Traffic Engineering with Estimated Traffic Matrices

Matthew Roughan Mikkel Thorup Yin Zhang
AT&T Labs – Research,
180 Park Avenue, Florham Park, NJ, 07932.
{roughan,mthorup,yzhang}@research.att.com

ABSTRACT

Traffic engineering and traffic matrix estimation are often treated as separate fields, even though one of the major applications for a traffic matrix is traffic engineering. In cases where a traffic matrix cannot be measured directly, it may still be estimated from indirect data (such as link measurements), but these estimates contain errors. Yet little thought has been given to the effects of inexact traffic estimates on traffic engineering. In this paper we consider how well traffic engineering works with estimated traffic matrices in the context of a specific task; namely that of optimizing network routing to minimize congestion, measured by maximum link-utilization. Our basic question is: *how well is the real traffic routed if the routing is only optimized for an estimated traffic matrix?* We compare against optimal routing of the real traffic using data derived from an operational tier-1 ISP. We find that the magnitude of errors in the traffic matrix estimate is not, in itself, a good indicator of the performance of that estimate in route optimization. Likewise, the optimal algorithm for traffic engineering given knowledge of the real traffic matrix is no longer the best with only the estimated traffic matrix as input. Our main practical finding is that the combination of a known traffic matrix estimation technique and a known traffic engineering technique can get close to the optimum in avoiding congestion for the real traffic. We even demonstrate stability in the sense that routing optimized on data from one day continued to perform well on subsequent days. This stability is crucial for the practical relevance to off-line traffic engineering, as it can be performed by ISPs today.

Categories and Subject Descriptors

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

General Terms

Measurement, Performance

Keywords

Traffic Matrix Estimation, Traffic Engineering, SNMP, OSPF, MPLS

IMC'03, October 27–29, 2003, Miami Beach, Florida, USA.

1. INTRODUCTION

Estimating an Internet traffic matrix has received considerable attention in recent years. A traffic matrix provides the volume of traffic between every pair of ingress and egress points over a given time interval. Such information is essential to a variety of operational tasks ranging from router/link failure analysis to capacity planning and traffic engineering, for instance by route optimization.

When direct flow-level measurements are available, accurate traffic matrices can be derived following the approaches detailed in [1]. Unfortunately, direct measurements require additional infrastructure support and it can be prohibitively expensive to instrument the entire IP network to collect such data. Recently, progress has been made on traffic matrix estimation and several methods [3, 4, 5] have been proposed that attempt to derive traffic matrices from the link load data, which can be easily obtained via the Simple Network Management Protocol (SNMP). We call such a technique an SNMP-based *traffic matrix estimator*. These algorithms have been validated against real (but partial) traffic matrices (obtained through direct measurements) using common metrics such as mean error computed over all source-destination pairs. The resulting estimates contain errors of varying magnitude depending on the traffic matrix estimator applied. It is, however, not directly clear what impact these errors have on operational tasks, as different tasks may have quite different tolerance to the types and magnitude of the errors. For example, if all errors were concentrated on a single critical link, this could have a big impact on performance, yet a negligible impact using most standard error measures.

In this paper, we attempt to establish a direct connection between SNMP-based traffic matrix estimators and one particular network operational task: traffic engineering to minimize congestion. That is, we are interested in the following operational performance measure:

> *If traffic engineering is done based on the estimated traffic matrix, how well does it perform on the real traffic matrix?*

Several traffic engineering techniques have been presented that optimize routing to minimize congestion [6, 7, 8, 9, 10, 11]. We call such a technique a *routing optimizer*. It sets the routing parameters of a network for a given traffic matrix so as to minimize congestion for that traffic matrix. The routing parameters determine, for each source-destination pair, the fraction of traffic going on different paths from the source to the destination. Typically, in the past, routing optimizers were evaluated using synthetic traffic matrices. In this paper we feed the routing optimizer an estimated traffic matrix while measuring the performance of the routing on the real traffic matrix.

In this paper, as in [9], max-utilization is picked as the easiest to appreciate measure for congestion. The *utilization* of a link is the ratio of its load over its capacity, and the *max-utilization* is the maximum utilization over all links in the network. Other works have focused on more sophisticated cost functions, summing costs over all links in the network (see, e.g., [8, 7]), but these are less easy to understand, and might obscure the point that the performance of estimation and optimization combined is not easily extrapolated from the performance of one by itself. Some intuition may be gained from the finding in [12] that their routing optimizer was robust to $\pm 50\%$ random errors, multiplying each individual demand with a random value from $[0, 2]$. We consider 50% a large mean error, and yet it only affected the max-utilization by about 10%. For contrast, if in a large network, we only changed the demands that used a specific highly utilized link, this would have a large impact for the max-utilization, yet a negligible impact on the average error.

We deliberately treat both traffic estimators and routing optimizers as black-boxes that we combine in a plug-and-play manner. Both sides are based on previously published techniques. The contribution of this paper is to see what happens when the two sides are combined. Tests were performed using simulations based on data from an operational tier-1 ISP. We found that, in itself, the magnitude of errors in the traffic matrix estimate was not a good indicator of the performance of that estimate in our traffic engineering tasks. Likewise, the traffic engineering algorithm that performs best knowing the real traffic matrix was no longer the best with estimated traffic matrices. Our main practical finding was that combining the OSPF routing optimizer technique from [7] with the tomogravity traffic matrix estimator from [4], we got close to the minimal max-utilization for the real traffic. The above OSPF routing can also be implemented with IS-IS and MPLS, making it broadly applicable to todays IP network.

To further test the applicability of our combination, we took an OSPF routing solution based on estimated traffic matrices from one day, and tested this routing on the real traffic over the following week. We found that the routing continued to perform well. Thus our approach was not only robust to estimation, but also reasonably stable over time. This later property is crucial for realistic off-line implementations in today's IP networks, where changing link weights frequently can result in network performance degradation.

Contents. The paper is divided as follows. In §2, we discuss the different routing optimizers considered, and in §3, we discuss the different traffic matrix estimators. In §4, we present our experimental methodology. Our results are presented in §5, followed by some practical considerations in §6. Then we have some reflections over limitations of the paper in §7, and finally we end with concluding remarks in §8.

2. ROUTING OPTIMIZERS

In this section, we discuss the different routing optimizers considered. We note that these are all based on published work, and the reader will be referred to the relevant publications for most technical details. The interesting new aspect is what happens when the optimizers, viewed as black-boxes, are applied to estimated traffic matrices and tested on real traffic matrices.

2.1 General routing with MPLS

In the most general form of routing, traffic from a source to a destination may be split arbitrarily over all possible paths between source and destination. Finding a general routing minimizing max-utilization is an instance of the classical multicommodity flow problem which can be formulated as a linear program [13, Chapter 17]. As described by Mitra and Ramakrishnan [6], the linear program solution can be implemented with the quite recent Multi-Protocol Label Switching (MPLS) protocol [14]. Essentially, each path used is implemented as a label-switched path that the source uses for a certain fraction of its traffic to the destination. We used the commercial linear programming package CPLEX version 6.5 to solve the standard linear program to minimize the max-utilization for a given traffic matrix, and we refer to this as the *MPLS optimizer*.

The MPLS optimizer is optimal in that if applied to the *true* traffic matrix, it gives the best possible performance among all routing protocols with the given traffic matrix and network, using the max-utilization as the only performance criteria. Other possible criteria such as feasibility of the implementation, robustness to link-failures, etc., are not considered.

However, what happens if the MPLS optimizer finds the optimal MPLS solution for the estimated traffic matrix and then applies it to the real traffic? The MPLS solution tells us exactly how traffic should be split over different paths from source to destination, and this splitting is now applied to the real traffic matrix. How good is the resulting routing compared with the above optimal MPLS routing for the real traffic matrix? Put conversely, how sensitive is the optimal solution to errors in estimating the traffic matrix? As we shall see, the answer is 'quite sensitive'.

In fact the MPLS optimizer can have some strange results when the inputs have errors. The algorithm may, without penalty, allow route loops for traffic matrix elements of small magnitude, as long as these loops do not affect the max-utilization objective function. This is the result of focusing the optimization on only minimizing the maximum utilization — a loop in a small traffic matrix element has zero penalty under such an objective function. However, if this small traffic matrix element contains errors, the loop will amplify the error when the traffic is routed.

For an example of a more robust optimization we tried modifying the objective function above to include a penalty for loops, and refer to the resulting algorithm as the *MPLS* optimizer*.

We note, however, that MPLS can implement any possible routing, so even if the above concrete MPLS optimizers do not work well with estimated traffic matrices, this does not imply that MPLS in itself cannot be made robust with respect to estimation. Also, in all fairness, it should be mentioned that the context for the optimal MPLS solutions in [6] was a matrix of virtual leased lines where the ISP commits to a certain amount of traffic for each source destination pair. These commitments are fixed in contracts, and can be honored as is.

2.2 Traditional shortest path routing

The most commonly used intra-domain Internet routing protocols today are the *shortest path* Interior Gateway Protocols (IGP): Open Shortest Path First (OSPF) [15] and Intermediate System-Intermediate System (IS-IS) [16]. In these protocols, which are functionally the same, each link is associated with a positive weight, and the length of a path is defined as the sum of the weights of all the links on that path. Traffic is routed along the shortest paths. In cases of ties where several outgoing links are on shortest paths to the destination, the flow is split roughly evenly.

By default, Cisco routers [17] set the weight of a link to be inversely proportional to its capacity — we refer to this setting as the *InvCap* weight setting. The weights of the links, and thereby the shortest path routes, can be changed by the network operators to optimize network performance.

Over the years, many methods [7, 8, 9, 10, 11] have been presented that compute a set of link weights that minimize congestion in the resulting shortest path routing of a given traffic matrix. We shall refer to such a method as an *OSPF optimizer*, though the results could equally be applied to IS-IS routing. We use the approach described in [7, 12], which is based on so-called local search techniques [18]. The method uses heuristics to iteratively improve the weight setting, changing one or a few weights in each iteration. As a standard, we ran it for 5000 iteration, taking about 5 minutes of simulation time. The problem of finding an optimal weight setting is NP-hard [7], and so we cannot guarantee finding the true optimum. The quality of the final weight setting is affected by random choices made through the iterations, giving some variance in the quality of the outcome. For example, it is possible that we, by chance, get a better weight setting for the true traffic matrix from the estimated traffic matrix than we would get from the real traffic matrix itself, but the results below show that this random variation is not very important in practice.

Of course, as argued carefully in [12], it is not attractive to optimize the weight setting on-line as the demands change. As in [12], our weight optimizer works for multiple traffic matrices. Even more importantly we will consider the impact of using the optimized routes as a *permanent weight setting*. This permanent weight setting is then tested on the true traffic matrices of the subsequent days.

3. ESTIMATING TRAFFIC MATRICES FROM LINK DATA

This section describes three methods for estimating traffic matrices from link load data. The first two methods are based on so called "Gravity models" while the third uses (in addition) "Network tomography" methods. Although it might be appealing to test some more complex algorithms, the sub-sample of possibilities presented here is sufficient to illustrate the points of interest. What's more we find a near optimal combination of estimation and routing optimization algorithms in any case, so there is little to be gained in using a more complex method.

This section is not intended to provide a detailed description of the estimator algorithms (which may be found in [4]). This is not intended as a study of the estimators. The novel aspect is what happens when the estimators are combined with routing optimizers and tested on real traffic matrices. The description here is to provide some insight into the relationship between the three algorithms tested.

Gravity models [19, 20, 21], are often used by social scientists to model the movement of people, goods or information between geographic areas [20, 21]. Recently, variations on gravity models have also been proposed for computing traffic matrices [3, 4, 5].

At the heart of the gravity model approach is a proportionality assumption: the amount of traffic from a given source to a given sink is proportional to the total traffic to the output sink, independent of source. For example, in a gravity model for car traffic between cities the relative strength of the interaction between two cities might be modeled as proportional to the product of the populations divided by a distance related "friction" term. Similarly, the simplest possible gravity models for the Internet assume that the traffic exchanged between locations is proportional to the volumes entering and exiting at those locations, though in this case we assume the distance related term is a constant because interactions in the Internet are less distance sensitive. This simple model of the Internet is used in [22], and we refer to it as the *simple gravity model*.

It is possible to generalize the simple gravity model in a number of ways [3, 4, 5] to take into account additional information provided by detailed link classification and routing policies. [3, 4, 5] have shown these gravity models to be significantly more accurate than the simple gravity models. We test the *generalized gravity model* of [4] in which additional information on points of ingress and egress for traffic flows can be incorporated to explicitly model hot-potato routing for traffic exchanged with peer networks.

By appropriate normalization, the gravity model solution is guaranteed to be consistent with the measured link loads at the network edge, but not necessarily so in the interior links. Alternatively, network tomography methods explicitly include the information measured from internal links. This information can be written as a set of linear constraint equations

$$\mathbf{x} = \mathbf{A}\mathbf{t}, \qquad (1)$$

where $\mathbf{x}$ is a vector of the link measurements, $\mathbf{t}$ is the traffic matrix written as a column vector, and $\mathbf{A}$ is the routing matrix, whose terms give the fraction of traffic from a particular origin/destination pair that traverse each link.

In practice this set of equations is ill-posed, and so to deal with this difficulty tomographic techniques from other fields have been used. For a detailed description and comparison (using simple metrics) of a number of these methods see [5]. We shall consider a single such algorithm, *tomogravity*, [4] which displays good properties in terms of scaling, estimation accuracy, speed of computation, and robustness to errors. The method uses the generalized gravity model above as a prior (a kicking off point) and refines it using a tomographic technique to select an estimate of the traffic matrix $\hat{\mathbf{t}}$, that satisfies the constraint equations, but that is closest to the gravity model according to some distance metric.

4. EXPERIMENTAL METHODOLOGY

4.1 Ideal

In this context it is possible to generate arbitrarily bad results for any particular algorithm by choosing pathological topologies or traffic matrices, but the important question is how well these algorithms perform on real data. The ideal experiment to test the use of traffic engineering on estimated traffic matrices would have SNMP link traffic measurements, a perfect traffic matrix, and exact topology information, all from exactly the same moment in time. Finally, the new routing computed should be tested in the real network back at the time when the measurements were made. Unfortunately, most of this is impractical.

Each different type of data has limitations, and practical constraints in how it may be collected. For instance

- Currently we do not have high-resolution traces of the network topology, and so we only have snapshot views of the network;

- Flow-level data (which is the easiest starting point for deriving a traffic matrix) is not generated as a traffic time series, but rather an overlapping set of flows, and in many cases can only be collected on a sampled basis. Furthermore, flow-level data can be hard to collect in places because it is a feature of a router, and not all routers support this feature, or its use conflicts with other features. Further, in some cases, collecting flow-level measurements might result in a reduction in forwarding performance (which is highly undesirable). Furthermore, flow-level data for an entire network can be vast — potentially terabytes per day — and handling this volume of data is daunting in and of itself.

- SNMP link data have many limitations — for instance missing data (SNMP uses unreliable UDP transport), incorrect data (through poor router vendor implementations), and a coarse sampling interval (five minutes is typical).
- Experimenting with the routing of a real operational tier-1 ISP is not an option. We have to conduct our investigation with simulations.

The network traffic also exhibits strong daily, and weekly cycles, and so averaging results over intervals longer than one or two hours is not very meaningful.

It is difficult to overstate the importance of consistency in the data. We do not wish the results here to be due to artifacts in the data, but the above problems make it seemingly impossible to generate a realistic, completely consistent set of test data. However, [4] presents an alternative methodology when testing their estimation algorithm, which we adapt here. In the following section we describe the data we have available, and the methodology used to test how well traffic engineering works using estimated traffic matrices.

Also, comparisons against the current routing in the real network are interesting, but would reveal proprietary information. Instead, as a benchmark, we here compare our performance against Cisco's [17] default InvCap weight setting for OSPF.

4.2 Inputs

This paper does not directly consider SNMP data for the reasons above. It would be unreasonably difficult to collect SNMP traffic statistics consistent with the traffic matrix and topology information available. The approach used here is to use

- sampled flow-level data, and
- topology and routing information as derived from [23].

The 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. Combined with topological and routing information (as in [1]) one may derive a traffic matrix from such information.

As noted above it is hard to have complete flow-level coverage of the network. In the data sets used here we cover around 80% of the edge of a large tier-1 IP network, including all the traffic on inter-peer links. The traffic matrices generated using this data will therefore be partial, in the sense that we are missing some rows from the true traffic matrix. However, the resulting traffic matrix is still a real traffic matrix (covering around 80% of the network traffic) on the real network topology, and so is as good a possible set of measurements as are currently available (for instance in [5] only three rows of the traffic matrix were available). This traffic matrix is what we shall refer to as the *true traffic matrix* throughout the rest of the paper.

The nature of flow-level data makes it only possible to approximate time-series data. Flow-level information contains the start and stop time of the flow, and the number of packets/bytes, but not when the packets were sent within the flow. Given that some flows can continue for hours, it is only practical to look at time series of the order of the timeouts used to flush current flows. Note that there is no inherent reason why the timeouts will occur at the same time at different routers, and so to use commensurate time series, one must average over longer intervals than the timeout to obtain useful data (using a more sophisticated interpolation scheme runs the risk of introducing artifacts into the data). In Cisco Netflow, the timeouts are of the order of 15 minutes, and so we consider time series at a one hour time scale, allowing (with not too much approximation) for these intervals to be offset at different routers. Over longer intervals the traffic is non-stationary (showing strong diurnal cycles) and so we do not wish to use longer time averages if possible.

The topology and routing information are derived from information gathered from the same network using the methods of [1]. Given these traffic matrices and the network topology, we need only a consistent set of link load measurements to proceed.

4.3 Methodology

The problem of providing a consistent set of traffic, topology and link measurement data can be solved as follows. We simulate the OSPF routing using the existing topology and link weights (and area structure). The existing link weights are those currently set by the network operator. From this we may compute a routing matrix $\mathbf{A}$, and then derive a set of link measurements $\mathbf{x}$ from (1). Thus the traffic matrix $\mathbf{t}$, the routing matrix $\mathbf{A}$ and the measured link loads $\mathbf{x}$ are all consistent.

We can then perform the estimation procedure to compute $\hat{\mathbf{t}}$, the traffic matrix estimate. This approach allows us to work with a problem for which we have both a real estimate and the true traffic matrix.

To help the reader understand the issues involved, we shall summarize the errors in the estimated traffic matrices in Section 5.1. However, the point of this paper is that simply looking at these errors is not enough to understand whether a traffic matrix estimate is "good". To really understand whether an estimate is good, one must assess how well it performs in operational tasks.

The task we assess here is traffic engineering — in particular the task of optimizing the network routing to make the network more efficient in its use of resources (and hence reduce congestion). To do this we use one basic approach. We compute routing by applying a routing optimizer to the estimated traffic matrix. We then assess how well these routes work for the real traffic matrix.

Specifically, consider the task of optimizing the OSPF weights in a network. Based on the traffic matrix estimate, we optimize the weight setting:

$$\hat{\mathbf{w}} = \text{OSPF-weight-optimizer}(\hat{\mathbf{t}}).$$

An OSPF simulator takes the new weights and finds the corresponding optimized routing matrix

$$\tilde{\mathbf{A}} = \text{OSPF-route-simulater}(\hat{\mathbf{w}})$$

Finally, we apply this new routing to the original true traffic matrix $\mathbf{t}$ so as to get a set of link loads:

$$\hat{\mathbf{x}} = \tilde{\mathbf{A}}\ \mathbf{t}.$$

The max-utilization optimized for the estimated traffic matrix $\tilde{\mathbf{t}}$ but applied to the true traffic matrix $\mathbf{t}$ is then

$$\text{max-utilization}(\hat{\mathbf{t}}; \mathbf{t}) = \max_i \frac{\hat{x}_i}{C_i},$$

where C_i are the link capacities. The whole procedure behind the experiments is illustrated in Figure 1.

We will also compare the results with those under alternative routing, for instance, using a routing matrix derived using MPLS optimization from the true or estimated traffic matrix, and the InvCap routing. The only difference is the mechanism used to generate the optimized routing matrix $\tilde{\mathbf{A}}$ from the estimated traffic matrix $\hat{\mathbf{t}}$.

We concern ourselves with optimizing the routing of the inter-PoP backbone-router network containing on the order of one hundred routers, with a few links per router. Link based traffic matrix estimates are difficult to obtain on any finer granularity than this,

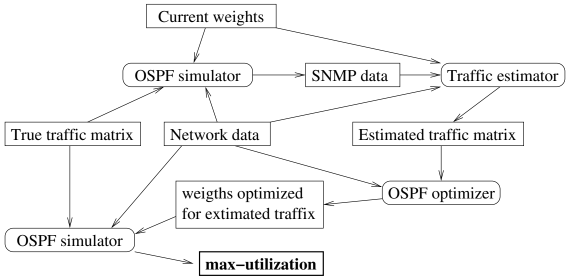


Figure 1: Diagram over experiment.

and OSPF allows a hierarchical routing based on areas, which can naturally be used to isolate the routing of local intra-PoP traffic from inter-PoP traffic.

5. RESULTS

The results shown here are derived from the backbone of a tier-1 ISP network. We present results over the course of one day (the 17th of August 2002) to show the effects of the changing matrix over the course of the day. We will also show results from a separate segment of data to illustrate the performance of routing prediction. The data is broken into one hour data sets over which the traffic matrix is approximately stationary.

For proprietary reasons, max-utilizations reported in this paper are scaled so that their absolute value for the operational backbone cannot be deduced. Such scaling does not affect the relative performance of the different schemes. Proprietary reasons also prevent us from exposing the performance of the OSPF weight settings used in the operational network.

5.1 Errors in Traffic Matrix Estimates

A detailed general analysis of the errors in the different traffic matrix estimates is presented in [4]. For reference in this paper we provide some simple measurements of the errors. In Figure 2 we present relative error of estimated traffic matrices versus true traffic matrices. That is, for each hour we compute the sum (over the source-destination pairs) of the absolute value of the error between estimated and true traffic, and divide this sum by the total traffic. We see that tomogravity is more than twice as good as general gravity, which is more than twice as good as simple gravity. These findings are consistent with those reported in [4].

In Figure 3 (a) we present an alternative representation of the estimates more comparable with later figures on max-utilization. A simple-minded hypothesis is that optimizing over the true traffic matrix, the max-utilization is going to be proportional to mean traffic and that if we optimize over an estimated traffic matrix, the performance is degraded by mean error. Figure 3 (a) shows the mean traffic plus the mean (absolute) errors for each of the data sets over the course of the day. If our simple-minded hypothesis is true, the curves should roughly match those of max-utilization achieved with the estimated traffic matrices.

5.2 Max-utilization versus mean errors

We now test how well the estimated traffic matrices perform on max-utilization. We apply the OSPF optimizer to each estimated demand matrix, including the true traffic matrix, and test the resulting routing on the true traffic matrix. The resulting max-utilizations are depicted in Figure 3 (b). At this stage, we could also have applied an MPLS optimizer, but as we shall see shortly, these are not as reliable. Averages and maximums over the 24 hours are contained in Table 1 along with other data. More precisely, the table reports the

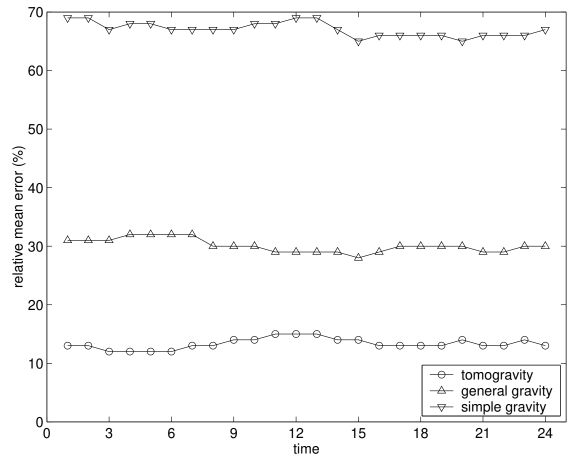


Figure 2: Relative error for the different traffic estimates for each of the 24 hours. The circles show the tomogravity estimate, and the $\triangle$ and $\bigtriangledown$ show the general and simple gravity model estimates. For tomogravity, the average relative error is 0.13, for general gravity it is 0.30, and for simple gravity it is 0.67.

- **Average Max-Utilization (AMU):** the average over all hours of the max-utilization for the relevant method.
- **Max Max-Utilization (MMU):** the largest max-utilization for the method over all hours.

Both quantities are reported as percentages of the MMU for InvCap (recall that we for proprietary reasons cannot give the absolute numbers). The former metric gives an average view of performance, while the latter is a type of worst case comparison.

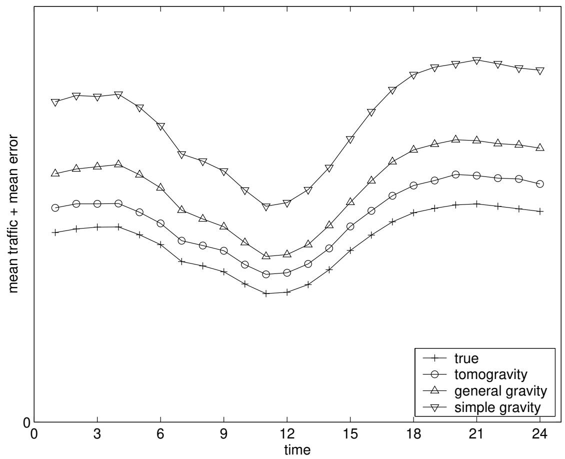
The most interesting thing to observe is that when it comes to the max-utilization in Figure 3 (b), the performance of simple gravity and general gravity is roughly the same. As calculated in Table 1, simple gravity slightly outperforms general gravity both on the average and in the worst-case. This is in sharp contrast to the findings in §5.1 that the errors of general gravity are half as large as those of simple gravity. Hence, the error improvement of general gravity does not help reduce the max-utilization. The error measurements in [4] also rate general gravity over simple gravity, so this is a strong counter example to the idea that one can use simple error measurements to make general conclusions about the performance of traffic estimates in traffic engineering.

The above being said, we do see that the tomogravity estimates perform best both with respect to errors and with respect to max-utilization, on the average getting within 6% of OSPF optimization based on the true traffic matrix. This is quite small compared with the 13% average error shown in Figure 2. We also note that we actually perform slightly better with the estimated than with the true traffic in hour 8. This clearly illustrates the point from §2.2 that the OSPF optimizer is only a heuristic, not guaranteed to find optimal solutions.

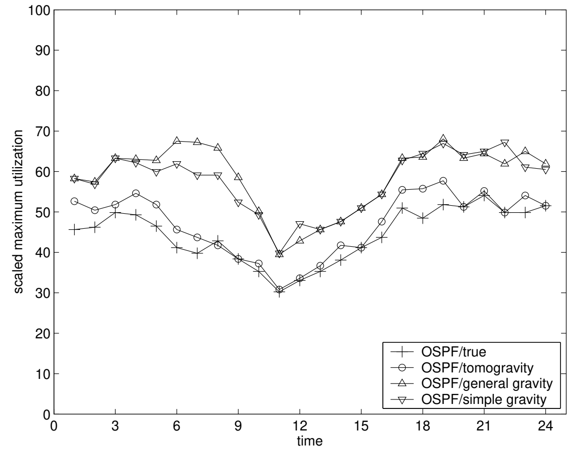
5.3 Sensitivity of optimizers to estimates

We next see how well the OSPF optimization compares with:

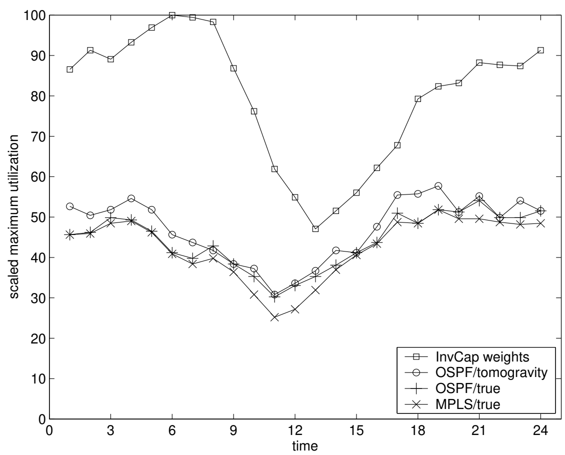
- inverse capacity weights: in this default weight setting the weights are the inverse of the capacity of links. This is a



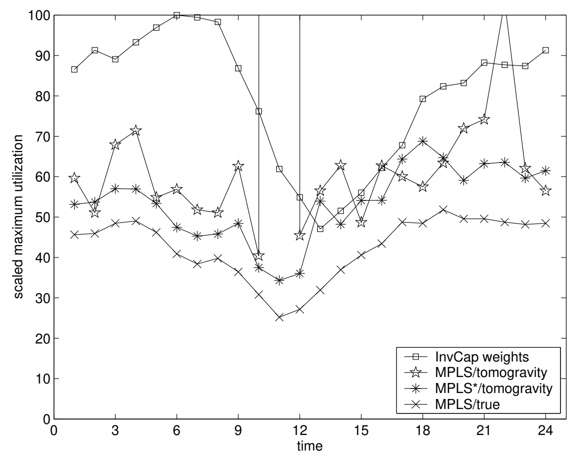
(a) Mean traffic plus the mean absolute errors for each method of estimation. The '+' signs show the magnitude of the true traffic matrix (with zero errors), the circles the tomogravity estimate, and the $\triangle$ and $\bigtriangledown$ show the general and simple gravity model estimates.



(b) The '+' signs show the performance of the OSPF optimization using the true traffic matrix, and the circles, $\triangle$ and $\bigtriangledown$ show the OSPF optimization on the tomogravity, general and simple gravity model estimates, respectively.



(c) The 'x's show the optimal MPLS routing, the '+' signs show the OSPF optimization using the true traffic matrix, and the squares show the result of inverse capacity OSPF weights. The circles show the result of the true traffic matrix being applied with weights from OSPF optimization using the estimated traffic matrices.



(d) As before, the 'x's show the optimal possible routing, and the squares show the result of inverse capacity OSPF weights. The stars show the result of using the estimated traffic matrix in MPLS optimization, and then applying the results to the true traffic matrix. Note that in one case the result is several times worse than can be displayed on this graph, and goes off the scale. Similarly, the '*'s show the performance of the modified MPLS algorithm.

Figure 3: Results for the 1 hour data sets over the 17th of August 2003. Figure (a) shows a comparison of errors in the estimates, (b)–(d) show the maximum utilizations of the network under various routing optimization. Note that the y-axes for (b)–(d) are all scaled in the same way so that the MMU of the InvCap weights is 100.

reasonable attempt at optimization given no other network information.

- MPLS and MPLS* optimization: both optimally minimizing the max-utilization with respect to the given traffic matrix.

- true traffic matrix: in which we apply the optimization algorithms above to the true traffic matrix elements.

Figure 3 (c) shows the maximum utilizations for each hour of the day, under various routing schemes. The '$\times$' signs show the optimal possible routing, i.e., the result of applying the MPLS optimizer to the true traffic. The '+' signs show the OSPF optimization using the true traffic matrix, and the squares show the result of inverse capacity OSPF weights. The 'o's show the result of the true traffic matrix being applied with weights from OSPF optimization using the estimated traffic matrices using the tomogravity method.

We can see in Figure 3 (c) that the OSPF optimization algorithm ('+'s) comes very close to the optimal possible solution ('$\times$'s) in most cases, and is always a significant improvement over the default InvCap weights. Furthermore, weights found from the estimated traffic matrix ('o's) perform only slightly worse than weights found from the true traffic matrix. In fact, as seen in Table 1, even when the OSPF optimizer is applied to the tomogravity estimated traffic matrix, it gets within 11-12% of the optimum solution. This is clear evidence that the errors in the tomogravity estimated traffic matrix are not very important from our particular traffic engineering perspective, and that SNMP link statistics could be used to estimate traffic matrices of considerable use.

Another interesting point is that whereas the optimized routings generally follow the developments in the mean traffic ('+'s) in Figure 3 (a), the InvCap routing does not. This shows that the traffic is not just scaled up and down during the day, but that it is also shifted around in ways that impact InvCap differently, yet which can be accommodated by the optimizers.

Figure 3 (d) shows the same optimal, and InvCap results, but compares them to the MPLS and MPLS* optimization using estimated data. Note that in one case using the MPLS optimization, the result is several times worse than can be displayed on this graph. According to Table 1 it goes off by a factor 400, and a more detailed investigation revealed that this was due to a loop, as discussed in §2.1. The MPLS* optimizer avoids this problem, but given the tomogravity estimates, it is still twice as far from the true optimum as the OSPF optimizer. Thus we have a strong counter example to our simple-minded hypothesis that the optimization that performs best given the true traffic would also perform best on estimates.

optimization method	traffic matrix	performance (%) AMU	MMU
InvCap	N/A	79.9	100.0
OSPF	simple gravity Model	57.5	67.2
OSPF	general gravity model	58.6	68.1
OSPF	tomogravity	47.1	57.7
OSPF	true	44.4	54.1
MPLS	tomogravity	1735.5	40259.7
MPLS*	tomogravity	53.5	68.8
MPLS	true	42.5	51.8

Table 1: The AMU and MMU of each method as a percentage of the MMU for the InvCap weights.

5.4 Further merits

We note that many of the above results sound similar to those reported in [7, 12], but the major difference is that the weight optimization is done based on an estimated traffic matrix while measured on a true traffic matrix, thus giving us the desired feed-back on the operational value of the estimated traffic matrix.

Another very important difference to all the previous work on OSPF optimization is that our results are based on measurements from a real operational network. The closest to this was the work from [7] that used a proposed AT&T WorldNet backbone with projected demands. It is only very recently that good measurements of traffic matrices have become possible on a sufficiently large scale. A secondary merit of this paper is thus to substantiate previous weight optimization work the with first real data.

6. PRACTICAL CONSIDERATIONS

While the results above show the OSPF optimization to work well with the tomogravity estimates, there are a number of considerations before any method can be consider to be practical. This section will address some of these issues.

6.1 Faster OSPF optimization

The OSPF optimization algorithm is quite fast — on the current network it runs in around 330 seconds (for 5000 iterations in Figure 3, on a Sun 900 MHz Ultrasparc-III Copper processor). However, the majority of the benefits of the optimization algorithm used here come early in the algorithm [7, 12] when applied to real data. It is worth testing whether this is also the case when we use an estimated traffic matrix. It is indeed the case, as is shown by Figure 4, which shows the same set of data with the results of the weights after 5000 iterations and only 1000 iterations. The relative increase in maximum utilization (due to 5 times fewer iterations) is always below 5%. In contrast, the time for 1000 iterations is only about one fifth of that for 5000.

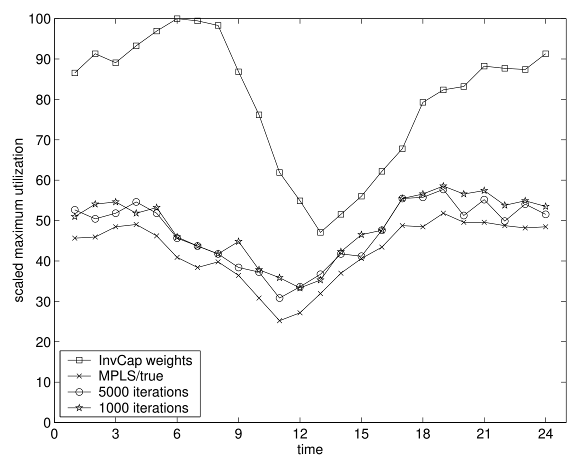


Figure 4: Results of route optimization as in Figure 3 (c), and with only 1000 iterations of the OSPF route optimization algorithm.

6.2 Global Optimization

Frequent changes to OSPF link weights are highly undesirable. Apart from the management complexity this induces on the network, OSPF routing takes at least seconds to re-converge after each change. While this may seem a short time, on a large network carrying 100's of Terabytes per day, changing the OSPF routing each hour could easily result in the loss of a 100 GBs of traffic or more per day.

It is, however, possible to use the optimization method above to compute an optimum routing for all 24, one hour traffic matrices for one day. More precisely, in [12] a technique is described that seeks a single global weight setting that works well for multiple demand matrices, obtaining a good max-utilization for all of them. We applied this technique as a black-box. This provides another interesting test of how well the traffic matrix estimates perform. We compute such a set of weights from the tomogravity estimated traffic matrices for one day, and then determine how well the weights perform for each hour of the day using the real traffic matrix. Figure 5 shows these results.

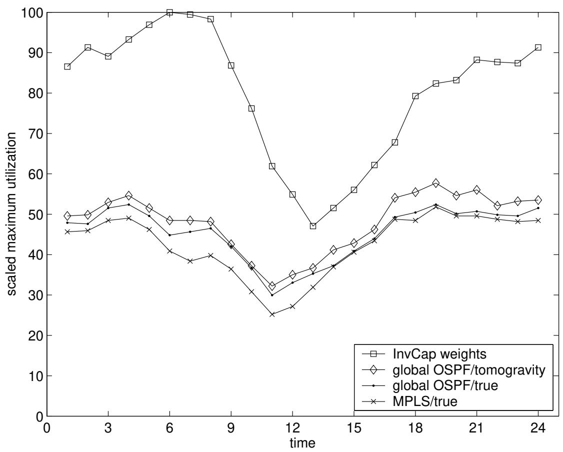


Figure 5: Global optimization over a 24 hour period. Once again the crosses represent the optimal solution for each one hour data set, and the squares represent inverse capacity weights. The points, and diamonds represent the global optimum weights calculated from the set of real and estimated traffic matrices, respectively, applied to each one hour traffic matrix.

The figure also gives the results optimized over the true traffic matrices for the 24 hours, and the hourly optimum. The results for the estimated and true traffic matrices are good, even in comparison to the hourly optimum. Comparing to the latter is unfair in this case because it is easier to optimize for a single hour than over the whole day. We use them instead of the global optimum MPLS solution because the computational cost for the global optimum MPLS solution was prohibitive (it involves a linear program with order 30 million variables and 50 million constraints). The OSPF optimization algorithm, on the other hand, is still quite fast. It is considerably slower when optimizing over 24 traffic matrices, taking around 6500 seconds to run 5000 iterations, but this is in fact better than 24 times the speed of the individual hourly optimization, which actually reduces the overall computation for the day.

Table 2 summarizes the results, and provides a comparison to those in the previous section. We can see that the global solution is very close to the individual solution in performance.

optimization method	traffic matrix	performance (%) AMU	MMU
Global OSPF	tomogravity	48.1	57.7
Global OSPF	true	45.4	52.4
Adaptive MPLS	true	42.5	51.8

Table 2: The AMU and MMU of the global routing optimization over one day.

6.3 Predictive Optimization

As noted in §6.2 it is highly desirable to change link weights infrequently. In the previous section we showed that we could derive a set of weights which work well for an entire day's set of traffic matrices. The question arises "how often need one change these weights?" Clearly, the answer is no more than once per day. However, if one wishes to change them even less frequently, we must be concerned with how well today's set of weights will perform in the future, that is, the predictive strength of the optimized weights.

This is a much harder problem than we have considered so far because any large, operational network changes continuously: new links are added, and old links re-homed, or retired. Note that in our data-sets we obtain routing and topology information at 24 hour intervals, so for the purposes of the previous examples the network appeared unchanged over the 24 hour period, although in fact it may have changed. Hence, even for 24 hour global optimization the estimated weights must be robust against network changes.

Obviously, in a full system for optimizing network routing there would be careful consideration given to weights for new links, and some kind of readjustment when the topology of the network is otherwise changed. However, we shall consider the very simple case where we assign maximal possible weights to new links (cost them out), which results in routing only over the previously existing network. This is obviously sub-optimal, but as we shall show, the weight optimization algorithm still performs well.

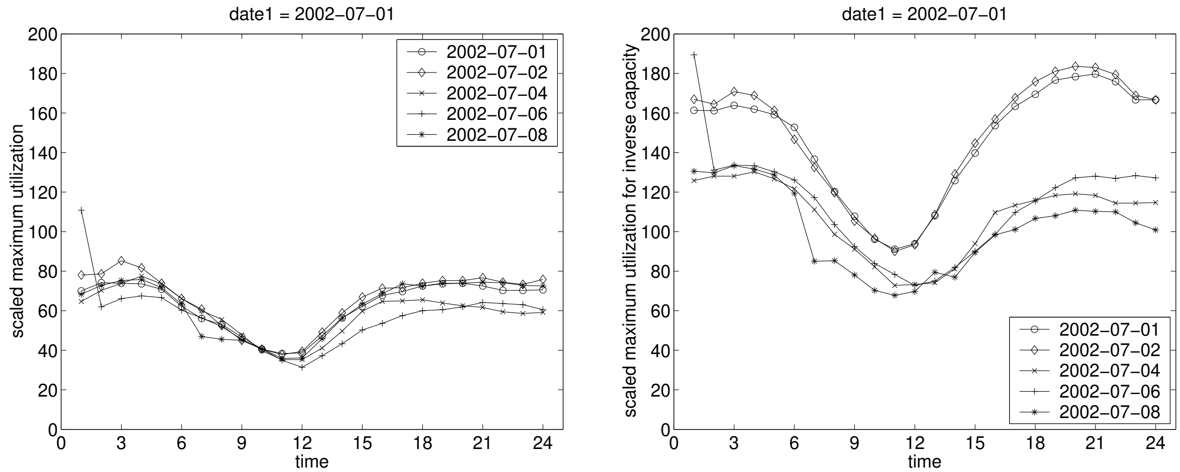
Figure 6 (a) shows the result of applying the weights derived from global optimization of the estimated traffic matrices from the 1st of July, 2002, to the real traffic on a series of days from the 1st to the 8th of July. We chose this period because there were no backbone network changes until the 8th, when a large link change occurred. For comparison Figure 6 (b) shows the result of inverse capacity weighting using the same scale for the y-axis.

The results show that the method consistently and significantly outperforms inverse capacity weighting, despite using weights optimized for a different day. This is despite the changes in the traffic matrices, which typically exhibit strong weekly variations as well as daily variations. Most notably, some sudden change in traffic behavior occurred between midnight and 1am on Saturday the 7th, causing high maximum utilization under inverse capacity weighting (the first cross in Figure 6 (b)). Despite such a sudden change, the method works very well, reducing the maximum utilization by nearly 50%.

The results also show that the inverse capacity weighting performance varies considerably during the week. As we know the network does not change during the period before the 8th, these changes must be the result of changes in the traffic matrix. For instance, because of the weekly cycle in the traffic, or potentially because of changes in the the Border Gateway Protocol (BGP) policy, which controls routing along a large part of the edge of the network, and hence where traffic departs the network.

The stability of the weight setting chosen using the estimated traffic matrices in the face of such changes in the traffic is a profound benefit of this approach.

These results shows that the results above are not just theoretical possibilities, but provide a robust and stable solution to a real traffic engineering problem. The solution is based on the traffic matrices estimated from link data (such as one would get from SNMP link measurement), and is practical both in computational load, and robustness to typical sources of variation found in real networks.



(a) OSPF optimization. (b) InvCap weights.

Figure 6: Results based on tomogravity estimates and OSPF optimization on the 1st of July, 2002. The results show the maximum utilization for the true traffic matrix on each of five days, over the course of a week, using the weights from the first day. Figure (a) shows the OSPF optimization weights, and (b) shows the InvCap weights — note that both figures have the same y-axis scale.

7. REFLECTIONS

In this section we reflect over limitations in the results presented, addressing various natural points of criticism.

7.1 Max-utilization

As mentioned in the introduction, we picked max-utilization traffic engineering objective because it is easy to understand and appreciate, not because it is the best or most important traffic engineering objective. Indeed, there is no single most important objective. Focusing on max-utilization, we have produced some very promising results, and we hope that this will inspire others to perform similar studies for other important traffic engineering objectives.

7.2 Proprietary scaling

For proprietary reasons, we have scaled all max-utilization by a secret factor so as to hide the true max-utilization in our network. Obviously, the absolute value is interesting. For example, a typical operational requirement is that the max-utilization is below 60%. Hence, improving the MMU as in Table 1 from 100% with inverse capacity to 57.7% is really important. However, if the scaling was a factor 2, then the corresponding improvement from 50% to 28.8% is irrelevant. Nevertheless, traffic grows over time and ability to improve max-utilization means that we can accommodate far more growth before we have to invest in new equipment.

7.3 The InvCap bench-mark

There are several reasons why we have used InvCap as a benchmark. First, it is a vendor recommended default, hence a natural starting point for traffic engineering. Second, in [7], InvCap was found to be a good default in the sense being as good or better then other simple heuristics like unit weights, or link weights proportional to physical lengths.

More interestingly, [12] found that InvCap weights were as good as a set of weights optimized for one traffic matrix $\mathtt{t}_1$ and applied to an independently generated traffic matrix $\mathtt{t}_2$. The test tells us that using a clue-less weight setting is at least as bad as the InvCap default. Hence, improvements over InvCap indicate how much better our routing is compared to a clue-less one.

As a matter of fact, with no clue about the traffic, we are not aware of any general technique performing better than InvCap in practice, not even with general MPLS routing. Recently [24], there has been interesting theoretical work on oblivious routing that gives max-utilizations within a factor $O(\log^3 n)$ from optimality for all possible demand matrix. This is very surprising and impressive theoretical result, but in practice, for realistic IP networks and demand matrices, it appears that InvCap gets within a factor 2 from optimality, which is much better. The difference is, of course, that the theoretical result has to take all the most pathological networks and demand matrices into account. Recently in [25], oblivious routing has been tested on some concrete networks, getting within a factor 2 from optimality for all possible demand matrices. This is again theoretically interesting, but the method is too slow to deal with networks with more than 40 nodes. Moreover, an MPLS implementation of the solution would require a cubic number of labels, which is prohibitive for large networks. For contrast, this paper is focused on techniques that work in practice for large ISP networks. Indeed the methods are already in use in a decision support system for the human network operators responsible for the network configuration.

7.4 The data sets

There are two major problems in the experiments reported. First, it is unfortunate that our basic data are proprietary so that other researchers cannot reproduce our results and continue the work with these data. Also, all our data are from the same network. We did perform many experiments with this network. For example, there is more than a month and many topological changes between the experiments reported in §5 and those reported in §6. Nevertheless we would like to see realistic experiments from other networks. Unfortunately, for real networks, most of these data are proprietary.

From [26] we do have good public estimates of the basic topology of many major IP networks, except that link capacities are not so easy to obtain. Also, we have established models for generating synthetic topologies like those in [27].

The main problem, however, is to get good traffic matrices. Real networks are highly tuned to the expected traffic so one cannot just generate the traffic matrix independently. We simply don't know of any good method for generating demand matrices to go with the topologies from [26]. Simple minded ideas like saying that high degree routers originate a lot of traffic don't work, for these could just be hubs in the middle of nowhere.

In [7, 12], a noisy gravity based model from [28] is used to generate synthetic traffic matrices to go with the synthetic topologies from [27]. It is easy to test our techniques on these data, but there is a major caveat; namely that when we use a gravity based method for generating the traffic, then the simple gravity estimation will be very accurate. In fact, we get very similar results to those reported in [12] for noisy data. The answer is that for these kinds of synthetic data, simple gravity gets within a few percent of optimality, which is much better than what we reported for our real network.

The above illustrates an inherent difficulty in generating synthetic demand matrices for our kind of experiment. If we know the generator, we cheat if we exploit that in our estimation, but we also cheat ourselves if our estimator does not exploit the generation. The basic point is that an estimator is supposed to exploit what we think is a good model for how traffic arises, and this thinking is difficult to model.

One could, of course, argue that our data sets are too thin, and that research in traffic engineering should wait until someone has found a way of getting better data. Finding better data is clearly a very important problem of independent interest, but why wait? Even though our data are not conclusive, they are promising enough to be of interest for other ISPs. As they get applied, we will get a better understanding of how they work in the real world. Also, as better public data emerges, it is trivial to test them in our framework. We just have to rerun the programs.

8. CONCLUSION

We set out in this paper to provide a genuine measure for assessing the practical accuracy of traffic matrix estimation. Simple metrics are unsatisfactory because one may form any number of them, and they may return different results, depending on what aspect of the traffic matrix is given importance. Hence we wished to provide a direct connection to a practical problem as a means of assessing the quality of the results. The means chosen was to test the routing optimization based on estimated traffic matrices when used with the true traffic matrix.

Experimenting with data from a large tier-1 ISP, we found something more, namely that the combination of tomogravity and OSPF weight optimization was a powerful and practical method for traffic engineering.

The result arose because the OSPF optimization method was quite robust to the types of errors found in the traffic matrix estimates. MPLS style optimization designed to obtain the very best possible routing was much less robust. Hence, even if MPLS is used, it makes sense to use a more robust method to determine the routes, for instance IGP routing.

The OSPF optimization method had other desirable properties, such as the ability to optimize weights for a range of traffic matrices (say over a day) and also to provide weights that worked a whole week into the future.

The other side of these results was the finding that the performance of the traffic matrix estimates was not a direct function of the magnitude of the errors in the traffic matrix estimates. This shows the importance of considering the combination of traffic estimation and route optimization before making any conclusions on how they will work together. Arbitrary error measurements are useful for superficial comparison, but do not tell the true story as far as practical applications goes.

In the future we wish to examine alternative optimization methods, and traffic matrix estimation algorithms, but given the quality of the results here, for the data considered, we do not see much hope for improved algorithms, only additional insight into the problem.

9. REFERENCES

[1] 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*, vol. 9, no. 3, pp. 265–279, 2001.

[2] 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. An earlier version appeared in SIGCOMM'00.

[3] 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 *Internet Measurement Workshop*, 2002.

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

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

[6] D. Mitra and K.G.Ramakrishnan, "A case study of multiservice, multipriority traffic engineering design for data networks," in *Proc. IEEE GLOBECOM*, pp. 1077–1083, 1999.

[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, pp. 519–528, 2000.

[8] K. Ramakrishnan and M. Rodriguez, "Optimal routing in shortest-path data networks," *Lucent Bell Labs Technical Journal*, vol. 6, no. 1, 2001.

[9] F. Lin and J. Wang, "Minimax open shortest path first routing algorithms in networks supporting the SMDS services," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 2, pp. 666–670, 1993.

[10] M. Ericsson, M. Resende, and P. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *J. Combinatorial Optimization*, vol. 6, no. 3, pp. 299–333, 2002.

[11] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup, "A memetic algorithms for OSPF routing," in *Proc. 6th INFORMS Telecom*, pp. 187–188, 2002.

[12] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE Journal on Selected Areas in Communications (Special Issue on Recent Advances on Fundamentals of Network Management)*, vol. 20, no. 4, pp. 756–767, 2002.

[13] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, New Jersey: Prentice Hall, 1993.

[14] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture." Network Working Group, Request for Comments, `http://search.ietf.org/rfc/rfc3031.txt`, 2001.

[15] J. T. Moy, "OSPF version 2." Network Working Group, Request for Comments: 2328, `http://search.ietf.org/rfc/rfc2328.txt`, April 1998.

[16] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments." Network Working Group, Request for Comments: 1195, `http://search.ietf.org/rfc/rfc1195.txt`, December 1990.

[17] Cisco, "Configuring OSPF," 2001. Documentation at `http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/ip_c/ipcprt2/1cdospf.htm`.

[18] E. H. L. Aarts and J. K. Lenstra, eds., *Local Search in Combinatorial Optimization*. Discrete Mathematics and Optimization, Chichester, England: Wiley-Interscience, 1997.

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

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

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

[22] A. Dwivedi and R. Wagner, "Traffic model for USA long-distance optimal network," in *Proc. Optical Fiber Communication Conference (OFC)*, pp. 156–158, 2000.

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

[24] H. Räcke, "Minimizing congestion in general networks," in *Proc. 43rd IEEE Symp. Foundations of Computer Science*, 2002., pp. 43–52.

[25] D. Applegate and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs," in *Proc. ACM SIGCOMM*, 2003.

[26] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, 2002.

[27] K. Calvert, M. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, June 1997.

[28] P. L. Toint. Transportation modeling and operations research: A fruitful connection. In M. Labbé, G. Laporte, K. Tanczos, and P. L. Toint, editors, *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, volume 166 of *NATO ASI series: Ser. F, Computer and systems sciences*, pages 1–27. Springer, 1998.