An Information-Theoretic Approach to Traffic Matrix Estimation

Yin Zhang Matthew Roughan Carsten Lund
AT&T Labs – Research
(yzhang,roughan,lund)@research.att.com

David Donoho
Stanford University
donoho@stanford.edu

ABSTRACT

Traffic matrices are required inputs for many IP network management tasks: for instance, capacity planning, traffic engineering and network reliability analysis. However, it is difficult to measure these matrices directly, and so there has been recent interest in inferring traffic matrices from link measurements and other more easily measured data. Typically, this inference problem is ill-posed, as it involves significantly more unknowns than data. Experience in many scientific and engineering fields has shown that it is essential to approach such ill-posed problems via "regularization". This paper presents a new approach to traffic matrix estimation using a regularization based on "entropy penalization". Our solution chooses the traffic matrix consistent with the measured data that is information-theoretically closest to a model in which source/destination pairs are stochastically independent. We use fast algorithms based on modern convex optimization theory to solve for our traffic matrices. We evaluate the algorithm with real backbone traffic and routing data, and demonstrate that it is fast, accurate, robust, and flexible.

Categories and Subject Descriptors

C.2.3 [**Computer-Communications Networks**]: Network Operations—*network monitoring*; C.2.5 [**Computer-Communications Networks**]: Local and Wide-Area Networks—*Internet*

General Terms

Measurement, Performance

Keywords

Traffic Matrix Estimation, Information Theory, Minimum Mutual Information, Regularization, Traffic Engineering, SNMP.

1. INTRODUCTION

A point-to-point *traffic matrix* gives the volume of traffic between origin/destination pairs in some network. Traffic matrices are required inputs for many IP network management tasks: for instance, capacity planning, traffic engineering and network reliability analysis. However, it is difficult to measure these matrices directly, and so there is interest in inferring traffic matrices from link load statistics and other more easily measured data [24, 23, 3, 16, 28].

Traffic matrices may be estimated or measured at varying levels of detail [15]: between Points-of-Presence (PoPs) [16], routers [28], links, or even IP prefixes [8]. The finer grained traffic matrices are generally more useful, for example, in the analysis of the reliability of a network under a component failure. During a failure, IP traffic is rerouted to find the new path through the network, and one wishes to test if this would cause a link overload anywhere in the network. Failure of a link within a PoP may cause traffic to reroute via alternate links within the PoP without changing the inter-PoP routing. Thus to understand failure loads on the network we must measure traffic at a router-to-router level. In general, the inference problem is more challenging at finer levels of detail, the finest so far considered being router-to-router.

The challenge lies in the ill-posed nature of the problem: for a network with N ingress/egress points we need to estimate the N^2 origin/destination demands. At a PoP level N is in the tens, at a router level N may be in the hundreds, at a link level N may be tens of thousands, and at the prefix level N may be of the order of one hundred thousand. However, the number of pieces of information available, the link measurements, remains approximately constant. One can see the difficulty — for large N the problem becomes massively underconstrained.

There is extensive experience with ill-posed linear inverse problems from fields as diverse as seismology, astronomy, and medical imaging [1, 2, 17, 18, 26], all leading to the conclusion that some sort of side information must be brought in, producing a result which may be good or bad depending on the quality of this information. All of the previous work on IP traffic matrix estimation has incorporated prior information: for instance, Vardi [24] and Tebaldi and West [23] assume a Poisson traffic model, Cao et al. [3] assume a Gaussian traffic model, Zhang et al. [28] assume an underlying gravity model, and Medina et al. [16] assume a logit-choice model. Each method is sensitive to the accuracy of this prior: for instance, [16] showed that the methods in [24, 23, 3] were sensitive to their prior assumptions, while [28] showed that their method's performance was improved if the prior (the so called gravity model) was generalized to more accurately reflect realistic routing rules.

In contrast, this paper starts from a regularization formulation of the problem drawn from the field of ill-posed problems, and derives a prior distribution that is most appropriate to this problem. Our prior assumes source/destination independence, until proven otherwise by measurements. The method then blends measurements with prior information, producing the reconstruction closest to independence, but consistent with the measured data. The method proceeds by solving an optimization problem that is understandable and intuitively appealing. This approach allows a convenient implementation using modern optimization software, with the result that the algorithm is very efficient.

We test the estimation algorithm extensively on network traffic and topology data from an operational backbone ISP. The results show that the algorithm is fast, and accurate for point-to-point traffic matrix estimation. We also test the algorithm on topologies generated through the Rocketfuel project [21, 14, 22] to resemble alternative ISPs, providing useful insight into where the algorithm will

SIGCOMM'03, August 25–29, 2003, Karlsruhe, Germany.

work well. One interesting side result is that there is a relationship between the network traffic and topology that is beneficial in this estimation problem. We also test the sensitivity of the algorithm to measurements errors, demonstrating that the algorithm is highly robust to errors, and missing data in the traffic measurements.

Our approach also allows us to address the problem of estimating point-to-multipoint demand matrices. As shown in [8], point-to-point traffic matrices are not always enough for applications. Under some failures the traffic may actually change its origin and destination; its network entry and exit points. The point-to-point traffic matrix will be altered, because the point-to-point traffic matrix describes the "carried" load on the network between two points. The *demand matrix*, which describes the "offered" load for an IP network, is point-to-multipoint. To understand this, consider a packet entering a backbone ISP through a customer link, destined for another backbone ISP's customer. Large North-American backbone providers typically are connected at multiple peering points. Our packet could reach its final destination through any of these peering links; the actual decision is made through a combination of Border Gateway Protocol (BGP) and Interior Gateway Protocol (IGP) routing protocols. If the normal exit link fails, then the routing protocols would choose a different exit point. In a more complicated scenario, the recipient of the packet might be multi-homed — that is, connected to more than one ISP. In this case the packet may exit the first ISP through multiple sets of peering links. Finally, even single homed customers may sometimes be reached through multiple inter-AS (Autonomous System) paths.

Given the complexity and ill-posed nature of the point-to-multipoint problem, one is tempted to throw his arms in the air and say: "we cannot solve the point-to-multipoint problem with link level data; we need better information (for instance from Netflow [8])." This paper shows, however, that by adopting the regularization approach above it is possible to make some progress towards solving this problem. We cannot estimate demand matrices at the ideal level of detail (prefix level), because the data at our disposal (SNMP link loads) cannot distinguish prefixes. However, the operational realities of large networks make a simplification to router level practical, and useful. Using these simplifications we present a method for estimating the point-to-multipoint demand matrices, though in this paper we only test these implicitly to make the results more directly comparable to previous work.

An advantage of the approach used in this paper is that it also provides some insight into alternative algorithms. For instance, the simple gravity model of [28] is equivalent to complete independence of source and destination, while the generalized gravity model corresponds to independence conditional on source and destination link classes. Furthermore, the algorithm of [28] is a first-order approximation of the algorithm presented here, explaining the success of that algorithm, and suggesting that it also can be extended to measure point-to-multipoint demand matrices. Our method opens up further opportunities for extensions, given the better understanding of the importance of prior information about network traffic and how it can be incorporated into the process of finding traffic matrices. For instance, an appealing alternative prior generation procedure is proposed in [16] (this idea is suggested in [16] but the mechanism to do so is not explored). Alternatively, the Bayesian method of [23] can be placed into the optimization framework here, with a different penalty function, as could the methods of [24, 3].

Finally, we examine some alternative measurement strategies that could benefit our estimates. We examine two possibilities: the first (suggested in [16]) is to make direct measurements of some rows of the traffic matrix, the second is to measure local traffic matrices as suggested in [25]. Both result in improvements in accuracy, however, we found in contrast to [16] that the order in which rows of the traffic matrix are included does matter — adding rows in order of the largest row sum first is better than random ordering.

To summarize, this paper demonstrates a specific tool that works well on large scale point-to-point traffic matrix estimation, and can be extended in a number of ways, for instance to compute point-to-multipoint demand matrices. The results show that it is important to add appropriate prior information. Our prior information is based on independence-until-proven-otherwise, which is plausible, computationally convenient, and results in accurate estimates.

The paper begins in Section 2 with some background: definitions of terminology and descriptions of the types of data available. Section 3 describes the regularization approach used here, and our algorithm, followed by Section 4, the evaluation methodology, and Section 5, which shows the algorithm's performance on a large set of measurements from an operational tier-1 ISP. Section 6 examines the algorithm's robustness to errors in its inputs, and Section 7 shows the flexibility of the algorithm to incorporate additional information. We conclude the paper in Section 8.

2. BACKGROUND

2.1 Network

An IP network is made up of routers and adjacencies between those routers, within a single AS 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 1. We refer to routers and links that are wholly internal to the network as *Backbone* Routers (BRs) and links, and refer to the others as *Edge* Routers (ERs) and links.

One could compute traffic matrices with different levels of aggregation at the source and destination end-points, for instance, at the level of PoP to PoP, or router to router, or link to link [15]. 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 topology information [15]. We may further specify the traffic matrix to be between BRs, by aggregating up to this level.

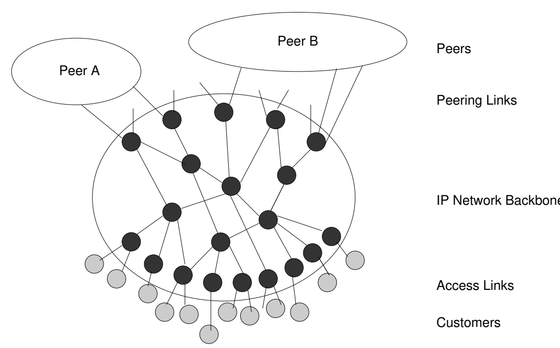


Figure 1: IP network components and terminology

In addition, it is helpful for IP networks managed by Internet Service Providers (ISPs) to further classify the 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. Today traffic to peer networks is largely focused on dedicated peering links, as illustrated in Figure 1. Under the typical routing policies implemented by large ISPs, very little traffic will transit the backbone from one peer 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 operations.

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, or a route monitor such as [19]. In our investigation, we employ a routing simulator such as in [7] that makes use of this routing information to compute a routing matrix. We also simulate load balancing across multiple shortest paths.

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 (a poller).

The properties of data gathered via SNMP are important for the 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 also 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 mitigated by using hourly traffic averages (of five minute data), and we shall use this approach. The problems with the finer time-scale data make time-series approaches to traffic matrix estimation more difficult.

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 port numbers. This level of granularity is sufficient to obtain a real traffic matrix [8], 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 Information Theory

Information theory is of course a standard tool in communications systems [12], but a brief review will set up our terminology. We begin with basic probabilistic notation: we define $p_X(x)$ to mean the probability that a random variable X is equal to x. We shall typically abuse this notation (where it is clear) and simply write $p(x) = p_X(x)$. Suppose for sake of discussion that X and Y are independent random variables, then

$$p(x, y) = p(x)p(y), \tag{1}$$

i.e. the joint distribution is the product of its marginals. This can be equivalently written using the conditional probability

$$p(x|y) = p(x). \tag{2}$$

In this paper we shall typically use, rather than the standard random variables X and Y, S and D, the source S and the destination D of a packet (or bit). Thus $p(d|s)$ is the conditional probability of a packet (bit) exiting the network at $D = d$, given that it entered at $S = s$, and $p(d)$ is the unconditional probability of a packet (bit) going to $D = d$.

We can now define the Discrete Shannon Entropy of a discrete random variable X taking values x_i as

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i), \tag{3}$$

The entropy is a measure of the uncertainty about the outcome of X. For instance, if $X = x_1$ with certainty, then $H(X) = 0$, and $H(X)$ takes its maximum value when X is uniformly distributed — that is, when the uncertainty about its value is greatest.

We can also define the conditional entropy of one random variable Y with respect to another X by

$$H(Y|X) = -\sum_j p(x_i) \sum_i p(y_i|x_i) \log_2 p(y_i|x_i), \tag{4}$$

where $p(y_i|x_i)$ is the probability that $Y = y_i$ conditional on $X = x_i$. $H(Y|X)$ can be thought of as the uncertainty remaining about Y given that we are informed of the outcome of X. Notice that the joint entropy of X and Y can be shown to be

$$H(X, Y) = H(X) + H(Y|X). \tag{5}$$

We can also define the Shannon information

$$I(Y|X) = H(Y) - H(Y|X), \tag{6}$$

which therefore represents the decrease in uncertainty about Y from measurement of X, or the information that we gain about Y from X. The information is symmetric, $I(X|Y) = I(Y|X)$ and so we can refer to this as the *mutual information* of X and Y, and write as $I(X, Y)$. Note that $I(X, Y) \geq 0$, with equality if and only if X and Y are independent — when X and Y are independent X gives us no additional information about Y.

The mutual information can be written in a number of ways, but here we write it

$$I(X, Y) = \sum_{x,y} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} = K(p_{x,y} || p_x \times p_y), \tag{7}$$

where $K(f||g) = \sum_i f_i \log(f_i/g_i)$ is the Kullback-Leibler divergence of f with respect to g, a well-known measure of distance between probability distributions.

Discrete Entropy is frequently used in coding because the entropy $H(X)$ gives a measure of the number of bits required to code the values of X. That is, if we had a large number n of randomly-generated instances $X_1, X_2, \ldots, X_n$ and needed to represent this stream as compactly as possible, we could represent this stream using only $nH(X)$ bits, using entropy coding as practiced for example in various standard commercial compression schemes.

Entropy has also been advocated as a tool in the estimation of probabilities. Simply put, the *maximum entropy principle* states that we should estimate an unknown probability distribution by enumerating all the constraints we know it must obey on 'physical' grounds, and searching for the probability distribution that maximizes the entropy subject to those constraints. It is well known that the probability distributions occurring in many physical situations can be obtained by the maximum entropy principle. Heuristically, if we had no prior information about a random variable X, our uncertainty about X is at its peak, and therefore we should choose a distribution for X which maximizes this uncertainty, or the entropy. In the case where we do have information about the variable, usually in the form of some set of mathematical constraints C, then the principle states that we should maximize the entropy $H(X|C)$ of X conditional on consistency with these constraints. That is, we choose the solution which maintains the most uncertainty while satisfying the constraints. The principle can also be derived directly from some simple axioms which we wish the solution to obey [20].

2.4 Ill-Posed Linear Inverse Problems

Many scientific and engineering problems have to solve inference problems which can be posed as follows. We observe data $\mathbf{y}$ which are thought to follow a system of linear equations

$$\mathbf{y} = A\mathbf{x}, \tag{8}$$

where the n by 1 vector $\mathbf{y}$ contains the data, and the p by 1 vector $\mathbf{x}$ contains unknowns to be estimated. The matrix A is an n by p matrix. In many cases of interest $p > n$, and so there is no unique solution to the equations. Such problems are called *ill-posed linear inverse problems*. In addition, frequently the data are noisy, so that it is more accurate to write

$$\mathbf{y} = A\mathbf{x} + \mathbf{z}. \tag{9}$$

In that case any reconstruction procedure needs to remain stable under perturbations of the observations. In our case, $\mathbf{y}$ are the SNMP link measurements, $\mathbf{x}$ is the traffic matrix written as a vector, and A is the routing matrix.

There is extensive experience with ill-posed linear inverse problems from fields as diverse as seismology, astronomy, and medical imaging [1, 2, 17, 18, 26], all leading to the conclusion that some sort of side information must be brought in, producing a reconstruction which may be good or bad depending on the quality of the prior information. Many such proposals solve the minimization problem

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 J(\mathbf{x}), \tag{10}$$

where where $\|\cdot\|_2$ denotes the L_2 norm, $\lambda > 0$ is a regularization parameter, and $J(\mathbf{x})$ is a penalization functional. Proposals of this kind have been used in a wide range of fields, with considerable practical and theoretical success when the data matched the assumptions leading to the method, and the regularization functional matched the properties of the estimand. These are generally called *strategies for regularization of ill-posed problems* (for a more general description of regularization see [11]).

A general approach to deriving such regularization ideas is the Bayesian approach (such as used in [23]), where we model the estimand $\mathbf{x}$ as being drawn at random from a so-called 'prior' probability distribution with density $\pi(\mathbf{x})$ and the noise $\mathbf{z}$ is taken as a Gaussian white noise with variance σ^2. Then the so-called posterior probability density $p(\mathbf{x}|\mathbf{y})$ has its maximum $\hat{\mathbf{x}}$ at the solution of

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + 2 \cdot \sigma^2 \log \pi(\mathbf{x}). \tag{11}$$

Comparing this with (10) we see that penalized least-squares problems as giving the most likely reconstructions under a given model. Thus the method of regularization has a Bayesian interpretation, assuming Gaussian noise and assuming $J(\mathbf{x}) = \log \pi(\mathbf{x})$. We stress that there should be a good match between the regularization functional J and the properties of the estimand — that is, a good choice of prior distribution. The penalization in (10) may be thought of as expressing the fact that reconstructions are very implausible if they have large values of $J(\cdot)$.

Regularization can help us understand approaches such as that of Vardi [24] and Cao et al. [3], which treat this as a maximum likelihood problem where the $\mathbf{x}$ are independent random variables following a particular model. In these cases they use the model to form a penalty function which measures the distance from the model by considering higher order moments of the distributions.

2.5 Shrinkage Estimation

An alternative reasoning behind regularization is that in estimating large numbers of parameters (as in the problem above), 'shrinking' an otherwise valid estimates towards a special point results in substantial reductions in mean-squared error. As a simple example, suppose we have noisy data $\mathbf{y} = \mathbf{x}+\mathbf{z}$, where $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{z}$ are all $n \times 1$ vectors. We wish to recover the vector $\mathbf{x}$, where $\mathbf{z}$ represents Gaussian white noise $N(0, 1)$. The raw data components y_i are unbiased minimum variance estimators of the corresponding components x_i of the estimand $\mathbf{x}$, so it is tempting to believe that $\mathbf{y}$ is the optimal estimate of $\mathbf{x}$. In fact, if n is large, it is possible to do substantially better than using $\mathbf{y}$. We should instead solve the penalized problem

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \hat{\lambda}^2 \|\mathbf{x}\|_2^2, \tag{12}$$

where $\hat{\lambda} = \frac{n}{\|\mathbf{y}\|_2^2}$ is a measure of the dataset's size in mean-square (or rather its reciprocal). The solution is a compromise between fidelity to the measured data $\mathbf{y}$ and closeness to the origin, and has the simple form $\hat{\mathbf{x}}^* = \frac{1}{1+\hat{\lambda}}\mathbf{y}$. This reconstruction is obtained simply by 'shrinking' the raw data $\mathbf{y}$ towards zero. It turns out that for large n this shrunken estimator is always better than the 'obvious' unbiased estimate $\mathbf{y}$, in the sense that it always has a lower mean-squared error. This qualitative conclusion remains true if we shrink towards some other fixed point, though it is better to shrink towards a point close to the $\mathbf{x}$ we are trying to estimate. For a fuller discussion of shrinkage estimation, see for example [13, 6]. For now, simply note that shrinkage of a very high-dimensional estimand towards a chosen point can be helpful. Note that no Bayesian assumption is being made here: whatever the underlying estimand may be, shrinkage is an improvement, regardless of our prior beliefs about which vectors $\mathbf{x}$ are plausible. The key assumption is that we are trying to estimate a vector with many components, all affected by noise.

3. REGULARIZATION OF THE TRAFFIC ESTIMATION PROBLEM USING MINIMUM MUTUAL INFORMATION

The problem of inference of the end-to-end traffic matrix is massively ill-posed because there are so many more routes than links in a network. In this section, we develop a regularization approach using a penalty that seems well-adapted to the structure of actual traffic matrices, and which has some appealing information-theoretic structure. Effectively, among all traffic matrices agreeing with the link measurements, we choose the one that minimizes the mutual information between the source and destination random variables.

Under this criterion, absent any information to the contrary, we assume that the conditional probability $p(d|s)$ that a source s sends traffic to a destination d is the same as $p(d)$, the probability that the network as a whole sends packets or bytes to destination d. There are strong heuristic reasons why the largest-volume links in the network should obey this principle — they are so highly aggregated that they intuitively should behave similarly to the network as a whole.

On the other hand, as evidence accumulates in the link-level statistics, the conditional probabilities are adapted to be consistent with the link-level statistics in such a way as to minimize the mutual information between the source and destination random variables.

This Minimum Mutual Information (MMI) criterion is well-suited to efficient computation. It can be implemented as a convex optimization problem; in effect one simply adds a minimum weighted entropy term to the usual least-squares lack of fit criterion. There are several widely-available software packages for solving this optimization problem, even on very large scale problems; some of these packages can take advantages of the sparsity of routing matrices.

3.1 Traffic-Matrix Estimation

Let $N(s, d)$ denote the traffic volume going from source s to destination d in a unit time. Note that $N(s, d)$ is unknown to us; what can be known is the traffic $T(l)$ on link l. Let $A(s, d; l)$ denote the routing matrix, i.e. $A(s, d; l)$ gives the fraction of traffic from s to d which crosses link l (and which is zero if the traffic on this route does not use this link at all). The link-level traffic counts are

$$T(l) = \sum_{s,d} A(s, d; l)N(s, d), \quad \forall l \in L, \tag{13}$$

where L is the set of backbone links. We would like to recover the traffic matrix $N(s, d)$ from the link measurements $T(l)$, but this is the same as solving the matrix equation (8), where $\mathbf{y}$ is a vector containing the traffic counts $T(l)$, $\mathbf{x}$ is a vectorization of the traffic matrix, and A is the routing matrix. A is a matrix which is $\#L$ by $(\#S \times \#D)$, where there are $\#L$ link measurements, $\#S$ sources, and $\#D$ destinations.

3.2 The Independence Model

We propose thinking about $N(s, d)$ in probabilistic terms, so that if a network carries N end-to-end packets (or bits) total within a unit time then the number of packets sent from source s to destination d, $N(s, d)$ say, is a random variable with mean $N \cdot p(s, d)$, with $p(s, d)$ the joint probability that a randomly chosen one of the N packets (or bits) goes from s to d. We consider the marginal probabilities

$$p_S(s) = \sum_d p(s,d), \quad (14)$$

$$p_D(d) = \sum_s p(s,d), \quad (15)$$

the chance that a randomly-chosen packet (bit) enters the network at s, and the chance that a randomly chosen packet (bit) departs at d, respectively. We can expand this notation to measure sets:

$$p_{S,D}(Q_s, Q_d) = \sum_{s \in Q_s} \sum_{d \in Q_d} p(s,d), \quad (16)$$

for all sets of source and destination links Q_s, Q_d, and similarly for the marginal probabilities p_s and p_d.

We let S be the random variable obtained looking at the source of a random packet (or bit), and let D denote the destination. Suppose for sake of discussion that S and D are independent random variables. Then (2) means that, given that a packet (bit) originates at $S = s$, it is no more likely to go to $D = d$ than would a randomly-chosen packet (bit) originating anywhere in the network. For networks containing a few extremely high volume links carrying very large fractions of the packets, the assumption (2) should work well for the very largest circuits, since they have been so highly aggregated that their behavior may be very similar to the network as a whole.

Note that the independence of source and destination is equivalent to the simple *gravity model* which has been discussed in the Internet measurement community; the model has the form

$$N(s,d) \approx \text{Const } N(s)N(d) \quad (17)$$

where $N(s)$ is the traffic entering at s, and $N(d)$ is the traffic exiting at d. While there is experience with the gravity model above and some success in its application, it is also known that it gives results that are not as accurate as may be obtained using additional information [16, 28].

Section 2 suggests that regularization is a way of using prior information in conjunction with link measurements to help decide which traffic matrices from the set satisfying (8) are more plausible. We propose using a regularization functional that uses the independence/gravity model as a point of departure, but which considers other models as well. Recall from our discussion of information theory that independence of source and destination is tantamount to the statement that the mutual information vanishes: $I(S,D) = 0$. Recall also that $I(S,D) \geq 0$. It follows that the penalty functional on traffic matrices $p(s,d)$, given by

$$J(p) \equiv I(S,D),$$

has $J(T) \geq 0$ with equality if and only if S and D are independent.

This functional has an interpretation in terms of the compressibility of addresses in IP headers. Suppose we have a large number of IP headers — abstracted to be simply source/destination address pairs (s_i, d_i), $i = 1, \ldots, N$. We want to know: what is the minimal number of bits required (per header) to represent the source destination pair. It turns out that this is just $H(S) + H(D) - I(S,D)$. Now if we simply applied entropy compression to the S_i and D_i streams separately, we would pay $H(S) + H(D)$ bits per header to represent headers. Hence the functional $I(S,D)$ measures the number of bits of additional compression possible beyond the separate compression of source and destination based on traditional entropy compression. This extra compression is possible because of special dependencies that make IP messages more likely to go in certain source/destination pairs than we would have expected by independence. In fact measurements of $H(S)$ and $H(D)$ (on real datasets described below) are typically around 5, while $I(S,D)$ is very small, typically around 0.1. This suggests that the independence assumption is a reasonable fit to the real data, at least on average. There may be some links for which it is not, but the MMI method specifically allows for correction to these (see below).

Suppose we adopt a Bayesian viewpoint, assigning an *a priori* probability $\pi(p)$ to the traffic matrix p that is proportional to $2^{-J(p)}$. Then we are saying we regard as *a priori* implausible those traffic matrices where much higher compression is possible based on joint source-destination pairs as compared to compression of sources and destinations separately. Each bit saved reduces our *a priori* likelihood by about a factor $1/2$.

3.3 Regularization Method

We propose now to reconstruct traffic matrices by adopting the regularization prescription (10) with the regularization functional $J(p) = I(S,D)$. Translating (10) into traffic-matrix notation, we seek to solve

$$\text{minimize} \sum_l \left(T(l) - N \sum_{s,d} A(s,d;l) p(s,d) \right)^2 + \lambda^2 I(S,D), \quad (18)$$

Recalling the Bayesian interpretation of regularization, we are saying that we want a traffic matrix which is a tradeoff between matching the observed link traffic counts and having *a priori* plausibility, where our measure of plausibility, as just explained, involves the 'anomalous compressibility' of source-destination pairs. The traffic matrix obtained as the solution to this optimization will be a compromise between two terms based on the size of λ, which is a proxy for the noise level in our measurements. Note that

$$I(S,D) = \sum_{d,s} p(s,d) \log \frac{p(s,d)}{p(s)p(d)} = K(p(s,d)||p(s)p(d)), \quad (19)$$

where $K(\cdot||\cdot)$ again denotes the Kullback-Leibler divergence. Here $p(s)p(d)$ represents the gravity model, and $K(\cdot||\cdot)$ can be see as a distance between probability distributions, so that we can see (18) as having an explicit tradeoff between fidelity to the data and deviation from the independence/gravity model. Note also that the Kullback-Leibler divergence is the negative of the relative entropy of $p(s,d)$ with respect to $p(s)p(d)$, and so this method also has an interpretation as a maximum entropy algorithm.

Both terms in the above tradeoff are convex functionals of the traffic matrix p. Hence, for each given λ, they can be rewritten in constrained optimization form:

$$\begin{array}{c} \text{minimize } K(p(s,d)||p(s)p(d)) \text{ subject to} \\ \sum_l (T(l) - N \sum_{s,d} A(s,d;l) p(s,d))^2 \leq \chi^2. \end{array} \quad (20)$$

Here $\chi^2 = \chi^2(\lambda)$ is chosen appropriately so that the solution of this problem and the previous one are the same, at the given value of λ. The problem is saying: among all traffic matrices adequately accounting for the observed link counts, find the one closest to the gravity model. It can also be viewed as saying: shrink away from the observed link counts towards the gravity model.

Thinking heuristically, we are trying to estimate a very large number of unknowns, so shrinkage towards the gravity model can be expected to be error-reducing, providing it is performed appropriately (as here). Based on the experience of statisticians with shrinkage estimation, it seems that we can expect this procedure to provide at least some improvement in mean-squared error even though the gravity model assumption may not be valid.

If the noise level in the data is small, of course, then the solution will not be allowed to be very close to the gravity model. In the limit, as the noise level goes to zero, we obtain the solution by minimizing $K(p(s,d)||p(s)p(d))$ subject to the constraints (13). In effect we are looking for the most nearly independent version of $p(s,d)$ subject to generating the observed traffic statistics.

Note that in all these optimization problems, there are additional constraints (on any probability distribution) such as non-negativity, normalization, and (14) and (15). We leave all these implicit.

3.4 Algorithm

The problem we attack in this paper is the BR-to-BR traffic matrix. While this problem is an order of magnitude more complex than a PoP-to-PoP traffic matrix, a router-to-router traffic matrix is absolutely necessary for many network engineering tasks. A PoP-to-PoP traffic matrix is useful when designing a network from scratch, but typically, in a real network changes are incremental, and so we need to see how these changes affect traffic at the router level. We use techniques from [28] to reduce the size of the problem initially, by removing redundant information, and a large number of traffic matrix elements that we know to be zero from routing information. This processing does not improve accuracy, but does speed up later computations.

To make the exact formulation explicit, we define

$$\begin{aligned} x_i &= N(s_i, d_i), & (21)\\ y_j &= \text{traffic counts } = T(l_j), & (22)\\ g_i &= N(s_i)N(d_i), & (23) \end{aligned}$$

where

$$\begin{aligned} N &= \text{total traffic in network} & (24)\\ N(s_i) &= \text{total traffic originating at } s_i & (25)\\ N(d_i) &= \text{total traffic departing at } d_i & (26) \end{aligned}$$

and we define the column vectors $\mathbf{x}$, and $\mathbf{y}$ with elements x_i and y_i, respectively. Our formulation is

$$\min_x \left\{ ||\mathbf{y} - A\mathbf{x}||^2 + \lambda^2 \sum_{i:\ g_i>0} \frac{x_i}{N} \log\left(\frac{x_i}{g_i}\right) \right\} \qquad (27)$$

subject to $x_i \geq 0$.

Note that $g_i = 0$ if and only if the traffic at the source or destination is zero, and so $x_i = 0$. The additional constraints on the marginal distributions are satisfied by supplementing the routing matrix, and measurements to ensure that they include these constraints.

This penalized least-squares formulation has been used in solving many other ill-posed problems, and so there exist publicly available software in Matlab (such as routine MaxEnt in Per Christian Hansen's Inverse Problems Toolbox [9, 10]) to solve small-scale variants of such problems. Our problems are, however, large in scale and not suited to such basic implementations. The problem of solving such large-scale traffic matrices is only possible if we can exploit one of the main properties of routing matrices: they are very sparse — the proportion of exact zero entries in each column and row is overwhelming. Accordingly, we use PDSCO [5], a MATLAB package developed by Michael Saunders of Stanford University, which has been highly optimized to solve problems with sparse matrices A. PDSCO has been used (see e.g. [5]) to solve problems of the order 16,000 by 256,000 efficiently. We have found that its performance is very good (taking no more than a few seconds) even on the largest problems we consider here.

In principle, the choice of λ depends on the noise level in the measurements, but in our results below we show that the results are insensitive to this parameter, and so its exact choice isn't important.

An interesting point is that if one were to have additional information such as used in the choice model of [16] then this could also be incorporated by conditioning the initial model $P_{S,D}(s,d)$ on this information (for an example of this type see Section 3.5). This would amount to a kind of shrinkage, this time not towards the gravity model, but instead towards a model incorporating more side information. Alternatively, such information could be included in the constraints underlying the optimization (as shown in Section 7).

3.5 Inter-domain Routing

3.5.1 *Zero Transit Traffic*

The above algorithm assumes that independence of source and destination is a reasonable starting model. However, there are good reasons we may want to modify this starting model. In real backbone ISPs, routing is typically asymmetric due to hot-potato routing — traffic from the customer edge to peers will be sent to the "nearest" exit point, while traffic in peer networks will do likewise resulting in a different pattern for traffic from peering to customers. Also there should be no traffic transiting the network from peer to peer [28]. Both factors demand departures from pure independence.

Suppose we assume there is zero transit traffic. We suggest that *conditional independence* of source and destination, *given appropriate side information*, will be more accurate than pure independence. More specifically, suppose we have available as side information, the source and destination class (access or peering). We would then model the probabilities of a packet (bit) arriving at s and departing at d as conditionally independent *given the class of arrival and destination link*. In Appendix A we show that this results in the following model, assuming A and P are the sets of access and peering links, respectively.

$$p_{S,D}(s,d) = \begin{cases} \frac{p_S(s)}{p_S(A)} \frac{p_D(d)}{p_D(A)}(1 - p_S(P) - p_D(P)), & \\ \qquad\qquad\qquad \text{for } s \in A, d \in A, & \\ p_S(s)\frac{p_D(d)}{p_D(A)}, \quad \text{for } s \in P, d \in A, & \\ \frac{p_S(s)}{p_S(A)} p_D(d), \quad \text{for } s \in A, d \in P, & \\ 0, \qquad\qquad \text{for } s \in P, d \in P. & \end{cases} \qquad (28)$$

to which we can naturally adapt the algorithm above (by modifying g_i). We note that the algorithm is then 'shrinking' the observed data in the direction, not of a pure gravity model, but a realistic modification of it.

3.5.2 *Point to Multipoint*

As noted in the introduction a point-to-point traffic matrix is not suitable for all applications. Sometimes we need a point-to-multipoint demand matrix, for instance, when we want to answer questions about the impact of link failures outside the backbone, e.g. "would a peering link failure cause an overload on any backbone links?" In this case, traffic would reroute to an alternate exit point, changing the point-to-point traffic matrix in an unknown way. However, the point-to-multipoint demand matrix would remain constant.

Ideally such a matrix would be at the prefix level, but a number of operational realities make an approximation to router level useful for many engineering tasks. The first such reality is that backbone networks that exchange large traffic volumes are connected by private peering links as opposed to Internet Exchange Points. This allows us to see the proportion of traffic going to each individual peer using only SNMP link measurements, so we can partition traffic per peer. The second such reality is that the BGP policies across a set of peering links to a single peer are typically the same. Therefore, the decision as to which peering link to use as the exit point is made on the basis of shortest IGP distance. This distance is computed at the link level, as opposed to BGP policies, which can act at the prefix level. While we cannot test that this property is true for all large ISPs (and in general it is not always true even on the network from which we have measurements), the methodology above does not need this, because the algorithm above only uses this as a prior, to be corrected through the use of link (and other) information.

The step required to generate a point-to-multipoint demand matrix requires consideration of the control ISPs have over interdomain routing. Interdomain routing gives an ISP little control over where traffic enters their network, so we shall not make any changes to (28) for access-to-access, and peering-to-access traffic. However, a provider has considerable control over where traffic will leave their network across the peering edge. Traffic destined for a particular peer may be sent on any of the links to that peer.

The result is that we must modify (28) for access-to-peer traffic. We do so by not specifying which link d in the set of links to peer i (i.e. P_i) is used for traffic leaving the network to peer i. We can do this formally by not specifying $p_{S,D}(s,d)$ for $s \in A, d \in P$ but rather $p_{S,D}(s,P_i)$ for all peers i. This simple point-to-multipoint model can then be used in the estimation through using

$$p_{S,D}(s,P_i) = \frac{p_S(s)}{p_S(A)} p_D(P_i), \tag{29}$$

for $s \in A$, in place of the access-to-peering equation from (28). We do not determine the exit point in the estimates. The algorithm can then proceed by minimizing the mutual information of the final distribution with respect to (28) and (29). The exit points are implicit in the routing matrix used in the optimization (27), but are left undetermined in the estimate, and can therefore be fixed only when applied to a particular problem.

We should also note that this is a quite general extension. We use it here on sets of peering links P_i, but in a network with different policies, we can partition the peering links in some different fashion (even through a non-disjoint partition) to reflect some particular idiosyncrasies in routing policy.

3.6 Relationship to Previous Algorithms

The work in this paper presents a general framework, within which we can place a number of alternative methods for estimating IP traffic matrices. For instance, by taking a linear approximation to the log function in the Kullback-Leibler information distance information and exploiting the fact that $\sum_x [f(x) - g(x)] = 0$ we get

$$\begin{aligned} K(f||g) &\approx \sum_x f(x) \left[\frac{f(x) - g(x)}{g(x)} \right] - \sum_x [f(x) - g(x)] \\ &= \sum_x \left[\frac{f(x) - g(x)}{\sqrt{g(x)}} \right]^2 . \end{aligned} \tag{30}$$

From this we can see that the MMI solution may be approximated by using a quadratic distance metric (with square root weights) as was applied in [28]. This explains the success of that approach, as well as the need to use square root weights for best performance. The conditional independence of Section 3.5 explains the use of the generalized gravity model as an initial condition in [28].

The quadratic optimization is convenient, because it can be simply solved using the Singular Value Decomposition (SVD) [28], with non-negativity enforced by a second step using Iterative Proportional Fitting (IPF) [3]. In this paper we will compare the performance of the pure MMI approach, its quadratic approximation, and the previous method (referred to here as SVD-IPF), and we see that the approximation works well in the cases considered. We defer the comparison with maximum likelihood approaches ([24, 3, 16]) to future work, because scaling these methods to the size of problem described here requires additional techniques (for instance see [4, 27]) that have only recently been developed.

The point of interest here is that the MMI principle above produces (an approximation of) the algorithm previously derived from an initial gravity model solution. However in the case of the MMI solution, the principle precedes practice — that is, the decision to regularize with respect to a prior is not an arbitrary decision, but a standard step in ill-posed estimation problems. The close approximation has a practical impact in that we can use the fact that [28] already demonstrated that the conditional independence of Section 3.5 to be a better prior than complete independence. We use this fact here by using (28) and (29) in the remainder of the paper.

4. EVALUATION METHODOLOGY

In this paper, we apply the traffic matrix benchmarking methodology developed in [28] to real Internet data to validate different algorithms. One major advantage of the methodology in [28] is that it can provide a *consistent* data set that is as realistic as practically possible. Below we provide an overview of this methodology, followed by a summary of the performance metrics we use.

4.1 Validation Methodology

The approach of [28] used sampled flow level data, and topology and routing information as derived from [7]. Flow level data contains details of numbers of packets and bytes transferred between source and destination IP addresses, and also gives information such as the interface at which the traffic entered our network. Combining these datasets one may derive a traffic matrix [8].

The resulting traffic matrix in our experiments covers around 80% of the real network traffic (including all the peering traffic) on the real topology of a large operational tier-1 ISP. Following [28], we compute the traffic matrices on one hour time scales to deal with some limitations of the measurements. Given these traffic matrices and the network topology and routing information, we only need a consistent set of link load measurements to proceed.

[28] solves the problem of providing a consistent set of traffic, topology and link measurement data as follows. Simulate the network routing using the available topology and routing information. From this we may compute a routing matrix $\mathbf{A}$, and then derive a set of link measurements $\mathbf{y}$ from (8). Thus the traffic matrix $\mathbf{x}$, the routing matrix $\mathbf{A}$ and the measured link loads $\mathbf{y}$ are all consistent. We can then perform the estimation procedure to compute $\hat{\mathbf{x}}$, the traffic matrix estimate.

Part of the goal of this paper is to extend understanding of previous methods, and so we apply the pre-existing methodology for testing traffic matrices. However, this method does not explicitly validate point-to-multipoint traffic matrices. We compute the point-to-multipoint traffic matrix, and then collapse this down to a point-to-point traffic matrix for comparison with the real traffic matrix. The result is an implicit validation of the multipoint estimates.

The validation approach allows us to work with a problem for which we know the "ground truth" — the real traffic matrix. It can also be extended in several different ways. For example, it allows one to take a traffic matrix and apply it on an arbitrary topology, for instance a simulated network such as a star, or a measured topology such as those produced by Rocketfuel [21, 14]. Thus we can gain insight into the effect of different topologies on the performance of the algorithm. We may also introduce controlled measurement errors to assess the algorithm's robustness, or simulate alternative measurements to see their impact in a rigorous manner.

4.2 Performance Metrics

In this paper we use two basic methods for assessing and comparing the results. The first method is to estimate the relative error (that is, the average of the absolute value of the errors, relative to the average traffic matrix element). The second method is to plot the Cumulative Distribution Function (CDF) of the errors relative to the average traffic matrix element. However, many elements of a router to router traffic matrix are zero due to routing constraints, and these constrained elements are easy to estimate. This results in a large number of entries to the traffic matrix with near zero error. To more accurately indicate the errors on the positive elements we separate the zero and non-zero elements and compute their errors separately. The errors on the zero elements are very small (99% of the errors are below 1%), and so we shall not display these separately here. We shall report the relative errors of the positive elements.

5. PERFORMANCE

In this section, we first examine the algorithm's sensitivity to the choice of λ, and then compare the accuracy of different algorithms.

5.1 Sensitivity to the Choice of λ

The choice of the parameter λ determines how much weight is given to independence, versus the routing constraint equations. In

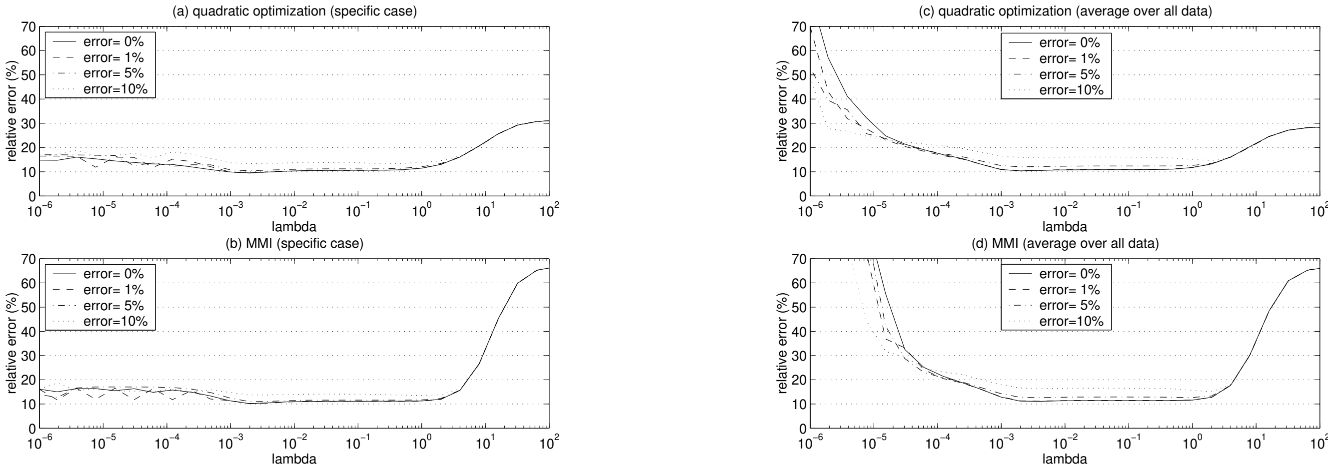


Figure 2: The relative errors for the quadratic and MMI algorithms for a given value of λ.

our experiments, we find that the algorithm's performance is not sensitive to the choice of λ. Figure 2 shows the relative error in the estimates for varying λ. Figure 2 (a) and (b) show the results for the quadratic and MMI algorithms respectively, for a single-hour data set given different levels of error in the input measurements (see below for details of the introduced measurement errors). Figure 2 (c) and (d) show the average results over a month of data.

Most notably, in each graph there is a distinct region where the curves are all quite flat, and that this region is largely the same regardless of the error level. Thus the choice of λ is insensitive to the level of noise in the measurements, and it is easy to choose a good value. We choose a value from the middle of the insensitive range, $\lambda = 0.01$ throughout the rest of the paper, as this performed well, not just in the average (which one can see from Figure 2 (c) and (d)), but also in the worst case. The impact of choosing a single value of λ, rather than the optimal value for each case is shown in Table 1. The table shows for varying levels of error (or noise) in the input measurements the reduction in accuracy due to the use of a fixed λ rather than the optimal value. The table presents two measures: the maximum and average accuracy reduction over all of the data sets.

Note that in the worst case the MMI algorithm is only a few percent worse for not using the optimal value of λ and typically is very close to optimal. The quadratic algorithm is marginally more sensitive to the correct choice of λ.

			accuracy reduction	
algorithm	noise	λ	maximum	average
MMI	0%	0.01	1.6%	0.3%
MMI	1%	0.01	1.6%	0.3%
MMI	5%	0.01	1.4%	0.3%
MMI	10%	0.01	2.9%	1.5%
quadratic	0%	0.01	1.9%	0.4%
quadratic	1%	0.01	1.7%	0.4%
quadratic	5%	0.01	1.9%	0.3%
quadratic	10%	0.01	3.7%	1.7%

Table 1: Impact of choosing a fixed value of λ rather than the optimal value. The table shows for the two algorithms, and various levels of noise in the measurements, the impact of choosing a fixed value of λ compared to the optimal value. The table shows the worst case and the average reduction in accuracy.

5.2 Comparison of Algorithms

We now apply the three algorithms described above (MMI, quadratic optimization, and SVD-IPF) to the problem of computing a BR-to-BR traffic matrix, in order to compare their performance. The results below are based on 506 data sets from the ISP in question, representing the majority of June 2002, and covering all days of the week, and times of day. Figure 3 shows the CDF of the relative errors for the three methods. We can see that their performance is almost identical. The mean relative error is 11.3%. Furthermore, note that more than 80% of the traffic matrix elements have errors less than 20%. The CDFs for individual data sets are very similar, but generally less smooth. All three algorithms are remarkably fast, delivering the traffic matrix in under six seconds. The fastest algorithm is SVD-IPF, which is about twice as fast as MMI, the slowest one. We also compare the three algorithms for robustness. The results are very similar, and are omitted here in the interest of brevity.

Note also that [28] showed a number of additional performance metrics for the SVD-IPF algorithm (which we can see has very similar performance to the MMI and quadratic algorithms). Those results indicated that not only are the errors on the flows reasonable, but also that the errors on the largest flows are small, and that the errors are stable over time (an important feature if the results are to be used to detect network events).

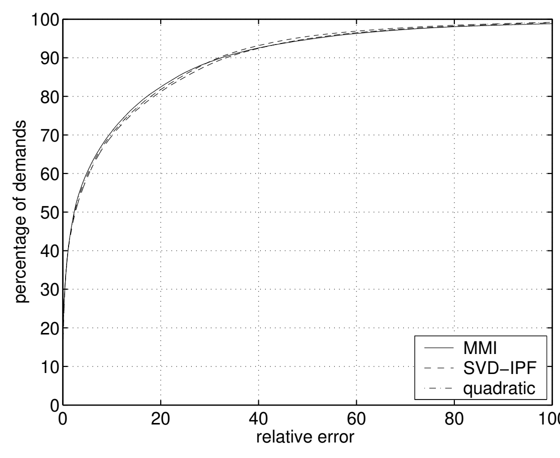


Figure 3: A comparison of the relative errors for the methods.

5.3 Topological Impact

In this section, we investigate the impact of different topologies on the performance of the algorithm. We use the ISP maps collected by Rocketfuel [21, 14, 22]. Since we also need IGP weights, we use the maps for three North American networks (Sprint, Abovenet, and Exodus), for which the IGP weights have been estimated by Rocketfuel. Note that these are not real weights from the networks of interest, but a set consistent with observed routing.

The Rocketfuel data do not contain the peering relationships of a network, and so we are limited to using the same initial conditional independence assumptions in our exploration of topology. This is not a problem here because we are primarily concerned with the impact of the internal network topology on the estimates.

The approach for testing the impact of topology is as follows. We map locations (origins and destination in the original network) to locations (in the Rocketfuel network) at the PoP level, and map (28) and (29) to this new network, assuming the same peering relationships, thus removing dependence on data we don't have access to.

More specifically, let $\mathcal{M} : A \to B$ denote a mapping from the original set of locations $i \in A$ to a set of Rocketfuel locations $j \in B$. Then the mapping of demands from one network to another is accomplished by

$$x_j^B = \sum_{i:\mathcal{M}(i)=j} x_i^A, \quad \forall j \in B, \tag{31}$$

and we map the g_i from (23) similarly. We consider two mappings, the first based on geographical location, which is provided in the Rocketfuel dataset. Geographical information does not provide any way of mapping from router to router in the new network, so we perform our mapping at the PoP level, and therefore also perform the estimation at this level). The second mapping is a random permutation that destroys the dependency between the traffic and the network topology.

5.3.1 Results based on geographical mapping

Figure 4 (a) shows the results of applying the MMI algorithm to the three Rocketfuel networks, where the mapping from location to location is done on the basis of nearest geographical equivalent[1]. That is, our mapping is given by

$$\mathcal{M}(i) = j, \text{ where } d(i,j) \le d(i,k)\ \forall k \in B,$$

where $d(i,j)$ is the geographic distance between PoPs i and j. The figure also shows the PoP level results for the original ISP (the results above were for BR-to-BR traffic matrices). One can see varying levels of performance for the different topologies, but it is generally similar to or better than the performance we see in Figure 3[2].

Our aim here is to understand what features of the topology have impact on the estimation algorithm, and to this end we can consider two illustrative examples: simple 20 node star and clique topologies. In the star, all PoPs are connected by a single hub, and in the clique, all PoPs have direct connections to each other. We intentionally make these control cases very simple so that we know exactly what is going on. The results are shown in Figure 4 (b). The performance on the star topology is poor, while on the clique the performance is almost perfect. The results stem from the fact that in the clique topology the link data gives us the traffic matrix. In this case, the initial MMI estimate of the traffic matrix is almost completely overridden by the information from link data. In the case of the star, there is no additional information contributed by the link data, and so we see how well the independence assumption performs on the input traffic matrix.

Table 2 provides a comparison between the different networks. The table shows, for each network, the number of North American PoPs (excluding the degree one nodes), and the number of inter-PoP logical links (note that multiple physical links are mapped to a single logical link here because these represent redundant information). The table also shows the resulting number of unknowns (traffic matrix elements to be estimated) relative to the number of measurements (or links), and average estimation errors. Clearly we can see a direct relationship between the ratio of unknowns to measurements, and the performance of the algorithm.

This illustrates the basis for the MMI method. It will work best where either the conditionally independent estimate is good to start with, or the topology has sufficiently diverse links to allow for the results to be accurately refined. The networks measured by Rocketfuel appear to have such diversity.

Network	PoPs	links	unknowns per measurement	error (%) geo.	error (%) rand.
Exodus	17	58	4.69	12.58	20.07
Sprint	19	100	3.42	8.06	18.93
Abovenet	11	48	2.29	3.76	11.74
Star	N	$2(N-1)$	$N/2 = 10$	24.02	24.02
Clique	N	$N(N-1)$	1	0.18	0.18
ISP	-	-	3.54–3.97	10.55	-

Table 2: The table shows, for the three Rocketfuel PoP level topologies: the number of PoPs (excluding degree one PoPs), inter-PoP links (parallel links aggregated), and the number of unknowns per link measurement. The table also shows the values for Star and Clique topologies with N nodes ($N = 20$ in the examples), and for the original ISP. The final two columns of the table give the performance (relative mean error) of the MMI algorithm on each topology for the geographic and random mappings. Note that the results for the ISP are at PoP level, obtained by aggregation from BR-BR traffic matrices, so the random mapping is not available.

5.3.2 Results based on random mapping

However, there is more to the problem than this. In fact it appears that there is a relationship between the network traffic, and the network topology that benefits the performance of the algorithm. Figure 4 (b) also shows the result of mapping the locations in the original ISP to the Rocketfuel ISPs using a random permutation (the figure is based on 100 random permutations of 24 data sets drawn from one day in June). The performance under a random mapping is worse than under a geographical mapping. The last column of Table 2 confirms this finding.

This is interesting because, typically in large networks, regions of the network with higher demand tend to have more connections to the other PoPs (in the measured network the correlation coefficient between node degree and traffic volume was 0.7). A higher degree at a node results in more information about the corresponding row of the traffic matrix, and thence a better estimate of this row. Good estimates of the larger elements make it easier to estimate other elements elsewhere in the network, and so we get a better overall result. This naturally leads to better estimates when the traffic is correlated to the network degree, but when we perform the random mapping, the correlation no longer holds. We shall see later that this property has an impact on the design of network measurement infrastructure to further improve traffic matrix estimates: it is better to put measurement infrastructure in the nodes with the largest traffic volume.

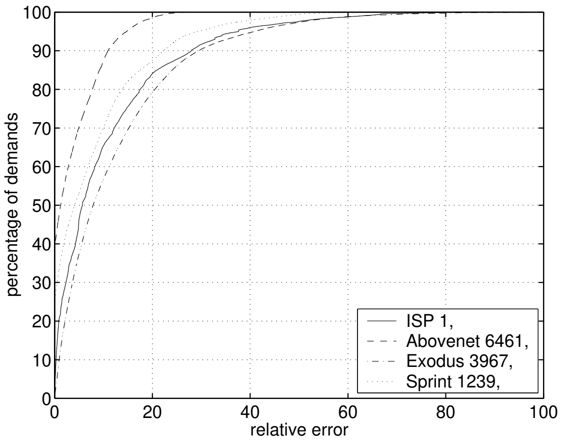
Also interesting is the fact that this finding adds credibility to the choice model idea presented in [16]. The choice model asserts that features of the network (such as the number of links) are correlated with the attractiveness of that node as a destination, and we can confirm that finding here, at least with respect to the number of links.

6. ROBUSTNESS

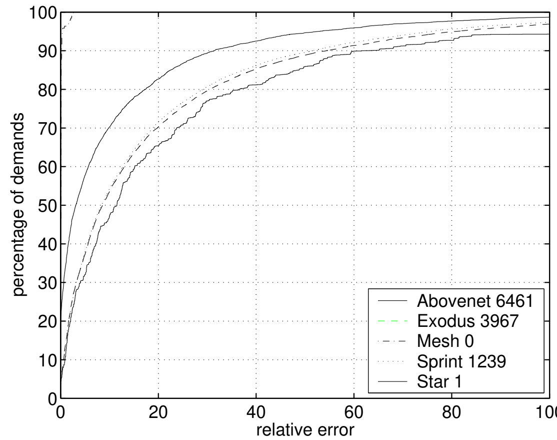
A critical requirement for any algorithm that will be applied to real network data is robustness. In general this refers to the sensitivity of an algorithm to violations of the algorithm's assumptions (implicit and explicit). In the MMI method, the only assumptions are that the MMI criteria is a reasonable approach (verified above)

[1] When performing the PoP level mapping we exclude nodes of degree one as these are often minor regional nodes.

[2] The unknowns in the Rocketfuel data, and the lack of traffic data from the other networks mean that the convenient labels Sprint, Exodus, or Abovenet should not be interpreted as saying that we have tested the algorithm on those networks directly.



(a) geographical mapping



(b) random mapping

Figure 4: Results on Rocketfuel, and simulated topologies.

and that the input data are correct. Network data are often error prone, and there can be missing data, and so we must consider how robust the algorithm is to such errors. In the following sections we consider the impact of incorrect or missing link data, and incorrect routing data on the MMI algorithm. Only the latter form of incorrect input data has an important impact on the results of the algorithm.

6.1 Incorrect Link Data

Like any measurements, SNMP link data contain errors. Therefore, we shall introduce a range of errors, and study their impact. Comparisons with flow level data have shown that errors in either source are not generally large, and the sources of such errors lead one to believe that they will not be strongly correlated. Hence we shall introduce independent Gaussian errors to the measurements $\mathbf{y}$ and compare with the zero error case. More specifically, take the error in the measurement of link i to be $\varepsilon_i \sim N(0, \sigma)$, where $N(0, \sigma)$ is the normal distribution with mean 0 and standard deviation σ. We vary σ from 0 to 0.1, with the latter corresponding to quite large relative errors in the measurements (remember the 95th percentiles of the normal distribution lie at $\pm 1.96\sigma$.)

Also note that errors on access and peering links will have minimal impact on a BR to BR traffic matrix because the data from access links is aggregated across many links (to form the traffic volumes entering and exiting the network at a router) and so we only consider here errors in the backbone-link traffic measurements.

Figure 5 shows the CDF of the results given different noise levels. Clearly noise impacts the results, but note that the additional errors in the measurements are actually smaller (for the most part) than the introduced errors in the measurements. This is likely due to the redundant link constraints, which provide an averaging effect to reduce the impact of individual errors. Table 3 presents a summary.

noise level (σ)	0	0.01	0.05	0.10
relative errors	11.26%	11.63%	14.00%	18.01%

Table 3: The relative errors given a particular noise level.

6.2 Missing Link Data

We next consider the impact of missing data, for instance missing because a link was not polled over an extended interval. A few missing data points can be replaced using interpolation; trading missing data for data with some error. Furthermore, ERs are typically connected very simply to the backbone (typically by sets of redundant links), and almost all ($> 99\%$) of ER traffic is between the backbone and the edge. Thus if data are missing from a single edge link we may estimate the corresponding traffic using measurements of the traffic between the ER and the backbone. Thus, except in the

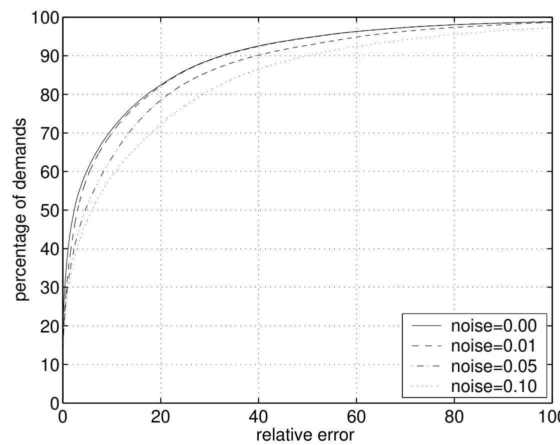


Figure 5: Relative errors for MMI given measurement noise.

rare case where we miss multiple edge links, we need only consider missing backbone link data.

Figure 6 shows the effect of missing the top N backbone links (rated in terms of traffic on those links). The results are shown for the 24 data sets from each of three days in June. The results show that despite loosing the links with the largest traffic, the results are hardly impacted at all (though the step appears because one of these links is actually important). This suggests that there is generally enough redundant information in the network to compensate for the missing links (except in one case).

6.3 Incorrect Routing Data

A third source of data in which we may find errors is the routing matrix. Errors in this matrix can have a large impact on the performance of estimation methods, because if we have errors in a significant number of routes, this corresponds to changing many elements of the matrix from 1 (in the absence of load sharing) to zero and visa versa. However, as in all other reports on traffic matrix estimation, we assume the routing matrix input is accurate. This assumption is reasonable because there are good methods for reliably obtaining routing information (for instance see [19]).

7. ADDITIONAL INFORMATION

One major benefit of adopting the information theoretic approach describe here is that it provides a natural framework for including additional information. In this section, we examine the impact of two sources of information: (i) flow level data at some locations, and (ii) the local traffic matrix at a router [25].

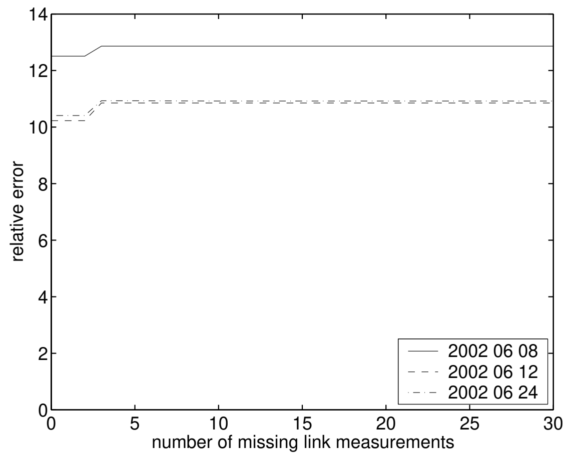


Figure 6: The impact of missing data on the relative errors for three days (each comprising 24 data sets).

7.1 Flow Level Data

In this section we consider the impact of having flow level data at some locations, which gives the rows of the traffic matrix for those locations. This inclusion was explored in [16] in a simulation. They showed that the methods of [23, 3] provided improvements to traffic matrix estimates roughly in proportion to the number of rows measured, but that it did not matter whether one selected the rows to be measured randomly, or in order of largest row sum.

Flow level information can be included in our algorithm by simply including additional constraint equations. Results are presented for three separate days of data, each consisting of twenty four, one-hour data sets. We compare the error in the estimates as we include a variable number of known rows of the traffic matrix, both in row sum order, and randomly. Figure 7 shows the results. In the random-ordering case, we see an approximately linear improvement as additional information is included, but in contrast to the results of [16] row sum order is significantly better. In fact, once 10 rows are included, the error for the row sum case is about half that of the random ordered case, and this ratio improves until we have included around half of the rows, when the error for the row sum ordered case becomes negligible. One possible reason why these results do not agree with [16] is that the traffic matrices used in the simulation were not as skewed as those observed in real networks.

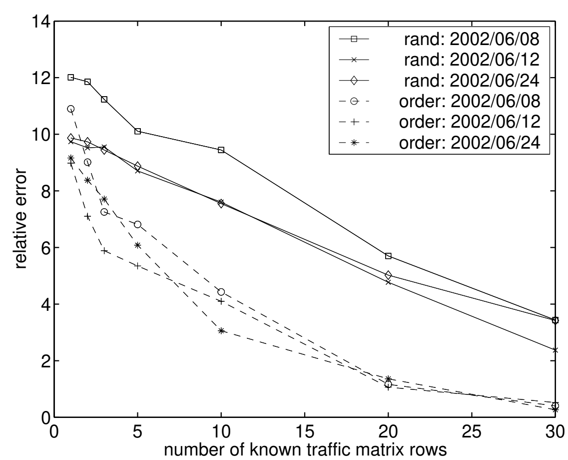


Figure 7: Effect of addition of known traffic matrix rows. Dashed lines show largest row sum ordering, and solid show random order. There are over 60 rows in the traffic matrix.

The result is a clear win for measuring flow, or packet level data. Such data on a fraction of the network may provide a disproportionate improvement in the estimates. The results were similar even when errors were added to the flow level measurements, and so sampled flows may also provide practical improvements.

7.2 Local Traffic Matrices

Another appealing alternative to collect additional information with minimal cost is to collect local router traffic matrices. That is, for the router to keep a table of traffic from in-interface to out-interface. As shown in [25], the collection of local traffic matrices only requires minimal changes to router hardware, and can be included in our algorithm as constraints. Figure 8 shows the CDF including local traffic matrices, and Table 4 shows a summary of the results in comparison to those without local traffic information. Notice that the results with a local traffic matrix, are not only better, but also less sensitive to measurement errors.

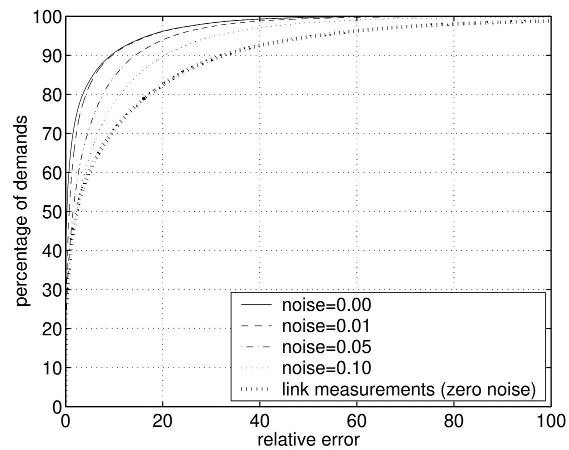


Figure 8: The result of including local traffic matrices, for varying error levels. Also included as a baseline is the zero noise, links measurement case from Figure 3.

noise level (σ)	0	0.01	0.05	0.10
with local TM	3.06%	3.40%	5.04%	7.3%
w/o local TM	11.26%	11.63%	14.00%	18.01%

Table 4: The relative errors given a particular noise level, with and without local traffic matrix data.

The star topology illustrates why a local traffic matrix helps. In that case, a local traffic matrix at the hub router provides the traffic matrix directly. In reality the network is not a star, so a large amount of additional information is redundant. In our problem, the number of constraints is of the order of a factor of 20 times the simple link measurement constraints, but the number of independent constraints is only roughly doubled. However, this redundant information is still useful because it makes the algorithm more robust to noise in the measurements, as seen in Table 4.

These results show that it is quite practical to improve the traffic matrix estimates above by incorporating additional information.

8. CONCLUSION

To summarize, we present a new approach to traffic matrix estimation for IP networks. We demonstrate on real data that the method has nice properties: it is fast, accurate, flexible, and robust. In addition, this paper provides some insight into the problem of traffic matrix estimation itself. In particular, by testing the method on Rocketfuel topologies we provide some measure of what aspects of a network make the problem easier or harder: estimates on more highly meshed networks were more accurate. Further, we found that the relationship between the traffic volumes and the topology played a significant role in the accuracy of the estimates. Apart from this, the method also provides additional insight into a broad range of approaches to traffic matrix estimation.

There is still considerable work to do in this area: for instance, the choice of priors is interesting. It is known that regularization and shrinkage approaches improve estimates even when the prior to which we shrink is arbitrary. However, it is also known that a better prior results in a better estimate. While the prior used here seems adequate, one may be able to do better (for instance by using [16]). Other areas of future work include, understanding why the methods are so insensitive to the value of λ, and performing further validations of the method, on alternate data sets (including different traffic patterns), and direct point-to-multipoint validation.

Acknowledgments

We would like to thank Michael Saunders for the PDSCO code used here, George Varghese for suggesting the collection of local traffic matrices, Dina Katabi and anonymous reviewers for their helpful comments, and the people who helped collect some of the data used here, in particular Joel Gottlieb, and Fred True.

9. REFERENCES

[1] M. Bertero, T. Poggio, and V. Torre. Ill-posed problems in early vision. In *Proc. of the IEEE*, 76:869–889, 1988.

[2] I. J. Craig and J. C. Brown. *Inverse Problems in Astronomy: A Guide to Inversion Strategies for Remotely Sensed Data*. Adam Hilger, Boston, 1986.

[3] J. Cao, D. Davis, S. V. Wiel, and B. Yu. Time-varying network tomography. *J. Amer. Statist. Assoc*, 95(452):1063–1075, 2000.

[4] J. Cao, S. V. Wiel, B. Yu, and Z. Zhu. A scalable method for estimating network traffic matrices from link counts. Preprint. Available at `http://stat-www.berkeley.edu/~binyu/publications.html`.

[5] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[6] B. Efron and C. Morris. Stein's paradox in statistics. *Scientific American*, 236(5):119–127, 1977.

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

[8] 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*, pages 265–279, June 2001.

[9] P. C. Hansen. Regularization tools (for Matlab). `http://www.imm.dtu.dk/~pch/Regutools/index.html`.

[10] P. C. Hansen. Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems. *Numerical Algorithms*, 6:1–35, 1994.

[11] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.

[12] G. Jumarie. *Relative Information*. Springer-Verlag, 1990.

[13] E. Lehmann. *Theory of Point Estimates*. Wiley, New York, 1983.

[14] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[15] 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.

[16] 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.

[17] A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40(3), 1998.

[18] R. Parker. *Geophysical Inverse Theory*. Princeton University Press, Princeton, NJ, 1994.

[19] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb. A case study of OSPF behavior in a large enterprise network. In *ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, 2002.

[20] J. Skilling. The axioms of maximum entropy. In G. J. Erickson and C. R. Smith, editors, *Maximum-Entropy and Bayesian Methods in Science and Engineering*, Volume 1: Foundations, pages 173–187. Kluwer Academic Publishers, 1988.

[21] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.

[22] N. Spring, R. Mahajan, D. Wetherall, and H. Hagerstrom. Rocketfuel: An ISP topology mapping engine. `http://www.cs.washigton.edu/research/networking/rocketfuel/`.

[23] C. Tebaldi and M. West. Bayesian inference on network traffic using link count data. *J. Amer. Statist. Assoc*, 93(442):557–576, 1998.

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

[25] G. Varghese and C. Estan. The measurement manifesto. Technical Report CS2003-0747, UCSD, 2003.

[26] G. Wahba. Constrained regularization for ill posed linear operator equations, with applications in meteorology and medicine. In *Statistical Decision Theory and Related Topics III*, 2:383–418, Academic Press, 1982.

[27] B. Yu. Maximum pseudo likelihood estimation in network tomography. In *NISS Internet Tomography Technical Day*, Research Triangle Park, March 28 2003. Available at `http://www.niss.org/affiliates/internet030328/presentations20030328.html`.

[28] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *ACM SIGMETRICS*, San Diego, USA, June 2003.

APPENDIX

A. CONDITIONAL INDEPENDENCE

In Section 3.5 we present a result based on conditional independence, rather than simple independence. Zero transit traffic makes it is more reasonable to adopt a conditionally independent model in which the probabilities of a packet (bit) arriving at s and departing at d given the class of arrival and destination link (peering or access) are independent:

$$\begin{aligned} p_{S,D}(s,d|S \in C_s, D \in C_d) = \\ p_S(s|S \in C_s, D \in C_d)\, p_D(d|S \in C_s, D \in C_d), \end{aligned} \tag{32}$$

where C_s, and C_d are the source the destination's link class, respectively. We know

$$p_{S,D}(s,d) = p_{S,D}(s,d|S \in C_s, D \in C_d)\, p_{S,D}(C_s, C_d) \tag{33}$$

The source and destination links only depend on the class of the source and destination respectively, so

$$p_S(s|S \in C_s, D \in C_d) = p_S(s|S \in C_s), \tag{34}$$

$$p_D(d|S \in C_s, D \in C_d) = p_D(d|D \in C_d). \tag{35}$$

Furthermore, from the definition of conditional probability

$$p_S(s|S \in C_s) = p_S(s) \,/\, p_S(C_s), \tag{36}$$

$$p_D(d|D \in C_d) = p_D(d) \,/\, p_D(C_d), \tag{37}$$

with the result

$$p_{S,D}(s,d) = \frac{p_S(s)}{p_S(C_s)} \frac{p_D(d)}{p_D(C_d)} p_{S,D}(C_s, C_d) \tag{38}$$

If the class of source and destination were independent, then (38) would result in the independent model $p_{S,D}(s,d) = p_S(s)p_D(d)$. However, noting that all traffic from peering must go to access, and likewise, all traffic to peering links comes from access, and further that the four probabilities must add to one, we get.

$$\begin{aligned} p_{S,D}(P,P) &= 0 \\ p_{S,D}(P,A) &= p(d \in A|s \in P)\, p_S(P) = p_S(P) \\ p_{S,D}(A,P) &= p(s \in A|d \in P)\, p_D(P) = p_D(P) \\ p_{S,D}(A,A) &= 1 - p_S(P) - p_D(P). \end{aligned}$$

Substituting into (38) we get (28).