Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads

Yin Zhang   Matthew Roughan   Nick Duffield   Albert Greenberg

AT&T Labs— Research, Shannon Laboratory
180 Park Avenue, Florham Park, NJ 07932

{yzhang,roughan,duffield,albert}@research.att.com

ABSTRACT

A matrix giving the traffic volumes between origin and destination in a network has tremendously potential utility for network capacity planning and management. Unfortunately, traffic matrices are generally unavailable in large operational IP networks. On the other hand, link load measurements are readily available in IP networks. In this paper, we propose a new method for practical and rapid inference of traffic matrices in IP networks from link load measurements, augmented by readily available network and routing configuration information. We apply and validate the method by computing backbone-router to backbone-router traffic matrices on a large operational tier-1 IP network – a problem an order of magnitude larger than any other comparable method has tackled. The results show that the method is remarkably fast and accurate, delivering the traffic matrix in under five seconds.

Categories and Subject Descriptors

C.2.3 [**Computer-Communications Networks**]: Network Operations—*network monitoring*

General Terms

measurement, performance

Keywords

Traffic Matrix Estimation, Traffic Engineering, SNMP

1. INTRODUCTION

A fundamental obstacle to developing sound methods for network and traffic engineering in operational IP networks today is the inability of network operators to measure the traffic matrix. A *traffic matrix* provides, for every ingress point i into the network and egress point j out of the network, the volume of traffic $T_{i,j}$ from i to j over a given time interval. Taken together with network topology, routing and fault data, the traffic matrix can provide a great deal of help in the diagnosis and management of network congestion [1]. On longer time scales, traffic matrices are critical inputs to network design, capacity planning and business planning.

Unfortunately, today's production systems for IP network measurement do not provide the inputs needed for direct computation of IP traffic matrices. Instead, these systems gather data on:

- resource utilization at network nodes and links, (e.g. link loads);
- end-to-end performance metrics for specific transactions, such as one way delay statistics for packets exchanged between measurement servers at the network edge;
- status and configuration of network topology and routing.

Though these measurements may reveal traffic anomalies or congestion problems, they do not in general reveal potential solutions. For instance, link load measurements may reveal congestion on a link, but shed little light on its cause, which in general requires understanding the traffic matrix.

The principal contribution of this paper is a simple, efficient, and accurate method for computing traffic matrix estimates for IP networks, from widely available data: link load and network routing and configuration data. The method draws on ideas from "gravity modeling" [2, 3, 4, 5, 6, 7] and "tomographic methods" [8, 9, 10, 11, 12]. It also makes use of network configuration data to dramatically reduce computational complexity.

We have validated the method against direct traffic matrix measurements from detailed flow level data on an operational tier-1 IP network, and the results show very good accuracy. It also has the appealing characteristic that additional information, say from flow level traces, may be included in a straight forward manner. The method is very fast, taking less than 5 seconds on a 336 MHz Ultrasparc-II processor to compute a backbone-router to backbone-router traffic matrix on the tier-1 IP network. The method, and its gravity model prior have been used in that network since 2001 for a variety of tasks ranging from traffic engineering to router/link failure analysis to capacity planning, with considerable success.

At their simplest, gravity models are based on the assumption of a simple proportionality relationship [2, 4]:

$$T_{i,j} \propto T_{i,*} \cdot T_{*,j} \tag{1}$$

where $T_{i,*}$ and $T_{*,j}$ denote the total traffic entering the network at i and exiting at j, quantities that can be obtained by summing link load data at the network edge. (See Section 3.1.) The gravity model assumes that the traffic component from or to a given site depends only the total traffic entering or leaving that site. By appropriate normalization, the gravity model solution is guaranteed to be consistent with measured link loads at the network edge, but not necessarily so in the interior links. Alternatively, tomographic methods are based on the system of linear equations:

$$\mathbf{x} = \mathbf{A}\mathbf{t} \tag{2}$$

where $\mathbf{t}$ is the traffic matrix (written as a column vector), $\mathbf{x}$ represents link loads, and $\mathbf{A}$ the network routing matrix – see Section 3.2 for details. In essence, equation (2) states that the traffic matrix must be consistent with network routing and measured link loads throughout the network, not just at the edge. However, this matrix equality is highly under-constrained, and so allows many solutions. Tomographic methods differ in how a single "best" solution is identified

SIGMETRICS'03, June 10–14, 2003, San Diego, California, USA.

from the possibilities. The majority of existing statistical tomography approaches (commonly referred to as "network tomography" methods) use models of the higher order statistics of the link load data to create additional constraints. In contrast, optimization-based tomography approaches (e.g., linear or quadratic programming) often attempt to find a solution that optimizes an objective function.

The method introduced in this paper refines and combines both gravity and tomographic methods:

1. We solve the gravity model using edge link load data. Additional information on routing between points of ingress and egress for traffic flows can be incorporated to obtain significant improvements. In the numerical results presented here, we incorporate information to model traffic exchanged with peer networks (Section 3.1.2).

2. As the final estimate of the traffic matrix, we apply quadratic programming to determine the solution in the space of those admitted by the tomography model closest to the solution obtained by the gravity model. This step utilizes all available link load data, and does not require (higher-order) statistics or additional traffic modeling assumptions. The key computational challenge is to compute the pseudo-inverse of the routing matrix A, which has high dimensionality. To overcome this challenge, we apply network configuration and routing data to dramatically decrease the problem dimension. Iterative proportional fitting is used to ensure the non-negativity of the results.

We term this method for computing IP traffic matrices the *tomogravity* method, for want of a better name.

The validation of the tomogravity method is based on a set of hourly traffic matrices derived from direct flow level measurements using the methodology described in [13]. These traffic matrices cover over 2/3 of a tier-1 IP backbone network (including all the peering traffic) over June 2002. Obtaining direct flow level measurement across large IP networks today is a far more taxing and complex task than link and router configuration measurement, due to limited and inconsistent router support for flow level measurement capabilities.

Figure 1 provides an indication of the accuracy of the method. The method is remarkably accurate for the all but the smallest entries in the traffic matrix. We note that the larger values in the traffic matrix dominate network and traffic engineering applications [14, 15, 13]. The majority of the traffic lies within $\pm 23\%$ relative error, and a more than 30% of the matrix elements have negligible error.

A more detailed examination of the data will show that the relative errors are largest for the smallest matrix elements, which fortunately do not have large absolute errors in general and are thus unlikely to matter in applications. The matrix elements of most importance – the largest values – are the most accurate. Further, all of the errors were found to be well-behaved, that is none have overly large absolute errors, and they vary smoothly over time, meaning their impact on operational tasks will not be dramatic, even where the errors are largest.

Even more importantly from an operations perspective, predictions based on the estimated traffic matrix are remarkably accurate. For instance, the relative accuracy of computed link loads based on the estimated traffic matrix are within a few percent of the real link loads. The method insures this will be the case, but interestingly, other results such as the distribution functions for the sizes of traffic matrix elements based on real and estimated data are almost indistinguishable as well.

The paper is organized as follows: we start in Section 2 with basic network concepts and terminology, and the features of the data we have available. This is followed by a detailed description of the tomogravity method in Section 3. The next section (Section 4) presents our validation results of the method, based on real network traffic matrices. Finally we conclude the paper in Section 5.

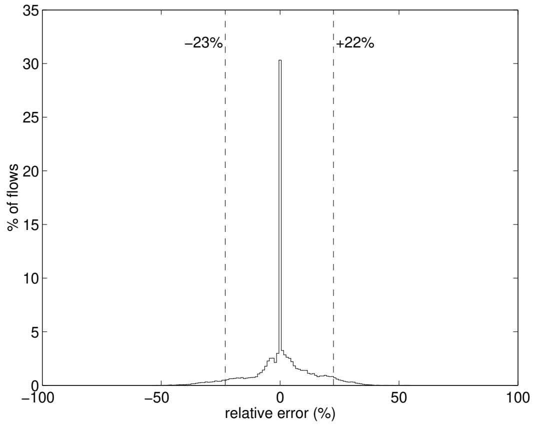


Figure 1: Relative errors of traffic matrix estimates compared to direct estimates of the traffic matrix (from flow level data) for the largest components of the traffic matrix (representing over 75% of the network traffic). Note that a significant proportion of the flows (more than 30%) have a negligible error. The two vertical dashed lines show the 5th and 95th percentiles of the distribution, showing that these lie within $\pm 23\%$.

1.1 Related Work

Tomographic methods have been widely and successfully applied, for example, in Computer Aided Tomography (CAT) scans, used in medical imaging. These methods differ in how they deal with the under-determination of the system of tomographic constraint equations. Optimization-based tomography approaches typically find a solution that optimizes an objective function, whereas network tomography approaches often use the higher order statistics of the link load data to create additional constraints.

Vardi [8] first put the ideas of network tomography into practice for computing traffic matrices in communications networks, with subsequent contributions by Tebaldi and West [9], and by Cao et al. [10]. There is a dual of this problem also referred to as network tomography in which link performance metrics are determined from path measurements [11, 12], but this does not directly concern us here.

Network tomography, in some sense, comprises determining the solution $\mathbf{t}$ to equation (2), or at least the parameters of some model of $\mathbf{t}$, from measurements of $\mathbf{x}$. As noted above, this system is highly under-constrained, and so the challenge is to choose the "best" solution from the space of possibilities. The typical approach has been to use additional modeling assumption to derive constraints from the higher order statistics of the traffic. For instance, Vardi [8], adopted a Poissonian model in which the X_i are independent Poissonian random variables of mean x_i. He observed that then the x_i are Poissonian random variables with covariance $\mathsf{Cov}(x_i, x_j) = \sum_k B_{ij,k} t_k$ where $B_{ij,k} = A_{ik} A_{kj}$. Thus, the equations $\mathbf{x} = At$ are supplemented with the equations $\mathbf{z} = Bt$ where z_{ij} denotes the measured covariance of the traffic rate across links i and j. We can write these compactly as

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} \mathbf{t}. \tag{3}$$

Vardi established that under realistic conditions on the matrix $\mathbf{B}$, the Poisson rates $\mathbf{t}$ are identifiable, in the sense that two different sets of rates $\mathbf{t}$ cannot give rise to the same asymptotic distributions of $\mathbf{x}$ and $\mathbf{z}$ for large numbers of probes. Cao et al. adopted a Gaussian model in which the variance of the t_i has a specified power-law dependence on the mean. The mean rates of this model are identifiable under the same conditions on $\mathbf{B}$.

Directly solving (3) for finitely many measurements is problematic. Due to statistical variability of the $\mathbf{x}$ and $\mathbf{z}$, the equations are generally inconsistent, while some of the z_i may be negative. To circumvent these problems, Vardi employed an iterative approach that uses the EM algorithm [16] to find approximate solutions. In the approach of Cao et al., a modified EM algorithm is used directly to find maximal likelihood parameters in the Gaussian model. Convergence is hastened by using second-order methods. Even with such methods, the complexity of the network tomography approaches grows as $O(R^5)$, where R is the number of regions, although a reduction to $O(R^2)$ is claimed for topologies in which the ingress-egress pairs can be partitioned over mostly disjoint regions [17].

Furthermore, Medina et al. [7] shows that the basic assumptions underlying the statistical models (Poisson or Gaussian) are not justified, and that the methods above may perform badly when their underlying assumptions are violated. The paper concluded [7] that none of the prior methods for computing traffic matrices was satisfactory for even a PoP to PoP traffic matrix on a large network, let alone a BR to BR matrix.

An alternative that is well known in social sciences for modeling commodity exchanges is the gravity model (See, for example [5, 6]). In network applications, gravity models have been used to model mobility in wireless networks [3], and the volume of telephone calls in a network [4]. Recently, variants of gravity models have been proposed for computing IP network traffic matrices [18, 7, 2]. For instance, [7] proposes an approach based on using the *choice models* to model PoP *fanouts*, which can be viewed as a variant of the gravity model approach. The paper also suggests using their method to generate priors to serve as inputs to statistical tomography techniques, but does not test this idea. An alternative generalization of the gravity model (detailed below) that explicitly models inter-peer routing was used by the authors of this current paper in capacity planning exercises in 2001 for an operational IP network.

As noted above, gravity models are typically based on edge data, and as such do not guarantee consistency with the observed link loads on the interior of the network. Of course, in their full generality, gravity models can indeed include such information through the use of the $R \times R$ friction matrix for R ingress/egress points (see Section 3.1 for details). But the inference of the complete friction matrix is a problem of the same complexity as inference of the traffic matrix itself. We prefer to solve the latter problem directly.

Gravity models (based on edge data) are $O(R^2)$ complexity in the worst case, for R ingress/egress points (as we need to compute R^2 elements in the traffic matrix), but the number of computations is actually rather small per term, and so these methods are fast on even the largest network sizes. Hence gravity models are quite appealing, but we shall demonstrate the model of [2] can be significantly improved by incorporating information from the internal link measurements.

2. BACKGROUND

2.1 Network

An IP network is made up of IP routers and IP adjacencies between those routers, within a single autonomous system or administrative domain. It is natural to think of the network as a set of nodes and links, associated with the routers and adjacencies, as illustrated in Figure 2. We refer to nodes and links that are wholly internal to the network as *backbone* nodes and links, and refer to the others as *edge* nodes and links.

In addition, it is helpful for IP networks managed by Internet Service Providers (ISPs) to further classify the edge. As shown in the figure, in general the network will connect to other autonomous systems and customers via edge links. We categorize the edge links into *access* links, connecting customers, and *peering* links, which connect other (non-customer) autonomous systems. A significant fraction of the traffic in an ISP is *inter-domain* and is exchanged between customers and peer networks. Traffic to peer networks is today largely focused on dedicated peering links, as illustrated in Figure 2. Under the typical routing policies implemented by large ISPs, very little traffic will transit the backbone from one peer network to another. Transit traffic between peers may reflect a temporary step in network consolidation following an ISP merger or acquisition, but should not occur under normal operating circumstances.

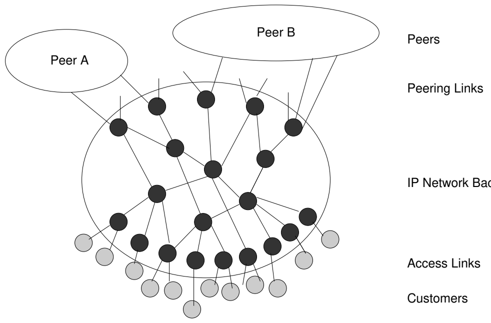


Figure 2: IP network components and terminology

In large IP networks, distributed routing protocols are used to build the forwarding tables within each router. It is possible to predict the results of these distributed computations, from data gathered from router configuration files. (The results provided here are based on router configuration files downloaded once daily). In our investigation, we employ a routing simulator such as in [19] that makes use of statically configured Border Gateway Protocol (BGP) and Interior Gateway Protocol (IGP) topology information gleaned from the configuration files. In operational IP networks, this information is quite stable on the time scales of interest.

2.2 Traffic Data

In IP networks today, link load measurements are readily available via the Simple Network Management Protocol (SNMP). SNMP is unique in that it is supported by essentially every device in an IP network. The SNMP data that is available on a device is defined in a abstract data structure known as a Management Information Base (MIB). An SNMP *poller* periodically requests the appropriate SNMP MIB data from a router (or other device). Since every router maintains a cyclic counter of the number of bytes transmitted and received on each of its interfaces, we can obtain basic traffic statistics for the entire network with little additional infrastructure support – all we need is an SNMP poller that periodically records these counters.

The properties of data gathered via SNMP are important for implementation of a useful algorithm – SNMP data has many limitations. Data may be lost in transit (SNMP uses unreliable UDP transport; copying to our research archive may introduce loss). Data may be incorrect (through poor router vendor implementations). The sampling interval is coarse (in our case 5 minutes). Many of the typical problems in SNMP data may be removed with minimal artifacts using simple techniques. For instance, using hourly traffic averages (with five minute data polls) mitigates the effect of missing data substantially. Slightly more sophisticated methods of anomaly detection and interpolation produce even better results, but we shall use simple hourly data for the purposes of this study, as hourly (or longer) data are commonly dealt with by many ISPs (with five minute or finer data kept for brief periods for trouble-shooting and alarming).

We use flow level data in this paper for validation purposes. This data is collected at the router which aggregates traffic by IP source and destination address, and TCP port numbers. This level of granularity is sufficient to obtain a real traffic matrix [13], and in the future such measurement may provide direct traffic matrix measurements, but at present limitations in vendor implementations prevent collection of this data from the entire network.

2.3 Terminology

For the purpose of computing traffic matrices, without loss of generality, we assume that all access and peering links terminate at *Edge Routers* (ERs), and that all remaining routers are *Backbone Routers* (BRs) that only terminate backbone links. (We can always insert dummy ERs to force the assumption to be true.) Figure 3 provides a simplified network topology to illustrate the terminology We make a further distinction that links between BRs are *core links*, and links between ER and BR are non-core links.

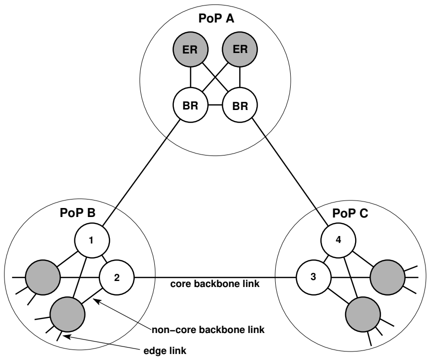


Figure 3: A simplified network topology to illustrate the terminology used here. Edge Routers (ERs) are shown shaded, while Backbone Routers (BRs) are unshaded.

Given two ERs E_i and E_j, the traffic between these edge routers T_{ij}^E is defined as the total amount of traffic that is enters the network at E_i and exits at E_j, with $\mathbf{T}^E = \{T_{ij}^E\}$ the associated matrix. We may also define traffic matrices between BRs $\mathbf{T}^B$ in a similar manner, where the elements refer to traffic entering and leaving the core. We will often refer to a vector form of the traffic matrix $\mathbf{t}$ in which the indices of the vector refer to source/destination pairs.

There may be more than one route between two routers even using only shortest paths. We assume that traffic will be evenly distributed across all such routes (though our method can be easily adapted to handle uneven distributions).

One could compute traffic matrices with different levels of aggregation at the source and destination endpoints, for instance, at the level of PoP to PoP, or router to router, or link to link [20]. In this paper, we are primarily interested in computing router to router traffic matrices, which are appropriate for a number of network and traffic engineering applications, and can be used to construct more highly aggregated traffic matrices (e.g. PoP to PoP) using routing information [20].

3. SOLUTION

In this section we provide our method, termed *tomogravity*, for computing the traffic matrix from link data. As its name indicates, the method consists of two basic steps – a *gravity modeling* step, and a *tomographic estimation* step:

1. In the gravity modeling step, an initial solution is obtained by solving a gravity model using edge link load data. We also incorporate static ISP routing policy information and explicitly model the traffic exchanged with peer networks.

2. In the tomographic estimation step, the initial solution is refined by applying quadratic programming to find a solution that minimizes the distance to the initial solution (in weighted least-square sense) subject to the tomographic constraints. We also apply knowledge of the network routing and topology configuration to significantly reduce the problem size. Iterative Proportional Fitting (IPF) is used to ensure non-negativity of the results.

Below we discuss each step in a separate subsection, followed by a brief summary of the complete algorithm.

3.1 Gravity Modeling

One of the simplest approaches to computing a traffic matrix is the *gravity model* [2, 3, 4, 5, 6]. Gravity models, taking their name from Newton's law of gravitation, are commonly used by social scientists to model the movement of people, goods or information between geographic areas [5, 6]. In Newton's law of gravitation the force is proportional to the product of the masses of the two objects divided by the distance squared. Similarly, in gravity models for cities, the relative strength of the interaction between two cities might be modeled as proportional to the product of the populations. A general formulation of a gravity model is given by the following equation:

$$X_{ij} = \frac{R_i \cdot A_j}{f_{ij}} \tag{4}$$

where X_{ij} is the matrix element representing the force from i to j; R_i represents the *repulsive* factors that are associated with "leaving" from i; A_j represents the *attractive* factors that are associated with "going" to j; and f_{ij} is a friction factor from i to j.

In our context, we can naturally interpret X_{ij} as the traffic volume that enters the network at location i and exits at location j, the repulsion factor R_i as the traffic volume entering the network at location i, and the attractivity factor A_j as the traffic volume exiting at location j. The friction matrix (f_{ij}) encodes the locality information specific to different source-destination pairs. The inference of all $R \times R$ friction factors is an equivalent problem of the same size as the inference of the traffic matrix itself. Accordingly, it is necessary to approximate the actual friction matrix using models with fewer parameters. In this paper, we shall assume a common constant for the friction factors, which is arguably the simplest among all possible approximation schemes. The resulting gravity model simply states that the traffic exchanged between locations is proportional to the volumes entering and exiting at those locations. Our results show that, remarkably, this gravity model when combined with detailed knowledge of ISP routing policies, is able to match the actual Internet data very well. One possible explanation for this is that geographic locality is not a major factor in today's Internet, as compared to ISP routing policies. As long as the gravity model captures the essence of the routing policies, it becomes very accurate and the choice of the friction factors is less critical.

Note that we do not expect our gravity model to accurately model the traffic between all source-destination pairs. In fact, one would naturally expect certain pairs of locations to stand out from the overall distribution, simply due to their specific characteristics (e.g. going through transoceanic links). A key insight of the tomogravity method is that we only need the gravity model to capture the overall distribution; we expect the tomographic estimation step to correct most of the violations in the assumptions underlying the gravity model and thus significantly improving the overall accuracy. It is certainly possible to further improve the method by using more accurate gravity models with additional parameters. However, as we show later in Section 4.3, the tomogravity method in its current simple form is already remarkably accurate. The margin for improvement may be limited.

Another important issue concerning the gravity model is the level of aggregation. Intuitively, we need the aggregation level to be sufficiently high so that the traffic exchanged between different locations is not sensitive to the detailed composition of the traffic. On the other hand, when the aggregation level is too high (e.g. PoP to PoP), ISP routing policies operating at a more granular level may have a profound impact and can introduce serious systematic distortion to the overall traffic pattern. In our tomogravity method, we apply gravity models at the link to link level, which is the finest level of resolution obtainable with SNMP data. We can easily use routing information to obtain more highly aggregated traffic matrices [20].

We formally present two (link to link) gravity models below. The first one is quite simple and is primarily used to provide insight into the approach. The second approach incorporates more realistic routing assumptions, and its performance is thereby much improved over the simple model.

3.1.1 A Simple Gravity Model

In this very simple gravity model, we aim to estimate the amount of traffic between edge links. Denote the edge links by l_1, l_2, ... We estimate the volume of traffic $T(l_i, l_j)$ that enters the network at edge link l_i and exits at edge link l_j. Let $T^{\rm in}_{\rm link}(l_i)$ denote the total traffic that enters the network via edge link l_i, and $T^{\rm out}_{\rm link}(l_i)$ the corresponding quantity for traffic that exits the network via edge link l_i. The gravity model can then be computed by either of

$$\begin{aligned} T(l_i, l_j) &= T^{\rm in}_{\rm link}(l_i)\frac{T^{\rm out}_{\rm link}(l_j)}{\sum_k T^{\rm out}_{\rm link}(l_k)}, \\ T(l_i, l_j) &= \frac{T^{\rm in}_{\rm link}(l_i)}{\sum_k T^{\rm in}_{\rm link}(l_k)} T^{\rm out}_{\rm link}(l_j). \end{aligned}$$

The first equation states that the traffic matrix elements $T(l_i, l_j)$ are the product of the traffic entering the network via edge link l_i and the proportion of the total traffic leaving the network via edge link l_j, while the second is reversed and is identical under traffic conservation – that is, the assumption that the interior network is neither a source, nor sink of traffic. While this assumption is violated (e.g., protocols running in the network interior act sources and sinks, and packet drops act as sinks) the volumes involved are insignificant in the network considered. Most notably the actual results from the two equations are almost identical.

3.1.2 Generalized Gravity Model

It is possible to generalize the simple gravity model of Section 3.1.1 to take into account a wide range of additional information provided by link classification and routing policy. In this Section, we will incorporate new information to model large ISP routing policies.

Typically, in large North-American ISPs, the majority of traffic is exchanged between network customers and network peers. The patterns and handling of customer and peer traffic are qualitatively different, and this has a large impact on the traffic matrix. Furthermore, this peering traffic has a large impact on every aspect of network design and engineering, and so estimating the associated traffic matrices is very important. In this Section, we adapt the gravity model to specifically differentiate between customer and peering traffic.

We assume that the network has a set of peers labeled $P_1, P_2, \ldots$, and exchanges traffic with peer P_i over a set of edge links dedicated to this peer. This is commonly termed *private peering* and is the dominant mode of peering for large IP backbones today. We also have a set of customer access links, labeled $a_1, a_2, \ldots$, and a set of peer links labeled $p_1, p_2, \ldots$. We denote the set of edge links carrying traffic to peer P_i by $\mathcal{P}_i$, and the set of all peer links by $\mathcal{P}$. We denote the set of all customer access links by $\mathcal{A}$. SNMP measurements provide volumes of traffic on all edge links, j: $T^{\rm in,out}_{\rm link}(j)$, where the superscripts *in* (*out*) denotes traffic into (out of) the backbone. The traffic entering, or exiting the network to peer P_i, is

$$T^{\rm x}_{\rm peer}(P_i) = \sum_{p \in \mathcal{P}_i} T^{\rm x}_{\rm link}(p),$$

where $x =$ in or out.

We will develop the equations for a gravity model under the following additional assumptions, which reflect dominant Internet and ISP routing policies:

- **Transit peer** (peering link to peering link) traffic. We assume that the volume of traffic that transits across the backbone from one peer network to another is negligible.

- **Outbound** (access link to peering link) traffic. We apply the proportionality assumption underlying gravity modeling on a peer-by-peer basis: that is, the traffic exiting to a specific peer comes from each access link in proportion to the traffic on that access link. We assume that all of the traffic from a single access link to the given peer exits the network on the same peering link (determined by the IGP and BGP routing configuration). We denote the exit peering link for traffic from access link a_i to peer P_j by $X(a_i, P_j)$. This may be derived from routing configuration information (See [13, 15].) The assumption is typically true in practice, except for example when short-term load balancing is performed. In such situations, our method could be supplemented with available statistics on the affected prefixes, though our experience is that the impact is small and does not affect the accuracy of the traffic matrix computations.

- **Inbound** (peering link to access link) traffic. A network operator has little control over the injection of traffic into its network from peer networks. Accordingly, we assume that the traffic entering from a given peering link is split amongst the access links in proportion to their outbound traffic.

- **Internal** (access link to access link) traffic. We assume that the fraction of internal traffic from a given access link a_i to a second access link a_j is proportional to the total traffic entering the network at a_i, and compute the traffic between the links by normalization.

Under these assumptions the outbound traffic from access link $a_i \in \mathcal{A}$ to peering link $p_m \in \mathcal{P}_j$ is

$$T_{\rm outbound}(a_i, p_m) = \begin{cases} \frac{T^{\rm in}_{\rm link}(a_i)}{\sum_{a_k \in \mathcal{A}} T^{\rm in}_{\rm link}(a_k)} \, T^{\rm out}_{\rm peer}(P_j), & \\ & \text{if } p_m = X(a_i, P_j), \\ 0, & \text{otherwise.} \end{cases}$$

The inbound traffic from peering link p_i to access link a_j is

$$T_{\rm inbound}(p_i, a_j) = T^{\rm in}_{\rm link}(p_i)\frac{T^{\rm out}_{\rm link}(a_j)}{\sum_{a_k \in \mathcal{A}} T^{\rm out}_{\rm link}(a_k)}.$$

The internal traffic from access link a_i to access link a_j is

$$T_{\rm internal}(a_i, a_j) = \frac{T^{\rm in}_{\rm link}(a_i)}{\sum_{a_k \in \mathcal{A}} T^{\rm in}_{\rm link}(a_k)} T^{\rm out}_{\rm internal}(a_j).$$

where

$$\begin{aligned} T^{\rm out}_{\rm internal}(a_j) &= T^{\rm out}_{\rm link}(a_j) - \sum_{p_k \in \mathcal{P}} T_{\rm inbound}(p_k, a_j) \\ &= T^{\rm out}_{\rm link}(a_j)\left(1 - \frac{\sum_{p_i \in \mathcal{P}} T^{\rm in}_{\rm link}(p_i)}{\sum_{a_k \in \mathcal{A}} T^{\rm out}_{\rm link}(a_k)}\right). \end{aligned}$$

The complexity of the algorithm is $O(N^2)$ in the number of edge links being considered, but the number of operations per term is small. Computation of the generalized gravity model for the complete network in question (of the order of 1000 routers) took less than 3 seconds (on a 336 MHz Ultrasparc-II processor). Despite the speed and simplicity of gravity models such as that expressed above, the models have the significant drawback of not guaranteeing consistency with the internal link measurements in the network. Remedying this leads us to the tomographic methods discussed next.

3.2 Tomography

Network tomography, as mentioned earlier, is the problem of determining the end-to-end traffic matrix from link loads. The link traffic is the sum of the traffic matrix elements that are routed across that link, and so we may see our problem as follows. We have a set of observables $\mathbf{x} = (x_1, x_2, \ldots, x_L)^T$, the traffic (as measured in packets or bytes) that traverses the L links of the network during some period, which derive from the traffic matrix following equation (2) where $\mathbf{t}$ is the traffic matrix, written as a column vector $\mathbf{t} = (t_1, t_2, \ldots, t_M)^T$, where the M traffic matrix elements, t_r, are the traffic between the rth source/destination pair, and $\mathbf{A} = \{A_{ir}\}$ is the $L \times M$ routing matrix where

$$A_{ir} = \begin{cases} F_{ir}, & \text{if traffic for } r \text{ traverses link } i \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where F_{ir} is the fraction of traffic from source/destination pair r that traverses link i. In some formulations of the problem, an additive term ϵ appears on the right hand side of equation (2) to model measurement error.

We need to solve the inverse problem to obtain $\mathbf{t}$. For general topologies and routing there are typically many more unknowns than constraints, and so equation (2) is highly under-constrained and does not have a unique solution. In order to infer $\mathbf{t}$ it is useful to adopt a traffic model of some kind – in the previous work on network tomography the model allows the addition of higher-order statistical constraints. Our approach is not to incorporate additional constraints, but rather to use the gravity model to obtain an initial estimate of the solution, which needs to be refined to satisfy the constraints. The details are presented below, but first it is important to reduce the size of the problem to make computation of the solution more manageable.

3.2.1 Reducing the Computational Complexity

One of the main problems with practical application of any tomographic technique is the size of the routing matrix. The chosen data samples in the study cover on the order of a thousand routers, which would lead to an intractably large problem (with millions of unknowns). One approach is to deal only with smaller problems. For example, [7] examines PoP to PoP traffic matrices. In this paper, we focus on computing BR to BR traffic matrices, which are generally much more useful for traffic engineering tasks such as load balancing, and are absolutely necessary for link/router failure analysis. We can directly derive PoP to PoP traffic matrices from BR to BR matrices using routing information. However, even if we only consider backbone routers there may be of the order of one hundred, leading to a problem with over ten thousand unknowns, which is orders of magnitude more than the available constraints on link traffic volume.

In this section we develop techniques that can significantly reduce the number of unknowns (in our case by a factor of 10 for the BR to BR traffic matrix) and thus making the computation of traffic matrix both more accurate and more efficient.

Our key observation is that many BR to BR matrix elements are empty as a result of the fact that there may be multiple BRs within each PoP, and so traffic will flow only between the closest of these (as determined by IGP routing). For instance consider the simplified topology shown in Figure 3. Here there are two BRs in each PoP, connecting ERs within the PoP with redundant links. Given shortest path routing (and equal link weights on backbone links), one can see that all of the traffic from PoP B, to PoP C will traverse the route through BRs 2 and 3, while there will be no traffic entering the backbone nodes at BR 1 and departing at BR 4. While this is a very simple example, in operational IP networks, the set of paths consistent with IP routing will typically be significantly less than the set of all paths between router pairs.

Consider the BR to BR traffic matrix, and denote the traffic matrix from BR B_i to B_j by T_{ij}^B. We use the following simple algorithm for removing all the empty demands:

1 Initially mark all elements of the BR to BR traffic matrix as empty
2 For each pair of ERs simulate the IGP as in [19] to find the shortest paths between the source and destination router;
3 For each path, let B_i and B_j be its first and last BRs respectively, and mark T_{ij}^B as not empty;
4 Remove all T_{ij}^B that remain empty. This step is equivalent to removing elements from $\mathbf{t}$ that will be zero because the corresponding route is not used (unless failures occur).

We can exploit the *topological equivalence* of ERs to avoid having to run IGP simulations for all possible pairs of routers, which can be prohibitive due to their large number. We consider two ERs to be *topologically equivalent* if they connect to the same (non-empty) set of BRs and the IGP weights on the corresponding links are the same[1]. We group such equivalent edge routers together, and consider them as a single *Edge Router Equivalence Class* (EREC). The key observation is that the routes between the component ERs of the same pair of ERECs will be the same except for the first and last links. Consequently, we only need to run one IGP simulation for each pair of ERECs. We found that computing routes on this basis reduces the computational burden by at least a factor of 20.

After eliminating all the empty demands we are able to reduce the number of unknowns by a factor of 10, thereby turning a highly under-constrained problem into a moderately under-constrained problem, while making the computation orders of magnitude faster.

The routing matrix is required for all methods except the simple gravity model. The time taken in computing the routing matrix dominates all other algorithms we use, taking two to three minutes. However, note that this cost can often be amortized over multiple traffic matrix computations because we only need to recompute the routing matrix when the network topology changes. The algorithm to reduce the problem size can be performed as part of computing the routing matrix, with a computational cost that is a very small marginal cost on top of computing the routing matrix itself.

3.2.2 Quadratic Programming

From the (link to link) gravity model, we can easily obtain an estimate of the BR to BR traffic matrix ($\mathbf{T}^B$) using routing information [20]. The link to link traffic matrix elements are routed across the backbone, and we then compute the traffic volumes between each pair of backbone routers (note that we loose the distinction between peering and access traffic in this step). We term the resulting estimate as the *gravity model solution* (denoted as $\mathbf{t}_g$). We then refine the gravity model solution by using a least-square solution that minimizes the Euclidean distance to the gravity model solution subject to the tomographic constraints. More specifically, we would like to solve the following quadratic program

$$\begin{array}{ll} \min & \| \mathbf{t} - \mathbf{t}_g \| \\ \text{s.t.} & \| \mathbf{A}t - x \| \text{ is minimized} \end{array}$$

where $\| \, . \, \|$ is the L_2 norm of a vector (i.e., the Euclidean distance to the origin).

We also investigate weighted least-squares solutions in which the projection onto the subspace is not orthogonal, but rather weighted by a function of the size of the estimated traffic matrix elements ($\mathbf{t}_g$). That is, we use $\| \ (\mathbf{t} - \mathbf{t}_g)/\mathbf{w} \ \|$ as the objective function to minimize in the above quadratic program, where $\mathbf{w}$ is the weight vector, and $/$ is the element-by-element vector division. We investigate three weighting schemes in this paper; the weight for each term of the traffic matrix is either: constant, linearly proportional to the terms in the gravity model traffic matrix, or proportional to the square root of the gravity model.

[1]There may be a multiple layer hierarchy of ERs within a PoP, and lower layer ERs pass traffic to BRs only through the higher layer ERs. We may remove the lower layer of ERs from consideration as we see their (network impacting) traffic at the higher layer.

Figure 4 illustrates the approach in a simple case with two unknowns (unknown traffic along two routes), and one constraint (one known link load). The constraint(s) specify a sub-space of the parameter space (in this case just a line), while the gravity model is one particular solution. The simple least-squares approach is just an orthogonal projection of the gravity model solution onto the constraint sub-space. The weighted least-squares solution gives different weights to different unknowns in the solution.

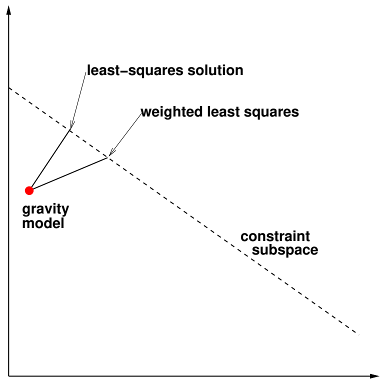


Figure 4: An illustration of the least-square solution. The point shows the gravity model solution, and the dashed line shows the subspace specified by a single constraint equation. The least-square solution is simply the point which satisfies the equation which is closest to the gravity model solution. The weighted least-squares solution gives different weight to different unknowns.

Note that the tomographic constraints may be ill-posed due to possible dependency among different link constraints. Furthermore, the constraints may not be satisfiable due to error and noise in the link load data or possible routing changes that are not captured by the topology data. The standard technique for dealing with ill-posed quadratic programs is to use Singular-Value Decomposition (SVD) of the routing matrix $\mathbf{A}$ to compute its pseudo-inverse. The resulting solution is the closest to the initial solution $\mathbf{t}_g$ among all solutions that minimize the discrepancy against the tomographic constraints ($\| \mathbf{A}t - x \|$). Routines to compute the pseudo-inverse are available in many numerical computing packages (for instance see [21]). We have implemented our method in Matlab, and the actual algorithm is so simple that it only takes 6 lines of code (see the Appendix).

The worst case complexity of the above algorithm is linear in the number of unknowns (elements of the traffic matrix), and quadratic in the number of constraints, however, in practice the complexity of singular value decomposition methods is generally less than this. For instance the SVD used in LAPACK [21], and thence Matlab (which uses these routines) are usually better than this complexity estimate would indicate. In reality, the algorithm is very fast, with the computation times on a 336 MHz Ultrasparc-II processor (running Solaris 7) all significantly under 2 seconds.

One additional locus of complexity is that the least-square algorithm may result in negative values, which are without physical meaning. One can avoid this by viewing the problem as a constrained optimization problem. However, a simple iterative procedure provides a fast and effective alternative. Specifically, we use Iterative Proportional Fitting (IPF) as suggested in [10] to ensure non-negativity. We do not model higher order statistics of the process, and therefore, the initial condition we use is not as complex as that in [10]. For the initial estimate, we simply use the traffic matrix estimated above, with zero replacing the negative elements of the matrix. IPF then proceeds by successively refining the estimate using the same method as [10]. In practice it only takes a few iterations to reduce errors in the constraint equations to the point at which they are negligible.

3.3 The Complete Algorithm

Below we summarize the complete tomogravity method.

1. Apply the generalized gravity model to obtain a link to link traffic matrix.
2. Transform the above traffic matrix into a BR to BR traffic matrix, and use it as our initial gravity model solution ($\mathbf{t}_g$).
3. Reduce the complexity of the tomography problem by removing empty BR to BR demands. Topological equivalence of ERs can be exploited to speed up the route computation by a factor of 20.
4. Apply singular value decomposition to solve the quadratic program and find a solution that minimizes its distance to $\mathbf{t}_g$ (in weighted least-square sense) subject to the tomographic constraints.
5. Replace negative values with zero, and perform IPF to obtain a non-negative solution that satisfies the constraints.

4. VALIDATION

In this section, we validate our methods using actual Internet data. We first describe the basic validation methodology. We then validate the underlying assumptions for the gravity models we use. This is followed by a detailed comparison of the quality of the results obtained by different methods. Finally, we present some preliminary results on the robustness of the tomogravity method.

4.1 Methodology

In an ideal environment, we would like to validate the results using a consistent set of real traffic matrix, topology, and link measurements. Unfortunately, we do not have the complete set of flow level data across the edge of the network (due to problems in the vendor implementations of flow collection). Furthermore, the routing data we have available comes in 24 hour snapshots, and the flow level data is sampled data, with its own limitations. Therefore, it is not possible for us to obtain a complete traffic and routing matrix consistent with the actual SNMP link measurements.

We solve the problem of providing a consistent set of traffic, topology and link measurement data as follows. We first derive a set of hourly (partial) traffic matrices based on direct flow level measurements using the methodology described in [13]. We then simulate OSPF routing using the topology and routing information. From this we may compute a routing matrix $\mathbf{A}$, and then derive a set of link measurements $\mathbf{x}$ from equation (2). Thus the traffic matrix $\mathbf{t}$, the routing matrix $\mathbf{A}$ and the measured link loads $\mathbf{x}$ are all consistent.

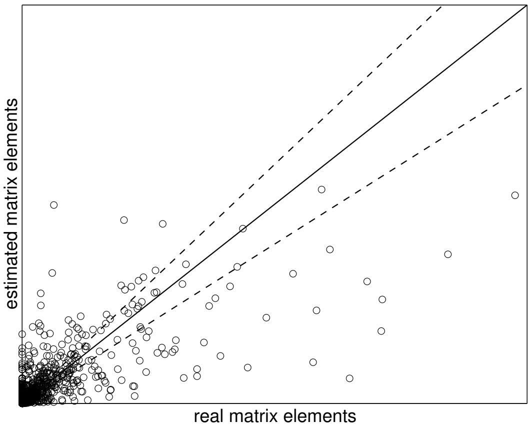
We can then estimate $\hat{\mathbf{t}}$ for a problem for which we know the "ground truth" – the real traffic matrix. This approach has the additional advantage that it isolates the impacts of measurement errors and inconsistencies from the performance of the algorithms (the topic of this paper). We can thereby study the impact of measurement errors separately, and more precisely, as we can induce the errors, rather than infering them from multiple (possibly inconsistent) data sets.

Note that we use use derived link load measurements for validation purposes only. In practice, our method is applicable to direct link data such as one would obtain from SNMP, and we have in fact applied them directly to SNMP data for operational tasks in capacity planning and network management.

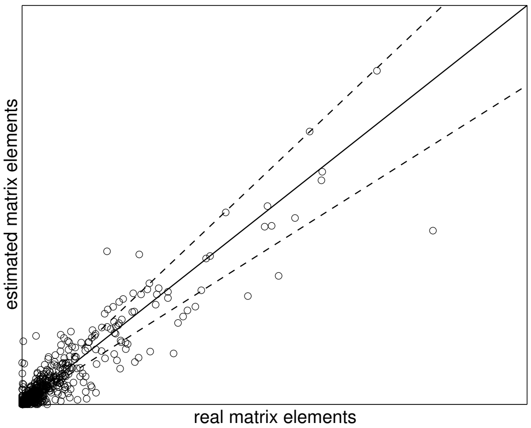
4.2 Gravity Models

In this section, we validate the basic assumptions underlying our gravity models, in particular the assumption on constant friction factors, which is the simplest possible approximation to the $R \times R$ friction matrix.

By simple rearrangements of the basic equations of the gravity model, it is clear that when the estimated friction factors are constant, the actual friction factors are constant if and only if the ratio between the estimated and the actual matrix elements are constant. Clearly the constant should be 1 as the gravity models maintain the correct total traffic volume through appropriate normalization. As a result, to test if the actual friction factors are constant, one only need to verify that the points (X^a_{ij}, X^e_{ij}) lie on a straight line passing through the origin. It is easy to show that the same analysis applies for the simple and the generalized gravity models introduced in Section 3.1.



(a) Simple gravity model



(b) Generalized gravity model

Figure 5: A comparison of the real traffic matrix elements to the traffic matrix elements estimated by the gravity model. The solid diagonal line shows equality, while the dashed lines show $\pm 20\%$.

Figure 5 illustrates the two link to link gravity models for one hour chosen randomly from June 2002. In each case we plot the true value of the elements of the traffic matrix versus the estimated value. Clearly, the closer the points cluster around the diagonal, the more accurate the model. It appears that the generalized gravity model is more accurate than the simple model.

To quantify the linear association between the real and the estimated matrix elements, we compute the correlation coefficient for each hourly traffic matrix, along with the least-square regression line. We need to verify two facts: i) the correlation coefficient is close to 1 (so that the points are closely centered around the regression line), and ii) the intercept of the regression line is close to 0.

Figure 6 shows the distribution of the correlation coefficients over the entire June 2002 dataset. For the simple gravity model, the correlation coefficient typically lies between 0.65 and 0.8, which are significant, but not as large as we would like. In contrast, the correlation coefficients for the generalized gravity model are consistently above 0.9, indicating very strong linear association.

We next compute the 95% confidence interval for the intercept of the regression line under the assumption that the regression residuals is Gaussian. We find that the 95% confidence interval covers 0 on

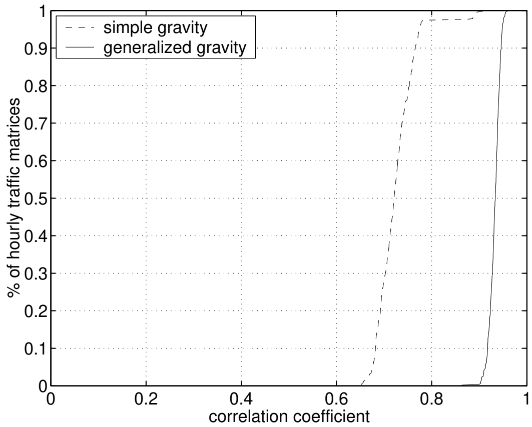


Figure 6: Distribution of the correlation coefficients between real and gravity estimate of traffic matrix.

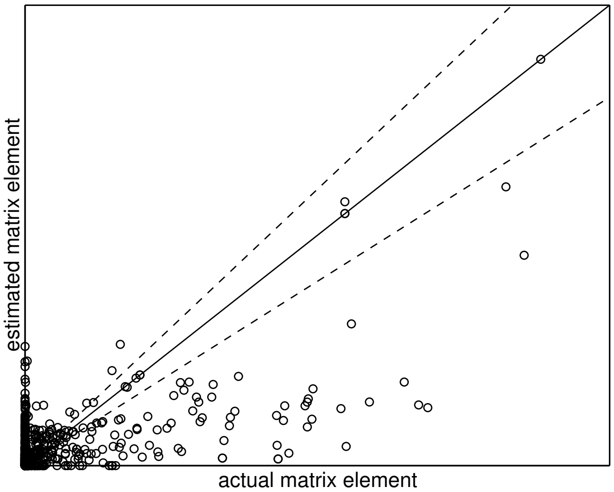
all 507 hourly traffic matrices for both the simple and the generalized gravity model. This indicates that the intercept is indeed close to 0.

These results show that the generalized gravity model is significantly more accurate than the simple gravity model, which highlights the importance of explicitly taking into account ISP routing policies in gravity modeling. Moreover, despite its simplistic assumptions about the traffic (i.e. proportionality, and constant friction factor), the model can capture the overall distribution of the matrix elements very well. As a result, the resulting estimate can provide a good starting point for the tomographic estimation step.

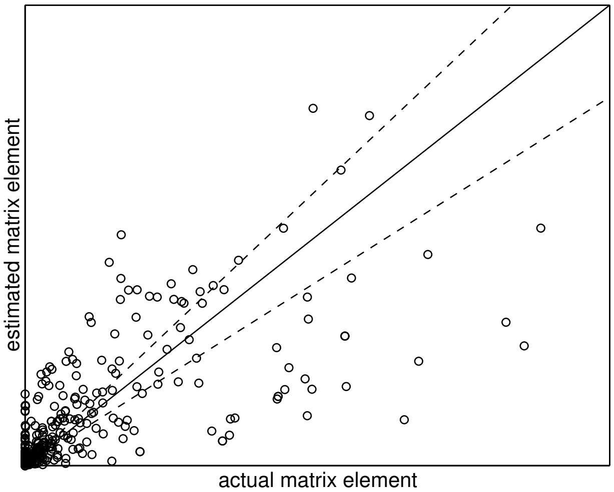
4.3 Performance

The main method of validation we use is to directly compare the estimated traffic matrix elements with the true traffic matrix. Figure 7 provides a comparison of methods for computing the gravity matrix for one hour chosen randomly from June 2002. In each case we plot the true value of the elements of the traffic matrix versus the estimated value. (The exact values of traffic on the network in question are considered proprietary so the scales are not present, but are not needed for the comparison to be valid.) The solid diagonal line shows equality and the dashed diagonal lines show $\pm 20\%$. For a base-line comparison we first show in Figure 7 (a) the least-square solution to equation (2), to demonstrate that a direct solution of the equation does not produce a good result. Figure 7 (b) shows the result of the simple gravity model, which are also fairly inaccurate overall, though better than the pure least squares approach. Figure 7 (c) shows the generalized gravity model, and we can see a dramatic improvement over the previous algorithms, though it is still not perfect. Finally, Figure 7 (d) shows the results with the generalized gravity model, and the least-squares solution (using square root weights). This final solution appears remarkably good.

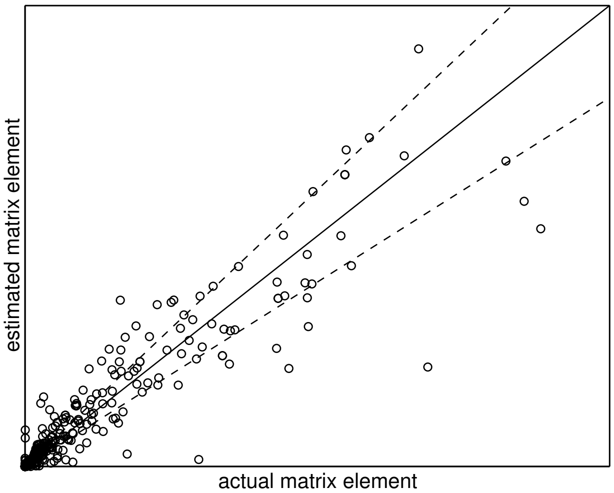
To quantify the quality of the results, it might seem ideal to establish a direct link to traffic and network engineering applications, for instance for capacity planning. However, these involve complex algorithms in their own right, and so we shall attempt to present some simple metrics for the quality of the results which have some linkage to the engineering requirements. The requirements (drawn from extensive conversation with engineers doing capacity planning and network design) can be approximately stated thus: we require reasonable relative accuracy for the larger elements of the traffic matrix, and not too large absolute errors for all matrix elements. How large is a reasonable error? In light of the other sources of error in Internet planning, and accuracy measured in a few ten's of percent is quite tolerable, especially if the errors for the larger flows are smaller. This is further backed up by work such as that in [22], which showed that



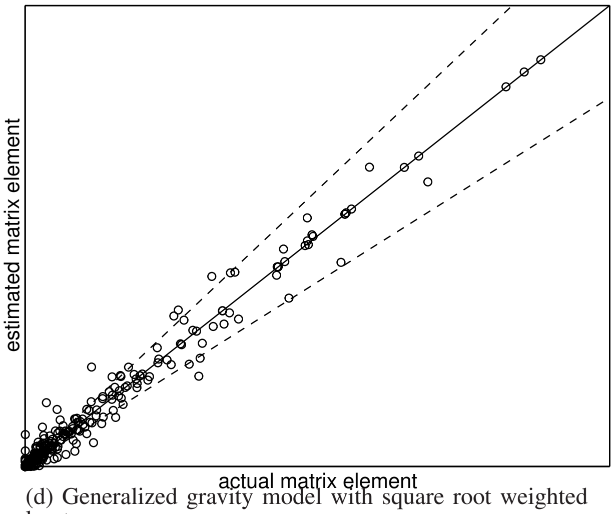
(a) Raw least squares



(b) Simple gravity model



(c) Generalized gravity model



(d) Generalized gravity model with square root weighted least squares

Figure 7: A comparison of the real traffic matrix elements to the estimated traffic matrix elements for four different algorithms. All plots are shown on the same scale. The solid diagonal line shows equality, while the dashed lines show $\pm$20%.

Algorithm			traffic matrix errors		core link errors	
initial solution	LSE	weights	RMSE (Mbps)	RMSRE	RMSE (Mbps)	RMSRE
Raw	yes	N/A	89	65%	174	37%
Simple Gravity	no	N/A	85	62%	260	54%
Simple Gravity	yes	const	27	22%	9	2%
Simple Gravity	yes	linear	28	24%	4	1%
Simple Gravity	yes	root	25	21%	4	1%
General Gravity	no	N/A	40	31%	117	24%
General Gravity	yes	const	16	13%	4	1%
General Gravity	yes	linear	16	13%	4	1%
General Gravity	yes	root	15	12%	4	1%

Table 1: Performance of the various algorithms over 507 individual hourly data sets from June 2002. The RMSREs are computed on the largest matrix elements comprising 75% of the network traffic, and the RMSE is computed on all the traffic matrix elements for the month.

at least one networking task (routing optimization) could still provide substantial capacity improvements even when the errors in the input traffic matrices were significantly larger than those reported here.

It seems that a single metric is unlikely to cover this adequately, so we shall present several. Two shown here are the Root Mean Squared Error (RMSE), and the Root Mean Squared Relative Error (RMSRE), defined below.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2}. \quad (6)$$

$$\text{RMSRE}(T) = \sqrt{\frac{1}{N_T}\sum_{\substack{i=1\\ x_i>T}}^{N}\left(\frac{\hat{x}_i - x_i}{x_i}\right)^2}. \quad (7)$$

The RMSE gives an overall metric for the errors in the estimates, and the RMSRE gives a relative measure, but note that the relative errors for small matrix elements are not really important for engineering, and so we when computing the RMSRE we take only matrix elements greater than some threshold T (and therefore when computing the mean normalized by the number of matrix elements greater than T, namely $N_T = \sum_{i=1,x_i>T}^{N} 1$).

We compute these metrics not just to compare the errors in the traffic matrices themselves, but also to compare the observables. That is we use the estimated traffic matrix to predict the link loads on core links, and compare with the real load on these links.

Table 1 summarizes the results for the different possible algorithms – in these results T is chosen so that 75% of the traffic is capture by the RMSRE. Note that this truncation removes elements of the traffic matrix whose magnitude is at most 5% of the capacity of the smallest link the the backbone network being considered. Hence, even quite large relative errors can be tolerated in these terms as long as the errors are not highly correlated (which should be guaranteed by satisfying the tomographic constraints).

The first three columns of the table specify the algorithm by its initial conditions, whether the LSE method is applied, and if it is, what weights are used. The quality metrics above are presented both for the errors in the estimated traffic matrix, and the estimated core link traffic, and are based on our entire set of data for June 2002 (though there are missing days in this data set the data set still contains over 500 one hour intervals of data).

The RMSE results are quoted in Mbps and these may all seem relatively small in comparison to the scale of the network – the backbone consists primarily of reasonably well utilized OC48 and OC192 links (2.5 and 10 Gbps respectively), but in particular note their relative values for comparison. The link errors are not as interesting as the traffic matrix errors in some respects, because there are many possible (incorrect) solutions which can produce the correct counts. However, it is important from a practical standpoint to get these at least approximately correct, and the results show that without a tomographic component, the link traffic estimates may be quite inaccurate. When a tomographic component is used the errors in the link loads are of the same magnitude as the targeted error level used in the IPF component of the algorithm, as required.

The results uniformly show[2] that (a) the simple gravity model is better than the raw least-squares algorithm, and that (b) using the least-squares with a gravity model initial condition provides a definite improvement, not just in the estimated link rates (which is what the method directly improves), but also in the traffic matrix estimates. Further the generalized gravity model is better than the simple gravity model. However the best result comes from tomogravity, using the generalized gravity model with the weighted least-squares method using square root weights (though the improvement over using other weightings is small).

[2]Other performance metrics investigated uniformly support the conclusion drawn from Table 1.

The statistics above appear to favor using the square root weighting scheme presented above. We investigate further to see whether this is really the case. In Figure 8 we vary T in computing the RMSRE and show the RMSRE for the three weightings considered, compared with the volume of traffic in the matrix elements above the threshold. The results show that no one scheme is a clear winner, but the square root scheme seems to win overall. We have investigated some other weighting schemes but have not found a superior one.

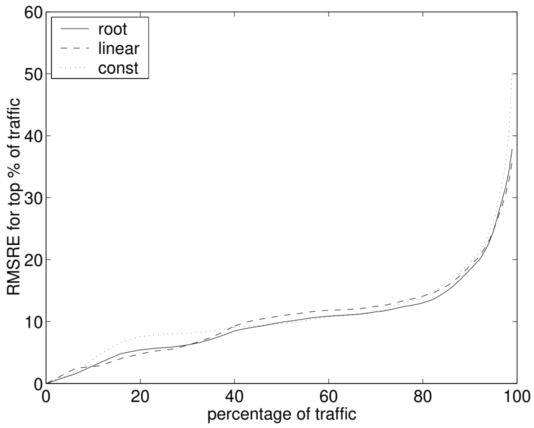


Figure 8: The RMSRE for varying threshold for the June 2002 data. The x-axis shows the proportion of the network traffic falling above the threshold, and the y-axis shows the RMSRE for the varying threshold. The results show that the RMSRE is smaller for the larger matrix elements.

The plot also provides another view into the performance of the algorithm, as the threshold used is varied. For instance, Figure 8 also demonstrates that the results are better for the larger matrix elements (as the threshold increases the volume of the thresholded matrix elements decreases, and the performance improves). The RMSRE actually drops below 5% for the top 20% of traffic. This is a *very* desirable result, but we can also see that a very large proportion of the traffic (over 90%) has a RMSRE less than 20%.

Figure 1 shows the distribution of errors for the generalized gravity model with square-root weighted least squares, and the particularly noteworthy point to take away from this plot is that more than 30% of the estimated matrix elements have a negligible error (less than 0.5%). The plot also shows the 5th and 95th percentiles of the distribution that lie within $\pm 23\%$. Furthermore, the plot shows that the results are not biased in one direction or the other. Thus, when we aggregate these results to higher levels, for instance PoP to PoP traffic matrices, the errors should not add to make things worse. We examine whether this is the case in the following figure.

Figure 9 shows the percentage of traffic (over the whole data set) for which the magnitude of the relative error below some value (for the generalized gravity model with square-root weighted least squares). Once again we see good results: more than 30% of the traffic has negligible relative error, and nearly 70% has an error less than 10% for the BR to BR traffic matrix. Further, as we have noted earlier, it is possible to derive PoP to PoP traffic matrices directly from the BR to BR traffic matrices. The dashed line in Figure 9 shows the equivalent curve for such matrices, showing that the performance is actually slightly better over the majority of the distribution for the PoP to PoP matrices. In fact, the RMSE for the PoP to PoP matrix is 10 Mbps, while the RMSRE is 11%.

Another method to assess the quality of the results is to look at the prediction our estimated traffic matrix would make of properties of the network. A simple example is presented in Figure 10, which shows the percentage of traffic in the largest N matrix elements. The

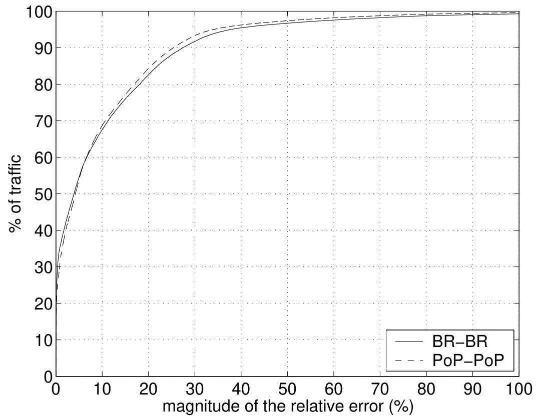


Figure 9: The percentage of traffic where the magnitude of the relative error below some value (x-axis) for the June 2002 data. The solid line shows the curve for the backbone router to backbone router matrix, and the dashed line shows the curve for the PoP to PoP traffic matrix. Note that for nearly 30% of the traffic the error is negligible.

plot shows how skewed the traffic matrix is, and how important it is to estimate the largest matrix elements correctly. The figure shows the actual distribution, and the estimated distribution – they are hard to distinguish because there is almost no discernible difference.

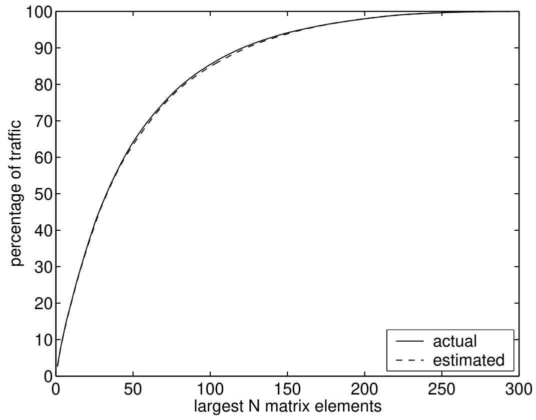


Figure 10: The percentage of traffic in the largest N matrix elements for the real traffic matrix (solid) and the estimated traffic matrix (dashed).

Finally, consider what happens to the estimates over time. We want to be able to answer questions about the variability of the traffic, or to detect trends or changes in the traffic matrix. It is therefore desirable for the results to remain stable – that is, for the errors in the estimated traffic matrix to remain approximately constant over time, even as the traffic matrix itself changes. Figure 11 shows two comparisons of elements of the traffic matrix over time, with their predicted values (note the gaps are where our archive is missing data). The first shows a case where the error is very small. The two curves (the actual and estimated matrix elements) are almost indiscernible, even though we can see clear diurnal variation in the traffic, and a profound drop in the traffic at one point where the network topology changed dramatically. The estimates track the traffic matrix perfectly. The second plot shows the results for a case that is not so accurate, but once again we see that the estimated value follows the variation of the true flow smoothly. Thus even where there are errors in the traffic matrix, the results can still be used for a range of operational tasks such as detecting traffic changes.

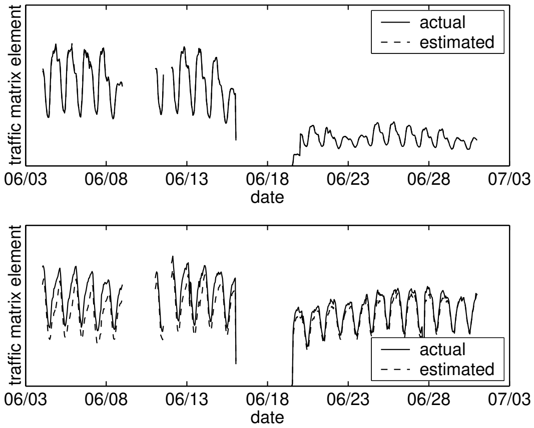


Figure 11: Comparison for two matrix elements over time. In the first case we see that the estimated value tracks the true value almost exactly, while in the second case although there is an error in the estimate, the error changes only slowly over time. Gaps in the data indicate missing data in our archive.

4.4 Robustness

In this section, we evaluate the impact of measurement errors and inconsistencies on the performance of the tomogravity method by inducing an error term ϵ to the tomography constraints:

$$\mathbf{x} = \mathbf{A}\mathbf{t} + \epsilon. \tag{8}$$

All the numerical results presented here are for the tomogravity method using the generalized gravity model with the weighted least-squares method using square root weights. The error term ϵ is formed by multiplying the $\mathbf{x}$ with a white noise term. Specifically,

$$\epsilon = \mathbf{x} * N(0, \sigma), \tag{9}$$

where $*$ is element-by-element vector multiplication, and $N(0, \sigma)$ is a vector with random entries drawn from a normal distribution with mean 0 and standard deviation σ. To ensure the non-negativity of $\mathbf{x} - \epsilon$, we cap the normal distribution at 1.

	traffic matrix errors		core link errors	
Noise Level	RMSE	RMSRE	RMSE	RMSRE
noise free	15 Mbps	12%	4 Mbps	1%
$\sigma = 0.01$	15 Mbps	13%	6 Mbps	1%
$\sigma = 0.02$	16 Mbps	14%	10 Mbps	2%
$\sigma = 0.04$	19 Mbps	17%	18 Mbps	4%

Table 2: Performance of the tomogravity method under different noise levels (0%, 1%, 2% and 4%) over 507 individual hourly data sets from June 2002.

Table 2 summarizes the performance of the tomogravity method under different levels of noise (0%, 1%, 2% and 4%)[3]. The results show that while the induced noise makes the estimated traffic matrix less accurate (as one would expect), the overall performance degradation is small – the additional estimation errors on both matrix elements and link observables are of the same size, or smaller than the introduced errors on the observables. Similar behavior is also evident in Figure 12, where we vary the threshold T in computing the RMSRE and show the RMSRE compared with the volume of traffic in the matrix elements above the threshold. While a more detailed study of robustness must thoroughly investigate the types of measurement errors seen in practice, these results show that the tomogravity method is robust to measurement errors on the observables.

[3] T is chosen to capture the top 75% traffic, as in Table 1.

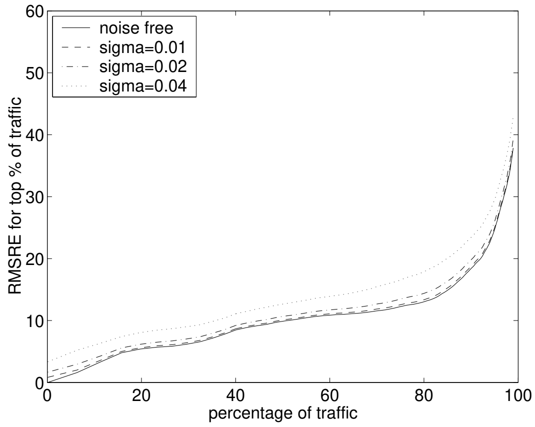


Figure 12: The RMSRE for varying threshold for the June 2002 data. The x-axis shows the proportion of the network traffic captured by the matrix elements that fall above the threshold, and the y-axis shows the RMSRE for the varying threshold. The results show the overall performance degradation due to induced noise is small.

5. CONCLUSION

We have presented a simple, fast and practical method called tomogravity for computing router to router traffic matrices in large IP networks, based on widely available link measurements. The tomogravity method combines the best features of gravity models and tomographic techniques. The method scales to handle networks larger than any previously attempted using alternative methods, and it is very fast (taking less than 5 seconds for a large network). Furthermore, this router to router traffic matrix may be used to directly derive an even more accurate PoP to PoP traffic matrix.

We have validated the method on the a large commercial IP backbone network, where direct measurements of the traffic matrix were available for June 2002. The results show remarkable accuracy for the larger elements in the traffic matrix: the estimates typically fall within a few percent of true values. These larger values dominate network and traffic engineering applications. However, the accuracy over all the matrix elements is still reasonable, and Tomogravity is now being used for a variety of capacity planning and management tasks in this network.

There are several avenues for further improvements. In particular, we would like to be able to include additional constraints in the solution. For instance, if one has partial flow level data available, this could provide additional constraints. Another avenue of exploration is the use of alternative starting points such as that proposed in [7].

Acknowledgments

We would like to thank Carsten Lund, Fred True, Joel Gottlieb, and Tim Griffin, whose work in collecting, and managing the data used here, made this research possible, Quynh Nguyen for motivation, encouragement and operational clue, and Jennifer Rexford, Aman Shaik and Chuck Kalmanek for many helpful comments.

6. REFERENCES

[1] M. Grossglauser and J. Rexford, "Passive traffic measurement for IP operations," in *The Internet as a Large-Scale Complex System* (K.Park and W. Willinger, eds.), Oxford University Press, 2002.

[2] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning (extended abstract)," in *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[3] D. Lam, D. Cox, and J. Widom, "Teletraffic modeling for personal communications services," *IEEE Communications Magazine: Special Issues on Teletraffic Modeling Engineering and Management in Wireless and Broadband Networks*, 35, pp. 79–87, February 1997.

[4] J. Kowalski and B. Warfield, "Modeling traffic demand between nodes in a telecommunications network," in *ATNAC'95*, 1995.

[5] J. Tinbergen, "Shaping the world economy: Suggestions for an international economic policy." The Twentieth Century Fund, 1962.

[6] P. Pyhnen, "A tentative model for the volume of trade between countries," *Weltwirtschaftliches Archive*, 90, pp. 93–100, 1963.

[7] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *ACM SIGCOMM*, (Pittsburg, USA), August 2002.

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

[9] C. Tebaldi and M. West, "Bayesian inference on network traffic using link count data," *J. Amer. Statist. Assoc*, 93, pp. 557–576, 1998.

[10] J. Cao, D. Davis, S. V. Wiel, and B. Yu, "Time-varying network tomography," *J. Amer. Statist. Assoc*, 95, pp. 1063–1075, 2000.

[11] A. Adams, T. Bu, R. Cáceres, N. Duffield, T. Friedman, J. Horowitz, F. L. Presti, S. Moon, V. Paxson, and D. Towsley, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications Magazine*, May 2000.

[12] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, May 2002.

[13] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Transactions on Networking*, pp. 265–279, June 2001.

[14] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proceedings of IEEE GLOBECOM '99*, (Rio de Janeiro, Brazil), pp. 1859–1868, 1999.

[15] N. Feamster, J. Borkenhagen, and J. Rexford, "Controlling the impact of BGP policy changes on IP traffic," Tech. Rep. 011106-02-TM, AT&T Labs – Research, Nov 2001.

[16] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm (with discussion)," *J. Roy. Statist. Soc. Ser.*, 39, pp. 1–38, 1977.

[17] J. Cao, S. V. Wiel, B. Yu, and Z. Zhu, "A scalable method for estimating network traffic matrices from link counts." preprint.

[18] J. G. Klincewicz. private communications, 2000.

[19] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "Netscope: Traffic engineering for IP networks," *IEEE Network Magazine*, pp. 11–19, March/April 2000.

[20] A. Medina, C. Fraleigh, N. Taft, S. Bhattacharyya, and C. Diot, "A taxonomy of IP traffic matrices," in *SPIE ITCOM: Scalability and Traffic Control in IP Networks II*, (Boston, USA), August 2002.

[21] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorense, *LAPACK Users' Guide*. SIAM, 3rd ed., 1999.

[22] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *To appear in IEEE JSAC Special Issue on Advances in Fundamentals of Network Management*, Spring 2002.

APPENDIX

A. MATLAB SOURCE CODE

```
% weighted least-squares estimate of the TM
% Input:
%    A   matrix A in constraints x=A*t
%    x   vector x in constraints x=A*t
%    tg  initial gravity model solution
%    w   weight vector
% Output:
%    t   estimated traffic matrix (as a vector)
%        that minimizes |(t-tg)./w|
%        among all t's that minimize |A*t-x|
function [t] = wlse(A,x,tg,w)
   % equivalently transform x=A*t into
   % xw=Aw*tw, where tw=(t-tg)./w
   xw = x - A*tg;
   [r, c] = size(A);
   Aw = A .* repmat(w', r, 1);
   % solve tw=Aw*tw by computing the pseudo-
   % inverse of matrix Aw (through svd)
   tw = pinv(full(Aw)) * xw;
   % transform tw back to t
   t = tg + w .* tw;
```