node i and egress node j are referred to as an *Origin-Destination (OD) pair* of the traffic flow. We refer to the aggregated traffic of an OD pair as an *OD flow*. The *traffic matrix* is thus the OD flows of all possible ingress and egress OD pairs. In this paper, we represent the traffic matrix in a vector form, where each element corresponds to the traffic volume of one OD flow. We illustrate our schemes and experiments at router level, while they can also be applied to interfaces and PoP levels. In the rest of the paper, the terms "node" and "router" are equivalent.

2.2 Imperfect data sources

The following measurement capabilities are deployed in most of commercial IP networks. Unfortunately, none of them alone is sufficient for providing direct and highly accurate measurement of traffic matrix. In addition, data collection is distributed among multiple network entities that are not fully synchronized, which results in noise in the data. To make matters worse, due to factors such as hardware and software errors, the quantities reported by SNMP and NetFlow measurements can deviate significantly from the actual quantity, which we regard as *dirty data* (distinguished from noise). The imperfectness of data, classified roughly into the following three categories, poses significant challenges to our goal of estimating traffic matrix accurately.

Link load measurements: The link load measurements are readily available via Simple Network Management Protocol (SNMP), which periodically polls statistics (e.g., byte counts) of each link in an IP network. The data is coarse-grained, the commonly adopted sampling interval is 5 minutes in operational IP networks. These link counts contain some noise since the measurement station cannot complete the MIB polling for thousands of network interfaces on hundreds of routers all at the same time at the beginning of the 5-minute intervals, making the actual polling intervals shifted as well as being longer or shorter than 5 minutes. We will show that this noise can be modeled as Gaussian random variables. In addition, the link counts can be lost during transit because SNMP uses UDP as the transport protocol, and may be incorrect due to hardware problem or software bugs. Such link counts are referred to as dirty data.

Flow level measurement: The traffic flow statistics are measured at each ingress node via NetFlow [10]. A *flow* is defined as an unidirectional sequence of packets between a particular source and destination IP address pair. For each flow, NetFlow maintains a record in router memory containing a number of fields including source and destination IP addresses, source and destination BGP routing prefixes, source and destination ASes, source and destination port numbers, protocol, type of service, flow starting and finishing timestamps, number of bytes and number of packets transmitted. These flow level information would be sufficient to provide direct traffic matrix measurement if complete NetFlow data were collected for the entire network. However, due to high cost of deploying and enabling flow level measurement via NetFlow, sampling is a common technique to reduce the overhead of detailed flow level measurement. The flow statistics are computed after applying sampling at both packet level and flow level. Since the sampling rates are often low, inference from the NetFlow data (through scaling) may be noisy. Also, NetFlow is often only partially deployed because products from some vendors do not support NetFlow in a way consistent to our needs (i.e., different from Cisco NetFlow specification) and some do not support it at all. Similar to SNMP data, NetFlow data may also be lost in transit, resulting in dirty data.

Topology and routing measurement: The network topology can be computed based on the configuration data of each router in an IP network. Both intra-domain (e.g., OSPF) and inter-domain (i.e., BGP) routing information are available via deployed monitors (e.g., [13]). Realtime access to these data allows us to compute the forwarding table at a given time within each router and identify topology and routing changes that affect traffic matrix.

2.3 Traffic matrix estimation

We formulate our problem of estimating traffic matrix from both SNMP link counts and NetFlow records as follows. Assume there are n OD flows and m links in an IP network. Let $\mathbf{X} = (x_1, x_2, \cdots, x_n)^T$ denote the real OD flows to be estimated and let $\mathbf{B} = (b_1, b_2, \cdots, b_m)^T$ denote the link loads when routing traffic demand $\mathbf{X}$ over the network. $\mathbf{B}$ and $\mathbf{X}$ are related by a routing matrix $\mathbf{A}$:

$$\mathbf{B} = \mathbf{A}\mathbf{X}$$

where $\mathbf{A}$ is an $m \times n$ matrix whose element on the j-th row and the i-th column, a_{ji}, indicates the fraction of traffic from flow i being routed through link j.

Let $\widehat{\mathbf{X}} = (\widehat{x_1}, \widehat{x_2}, \cdots, \widehat{x_n})^T$ be the estimated OD flow traffic matrix obtained from sampled NetFlow data (after compensating for the sampling), where $\widehat{x_i}$ is the estimator of x_i. Let $\widehat{\mathbf{B}} = (\widehat{b_1}, \widehat{b_2}, \cdots, \widehat{b_m})^T$ be the corresponding SNMP link load measurement adjusted by the length of polling intervals. Ideally, we would like to have

$$\widehat{\mathbf{B}} = \mathbf{A}\widehat{\mathbf{X}}$$

in which case we will simply use this $\widehat{\mathbf{X}}$ as our estimate. However, in practice, this is rarely true due to aforementioned noises and dirty data in both SNMP and NetFlow measurements. The question we are going to answer in this paper is: "What is the best estimate of $\mathbf{X}$ based on imperfect measurements of $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{B}}$ from NetFlow and SNMP counts respectively?"

3. METHODOLOGY

3.1 Modeling measurement noises

As we discuss in Section 2, both NetFlow and SNMP data can be inaccurate due to the sampling and polling processes used in the measurement. We refer to such measurement inaccuracies as *measurement noises*. In this section, we study the NetFlow sampling process in flow measurement and the SNMP polling process in link load measurement and present our models that precisely capture the measurement noise incurred in these processes. In particular, we define

$$\mathbf{X} = \widehat{\mathbf{X}} + \varepsilon^{\mathbf{X}}$$
$$\mathbf{B} = \widehat{\mathbf{B}} + \varepsilon^{\mathbf{B}}$$

where $\varepsilon^{\mathbf{X}}$ and $\varepsilon^{\mathbf{B}}$ are the measurement noises of NetFlow data and SNMP link loads respectively. We will show that accurately modeling these measurement noises enables us to derive good estimate of the traffic matrices. In addition, it is also helpful in distinguishing the dirty data (Section 5.2) from the measurement noises.

3.1.1 Noise ε^X in flow measurement

NetFlow typically uses packet sampling to reduce the processing load and storage space. We model this packet sampling process as Bernoulli trials.

Let $\mathbf{F} = (f_1, f_2, \cdots, f_n)^T$ be the observed byte counts of OD flows from NetFlow data (before compensation for sampling) and $\mathbf{R} = (r_1, r_2, \cdots, r_n)^T$ be the sampling rates of OD flows. We can compute $\widehat{\mathbf{X}}$, which is an unbiased estimator of $\mathbf{X}$, as follows:

$$\widehat{\mathbf{X}} = (\widehat{x}_1, \widehat{x}_2, \cdots, \widehat{x}_n) = (\frac{f_1}{r_1}, \frac{f_2}{r_2}, \cdots, \frac{f_n}{r_n})^T$$

The Mean Square Error (MSE) of estimator $\widehat{x}_i$, $1 \le i \le n$, is

$$\text{MSE}(\widehat{x}_i) = \frac{1 - r_i}{r_i} \sum_{k=1}^{w_i} \sum_{j=1}^{m_k^{(i)}} s_{i,k,j}^2 \qquad (1)$$

where w_i is the number of flows in the i-th OD flow, $m_k^{(i)}$ is the number of packets in the k-th flow of the i-th OD flow, and $s_{i,k,j}$ is the size of the j-th packet in the k-th flow of the i-th OD flow.

The above expression for MSE is a function of the size of each packet in the OD flow, which is undesirably expensive to compute in practice. Therefore, we adapt the following approximate (and upper bound) MSE derived by Duffield et al in [4]:

$$\text{MSE}(\widehat{x_i}) \approx \frac{(1-r_i)\widehat{x_i}s_{max}}{r_i}$$

where s_{max} is the largest packet size. In our paper, we use $s_{max} = 1,500$ bytes.

However, for a large operational ISP, storing and transmitting the data collected by the packet-sampled NetFlow are often still prohibitive due to its large volume. To make the data size manageable, [4] proposed a new IP flow measurement infrastructure which performs an additional flow-level sampling process, namely *smart sampling*, on the data collected by packet-sampled NetFlow. Conceptually, smart sampling selects a flow of x bytes with probability $\min(1, x/z)$, where z is a predefined threshold. In other words, flows of size greater than the threshold z are always collected, while smaller flows are selected with a probability proportional to their sizes.

As given in [4], the combined sample process (with both packet-level NetFlow sampling and flow-level smart sampling) has the following properties:

$$\widehat{x_i} = \sum_{k=1}^{w'_i} \max(z, \frac{c_{i,k}}{r_i}) \tag{2}$$

$$\text{MSE}(\widehat{x_i}) \approx \widehat{x_i}\left(\frac{(1-r_i)s_{max}}{r_i} + z\right) \tag{3}$$

where w'_i is the number of flows in the i-th OD flow after smart sampling, and $c_{i,k}$ is the observed size (by NetFlow) of the k-th flow of the i-th OD flow.

Finally, we approximate the measurement noise introduced by NetFlow sampling and smart sampling (if applicable) as Gaussian noises:

$$\varepsilon_i^X \sim N(0, \sigma_i^2)$$

where $\sigma_i^2 = \text{MSE}(\widehat{x_i})$.

A careful observation on the above will find that this model is not rigorous under one condition: when all flows of an OD flow have been missed from the sampling, the estimated traffic volume of the OD flow becomes zero according to the unbiased estimator, and so does the MSE of this estimate. In this case, the corresponding ε_i^X is not well defined. We now address this problem.

What we are interested in is the conditional distribution of the OD flow size given it is not sampled by either NetFlow or smart sampling. Consider the simple case where all packets for the OD flow have equal size, s. Let L_0, L_1 and L_2 denote the size (in number of packets) of an OD flow originally, after NetFlow packet sampling, and after smart sampling respectively. We can derive the conditional probability of $\Pr[L_0 = l|L_2 = 0]$ below. The derivation of (4) can be found in [21].

$$\Pr[L_0 = l|L_2 = 0] = \sum_{j=0}^{\min\{l, r_i z/s\}} \binom{l}{j} r_i^{j+1}(1-r_i)^{l-j} \frac{2 - \frac{2j}{r_i z/s}}{r_i z/s + 1}$$

From (4), we can compute the mean square error (MSE) when the observed OD flow is zero as

$$\text{MSE}_0 = s\text{E}[L_0^2|L_2 = 0]$$

and our definition of σ_i^2 becomes

$$\sigma_i^2 = \begin{cases} \text{MSE}(\widehat{x_i}) & \text{if we observe OD flow } i \text{ in the sampled data} \\ \text{MSE}_0 & \text{otherwise} \end{cases}$$

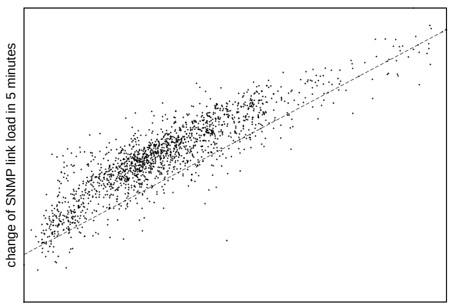


Figure 1: Traffic rate over 5-minute intervals (both x and y axis are in logscale)

3.1.2 *Noise ε^B in link load measurement*

The primary source of errors in SNMP link load measurement is due to the disalignment of polling intervals. Consider a target measurement for byte counts, b_i, over a link i, during interval $[t, t+l]$. The actual measurement, derived from two consecutive SNMP pollings for link i, however is on interval $[t+\Delta_1, (t+\Delta_1)+(l+\Delta_2)]$. We denote the result from this measurement as m_i. Note that the magnitude of Δ_1 and Δ_2 are typically much smaller than l (i.e., $|\Delta_1|, |\Delta_2| \ll l$): Δ_1 and Δ_2 is typically in the order of several tens of seconds and l is in the order of several minutes. If we assume that the traffic rate over link i in a short period of time (*e.g.,* $[t-l, t+2l]$) can be described as a Wiener process with parameter θ_i^2

$$b_i(t) - b_i(s) \sim N(0, \theta_i^2(t-s))$$

then

$$\widehat{b_i} = \frac{m_i l}{l + \Delta_2}$$

is an unbiased estimator for b_i. The mean square error is approximately

$$\text{MSE}(\widehat{b_i}) \approx |\Delta_1| l \theta_i^2$$

To quantify θ_i^2, we measure the difference in the average traffic rate of two consecutive polling intervals. Figure 1 shows the scatter plot of the difference in the average traffic rate of two consecutive 5-minutes intervals versus the traffic rate in the first 5-minute interval for a few thousands links in a large tier-1 ISP backbone network. We observe the trend of a linear relationship between the difference, which is proportional to θ_i^2, and the average traffic rate, which is $\frac{m_i}{l+\Delta_2} = \frac{\widehat{b_i}}{l}$. Therefore, we can further approximate the mean square error as

$$\text{MSE}(\widehat{b_i}) \approx |\Delta_1| \lambda \widehat{b_i}$$

where λ is a constant that can be derived by fitting the scatter plot. For the completeness of the model, we also define a small constant as the MSE in case a link load measurement is zero. However, we do not encounter this situation in the data we explored.

Similarly to ε^X, we then model the measurement noises of link load introduced in SNMP polling as Gaussian random variables:

$$\varepsilon_i^B \sim N(0, \mu_i^2)$$

where $\mu_i^2 = \text{MSE}(\widehat{b_i})$.

3.2 Estimating traffic matrix

Let us now revisit our problem formulation by combining the above model we derived.

$$\widehat{\mathbf{X}} = \mathbf{X} + \varepsilon^{\mathbf{X}} \quad (4)$$

$$\widehat{\mathbf{B}} = \mathbf{A}\mathbf{X} + \varepsilon^{\mathbf{B}} \quad (5)$$

where both $\varepsilon^{\mathbf{X}}$ and $\varepsilon^{\mathbf{B}}$ are zero-mean Gaussian random variables. Put into matrix representation, our system can hence be described as

$$\mathbf{Y} = \mathbf{H}\mathbf{X} + \mathbf{N} \quad (6)$$

where $\mathbf{H} = (\mathbf{I}; \mathbf{A})$ is an $(m + n) \times n$ matrix which vertically concatenates an identity matrix $\mathbf{I}$ and the routing matrix $\mathbf{A}$, $\mathbf{Y}$ is the concatenated vector of $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{B}}$ and $\mathbf{N}$ is the concatenated vector of $\varepsilon^{\mathbf{X}}$ and $\varepsilon^{\mathbf{B}}$. What we are looking for is a good estimator of the traffic matrix $\mathbf{X}$ from the observable $\mathbf{Y}$.

Our system in (6) fits well in the framework of the well-known Gauss–Markov theorem [7], which states that in a linear model in which the errors are uncorrelated and have expectation zero, the best linear unbiased estimators (BLUE) of the coefficients are the least-squares (LS) estimators. That is, among all unbiased estimators for $\mathbf{X}$, the one that minimizes the normalized residual errors (defined below), yields the smallest variance. Note that for Gauss–Markov theorem to hold, the errors need not to be normally distributed. We model the SNMP and NetFlow measurement noises as Gaussian only for the purpose of defining dirty data and distinguishing them from measurement noises, as we will discuss in Section 5.2.

The weighted LS estimator of $\mathbf{X}$ in (6) is found by the pseudo-inverse solution of the normalized equivalent of (6) in which the errors are homoscedastic:

$$\dot{\mathbf{X}} = (\mathbf{H}^T\mathbf{K}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{K}^{-1}\mathbf{Y} \quad (7)$$

Here $\mathbf{K}$ is the covariance matrix of $\mathbf{N}$ ($\mathbf{K} = \mathrm{E}[\mathbf{N}\mathbf{N}^T]$), which is a diagonal matrix as we assume all measurement errors are uncorrelated. To relate back to our models in Section 3.1.1 and 3.1.2, we denote $\mathbf{\Sigma^2} = (\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2)^T$ and $\mathbf{\Gamma^2} = (\mu_1^2, \mu_2^2, \cdots, \mu_m^2)^T$. The above LS estimator $\dot{\mathbf{X}}$ solves the quadratic optimization problem that aims at minimizing the total weighted squared-error in the observations, i.e.,

$$\text{minimize} \quad \|\frac{\mathbf{X} - \widehat{\mathbf{X}}}{\mathbf{\Sigma}}\|_2 + \|\frac{\mathbf{A}\mathbf{X} - \widehat{\mathbf{B}}}{\mathbf{\Gamma}}\|_2 \quad (8)$$

Note that the division in (8) is an element-by-element division where the numerator and denominator are vectors of same length. To compute $\dot{\mathbf{X}}$, we use the the Singular-Value Decomposition (SVD) routine in Matlab to solve for the pseudo-inverse, and similar to [19], we adopt Iterative Proportional Fitting (IPF) to avoid negative values of the traffic matrix, which are without any physical meaning.

The size of $\mathbf{H}$ could be very large in a large network, which makes the computational complexity (7) high. For example, in our experiment with a large tier-1 ISP backbone network, it can take as much as tens of minutes to obtain a solution on a 900 MHz processor. This may hurt the applicability of the above method to some applications such as online diagnosis and failure detection, which require faster response time. To satisfy the requirement of these applications we design the following technique which reduces the computational complexity significantly while retaining a high accuracy of the derived traffic matrix. The idea is straightforward. We first sort the OD flows by their sizes in $\widehat{\mathbf{X}}$. Then we divide them into two sets by comparing them to a threshold value T (e.g., 0.01% of the total volume). Let $\mathbf{X_L}$ be the subvector of $\mathbf{X}$ in which the corresponding OD flow has $\widehat{x_i} \geq T$, and let $\mathbf{X_S}$ be the subvector of the remaining $\mathbf{X}$ such that $\widehat{x_i} < T$. To speed up the computation, we hence focus only on obtaining a good estimate of $\mathbf{X_L}$ while treating $\mathbf{X_S}$ as known, which take values equal to their corresponding $\widehat{x_i}$. Our problem in (4) and (5) becomes

$$\widehat{\mathbf{X}}_\mathbf{L} = \mathbf{X_L} + \varepsilon^{\mathbf{X_L}}$$

$$\widehat{\mathbf{B}} - \mathbf{A_S}\widehat{\mathbf{X}}_\mathbf{S} = \mathbf{A_L}\mathbf{X_L} + \varepsilon^{\mathbf{B}}$$

where $\mathbf{A_L}$ and $\mathbf{A_S}$ are the submatrices (columns) of the routing matrix $\mathbf{A}$ that corresponds to $\mathbf{X_L}$ and $\mathbf{X_S}$ respectively. We apply the same solution technique in solving the above reduced system.

Here the threshold T should determine the desirable tradeoff between the computational complexity and the estimation accuracy. Fortunately, the OD flows in operational networks are often highly skewed [1]: a small number of OD pairs have very large traffic volume, while the majority of OD pairs have substantially low traffic between them. This is a very favorable property for our scheme. In our experiments, we can reduce the running time of the traffic matrix computation (for the same backbone network and on the same processor as above) to a few seconds by setting an appropriate threshold, meanwhile not comprising the overall accuracy by much (shown in Section 4).

We should note that the prior estimate of NetFlow measurement may significantly underestimate the volume of an OD flow due to unexpected errors. In this case, it is possible that an OD flow in $\mathbf{X_L}$ be mistakenly placed in $\mathbf{X_S}$, which contaminates the rest of the computation. We rely on dirty data detection (Section 5.2) to correct such problems.

4. EVALUATION

4.1 Data gathering methodology

We evaluate our techniques based on real network measurement data gathered from a large tier-1 ISP backbone network which consists of tens of Point of Presence (PoPs), hundreds of routers, and thousands of links, and carries over one petabyte of data traffic per day. Ideally, we would like to use both the real physical and logical (routing) network topology, and the true traffic matrix and link load information in our experiments. The former are readily available through the methods introduced in [13]. The latter, however, cannot be easily measured from the network, and is in fact the objective of this paper.

To construct a traffic matrix that is as close to reality as possible, we use the data collected from our deployed IP flow measurement collection infrastructure [4] which applies the sampled NetFlow with sampling rate $1/500$ and the smart sampling with threshold 20MB. The data were collected over one month period from 8/15/2005 to 9/18/2005. In fact, our measurement infrastructure has very good coverage on the periphery of the network – the aggregated traffic volume computed from the collected data accounts for over 90% of the total volume observed by SNMP. We aggregate flows into a set of hourly traffic matrices [5]. Due to sampling (sampled NetFlow + smart sampling), some of the elements in the traffic matrix are zero (i.e., the traffic between the corresponding OD pair during the hour is completely missed by sampling). We fill in each zero-valued element a small number drawn randomly according to the model described in Section 3.1.1, unless the traffic matrix element is prohibited by routing (e.g., the traffic from one peering link to another peering link), in which case we keep its value as zero. We then simulate OSPF routing to derive a routing matrix A, and project the above traffic matrices on A to derive the

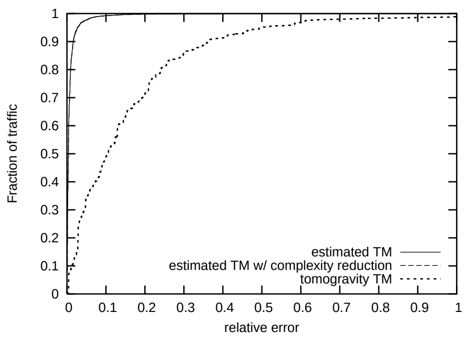


Figure 2: The weighted cumulative distribution of relative errors of estimated traffic matrices elements for our schemes and tomogravity method.

corresponding link load information. In this way, we obtain a set of "actual" traffic matrices, an accurate routing matrix, and a set of "actual" link load data that are all consistent with each other.

Once the "actual" traffic matrices and the link load data are available, we can simulate the measurement noises and obtain a set of NetFlow and SNMP measurement data. We introduce noises on an element-by-element basis, following the models described in Sections 3.1.1 and 3.1.2. Such "contaminated" data best capture the quality of the network measurement data in reality, and thus will be used as input data in evaluating our algorithms.

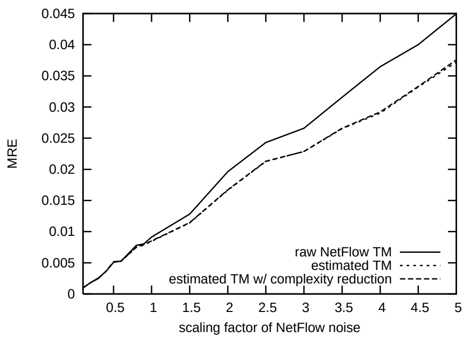
4.2 Experimental results

We first compare our approach in Section 3.2 and the tomogravity method introduced by Zhang et al. [19]. Figure 2 shows the weighted cumulative distribution (CDF) of relative errors of the estimated traffic matrix elements for these two methods, where the weight is the traffic volume of the OD flow. We observe a much better performance for our scheme: more than 90% of the traffic has negligible relative errors and almost all the traffic (i.e., 100%) has error less than 10% while the corresponding fractions for the tomogravity method are 20% and 50%, respectively. It demonstrates the advantage by making use of the NetFlow data in traffic estimation. Figure 2 also shows the cumulative distribution of relative errors when the complexity reduction scheme is in place. We set the threshold T to be 0.0005 times the volume of the the largest OD flow reported by NetFlow. The result is encouraging. The curve for complexity reduction scheme has no discernible difference compared to that without complexity reduction. This is because majority of OD flows are relatively small. However, the running time of traffic matrix computation is shortened from tens of minutes to several seconds. This suggests that we can focus only on the large flows in order to tradeoff for fast computation while not compromising the overall accuracy of the derived traffic matrices much.

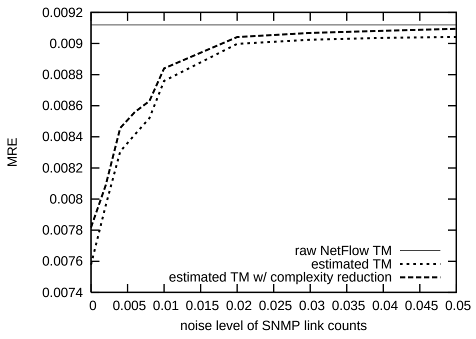
Figure 2 provides us a view on the error distribution of all traffic matrix elements. To measure the overall accuracy of the derived traffic matrices, we adopt the Mean Relative Error (MRE) metric, which is introduced in [6] and is defined as follows:

$$MRE = \frac{1}{N_T} \sum_{i:x_i>T} \left| \frac{\widehat{x_i} - x_i}{x_i} \right|$$

where N_T is the number of matrix elements that are greater than a threshold value T, i.e., $N_T = |\{x_i | x_i > T, i = 1, 2, \cdots, N\}|$. Consistent with [6], we choose the value of T so that the OD flows under consideration carries approximately 90% of the total traffic.



(a) Noise in NetFlow measurement



(b) Noise in SNMP measurement

Figure 3: MRE as a function of different levels of noise in NetFlow and SNMP measurements.

We now evaluate the impact of different levels of noise in NetFlow and SNMP measurements on the traffic matrix inference. In particular, we tune the noise level according to the model presented in Section 3. Figure 3(a) compares MREs of the estimated traffic matrix using our scheme (with and without complexity reduction) with that of the traffic matrix directly obtained from the raw NetFlow data (i.e., the prior of our scheme) under different levels of noise in NetFlow measurement. Here the standard deviation of the noise of SNMP link loads is set to 0.01 times of the actual link loads which is a typical value in operational networks. We vary the scaling factor of NetFlow noise. Scaling factor 0 corresponds to perfect NetFlow measurement with no sampling conducted and scaling factor 1 corresponds to the existing measurement setup in the tier-1 ISP network from which our data is collected, i.e., sampled NetFlow with $1/500$ sampling rate and smart sampling with 20MB threshold. A scaling factor of k corresponds to having a standard deviation k times of that from the existing setup, as the result of changing sampling rate or smart sampling threshold. We observe that MREs of both approaches increase as the scaling factor increases. Our scheme only improves to a very limited extend the estimation accuracy from the raw NetFlow data when the scaling factors are small. This can be explained as follows: NetFlow, although suffering from sampling errors, still provides good estimation of the traffic matrix when noise is small (i.e., sampling rate is high). Using a set of SNMP measurement, especially with poor quality, as part of the linear constraints does not provide a force that is strong enough to correct the sampling errors in NetFlow measurement, or on the other contrary, it may even make the result a little worse. However, when the accuracy from NetFlow degrades (scaling factor becomes large), the noisy SNMP measurement would guide the estimation closer to the actual values. This is manifested by the more pronounced improvement when the scaling factor becomes large (e.g., when the sampling rate becomes low). Note that we also compute the traffic matrices using tomogravity under the same settings. The MRE is 0.18 (not shown in Figure 3(a)), which is significantly higher than our scheme.

Next we study the impact of noise of SNMP measurement on the traffic matrix computation given a fixed noise level of NetFlow measurement (sampled NetFlow with $1/500$ sampling rate and smart sampling with 20MB threshold). Figure 3(b) shows MREs of the traffic matrix under different levels of noise in SNMP link load measurement. A noise level at α implies the standard deviation of the introduced Gaussian noise is α times of the true link load. We observe that the improvement from combining SNMP information to NetFlow measurement diminishes gradually as the noise in SNMP measurement increases. This is manifested by the reducing gap between the two curves, demonstrating the reduced power of recalibrating the NetFlow estimate by the noisy SNMP link loads. On the other hand, we find that once the SNMP measurement is not extremely distorted (e.g., $\alpha \leq 0.05$), our scheme always has an observable improvement in comparison to raw NetFlow traffic matrix (the horizontal line in Figure 3(b)). This is reassuring, since it suggests that our scheme by combining multiple data sources can be safely applied and it will not underperform methods that only use single data source (e.g., sampled NetFlow).

We also plot in Figure 3 the result when we apply our complexity reduction technique. Similar to Figure 2, the results indicate that our proposed complexity reduction technique introduces little inaccuracy in traffic matrix computation.

Our evaluation thus far has found that our scheme improves, but to a very limited extend, the accuracy of the raw NetFlow traffic matrix. This is due to the ideal setting of the measurement environment under evaluation. Next, we discuss more practical problems presented in operational networks, where the advantage of using multiple data sources becomes evident.

5. EXISTING CHALLENGES IN PRACTICE

Our inference techniques in Section 3 are based on the assumption that our observation of the SNMP data and NetFlow data are ideal. There are three aspects of this assumption. First, the SNMP data is considered perfect in the sense that there is no measurement error other than the small Gaussian noise. Second, the NetFlow data is considered perfect in the sense that all ingress points have NetFlow enabled and the gathering and accounting of NetFlow records is lossless and error-free (other than the sampling error). Third, there is a subtle assumption that network routing does not change during a measurement interval, since otherwise the routing matrix in (8) cannot be treated as a constant matrix. In practice, however, none of these three aspects will hold. In the following sections we will show how to enhance our previous scheme to handle the situation when the above assumption does not hold.

5.1 Partial NetFlow coverage

As discussed before, instrumenting a reliable IP flow measurement collection infrastructure (e.g., NetFlow) across the whole network is a difficult task for a large ISP. According to our experience it would take long time to obtain reasonably good coverage over the network periphery due to various operational issues (e.g., vendor implementation problems). Even when the measurement infrastructure is present, ensuring reliable data feed and timely processing is nontrivial. It is well expected to have NetFlow data missing for some OD flows. Therefore the model we setup in Section 3.1.1 cannot always be completely populated. In this section we describe our solution in resolving this challenge.

The basic idea is to fill in the traffic matrix elements, which are not covered by NetFlow, with our best estimate. So what is the next best thing when direct flow measurement is unavailable? The answer is the generalized gravity model. As shown in previous work [19, 20], generalized gravity model provides a reasonably good prior estimate of traffic demand. A simple gravity model works as follows. Let the total traffic volume going to an ingress router i be $T_{i,*}$. Then the traffic matrix element[3] $T_{i,j}$ is believed to be proportional to $T_{i,*} \cdot T_{*,j}$. This belief uniquely determines a traffic matrix vector $\mathbf{T}^{(g)}$, where "(g)" stands for gravity. However, $\mathbf{T}^{(g)}$ may not be consistent with the link count observations. Therefore, the tomogravity solution is to find a $\mathbf{T}$ that is closest to $\mathbf{T}^{(g)}$ in a weighted fashion (i.e., "minimizing $||(\mathbf{T} - \mathbf{T}^{(g)})/\mathbf{W}||_2$" where $\mathbf{W}$ is a weight vector[4]) among all traffic matrix vectors that are consistent with the link count observations (i.e., "subject to $||\mathbf{AX} - \mathbf{B}||_2$ being minimized"). It is determined empirically that setting weights w_i proportional to the square root of the estimation of x_i ($T_i^{(g)}$ in this case) results in the best estimation accuracy. The generalized gravity model enhances the simple gravity model by explicitly considering some routing policies, such as no transit for peers and hot-potato routing. It distinguishes edge links that connect to a customer (referred to as access links) and that connect to a peer (referred to as peering links), and then applies gravity assumption separately on traffic among the access links and between the access links and peering links. These enhancements enable the generalized gravity model to achieve good accuracy in the derived traffic matrices (around 30% relative error in a similar sized network [19]).

It is however very hard to blend the gravity model (simple as well as generalized) with the model derived from the NetFlow observations because the the gravity model is vaguely specified as "the probability model under which the above optimization procedure will produce a good estimator". The implicit probability model in this gravity model has never been made explicit in [19] and the later works. One of the contributions we made here is to make it explicit using our Equivalent Ghost Observation (EGO) method. Let x'_1, x'_2, ..., x'_n be the terms of the aforementioned $\mathbf{T}^{(g)}$ (in the matrix form) written into the vector form. [5] We can prove, again using the Gauss–Markov theorem, that if we take out all the beliefs of the gravity model (i.e., $T_{i,j} \propto T_{i,*} \cdot T_{*,j}$) and replace it with "ghost observations" x'_i where the error $x_i - x'_i$ is Gaussian with distribution $N(0, v_i^2)$ where v_i is proportional to the square root of x'_i, then the LS estimator of $(x_1, x_2, ..., x_n)^T$ is exactly the result of the optimization specified by the gravity model (with the beliefs). We refer to these nonexistent observations x'_i, $i = 1, 2, ..., n$, as EGO, as they are statistically equivalent to the beliefs in the gravity model.

Now blending our model with the gravity model becomes straightforward. For x_i terms that we have real (but noisy) observation from NetFlow, we use the probability model introduced in Sec. 3.1.1, that is, $x_i = \widehat{x}_i + \varepsilon_i^x$ and $\varepsilon_i^x \sim N(0, \sigma_i^2)$. For x_i terms that we do not have real observation, we use the aforementioned EGO x'_i, but model the estimation error as having distribution $N(0, \lambda\sigma_i^2)$. Then applying the Gauss–Markov theorem to this blended model results in an estimator that is found to be fairly accurate. Note

[3]Different from previous column vector notation, here $T_{i,j}$ denotes the volume of the traffic from ingress point i to egress point j.

[4]Here the division over W is an element-by-element division where the numerator and denominator are vectors of same length.

[5]We have also explored the scheme in which x'_i is determined by a "conditional" generalized gravity model, i.e., removing the traffic observed by NetFlow before applying gravity model. However, we find little performance difference using this variation in our experiments.

this estimator is no longer LS and BLUE since the gravity model (equivalently the EGO's) is only a belief that is not backed up by actual observations. Here λ is a normalization factor introduced to capture the relative credibility of EGO in comparison to the NetFlow observations. We will study the impact of different choices of λ in Section 6.1, where we find the overall result insensitive to the choice of λ in a fairly large "good range".

5.2 Removal of dirty data

The SNMP and NetFlow measurement, as the outcome of large scale complex measurement systems, inevitably suffer from a so-called *dirty data* problem. Unlike the data inaccuracy (Gaussian noise) discussed in Section 3, dirty data are result of unexpected hardware, software or transmission problems that cannot be modeled as measurement noises. For example, a reset of a network interface card during a collection interval may mess up the SNMP counters, thus producing completely bogus measurement result. More frequently than we would want, dirty data has caused many problems in network management including false alarms in anomaly detection, inaccurate traffic report and billing statement, and erroneous outcome from network analysis tools. In this section, we describe our algorithms to removing dirty measurement data by taking advantage of the multiple data sources that are available.

Since we define dirty data as measurement errors that cannot be captured by our noise model, it is natural to devise an iterative dirty data removal process as follows:

(i) let $\tilde{\mathbf{X}}$ be the result of solving (8); compute $\tilde{\mathbf{B}} = \mathbf{A}\tilde{\mathbf{X}}$;

(ii) compute measurement noise $\tilde{\varepsilon}^{\mathbf{X}} = \tilde{\mathbf{X}} - \hat{\mathbf{X}}$ and $\tilde{\varepsilon}^{\mathbf{B}} = \tilde{\mathbf{B}} - \hat{\mathbf{B}}$; all $\tilde{\varepsilon}_i^{\mathbf{X}}/\sigma_i$ and $\tilde{\varepsilon}_j^{\mathbf{B}}/\mu_j$ should be Gaussian $N(0,1)$ by our noise model, where $1 \le i \le n$ and $1 \le j \le m$.

(iii) if the biggest element in $|\tilde{\varepsilon}_i^{\mathbf{X}}|/\sigma_i$ and $|\tilde{\varepsilon}_j^{\mathbf{B}}|/\mu_j$ is above a threshold (e.g., 3.09), we mark it as dirty and set $\widehat{x}_i = \tilde{x}_i$ (or $\widehat{b}_j = \tilde{b}_j$).

(iv) repeats (i)-(iii) until no dirty data can be found.

The above approach, although intuitive, does not find good result. A possible reason to its poor performance is that the pseudo-inverse solution distributes the energy of dirty data to all possible explanations. In consequence, dirty data do not stand out.

Now we describe our approaches in identifying dirty data from contaminated measurement $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{B}}$. Our methodology is to detect dirty data by applying Occam's Razor principle, which aims at finding a simplest explanation for the observed phenomena. The intuition is as follows. If an OD flow from NetFlow is dirty, it may cause inconsistency with all SNMP link measurement on the path the OD flow traverses. On the other hand, if an SNMP link load is dirty, it is inconsistent with the total traffic of all OD flows routed through the link. Since dirty data are rare, the simplest explanation of the inconsistency identifies the source of dirty data.

Let ξ^X and ξ^B be the dirty data component in the measurement. We have

$$\mathbf{X} = \widehat{\mathbf{X}} + \varepsilon^{\mathbf{X}} + \xi^X$$

$$\mathbf{B} = \widehat{\mathbf{B}} + \varepsilon^{\mathbf{B}} + \xi^B$$

Since a non-zero $|\xi_i^X|$ should be much larger than σ_i (otherwise, it can be faithfully modeled by measurement noise), we expect such $|\xi_i^X| \gg |\varepsilon_i^X|$. Similarly, a non-zero ξ_j^B in SNMP data should have $|\xi_j^B| \gg |\varepsilon_j^B|$. We thus let dirty data component include both the measure noise and dirty data itself and distinguish them afterwards by comparing with $\mathbf{\Sigma}$ and $\mathbf{\Gamma}$. Let ξ be the concatenated vector of $\varepsilon^{\mathbf{X}} + \xi^X$ and $\varepsilon^{\mathbf{B}} + \xi^B$. The problem becomes

$$\text{minimize } ||\xi||_0 \quad \text{subject to } \mathbf{D} = \mathbf{C}\xi \tag{9}$$

where $||\xi||_0$ is the L_0 norm of vector ξ, $\mathbf{D} = \mathbf{A}\widehat{\mathbf{X}} - \widehat{\mathbf{B}}$ is the residual vector describing the inconsistency of NetFlow measurement and SNMP measurement, and $\mathbf{C} = (\mathbf{I}, -\mathbf{A})$ is an $m \times (m+n)$ matrix that horizontally concatenates an identity matrix $\mathbf{I}$ and the negation of routing matrix $\mathbf{A}$. It can be easily verified that $\mathbf{D} = \mathbf{C}\xi$ is equivalent to $\mathbf{B} = \mathbf{A}\mathbf{X}$.

The L_0 norm is not convex and is notoriously difficult to minimize. Therefore in practice one needs to either approximate the L_0 norm with a convex function or use some heuristics, for example the greedy algorithms proposed in [8, 18]. Here we propose two schemes: one is a greedy heuristic algorithm for L_0 norm minimization and the other is doing L_1 norm minimization which is a common approach to approximate L_0 norm minimization.

Greedy algorithm
The algorithm starts with an empty set $\mathbf{Z}$ of dirty data and then iteratively adds new dirty data to it. During each iteration, for each element $p \notin \mathbf{Z}$, the algorithm tests how much it can reduce the residual $\mathbf{D} - \mathbf{C}\xi$ by including p as a dirty data and chooses the element which can reduce the most L_2 norm of the residual, i.e., minimizing $||\mathbf{D} - \mathbf{C}\xi||_2$. The algorithm stops whenever either the residual energy falls below some tolerance factor or the number of dirty data exceeds some pre-defined threshold.

L_1 norm minimization
As shown in [3, 18], L_1 norm minimization results in the sparsest solution for many large under-determined linear systems and therefore is a common approach to approximate and convexify L_0 norm minimization. Here, we apply the same principle. That is,

$$\text{minimize } ||\xi||_1 \quad \text{subject to } \mathbf{D} = \mathbf{C}\xi$$

We can further transform the above into an unconstrained optimization problem by moving the constraints into the objective function in the form of a penalty term, i.e.,

$$\text{minimize } \theta||\xi||_1 + ||\mathbf{D} - \mathbf{C}\xi||_1$$

where $\theta \in [0,1]$ is a weight parameter that controls the degree to which the constraints $\mathbf{D} = \mathbf{C}\xi$ are satisfied. We find the optimization result not very sensitive to the choice of θ. Thus, we set it to 0.01 in the rest of the paper.

We can cast the above problem into the following equivalent Linear Programming (LP) problem, for which solutions are available even when $\mathbf{C}$ is very large, owing to modern interior point linear programming methods.

$$\begin{aligned} \text{minimize} \quad & \theta \sum_i \mu_i + \sum_j v_j \\ \text{subject to} \quad & \mathbf{D} = \mathbf{C}\xi + d \\ & u \ge \xi, u \ge -\xi \\ & v \ge d, v \ge -d \end{aligned}$$

5.3 Handling routing changes

Changes to the routing tables can happen anytime and generally do not align with the beginning of SNMP measurement intervals. If a route change happens within a polling interval, the corresponding measurement would reflect the total traffic volume of both before and after the route change. This creates problem for our problem formulation of $\mathbf{AX} = \mathbf{B}$, since the routing matrix $\mathbf{A}$ is no longer constant. Moreover, change of internal routing structure may have impact on the ingress and egress point of traffic demand, resulting in changes in the traffic matrix to be estimated. For example, traffic destinated to a peer may shift its egress point from one peering link to another due to hot-potato routing. To address this problem, we propose the following solution.

Assume the routing only changes once during that measurement epoch (the solution can be easily adapted to the case where there are more than one routing changes happening.). Let $\mathbf{A_1}$ and $\mathbf{A_2}$ be the routing matrix before and after the change. Let $\mathbf{X_1}$ and $\mathbf{X_2}$ denote the corresponding traffic matrices to be estimated. We have

$$\mathbf{A_1X_1 + A_2X_2 = B}$$

Since we can obtain the exact time of routing change from monitoring tools [13] and the flow records from NetFlow are timestamped, we can easily compute $\widehat{\mathbf{X_1}}$ and $\widehat{\mathbf{X_2}}$ if the flow measurement covers all OD flows. In the case where flow measurement is not complete, we have to rely on the generalized gravity model to populate the a priori estimate. However, since SNMP measurement is indivisible within a polling interval, we can only prorate the traffic volume for the duration when $\mathbf{A_1}$ or $\mathbf{A_2}$ prevails and so do their MSE vectors. Obviously, this introduces additional risk of inaccuracies of the estimated traffic of the OD flow, which we can compensate by increasing the value of λ in Section 5.1.

Similar to (6) the problem here also can be described by the following linear system.

$$\mathbf{Y = HX + N} \tag{10}$$

Here $\mathbf{X}$ is the concatenated vector of $\mathbf{X_1}$ and $\mathbf{X_2}$, $\mathbf{N}$ is the concatenated vector of $\varepsilon^{\mathbf{X_1}}$, $\varepsilon^{\mathbf{X_2}}$ and $\varepsilon^{\mathbf{B}}$, $\mathbf{H} = (\mathbf{I}; (\mathbf{A_1}, \mathbf{A_2}))$ is an $(m + 2n) \times 2n$ matrix which vertically concatenates an identity matrix $\mathbf{I}$ and the horizontal concatenation of the two routing matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, and $\mathbf{Y}$ is the concatenated vector of $\widehat{\mathbf{X}_1}$, $\widehat{\mathbf{X}_2}$ and $\widehat{\mathbf{B}}$.

The exact same solving procedure as in Section 3.2 can be followed to obtain the LS estimators of $\mathbf{X}_1$ and $\mathbf{X}_2$. Notice that the size of $\mathbf{H}$ here is even larger than that in Section 3.2 and therefore leads to a higher computational complexity. The situation is worse when more routing changes occur during the estimation interval as the size of $\mathbf{H}$ increases with the number of routing changes: $\mathbf{H}$ is an $(m + n \times i) \times (n \times i)$ matrix for i routing changes. Therefore the complexity reduction technique in Section 3.2 becomes increasingly important here. We will show in Section 6.3 that we can achieve very desirable accuracy within tens of seconds with the complexity reduction technique for a reasonably sized network.

6. EVALUATION OF ENHANCEMENTS

In this section, we present the evaluation results on the impact of the aforementioned three aspects of practical challenges on the accuracy of traffic matrix estimation, and correspondingly the performance of our proposed techniques to these challenges.

Unless specified otherwise, we use the following default parameters in this section: the noise level of SNMP link counts is 0.01; the scaling factor for the NetFlow noise is 1; the threshold for complexity reduction T is set such that around 1000 OD flows are selected in $\mathbf{X_L}$.

6.1 Incomplete NetFlow data

As we mentioned before, partial deployment of NetFlow and incomplete NetFlow measurement data are very common in operational networks. In this section, we evaluate our scheme towards estimating traffic matrices based on incomplete NetFlow data. Our evaluation also provides us insight on the effectiveness of various deployment strategies of the NetFlow measurement infrastructure for traffic matrix computation.

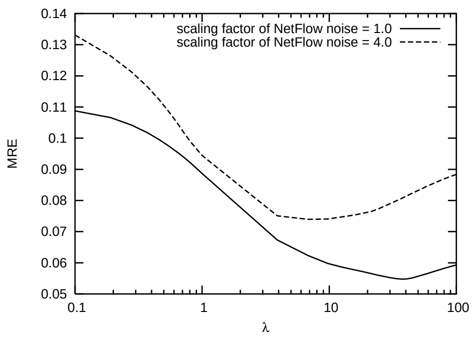


Figure 4: MRE under different values of λ (x axis is in logscale)

Recall that we have incorporated a parameter λ in our scheme in the presence of incomplete NetFlow measurement. λ captures the relative quality between the NetFlow measurement and the prior from the generalized gravity model. A larger value of λ corresponds to a more accurate NetFlow measurement in comparison to the generalized gravity model. Figure 4 evaluates the performance of our scheme with different values of λ.

In Figure 4, we study the estimation error (measured by MRE) with varying λ when the NetFlow measurement is only available at the top 20% edge routers (ranked by its total ingress traffic volume) assuming two different scaling factors, 1 or 4, of the NetFlow noise. We observe that, in both cases, there exists an optimal value for λ that achieves the minimum MRE. This value of λ best reflects the relative accuracy of sampled NetFlow and generalized gravity model defined in our solution framework. The optimal value of λ is found at around 40 when the scaling factor is 1 and at around 10 when the scaling factor is 4. This matches well with our expectation since the relative accuracy of sampled NetFlow is low when the NetFlow noise level is high (scaling factor 4), resulting in a reduced optimal penalty weight for generalized gravity model. The ratio in their optimal penalty weight ($40/10$) matches well with the inverse of their noise scaling factor of NetFlow ($4/1$). Furthermore, we observe that the performance of our approach is robust to the choice of λ at higher values (note that the x-axis is in logscale). In other words, the performance degradation due to using a λ value higher than the optimal is not dramatic. It suggests that when applying our approach in operation, one does not need to put too much effort in tuning the parameter λ to get a good performance. In the rest of the evaluation, we set $\lambda = 40$ (matching the scaling factor 1).

Now we evaluate the accuracy of our proposed scheme with NetFlow coverage on the ingress points ranking in top 20% by traffic volume (we will study other strategies later) and compare it with those from the generalized gravity model, tomogravity solution and the NetFlow measurement amended by EGO. Figure 5 plots the CDFs of the relative errors of the estimated volume of OD flows obtained by those approaches. We observe that our approach, which makes use of the NetFlow measurement at 20% of the ingress points, already achieves a significantly better accuracy than that of the tomogravity solution, which only depends on the SNMP measurement. This improvement is mostly due to the better prior estimate from the EGO-amended NetFlow than from the generalized gravity model, as indicated by the gap between the corresponding two curves. Finally, our optimization method further improves the prior estimate from the EGO-amended NetFlow to achieve an even higher accuracy. We have also plotted in Figure 5 the result when we apply our complexity reduction technique.

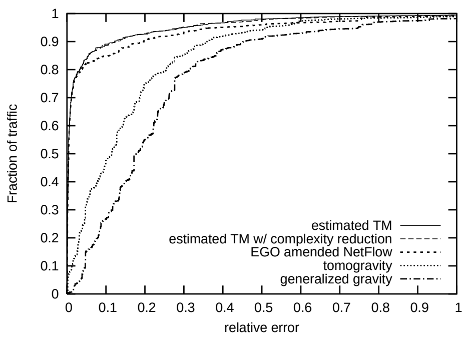


Figure 5: The fraction of traffic vs. the magnitude of the relative error.

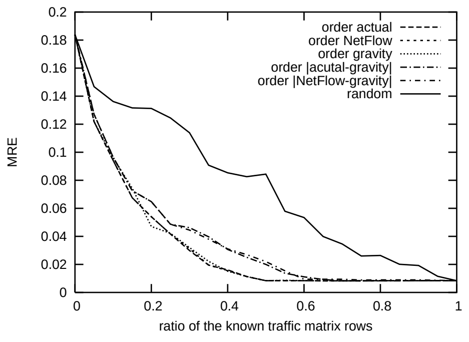


Figure 6: Impact of partial deployment of NetFlow on traffic matrix estimation.

The curve overlaps with that of the estimated traffic matrix without complexity reduction, indicating little inaccuracy has been introduced because of the approximation in the complexity reduction.

We next study the NetFlow deployment problem: given a fixed number (e.g., 20% ingress points) of ingress points that can deploy NetFlow, how to choose ingress points where the NetFlow measurement data yield the most accurate traffic matrix estimation? This problem is briefly studied in [9] and [20]. Here we revisit this problem with more comprehensive evaluation. In particular, we evaluate the following three different types of strategy that (i) randomly select x fraction of ingress points; (ii) select top x fraction of ingress points ranked by traffic volume generated from actual value, generalized gravity model and our scheme, respectively; and (iii) select top x fraction of ingress points ranked by the difference of traffic volume between the actual value and result of the generalized gravity model and between the result of our scheme and that of the generalized gravity model, respectively. Figure 6 shows the accuracy of traffic matrix estimation for various value of x. It is not surprising that random selection performs the worst. The strategies in (ii) (i.e., ranking ingress points by traffic volume) yield the best overall performance. In addition, we observe that the curves for strategies in (ii) and (iii) become flat after x is approaching 0.6. This indicates that deploying NetFlow measurement on more than 60% of ingress points only has marginal gain under careful deployment decision.

Instrumenting NetFlow measurement infrastructure on a set of ingress points to facilitate traffic matrix computation is an arduous engineering task, which involves careful testing, certification and performance tuning, etc.. We would like the set of ingress points for NetFlow deployment, which is based on most up-to-date traffic volume information, to be stable over a long time. In our evaluation, we define a stability function $f = |\alpha \cap \beta|/|\beta|$, where α and

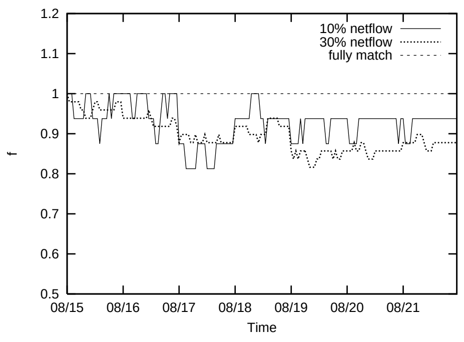


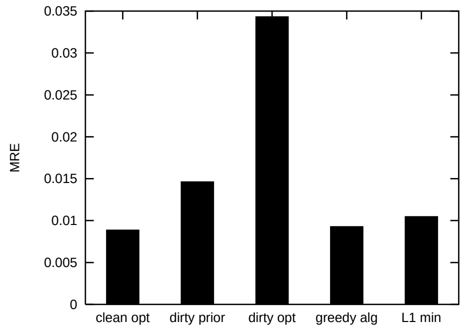
Figure 7: The stability of the NetFlow deployment decision.

β are two sets of ingress points that are selected base on traffic information at time t_α and t_β, respectively. An f value closer to 1 indicates that the two sets share a large number of overlapping elements. Figure 7 shows the change of value of f during one week period (August 15-21, 2005). In this case, t_β is set to 0:00 AM on August 15, 2005 and t_α varies from 0:00 AM on August 15 to 23:00 PM on August 21. Two sets of ingress points are selected: the ingress points ranking at top 10% and 30% in traffic volume, respectively. We observe that values of f for both sets are fairly high (always above 0.8). This implies that network administrators can make their deployment decision based on traffic volume without the risk of major re-deployment in short term.

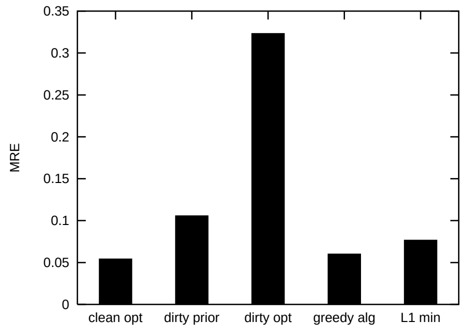
6.2 Dirty data

In order to evaluate our scheme in identifying and removing dirty data, we synthetically inject some dirty data into both the SNMP link loads and the sampled NetFlow measurement. In particular, we introduce two types of dirty data in SNMP link loads. In the first type, we scale up an SNMP measurement by a factor of 5 on a link chosen at random among the top 20% links (ranked by the traffic volume). In the second type, we scale down an SNMP measurement also by a factor of 5 on a link chosen randomly among the top 2% links. We introduce three types of dirty data into NetFlow measurements. In the first two types, we scale down a NetFlow measurement by a factor of 5 on an OD flow chosen at random among the top 0.2% and the top 2% OD flows respectively. In the third type, we scale up a NetFlow data by a factor of 10 on an OD flow chosen randomly among the top 20% of OD pairs. In the evaluation that we show next, the results are obtained when we inject one dirty data of each type into the measurement.

We plot the MREs of our estimation with and without dirty data in Figure 8 to illustrate the negative impact of the small number of dirty data. Figure 8(a) shows the results in which Netflow has complete coverage at the network periphery, while Figure 8(b) presents the case where NetFlow measurement is available at the top 20% edge routers (ranked by the total ingress traffic volume). We repeat our experiment with different choice of dirty data and present the average result in the graphs. We first look at the left three columns of these graphs. The first column represents the MRE of the estimated traffic matrix when there is no synthetically introduced dirty data; the second column represents the MRE of the Netflow or the EGO-amended gravity prior after we inject one dirty data of each type; and the third one corresponds to the MRE of the our least square estimate of the traffic matrix with those dirty data. We observe that the small number of dirty data have a strong disturbing effect to our quadratic optimization in (7): the optimization result with a handful of dirty data (the third column) is 3 times (in Figure 8(a)) or 6 times (in Figure 8(b)) worse than the result when



(a) complete NetFlow coverage



(b) 20% NetFlow coverage

Figure 8: Impact of dirty data on traffic matrix estimation.

dirty data is not present (the first column). It is interesting to observe that the contaminated Netflow or the EGO-amended gravity prior in the second column has a better accuracy than the third column. This suggests that dirty data in the SNMP measurement, as part of the optimization constraints, may steer the optimization result away from the actual traffic matrix, causing significant inaccuracy in traffic matrix estimation. In fact, dirty data phenomena has caused many problems in various applications in network management besides the traffic matrix estimation, and becomes quite a headache to network operators.

We now apply our techniques in Section 5.2 to remove dirty measurement data. We use a threshold of 3.09, which corresponds to 99.9 percentile of a standard Gaussian, as the cut off between dirty data and the normal measurement noise. Since for most of the applications of traffic matrix estimation, only the dirty data of significant size is of interest, we further filter the identified dirty data such that only those correspond to a data rate higher than 10Mbps are reported. When the sampled NetFlow covers the whole network, with the above configuration our scheme using the greedy algorithm identifies all of the five injected dirty data with 3 false positive on the average, and our scheme using L_1 norm minimization identifies all five injected dirty data with 22 false positives on the average (with repeated experiments with different base traffic matrices and random choices of dirty data). When NetFlow measurement only covers the top 20% edge routers ranked by the total ingress traffic volume, we find that our greedy algorithm can typically produce no false negatives and a small number (≤ 5) of false positives, while the L_1 norm minimization has a slightly worse performance, producing occasionally one false negative and a small number (≤ 35) of false positives.

After we identify the dirty data, we correct the corresponding elements (by subtracting the dirty components) in our problem formulation and then derive an estimate of the traffic matrix. The result is shown in Figure 8(a) and (b), where the fourth columns are the result when we use the greedy algorithms in identifying the dirty data and the fifth columns are the result when we apply L_1 minimization in identifying the dirty data. We find that both algorithms achieve comparable accuracy than that when dirty data are not present. Comparing the two methods, our greedy algorithm has a slightly better performance than the L_1 minimization.

6.3 Routing changes

In this section, we evaluate the impact of routing changes during a measurement period and the effectiveness of our proposed solution to this issue. Our experiment scenario is constructed as follows. We assume the measurement interval is of one hour length and a routing change occurs at the end of the 10^{th} minute. This routing change is a result of the failure of an internal link, which is chosen at random. We simulate the routing of the network before and after the failure to compute the corresponding routing matrices, $\mathbf{A_1}$ and $\mathbf{A_2}$. To obtain the traffic demand both before and after the failure, we pick the actual traffic matrices from two consecutive hours and prorate the traffic to populate the traffic matrix in the first 10 minutes and the last 50 minutes respectively. Finally, we compute the link load measurement over the hour, $\mathbf{B}$, by summing up the total traffic of both the first 10 minutes and the last 50 minutes on each link. The result is fed into our algorithm in Section 5.3 for traffic matrix estimation. We use the proposed complexity reduction technique (i.e., targeting at top $1,000$ elements in $\mathbf{X_1}$ and $\mathbf{X_2}$ respectively) to speed up the computation. In order to form the base for our comparison, we also consider the case in which there is no routing change during the hour. This can be constructed by simply using the traffic matrix and routing matrix from the first hour of the two consecutive ones. We next present the result of one example constructed using the above settings. Results of other cases based on different traffic matrices and choices of link failures are quantitatively similar.

Figure 9 shows the CDF of the relative error of the estimated volume of the OD flows. The two solid lines are the result derived solely based on SNMP link load, i.e., the traffic matrix being estimated entirely from the EGO of the prorated generalized gravity model, with and without routing change. We observe that the accuracy of the derived traffic matrix degrades significantly when routing change occurs during the measurement interval. This is because SNMP link load measurement does not contain sufficient information to distinguish the traffic before and after the routing change. The two dotted lines in Figure 9, however, show the result of our approach when NetFlow data is available at the top 30% edge routers. We make two observations here. First, we find that the estimation accuracy is significantly improved with the additional information from NetFlow. This echoes our observation in Section 6.1. Second, we observe that the performance degradation due to the routing change, shown as the difference between the case with routing change and the case without routing change (gap between the two dotted lines), is much less than that of the SNMP only scenario (gap between the two solid lines). Again, this demonstrates the effectiveness of our solution approaches, and more importantly the power of combining multiple data sources in deriving a good estimate of traffic matrix.

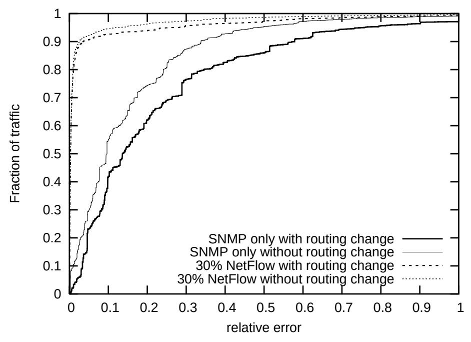


Figure 9: CDFs of the relative errors of the estimated TM elements with and without routing changes during a measurement interval

7. RELATED WORK

The problem of estimating traffic matrix attracts the effort of researchers since the late 1990s. The first a few techniques, proposed in [16, 2], adopt some simple statistical models for traffic matrix elements such as Poisson and Gaussian distribution. These techniques are highly sensitive to the apriori estimate of traffic matrices. The later work mostly focused on statistically inferring traffic matrices by combining more side information besides the backbone link loads. For example, Zhang et al. [19, 20] used SNMP data not only from backbone links but also from access and peering links in order to populate the generalized gravity model. Also in [11], the routing changes and link loads shift are intentionally induced to gather additional SNMP data. All of the above techniques are entirely based on the link loads from SNMP data. Their accuracies, usually around 10%, are not good enough for certain management tasks such as diagnosis of network fault. Some other techniques such as [5, 12, 22] are also designed to directly measure and estimate traffic matrices instead of statistically inferring based on link loads from SNMP data. They compress the information of the traffic going through a network using Cisco NetFlow or data streaming devices and estimate the corresponding traffic matrices based on the resulting compressed data set. So a natural thinking is whether we can combine SNMP data and flow measurements together to produce better results, which was raised briefly in previous work [9, 20] and studied comprehensively in this work. The work [14] also explores this problem but uses a different approach to adopt flow measurement data.

8. CONCLUSION

In this paper we present several novel inference techniques for robust traffic matrix estimation with both imperfect SNMP link counts and sampled NetFlow measurement. In contrast to previous work, our schemes take advantage of both NetFlow and SNMP link loads data to obtain better estimation. We find that under the existing configuration of sampled NetFlow the result from NetFlow is quite accurate and combining SNMP link loads with it only improves the estimation accuracy slightly if NetFlow is fully deployed in the network. However, when full deployment of NetFlow is not available – a common case in operational networks, our algorithm can improve estimation accuracy significantly even with a small fraction of NetFlow data.

More importantly, we show that dirty data can contaminate a traffic matrix. We design two novel algorithms to identify and remove dirty data in sampled NetFlow and SNMP data. This would benefit a number of important network management applications including the traffic matrix estimation. We show that by using our algorithms, the errors in traffic matrix estimation can be reduced by more than an order of magnitude. Finally, we observe that routing and topology change is also key factor that affects traffic matrix estimation. We develop a novel algorithm for estimating more accurate traffic matrices upon topology and routing changes.

To the best of our knowledge, this work is the first to offer a comprehensive solution which fully takes advantage of using multiple readily available but imperfect data sources. The experimental results based on real data obtained from a large tier-1 ISP backbone network provide valuable insight on the effectiveness of combining multiple data sources to estimate traffic matrices.

9. REFERENCES

[1] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Geographical and temporal characteristics of inter-pop flows: View from a single pop. *European Transactions on Telecommunications*, 2000.

[2] J. Cao, D. Davis, S. Vander Wiel, and B. Yu. Time-varying network tomography:router link data. *Journal of American Statistics Association*, pages 1063–1075, 2000.

[3] D.L. Donoho. For most large underdetermined systems of equations, the minimal l1-norm near solution approximates the sparsest near-solution. In *http://www-stat.stanford.edu/ donoho/Reports/*, 2004.

[4] N. Duffield and C. Lund. Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure. In *Proc. ACM SIGCOMM IMC*, October 2003.

[5] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE transaction on Networking*, June 2001.

[6] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large ip backbone– a comparison on real data. In *Proc. USENIX/ACM SIGCOMM IMC*, October 2004.

[7] S.M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993.

[8] A. Lakhina, K. Papagiannaki, and M. Crovella. Diagnosing network-wide anomalies. In *Proc. ACM SIGCOMM*, August 2004.

[9] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation:existing techniques and new directions. In *Proc. ACM SIGCOMM*, August 2002.

[10] White paper-netflow services and applications. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ neflct/ tech/napps_wp.htm.

[11] A. Nucci, R. Cruz, N. Taft, and C. Diot. Design of igp link weight changes for estimation of traffic matrices. In *Proc. IEEE INFOCOM*, March 2004.

[12] K. Papagiannaki, N. Taft, and A. Lakhina. A distributed approach to measure traffic matrices. In *Proc. ACM/SIGCOMM IMC*, October 2004.

[13] A. Shaikh and A. Greenberg. Ospf monitoring: Architecture, design and deployment experience. In *Proc. USENIX NSDI*, 2004.

[14] A. Soule, A. Lakhina, N. Taft, and K. Papagiannaki. Traffic matrices: Balancing measurements, inference and modeling. In *Proc. ACM SIGMETRICS*, August 2005.

[15] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft. How to identify and estimate the largest traffic matrix elements in a dynamic environment. In *Proc. ACM SIGMETRICS*, June 2004.

[16] Y. Vardi. Internet tomography: estimating source-destination traffic intensities from link data. *Journal of American Statistics Association*, pages 365–377, 1996.

[17] J. Wu, Z. Mao, J. Rexford, and J. Wang. Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. In *Proc. USENIX NSDI*, 2005.

[18] Y. Zhang, Z. Ge, A. Greeenberg, and M. Roughan. Network anomography. In *Proc. USENIX/ACM SIGCOMM IMC*, Oct 2005.

[19] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proc. ACM SIGMETRICS*, June 2003.

[20] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. ACM SIGCOMM*, August 2003.

[21] Q. Zhao, Z. Ge, J. Wang, and J. Xu. Robust traffic matrix estimation with imperfect information: Making use of multiple data sources. In *Technical Report*, April 2006.

[22] Q. Zhao, A. Kumar, J. Wang, and J. Xu. Data streaming algorithms for accurate and efficient measurement of traffic and flow matrices. In *Proc. ACM SIGMETRICS*, June 2005.