KTH VETENSKAP OCH KONST

KTH Electrical Engineering

Aspects of Proactive Traffic Engineering in IP Networks

ANDERS GUNNAR

Doctoral Thesis
Stockholm, Sweden 2011

TRITA-EE 2011:009
ISSN 1653-5146
ISBN 978-91-7415-870-0

KTH School of Electrical Engineering
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i telekommunikation tisdagen den 1 mars 2011 klockan 13.00 i sal F3, Kungl Tekniska högskolan, Lindstedtsvägen 26, Stockholm.

Tryck: Universitetsservice US AB

Swedish Institute of Computer Science

Swedish Institute of Computer Science, SE-164 29 Kista, SWEDEN
SICS Dissertation Series 54
ISRN SICS-D–54–SE
ISSN 1101-1335

Abstract

To deliver a reliable communication service over the Internet it is essential for the network operator to manage the traffic situation in the network. The traffic situation is controlled by the routing function which determines what path traffic follows from source to destination. Current practices for setting routing parameters in IP networks are designed to be simple to manage. This can lead to congestion in parts of the network while other parts of the network are far from fully utilized. In this thesis we explore issues related to optimization of the routing function to balance load in the network and efficiently deliver a reliable communication service to the users. The optimization takes into account not only the traffic situation under normal operational conditions, but also traffic situations that appear under a wide variety of circumstances deviating from the nominal case.

In order to balance load in the network knowledge of the traffic situations is needed. Consequently, in this thesis we investigate methods for efficient derivation of the traffic situation. The derivation is based on estimation of traffic demands from link load measurements. The advantage of using link load measurements is that they are easily obtained and consist of a limited amount of data that need to be processed. We evaluate and demonstrate how estimation based on link counts gives the operator a fast and accurate description of the traffic demands. For the evaluation we have access to a unique data set of complete traffic demands from an operational IP backbone.

However, to honor service level agreements at all times the variability of the traffic needs to be accounted for in the load balancing. In addition, optimization techniques are often sensitive to errors and variations in input data. Hence, when an optimized routing setting is subjected to real traffic demands in the network, performance often deviate from what can be anticipated from the optimization. Thus, we identify and model different traffic uncertainties and describe how the routing setting can be optimized, not only for a nominal case, but for a wide range of different traffic situations that might appear in the network.

Our results can be applied in MPLS enabled networks as well as in networks using link state routing protocols such as the widely used OSPF and IS-IS protocols. Only minor changes may be needed in current networks to implement our algorithms.

The contributions of this thesis is that we: demonstrate that it is possible to estimate the traffic matrix with acceptable precision, and we develop methods and models for common traffic uncertainties to account for these uncertainties in the optimization of the routing configuration. In addition, we identify important properties in the structure of the traffic to successfully balance uncertain and varying traffic demands.

Preface

When I started to write this thesis I quickly realized that it would be much longer than I first anticipated. It is indeed challenging to summarize years of research in a few pages. An academic thesis should address cutting edge research and is by definition not easily accessible to an average reader. However, I wanted to give readers not directly involved in this field a chance to understand the problems addressed. To this end I have added sections explaining the basics of how data is transferred over the Internet. Furthermore, I have included a short description of optimization. With this I want to convey why some optimization problems that appear straightforward are considered hard to solve, while other optimization problems that appear to be complicated are surprisingly simple to solve.

I suggest readers of this thesis to be selective in their reading. The content of some sections are well known to some readers, and can be omitted. Other sections use concepts known to people with a working knowledge of research in networking or a similar discipline, but might not be understood by someone without this background. Nevertheless, it is my intention that every reader should be able to find something fruitful to read in this thesis.

Anders Gunnar
Stockholm, January 2011

Acknowledgments

First I would like to express my sincere gratitude to my advisor Mikael Johansson for his encouragement and support during my time as a PhD student. Also, I am grateful to Gunnar Karlsson, for believing in me and accepting me as a PhD student before Mikael could become my advisor. I am also grateful to my manager at SICS, Bengt Ahlgren, for providing me the opportunity to pursue a PhD as part of my employment.

I am grateful to Henrik Abrahamsson for being an inspiring colleague and a good friend. My gratitude also goes to Laura Feeney and Thiemo Voigt for proof reading various versions of this thesis. Many thanks to Adam Dunkels for generously providing me with the LaTeX code for his thesis. Also, I would like to thank Steve Uhlig for acting as an opponent at my licentiate seminar but also for helping me to gain access to traffic data from the GEANT network. Thanks to all members in the NETS lab: Mudassar Aslam, Christian Gehrmann, Björn Grönvall, Ian Marsh, Oliver Schwarz and Javier Ubillos for contributing to an inspiring research environment. I am grateful to Karin Karlsson Eklund, Pablo Soldati and Anneli Ström for their help during my visits at the Automatic Control group at KTH.

Over the years I have had the opportunity to meet many inspiring persons. I would like to express my gratitude to Thomas Telkamp for providing me with traffic data from Global Crossings's global IP backbone network. Thanks to Mattias Söderqvist for his excellent master thesis where he wrote some of the software used in this thesis. Thanks to: Malin Forsgren, Daniel Gillblad, Sverker Jansson, Vicki Knopf, Martin Nilsson, Rebecca Steinert and many more at SICS for interesting discussions about research and other topics.

Finally, I would like to thank my family. My two sons Albin and Arvid for being there and letting me think about other things than computer networks. My wife Jenny for her patience and support. But above all, for her love and understanding.

Contents

Part I

Thesis

Chapter 1

Introduction

1.1 Scope and motivation

Originally, the Internet was designed for sharing research results using simple services such as email and file transfer. However, as the Internet evolved and was adopted by other sectors of society, new applications began to emerge. Many of these applications, such as streamed audio or video and voice transfer require a high degree of support from the network and introduce new service requirements such as bounded delay and limited packet loss. In addition, commercial interests have been incorporated into the provisioning of Internet services. Competition between Internet Service Providers (ISP) makes it important to reduce the cost of managing the network and to optimize the use of resources in the network. To manage the traffic situation in an efficient and reliable manner creates many new challenges for ISPs. New ways to monitor the traffic situation, along with improved techniques for configuring the routing to better control the traffic load, are becoming critical for achieving operational goals.

The subject of this thesis is *traffic engineering*. However, there is no universally adopted definition of this term. The meaning we give to traffic engineering is the process of measuring and controlling the traffic in the network to avoid congestion and to fulfill the service level agreements ISPs make with their customers. This includes monitoring of the traffic in the network as well as calculation and setting of routing parameters. Furthermore, congestion control and fair sharing of available communication resources are instrumental for the control of the traffic situation. In a long term perspective, traffic engineering also includes strategic planning of network topology and dimensioning of link capacity.

A key component of traffic engineering is the configuration of the routing function. In order to find a suitable routing setting, a number of steps needs to be executed; see Figure 1.1. The first step is to collect the necessary information about the network topology and the current traffic situation. Most traffic engineering methods need as input a traffic matrix describing the demand between each pair

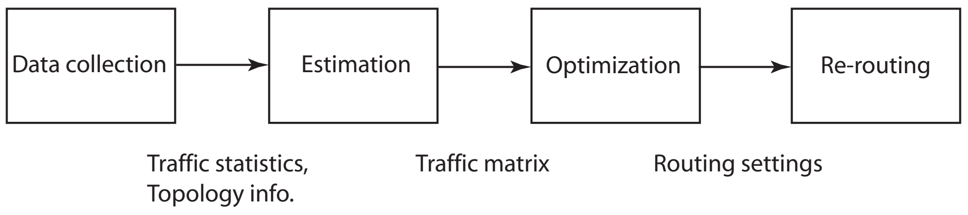


Figure 1.1: Traffic engineering

of nodes in the network. Obtaining the traffic matrix in a large IP backbone can be challenging since necessary measurement functionality is often not deployed in the network. Instead, the traffic matrix must be estimated from other available data. The traffic matrix together with network constraints such as network topology and link capacities are used as input to the optimization of the routing. The output from the optimization needs to be translated into parameter values of the routing protocol in use and distributed to the routers.

Internet traffic is often referred to as a "moving target" meaning that traffic volume and characteristics constantly change. To handle traffic variations we identify two approaches. Reactive traffic engineering solutions continuously monitor the state of the network and adapt the routing to handle changes in the traffic situation. This approach enables the network to handle unanticipated changes and to operate at an optimal (or at least favorable) point at all times. However, reactive traffic engineering requires close monitoring of the state of the network which imposes extra overhead. Hence, it is desirable to avoid frequent reconfigurations of network parameters to simplify network management. Proactive traffic engineering, on the other hand, aims to find static routing configurations that are able to cope with a large variety of traffic situations. The operation of the network is simple and controllable but performance may not be optimal in some situations.

In this thesis we address proactive traffic engineering and develop techniques for finding static routing configurations that can consistently maintain good network performance despite large traffic variations. We study efficient methods for estimating the traffic situation and demonstrate how estimation errors can be compensated for in the calculation of efficient routing settings. In addition, we describe algorithms to calculate routing settings that account not only for the current traffic situation but for a large number of possible traffic situations that can occur in the network. We identify sources of traffic uncertainties, develop mathematical models of the uncertainties and incorporate these models in an optimization problem. The outcome of the optimization is a routing setting that is able to absorb large variations in the traffic demands, i.e., a solution that is *robust* to traffic variations.

This thesis focuses on the management of the traffic situation within a network administered by a single organization, i.e., issues related to intradomain routing.

This assumption simplifies the proposed solutions since a single entity has control over involved parameters. For this reason we have excluded congestion control in this thesis since on the Internet this is handled from the end hosts of the connection which often reside in networks administered by different organizations. We focus on operational issues related to traffic fluctuations and assume network topology and link capacities are fixed. Hence, we omit problems related to network dimensioning and component failure.

1.2 Key contributions

This thesis addresses issues related to proactive traffic engineering in large IP backbone networks. We study efficient methods to determine the traffic situation as well as methods to optimize the routing function not only for normal operation, but for a variety of possible traffic situations that might occur in the network. Our evaluation of the proposed methods is performed on network topologies and traffic data from operational IP networks.

An important contribution of this thesis is the investigation of traffic matrix estimation techniques on real data. We evaluate a range of estimation methods for point-to-point traffic demands on a unique data set of measured traffic matrices. The data set consists of five minute measurements of each point-to-point traffic demand during a 24 hour period from a commercial Tier-1 IP backbone network. This allows us to do an accurate data analysis on the time-scale of typical link-load measurements and enables us to make a balanced evaluation of different traffic matrix estimation techniques. We explore some novel approaches to the problem and show that methods which rely on second order moments have poor performance due to slow convergence of the estimation of the covariances. The analysis indicates that regularized optimization from link load measurements gives an accurate estimation of the traffic situation.

Another important contribution is the development of routing optimization techniques that find proactive routing settings that are robust to the remaining traffic uncertainties. Although robust and proactive routing have been addressed before (e.g. [4, 71]), we present new models of traffic uncertainties that arise in many important networking problems. For instance, we demonstrate how traffic shifts caused by interdomain reroutes can be modeled and accounted for in the optimization. A particular novelty is the use of ellipsoidal uncertainty models, that are well tailored to stochastic estimation errors, and the development of associated robust routing optimization techniques with polynomial time complexity. Furthermore, in a stochastic setting it becomes clear that correlations in traffic demands play an important role for performance of load balancing under traffic uncertainty.

The robust optimization techniques result in routing settings that are implementable in MPLS-enabled networks. However, link state routing protocols are still widely used for intradomain routing in the Internet. Hence, we also study weight setting procedures for link state routing. We show that robust weight settings exist which have performance close to an optimal routing without the constraints imposed by link state routing.

1.3 Thesis outline

The rest of the thesis is organized as follows: Chapter 2 gives a short introduction to the design principles of the Internet and the functionality in the network important to this thesis. Chapter 2 also gives a short introduction to the mathematical optimization techniques relevant to our work. Chapter 3 describes the research areas addressed while related work is presented in Chapter 4. The following chapter contains a summary of included papers together with a description of the contributions of the author of this thesis. Concluding remarks and future work are described in Chapter 6. Finally, the second part of the thesis collects the five papers that contain the technical contributions.

Chapter 2

Technical and Mathematical Preliminaries

This chapter introduces some background material for the problems addressed and the solution approaches presented in this thesis. First we give a short introduction to computer networking and a description of some of the measurement functionality available for Internet traffic. This chapter also gives a brief description about how routing is performed in the Internet. Finally, we give a short summary of optimization techniques for problems with continuous and discrete variables.

2.1 Inter-networking in brief

To connect computers together and have them communicate there must exist a common language shared among all computers in the network. This shared language is specified in protocols that describes how information sent between computers is interpreted and what actions should be taken based on this information. Data is sent in packets, where a part of the packet (the header) contains protocol information and the other (the payload) contains the actual data that should be communicated. Each packet belonging to a connection between its source and destination host is routed independently of other packets belonging to the same connection. This is often referred to as *packet switching*. In contrast, telephone networks traditionally use *circuit switching* where an explicit path is set up between source and destination before any information is exchanged. However, modern telephone networks rely increasingly on packet switching as well.

The language of the Internet is the *Internet Protocol* (IP). The most important elements of IP packets are the source and destination address of the packet. Every computer connected to the Internet has a unique 32-bit IP address. The address provides a uniform way of identifying hosts in the network. Routers, the entities that forward the traffic between source and destination, base their routing decisions on the destination address.

The name Internet is derived from the technical term internetwork: to connect multiple networks into one. Hence, the Internet is a network of networks, where a large number of networks, each with a limited number of end-hosts and limited geographical reach, are connected to provide global connectivity of billions of devices. These networks are administered by different and often competing organizations known as Internet Service Providers (ISP). Consequently, the Internet is partitioned into subnetworks called Autonomous Systems (AS).

The Internet protocol is designed to rely as little as possible on the functionality in the underlying transmission technology to facilitate connection of networks using different transmission technologies. Instead, most of the complexity needed for providing a reliable and easy to use communication service is placed at the end hosts. The computational resources in the end host can deal with the problems introduced with packet switching such as retransmission of lost/delayed data packets or adjusting the sending rate of the source to remedy congestion in the network. The design with a primitive core that just forwards data and complex end hosts that provide the additional functionality required, known as the *end-to-end* principle, is another fundamental difference to the design of telephone network. In the telephone network, complexity is placed in the network and the end terminals are kept simple. One argument for placing functionality in the end hosts is that a computer is equipped with memory and a CPU that can be programmed to handle errors in the transmission of data. Traditional telephones lacked this functionality. Furthermore, a simple core makes the cost of transmitting data over the Internet small compared to transmission of data over the telephone network.

To simplify design and isolate implementation changes, the Internet has adopted a layered design. Each layer has a specified interface and is responsible for a communication service. How the interfaces are implemented is hidden to other layers. As long as the interface is not altered, implementation changes in the layers are kept isolated inside the layer. These layers are often described as a stack. The Internet protocol stack is called the TCP/IP reference model after its two most well known protocols. Originally the TCP/IP reference model contained four layers but has evolved to include the physical layer as a fifth layer.

- **Application layer:** This layer contains information about the application at the end host that uses the network to communicate with other applications at other hosts in the network.

- **Transport layer:** The transport layer contains most of the complexity that is needed in order to communicate between two hosts over a connectionless network. This include congestion control, sequence control, flow control and resending of lost data.

- **Network layer:** The main task of the network layer is routing, i.e. forwarding traffic towards the destination, and maintaining the necessary information to perform this routing.

- **Data link layer:** In the data link layer, traffic is sent over a single hop towards the destination without errors using a noisy channel.
- **Physical layer:** The physical layer is concerned with sending bits over a communication channel. Design issues include coding of bitstreams and delimiters for data packets.

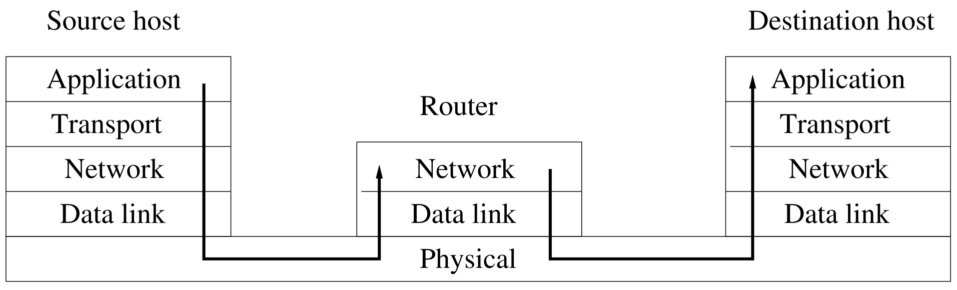


Figure 2.1: An example of how data is transmitted in the Internet

Figure 2.1 shows how data is sent from the source application program to the destination application. A packet leaving the sender host descends the protocol stack to the physical layer where it is transmitted to a neighboring router. Upon reception at an intermediate router the packet is propagated upward to the network layer where the router detects that it is not the destination of the packet. The destination address in the packet is used as a key in a routing table that keeps track of what outgoing link the packet should be forwarded on. The procedure is repeated hop by hop, until the packet reaches the destination where the packet propagates up to the application on the receiving host. A *port* and *protocol* number is included in the transport header to find the right application on the receiving host. When data is propagated downward in the protocol stack new headers are added. Header information in packets received from underlying layers is inspected and removed before packets are sent upwards in the protocol stack.

2.2 Measurement functionality for Internet traffic

There are basically two tools for measuring Internet traffic that are widely deployed in routers today. The most advanced is Cisco's flow measurement functionality Netflow (other router vendors offer similar functionality in their routers). Originally, Netflow was a pure flow measurement tool where flows are identified by source and destination addresses, protocol and port numbers. However for a backbone router, the number of flows in the flow table quickly grows to unmanageable proportions. Sampling is often used to reduce the computational burden. With sampling only a small fraction of the packets are selected for flow analysis.

Nevertheless, even with sampling the amount of data collected can be substantial, in particular if Netflow is enabled on every router in the network. Often the flow information needs to be processed with other information in order to derive the desired information about the traffic (e.g. [19]). With the introduction of version 9 of Netflow [50] it has become possible to use a much wider set of criteria for the definition of flows and much of the post-processing of flow measurements is no longer needed.

A more light-weight measurement functionality is link-counts provided by Simple Network Management Protocol (SNMP). The link-counts count the number of bytes sent on an outgoing interface in a router during a specified measurement period (often 5-15 minutes). Since there is one counter per link in the network the measurement information that needs to be sent over the network to the management station is much smaller than with Netflow measurements. However, since the information is aggregated at a much higher level than with flow measurements, the desired information about the traffic often has to estimated from the link-counts.

2.3 Internet routing

In order to be able to forward packets towards every destination host, routers need to maintain a large routing state. This is kept in the form of a routing table which contains network prefixes representing addresses to different networks together with a pointer to the interface that is used to forward packets to that destination. The use of prefixes enables aggregation of multiple addresses into one prefix leading to a large reduction of the routing state which needs to be maintained in routers. When a packet arrives at a router, the destination address in the packet is used as a key to find the longest prefix match in the routing table. The packet is forwarded on the interface associated with the matched prefix. The state in the routing table is maintained by the routing system. Since the Internet is partitioned into ASes, the routing is divided between *intradomain* routing inside an AS and *interdomain* routing between ASes.

Intradomain routing

The routing within an AS is managed by an Interior Gateway Protocol (IGP). Typically, the IGP is a link state routing protocol such as Intermediate System Intermediate System (IS-IS) or Open Shortest Path First (OSPF). Associated with each link is a weight reflecting the cost of sending traffic over the link. Routers announce topology information about which other routers they connect to in Link State Advertisements (LSA) and the weights of the associated links. LSAs are flooded in the network to allow each router to collect information about network topology and build a map of the network. The least cost path (shortest path in the given link metric) to each destination router in the network can be calculated using, e.g., Dijkstra's algorithm (cf. [33]). Figure 2.2 illustrates how paths are selected in link-state routing. The path from router A to C is A→E→D→C since this is the shortest path in the given link weights. In case of a router or link failure each router is

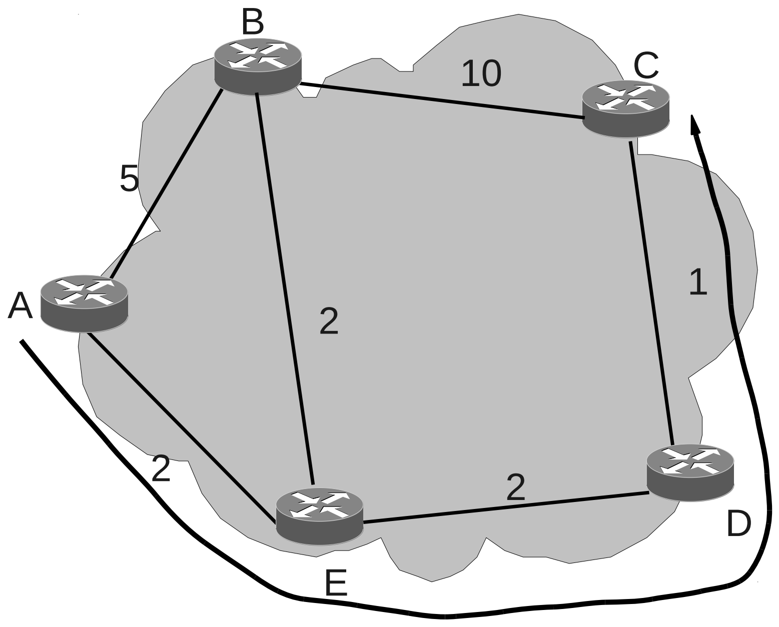


Figure 2.2: A small five node example network with link weights

able to calculate new routes using the link weights independently of other routers. Because of the shortest path principle the routing will be consistent and does not contain loops. In this thesis we refer to this type of routing as Shortest Path First (SPF) routing. A variant of shortest path routing is Equal Cost Multi-Path (ECMP). In ECMP traffic is split evenly over multiple paths with the same cost to destination. This technique offers a simple but blunt method to balance load over multiple paths. More details can be found in, e.g. [44].

Forwarding in SPF routing is based on the destination address only. All traffic from routers on the path from source to destination must follow the same path to the destination. This limits the possible routing paths that can be realized with SPF routing. However, more fine grained forwarding can be implemented with Multi Protocol Label Switching (MPLS). With MPLS Label Switch Paths (LSP) are set up between an ingress and egress node pair. The ingress router selects a label for an incoming packet based on some criterion such as destination, source/destination or traffic class. Packets following the same path are grouped in an Forwarding Equivalence Class (FEC). The packet is forwarded along the path based on the label until the packet reaches the egress router of the LSP where the label is removed. Since MPLS allows traffic to be forwarded arbitrarily in the network MPLS has loose restrictions on how paths are calculated. A commonly used approach is to use Constrained Shortest Path First (CSPF). In CSPF links in the network that do not meet a given criterion are removed from the routing calculations. The shortest paths are then calculated in the same manner as in Shortest Path Routing. More sophisticated routing can also be used in conjunction with MPLS. One powerful methodology for computing label-switched paths with certain optimality guarantees is to use Multi Commodity Network Flow (MCNF) optimization [3, 53]. The advantage with MCNF is that the resulting routing setting is optimal for a given objective but is more difficult to implement since traffic is often split between more than one MPLS path between ingress and egress routers.

Interdomain routing

To provide global connectivity, intradomain routing, operating within ASes, needs to be complemented by interdomain routing for connecting ASes and exchanges routing information. The current Interdomain routing protocol is called Border Gateway Protocol version 4 (BGP4) [26]. Usually ASes apply policies to the received routing information which reflect the business relation it has with the neighboring ASes. Business relations can be classified into customer, peer or provider. A customer AS pays a provider AS for connectivity to the rest of the Internet. However, ASes that exchange large amounts of traffic sometimes set up peering links to exchange traffic that originates in one AS and is destined to a network in the peering AS or one of its customer ASes. Today there is only a small group of ISPs that are not a customer of another ISP. This group of ISPs, known as *Tier-1* operators, peer with each other to obtain connectivity to the entire Internet. Figure 2.3 demonstrates different relationships between ISPs. At the top level in the picture are the Tier-1 ISPs that all peer with each other to gain connectivity information to the entire Internet. The second tier of operators (Tier-2) buy connectivity from Tier-1 operators but also sell connectivity to the entire Internet to other ISPs (Tier-3). Both Tier-2 and Tier-3 sometimes peer with each other to exchange traffic to reduce costs for traffic exchanged with provider ISPs. Furthermore, for resilience many ISPs buy connectivity from more than one provider. This is often called *multi-homing*. Multi-homing has implications on performance of the routing system. For instance, aggregation of network prefixes is aggravated leading to an increase of routing state. The BGP protocol is a path vector protocol where an AS announces to its neighboring ASes which networks it has a route to. In order to avoid routing loops the path of ASes of the prefix is included in the routing messages.

Due to multi-homing, routes to the same prefix are often available at multiple locations in an operators network. When an AS has more than one route to a prefix,

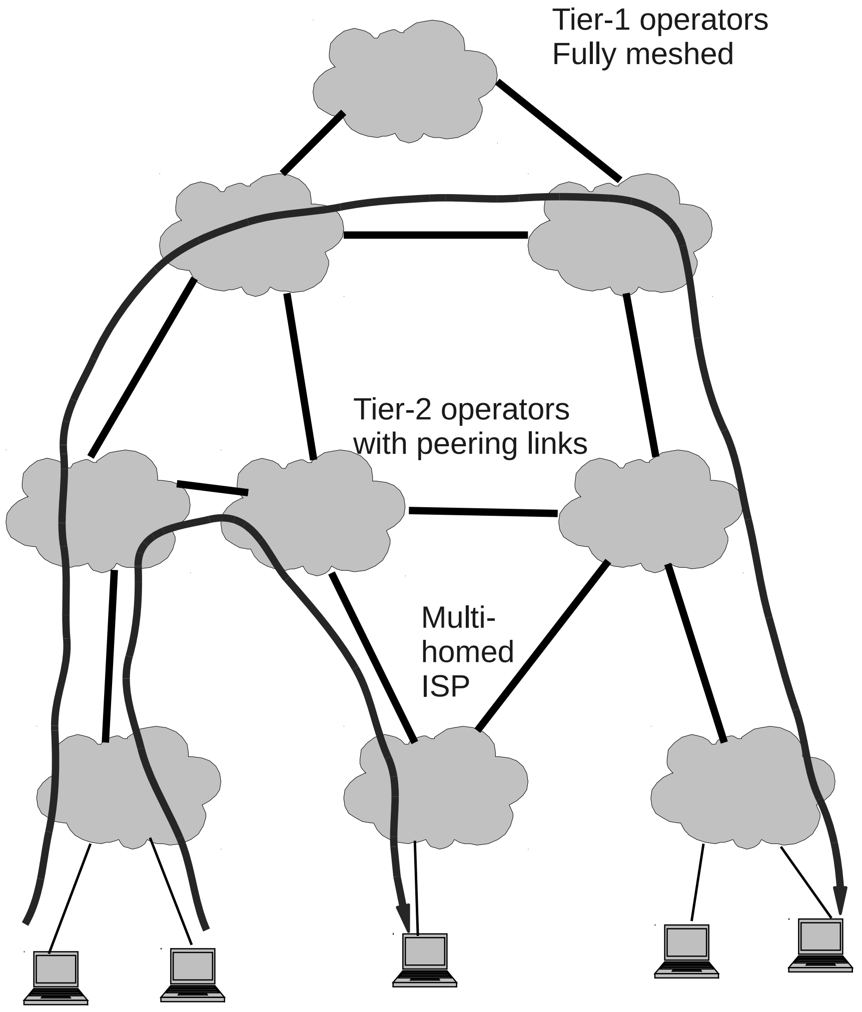


Figure 2.3: Business relations between ISPs and their implications on the paths taken by the traffic. Traffic on the shorter path use a peering link between two Tier-2 operators. Traffic on the longer path, on the other hand, has to propagate up to Tier-1 to reach the destination since peering ISPs do not transit traffic between peers

BGP has to select one route from the set of available routes as the preferred route. This is performed according to a decision process. The first step is to determine if there is a route to the egress point of the AS. Next BGP examines a number of BGP specific attributes. If BGP still is unable to select one route, the shortest distance according to IGP is considered. This is sometimes referred to as *hot-potato* routing [65]. Figure 2.4 illustrates this. Fluctuations in the routing caused by hot-potato routing are known to cause large traffic shifts in the network [63]. The final step is to use a vendor-specific tie-breaking. A detailed description of BGP4 can be found in [26].

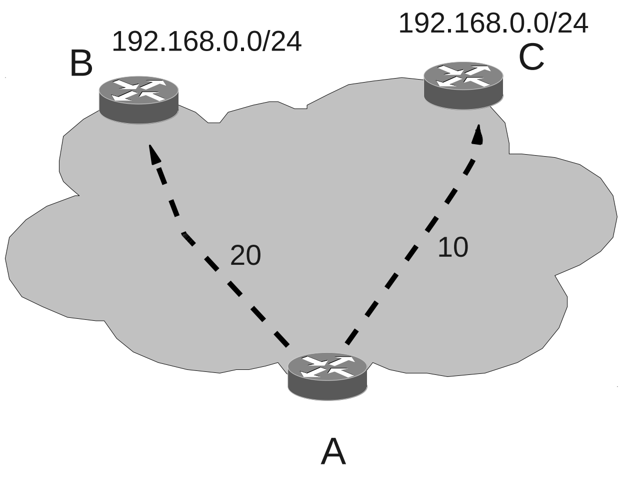


Figure 2.4: The network prefix 192.168.0.0/24 is announced by router B and C. Router A selects the route announced by router C since it is closest to A

2.4 Mathematical optimization techniques

Making the best possible use of available resources is essential for any engineering system. However, to optimize parameter settings has different meaning for different applications. For instance, a web server could be optimized for certain web-browsers, meaning that its performance has been tuned to provide the best possible experience for those browsers. In this thesis we use optimization in the context of *mathematical optimization*. A set of decision variables collected in a vector $x = [x_1, x_2, ..., x_n]^T$ is found such that a cost function is minimized/maximized under a set of constraints. The constraints express limitations in available resources or properties that must be present in the solution of the problem. The problem is formulated as follows:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, i = 1...m. \end{array} \tag{2.1}$$

Depending on the application, the decision variables in the vector x can either be continuous or only be allowed to take discrete values. Furthermore, the nature of the cost function and the constraints may or may not allow for efficient methods for finding an optimal solution. In the next sub-section we describe optimization problems in continuous variables followed by a brief introduction to optimization with discrete variables. The intention is to give the reader intuition about optimization and why some classes of optimization problems are tractable while other problems are more difficult to solve. Propositions are given without proof. Interested readers may confer references given in the text for further details.

Optimization in continuous variables

In general finding the optimal solution to Problem (2.1) is computationally intractable. However, for a class of optimization problems called *convex* problems there exist fast and accurate algorithms for solving large problems with tens of thousands or in some cases problems with hundreds of thousands of variables.

For many applications the set of feasible points determined from constraints form a convex set. A set $\mathcal{S}$ is said to be convex if every point in the line segment between two arbitrary points $x, y \in \mathcal{S}$ belongs to $\mathcal{S}$. This is illustrated in Figure 2.5 where the set to the left is a convex set and the set to the right is a non-convex set.

Related to convex sets are *convex functions*. A functions $f : \mathbb{R}^n \to \mathbb{R}$ is said to be convex on a convex set $\mathcal{S} \in \mathbb{R}^n$ if the following condition holds for every $x, y \in \mathcal{S}$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1+\theta)f(y) \tag{2.2}$$

for every $0 \leq \theta \leq 1$. Similarly, a function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *concave* on a convex set $\mathcal{S}$ if

$$f(\theta x + (1-\theta)y) \geq \theta f(x) + (1+\theta)f(y) \tag{2.3}$$

for $x, y \in \mathcal{S}$ and every $0 \leq \theta \leq 1$. Geometrically, these definitions mean that a line segment between the points x and y is either above the function f for convex functions or below f for concave functions. This is illustrated in Figure 2.6. It can easily be shown that if f is convex then $-f$ is concave and conversely, if f is concave then $-f$ is convex.

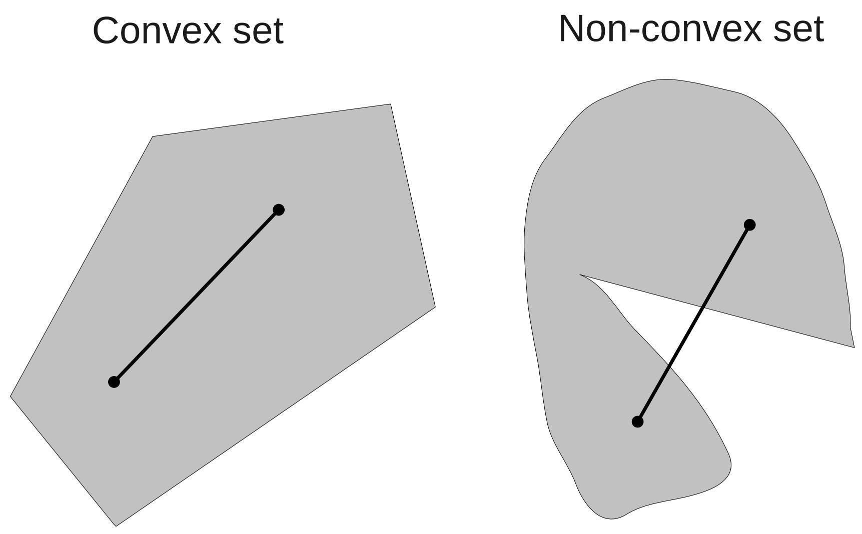


Figure 2.5: Examples of convex and non-convex sets

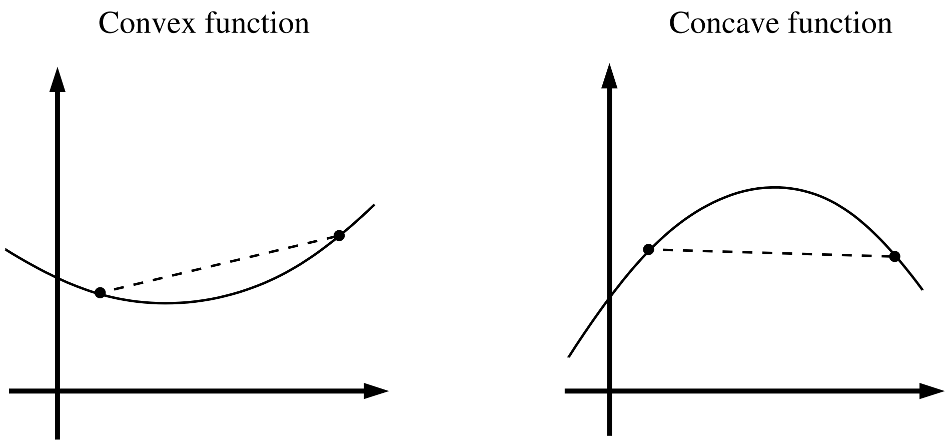


Figure 2.6: Examples of convex (left) and concave (right) functions

Using convex sets and convex functions we are ready to formulate a convex optimization problem as follows

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, i = 1, ..., m \\ & a_i^T x = b_i, i = 1, ..., p, \end{array} \tag{2.4}$$

where the functions $f_i, i = 0, ..., m$ are all convex functions. Note that the equality constraints are linear since these are the only equality constraints that yield a convex solution set.

The advantage that convexity brings to optimization is that a local minimum is also a global minimum. Hence, efficient search methods can readily be derived to find optimal solutions in polynomial time. Once a local minimum is found the search stops since this is also a global minimum.

Linear programming (LP), a sub-class of convex optimization problems has been studied since the 1940's. However, the theory for nonlinear problems made rapid progress in the 1980's and 1990's after publication of Karmarkar's groundbreaking paper [32]. Although, the theory has matured during recent years, many aspects of convex optimization are still open for active research. More details on the subject can be found in e.g. [7, 43, 45].

Optimization in discrete variables

Many applications only admit some decision variables to take values at discrete levels, typically integer values. These kind of problems are called Integer Programming (IP) problems when all decision variables are integer or Mixed Integer Programming (MIP) problems if a subset of the variables have integer restrictions. For example, consider the following single-link dimensioning problem. Assume that a demand of d Gbps should be served at the minimum cost. Three different link layer technologies are available represented by x_i, providing rate r_i at an investment cost of c_i. The problem of finding the most cost-effective investment that satisfies the demand can be written as

$$\begin{array}{ll} \text{minimize} & c_1x_1 + c_2x_2 + c_3x_3 \\ \text{subject to} & r_1x_1 + r_2x_2 + r_3x_3 \leq d \\ & x_i \in \{0, 1\}. \end{array} \tag{2.5}$$

Problems like this appear in network dimensioning, where the decision is both over the routing of the demand across the network as well as the investment decisions for all links.

One nice feature of the integer programming framework is that one can include additional logical constraints. Say, for example that technologies 1 and 2 are mutually exclusive. Then we impose the additional constraint that

$$x_1 + x_2 \leq 1.$$

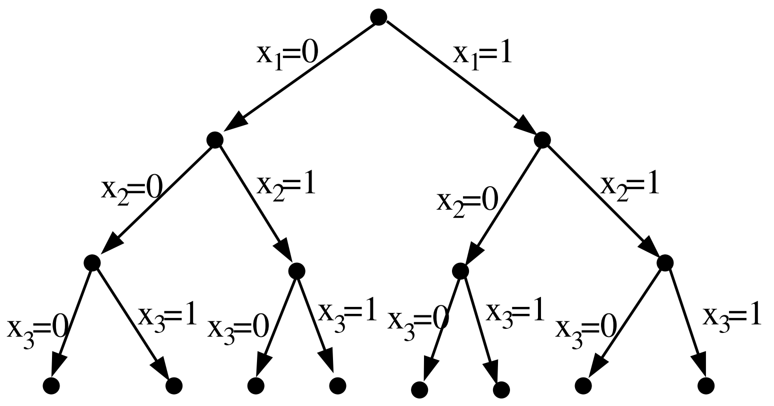


Figure 2.7: Example of an enumeration tree for three binary variables

The simplest way to solve a MIP problem is to enumerate all combinations of the variables, check which combinations satisfy the constraints and calculate the objective. This enumeration is readily represented as a tree where each possible combination of the binary variables are represented by the leafs of the tree. Figure 2.7 depicts an enumeration tree for a problem with three binary variables x_1, x_2 and x_3. However, the number of combinations grows exponentially and the computational burden quickly becomes prohibitive.

To avoid enumeration of all possibilities, branch-and-bound techniques proceed as follows: we start at the root of the enumeration tree by relaxing the binary restriction of the variables to allow them to take any value in the interval $[0, 1]$. The solution can be used as a lower bound on the optimal solution since the relaxed problem will always give a better solution than the original restricted problem. In addition to the relaxed problem there exists other ways to obtain a bound for the problem, e.g., Lagrangian relaxation of some of the constraints. If the bound is higher than the best bound previously examined then the node is closed from further expansion. We say that the node is bounded. Otherwise, more variables are set to zero and one to expand new children nodes which are examined. The quality of this approach depends on the bounds. To increase the quality of branch and bound, new constraints can be added to the relaxed problem to cut off parts of the solution space with suboptimal solutions. This however, increases the computational burden since the best bound needs to recalculated for each new constraint. Furthermore, performance is dependent on the order nodes in the search tree are examined. There are no standard methods that give acceptable performance for all problems. Instead intuition and previous experience from similar problems must be used to increase performance. In general MIPs are considered to be hard and only problems of moderate size (a few hundred variables) can be solved to optimality.

To solve large problems with discrete variables one often has to resort to a search heuristic. Search heuristics start from a valid solution and generate a sequence of new solutions that are evaluated based around the currently best estimate. If the new solution is better than the previous the new solution is used and the procedure is repeated until a stopping criterion is satisfied. To avoid the search getting caught in a local minimum worse solution are sometimes accepted. There are a number of well known search heuristics available including simulated annealing, tabu search and genetic algorithms to mention a few. Search heuristic give no guarantees for optimality of the solution but in many applications produce solutions with satisfactory performance.

More details on solution methods for optimization problems with discrete variables can be found in [43, 73]. A survey of different methods, including search heuristics, in connection to optimization in telecommunication can also be found in [53].

Chapter 3

Problem Areas

This chapter describes the problems addressed in this thesis. The text is intended to help the reader build intuition about the problems and solutions presented in the included papers. Details can be found in the papers in Part II of the thesis. We begin with basic modeling and notation for routing and traffic followed by some observations on Internet traffic characteristics.

3.1 Modeling networks and traffic

Representation of network traffic and topology

We represent the network topology with a graph where nodes represent routers, or groups of routers located in close proximity of each other, and edges represent communication links. The grouping of routers into a single node is motivated by the way that ISPs often organize their networks. ISPs typically group a number of routers in a point-of-presence (PoP) where customers connect their networks with the ISP network [47]. Typically, customer networks connect to an access router. The access router is connected to a high capacity backbone router within the PoP. For resilience, usually there is more than one backbone router in a PoP and the backbone routers are fully meshed within the PoP as shown in Figure 3.1. In addition, backbone routers are connected to other PoPs, usually in other cities, with high capacity links. Typically, ISPs have less than one hundred PoPs in their networks [59]. At a more detailed level the network can be studied at the router level. Then the size of the network grows since at this level the network may contain hundreds or even thousands of routers.

We let N be the set of nodes in the topology graph and E be the set of edges/links. To each link we associate a number c_l which describes the transmission capacity in bits/second of the link. The set of incoming and outgoing links from a node n is denoted $\mathcal{I}(n)$ and $\mathcal{O}(n)$, respectively.

A network with $|N|$ nodes has $P = |N|(|N| - 1)$ pair of distinct nodes that may communicate with each other. The aggregate communication rate between

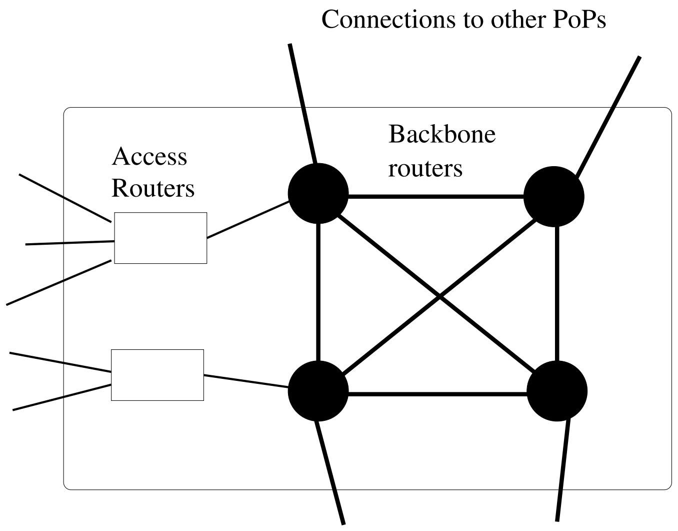


Figure 3.1: Example PoP with four fully meshed backbone routers and two access routers

any pair (s, d) of nodes is called the *point-to-point demand* between the nodes and is denoted by s_{sd}. The matrix $S = [s_{sd}]$ is called the *traffic matrix*. In many cases it is more convenient to represent the traffic matrix in vector form by enumerating all source-destination pairs, letting s_p denote the point-to-point demand of node pair p, and introducing $s = [s_p]$ to be the vector of demands for all source-destination pairs. We will use $o(p)$ and $d(p)$ to represent the origin and destination of source-destination pair p, respectively. The focus in this thesis is on PoP-to-PoP analysis of traffic. Although traffic can also be studied on the more detailed router-to-router level, or even link-to-link level [19], we will not consider such possibilities in this thesis.

Modeling routing

In its basic variation, SPF routes each source-destination flow on a single path as explained in Section 2.3. MPLS on the other hand, allows to define an arbitrary number of tunnels, each with a separate path, for each source-destination pair.

When modeling how traffic flows in the network, it is convenient to represent traffic volumes in terms of a traffic matrix. In SPF, the total traffic on link l is given by

$$t_l = \sum_p \rho_{lp} s_p$$

where ρ_{lp} is an indicator variable, taking the value 1 if traffic flow p is routed across link l and 0 otherwise. Letting $r_l = [\rho_{lp}] \in \mathbb{R}^P$ we can re-write this as

$$t_l = r_l^T s$$

and write the vector of traffic across links $t = [t_l]$ as

$$t = Rs. \tag{3.1}$$

Here, $R \in \mathbb{R}^{|E| \times P}$ is the routing matrix, whose columns indicate the links used to route traffic on a specific path.

For MPLS routing, and also for SPF with ECMP extension, traffic is balanced among multiple paths. We denote Π_p the set of paths between source destination pair p. Furthermore, we let $\alpha_{\pi p}$ represent the fraction of s_p sent over path π. All traffic is assigned to some path, i.e.

$$\sum_{\pi \in \Pi_p} \alpha_{\pi p} = 1.$$

This representation is readily linked with the routing matrix R by calculating

$$r_{lp} = \sum_{\pi \in \Pi_p} \rho_{l\pi} \alpha_{\pi p} \tag{3.2}$$

for each element in the matrix. The indicator variable $\rho_{l\pi}$ takes value one if link l is part of path π, and zero otherwise. By collecting the fractions connected to the source-destination pair p in a vector $\alpha_p \in \mathbb{R}^{|\Pi_p|}$ we define the block diagonal matrix

$$A = \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \alpha_p \end{pmatrix}$$

with $\sum |\Pi_p|$ rows and P columns. We let the matrix $\Pi \in \mathbb{R}^{|E|} \times \mathbb{R}^{\sum |\Pi_p|}$ be an indicator matrix with elements $[\Pi]_{l\pi} = \rho_{l\pi}$. Using this notation the routing matrix is related to the path notation by

$$R = \Pi A.$$

For MPLS routing, $\alpha_{\pi p}$ is in general a real number and r_{lp} is the fraction of source-destination traffic demand p routed over link l. With SPF routing on the other hand, $\alpha_{\pi p} \in \{0, 1\}$ and each column of the routing matrix has ones on the entries corresponding to the links in the single path between source and destination, and zeros on all other entries. Hence, for SPF routing we have $R = \Pi$, but for MPLS based routing these two matrices are in general different.

Traffic characteristics

Internet traffic has a rich variety of characteristics depending on location in the network and at what time scale the traffic is observed. For instance, Wide area network and Web traffic have been shown to possess self-similar properties (cf. [13, 51]). Basically, self-similarity means that traffic behavior is independent of the time scale the traffic is observed. If the traffic is bursty on the millisecond level it is bursty at the second level etc. However, for a network operator it is desirable to keep the routing stable in order to avoid oscillatory behavior of the traffic, minimize routing signaling and avoid instability in the routing system. Traffic engineering is preferably performed for a stable traffic situation.

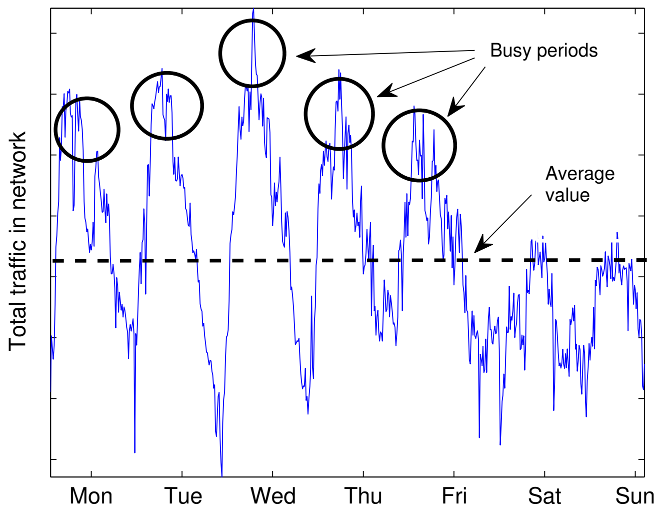


Figure 3.2: Total traffic sent in a large IP backbone for a seven day period (traffic normalized). Busy periods are illustrated with rings. Average value of all traffic is illustrated with a dashed line

Figure 3.2 shows total traffic in a large backbone during one week. A clear diurnal pattern appears in the plot but there also seems to be random fluctuations in the traffic. The randomness in traffic is different depending on the level of aggregation of the traffic. Traffic in a Tier-1 network usually displays a lower level of randomness than traffic in a local area network because of the higher level of aggregation in a Tier-1 network.

Classical traffic engineering methods use a single traffic matrix as input, but as we can observe in Figure 3.2 it is not obvious how to select this single traffic matrix. Using the average value of the traffic demands as shown in Figure 3.2 will potentially lead to overload for long periods of time. Alternatively, by identifying a busy period where traffic reaches its peak also face some difficulties. This will over-dimension the network since load is much lower for large periods of time. In addition, the busy hour demands might change over time. For instance, if we study a specific traffic demand as illustrated in Figure 3.3. The figure shows traffic intensity for a large source-destination demand in an IP backbone network during a three week period. The first two weeks the flow follows a stable diurnal pattern with regularly occurring peak values. However, at the beginning of the third week the flow suddenly becomes three times larger than before. This kind of disruptive behavior may cause overload but is not necessarily observable in the aggregated busy period of the network.

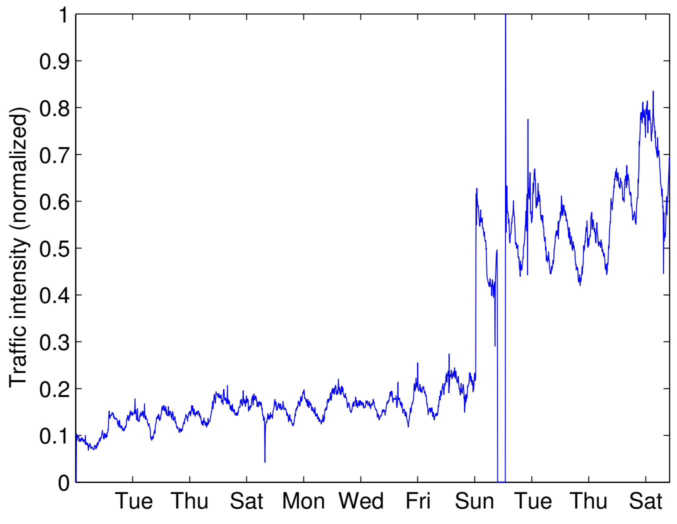


Figure 3.3: Sending rate for a large source-destination traffic demand during a three week period in a large IP backbone network

Deterministic traffic models

One of the simplest traffic models is the generalized *gravity model* [56, 76]. The model assumes that the traffic exchanged between nodes s and d is proportional to the total amount of traffic sent by s and total traffic received by d. The name gravity model refers to the fact that large senders and receivers are assumed to exchange large amounts of traffic similar to Newtons theory of gravitation where bodies of large mass exert a strong gravitational attraction on each other. We denote $t_{e(s)}$ the total amount of traffic injected in the network by source s and $t_{x(d)}$

the total amount of traffic received by destination d. In this notation, according to the gravity model the traffic between s and d is

$$\hat{s}_{sd}^{(p)} = C t_{e(s)} t_{x(d)} \tag{3.3}$$

where C is a normalization constant to make the sum of the traffic demands from the model consistent with the total measured traffic in the network. Point-to-point traffic demands obtained from the gravity model are denoted $\hat{s}_{sd}$ to indicate that it is an estimate of the true traffic demand and need not even be consistent with link load measurements obtained from SNMP. The estimate is rather crude since the gravity assumption tends to make the distribution of traffic uniform as indicated in the plot to the left in Figure 3.4. By comparing the two plots in Figure 3.4 where estimated and real traffic demands for the same network are shown we observe that the gravity estimate is not very accurate. To improve accuracy, information on business relations such as provider, customer and peering agreements can be added to the model [76].

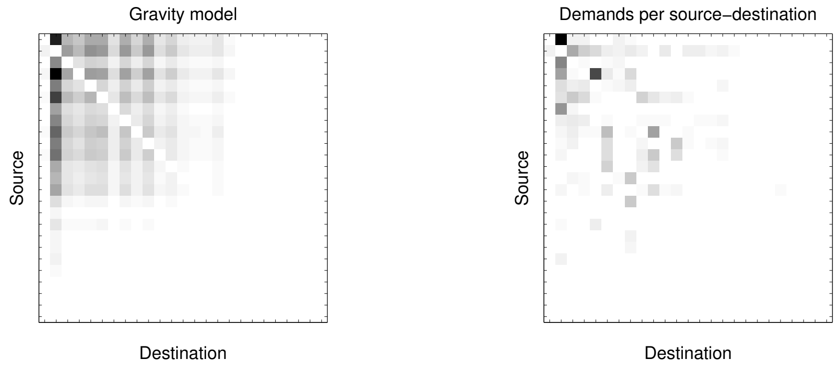


Figure 3.4: Spatial distribution of estimated traffic demands for gravity estimation (left) and real traffic demands (right) from a large IP backbone. Source nodes sorted in descending order for real traffic demands

Related to the gravity model is the *fanout model* [41, 42]. The fanout model can be seen as a probability distribution describing the probability that a packet injected in the network at s is destined to d. The fanout factors are expressed as

$$s_{sd} = \alpha_{sd} t_{e(s)} \qquad \sum_d \alpha_{sd} = 1 \tag{3.4}$$

determining the fraction of traffic injected at s destined to d. Note, that if the normalization constant C in (3.3) is set to

$$C = \frac{t_{x(d)}}{\sum_d t_{x(d)}}$$

the fanout model becomes identical to the gravity model.

These models are in general not very accurate to quantify the traffic demands. However, since the models are not based on link load measurements the gravity and the fanout models provide useful information as a prior guess of the traffic demands to estimation based on link load measurements. Section 3.2 describes the general ideas of point-to-point traffic demand estimation based on link load measurements. Paper A in this thesis presents details and evaluation of the estimation.

Stochastic traffic models

To capture the variability of the traffic during busy periods (or around the diurnal trend), it is natural to explore statistical models. Traffic demands are assumed to follow a given probability distribution, and parameters of the distribution such as mean and variance are adjusted to match the real data.

One of the simplest models is to assume that traffic demands follow a Poisson distribution, i.e., it is assumed that

$$s_p \sim \text{Poisson}(\lambda_p). \tag{3.5}$$

This model was suggested by Vardi [67] for point-to-point traffic demands. The estimation of the intensity λ_p from data is simplified by the fact that the mean and variance of the Poission distribution coincide. Although this distribution is widely used for e.g. telephone traffic it has been shown in a number of studies that traffic in the Internet is in general not Poissonian (cf. [25, 30, 51]).

Another tractable model is to assume that demands follow a normal (or Gaussian) distribution

$$s \sim \mathcal{N}(\lambda, \Sigma). \tag{3.6}$$

Here, λ is the vector of average traffic rates for the point-to-point demands while Σ is the corresponding covariance matrix. In the traffic estimation literature, it is often assumed that the mean and covariance are related by a scaling law (e.g. [8]),

$$\Sigma = \phi \text{diag}(\lambda^c) \tag{3.7}$$

where diag(λ) denotes a diagonal matrix whose value on the kth diagonal coincides with the kth component of the vector λ, and λ^c denotes that the elements of λ are raised to the cth power.The scaling law assumption makes the traffic demands statistically identifiable, but the associated estimation techniques are computationally more demanding than those for the Poisson assumption.

In Figure 3.5 we have plotted the relationship between mean and variance in logarithmic scale for traffic demands in a large IP network. The plot demonstrates a strong relationship between mean and variance of traffic demands and that the scaling law is a reasonable assumption. Furthermore, studies have shown that

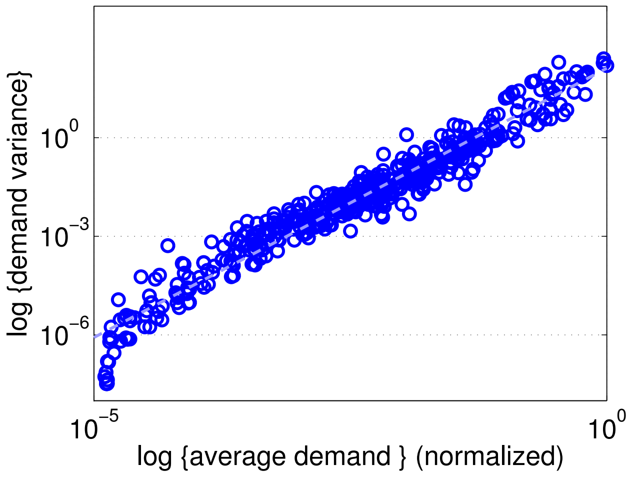


Figure 3.5: Relation between mean and variance of source destination traffic demands in an operational IP network measured every 15 minutes during a three week period

the Gaussian model formulated in (3.7) captures the behavioral of backbone traffic with good accuracy [25, 30].

Papers A and D in this thesis use statistical models for point-to-point traffic demands and study validity and implications of these assumptions.

Robust traffic models

As we have seen, it is often difficult to find a single traffic matrix that represents the traffic over time. This is especially true when large traffic shifts tend to occur. In these cases, it is more suitable to use multiple scenarios, or a worst-case model of traffic that does not specify a single traffic matrix but a full set of matrices to which the true traffic situation is guaranteed to belong.

Figure 3.6 represents one class of worst-case traffic models. Here, a set $\mathcal{S}$ is formed as the convex hull of a number of traffic scenarios. These traffic scenarios, which could for example be a time-series of measured traffic matrices, form the vertexes (extreme points) of the set. Formally, the set S is described as

$$\mathcal{S} = \{s = \sum_{v=1}^{V} \theta^{(v)} s^{(v)} \mid \theta^{(v)} \geq 0, \sum_{v=1}^{V} \theta^{(v)} = 1\} \tag{3.8}$$

where V is the number of extreme points. This is an efficient representation of a large set of traffic situations since all traffic scenarios inside $\mathcal{S}$ are accounted for

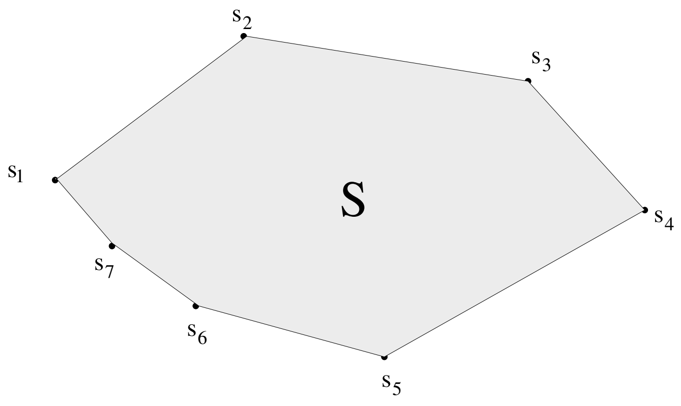


Figure 3.6: Convex hull of seven extreme points

and not only the extreme points. In many cases the traffic is not given as a set of extreme points but as a solution set of a number of intersecting half-spaces

$$\mathcal{S} = \left\{ s \mid a_i^T s \leq b_i, i = 1, \ldots, m \right\}. \tag{3.9}$$

In principle it is possible to calculate the extreme points of the polytope in (3.9) to derive the representation of (3.8). However, this is computationally demanding since the number of extreme-points grows exponentially with the dimension of the data set. Papers C and D discuss different traffic uncertainties occurring in IP networks that can be described by polyhedrons.

In some cases it is more natural to represent the traffic scenarios by an ellipsoidal model. An ellipsoid can be described by

$$\mathcal{S} = \{ s \mid (s - \lambda)^T M^{-1} (s - \lambda) \leq 1 \} \tag{3.10}$$

where M is a positive definite matrix. For instance, by using the concept of *likelihood regions* it is possible to quantify the most likely outcomes of a probability distribution. In particular, when traffic demands are assumed to follow the Gaussian distribution (3.6) the likelihood regions assume the shape of ellipsoids

$$\mathcal{S}_\alpha = \{ s \mid (s - \lambda)^T \Sigma^{-1} (s - \lambda) \leq \alpha^2 \}.$$

If s are samples from (3.6) it can be shown that the quantity

$$\alpha^2 = (s - \lambda)^T \Sigma^{-1} (s - \lambda)$$

follows the Chi-square distribution with P degrees of freedom. Thus, by setting $\alpha^2 = \chi^2(1 - \gamma, P)$, i.e., the upper $(100\gamma)\%$-point of the Chi-square distribution with P degrees of freedom we are able to form an $(100\gamma)\%$ confidence region for s. More details on confidence regions can be found in [11]. Details on ellipsoidal traffic models can be found in Paper D.

3.2 Traffic measurements and estimation

An estimate of the traffic situation is a prerequisite for optimizing the routing function. However, deriving point-to-point traffic demands in a large IP backbone can be a challenging task. One option is to use Cisco's flow measurement facility Netflow to collect flow records on each router. However, since the Internet is a connection less network the flow records need to be processed together with routing information data to derive the point-to-point traffic demands. For this purpose the flow records are sent to a central processing station, see Feldmann *et al.* [19] for details. The amount of state in terms of flow records and computational burden connected to measurements and processing with routing data make operators reluctant to use this method in large scale on a regular basis.

An alternative is to estimate the traffic matrix from link load measurements obtained from Simple Network Management Protocol (SNMP). However, since point-to-point traffic demands are not directly available from link loads we need to establish a connection between the measured link loads and the unknown traffic demands. The connection is the routing configuration encoded in the routing matrix R and the link-load relation $t = Rs$ as described in Section 3.1.

The *traffic matrix estimation problem* is simply the one of estimating the non-negative vector s based on the relation $t = Rs$ and knowledge about R and t. The challenge in this problem comes from the fact that this system of equations tends to be highly underdetermined: there are typically many more source-destination pairs ($\mathcal{O}(|N|^2)$) than links in a network ($\mathcal{O}(|N|)$), and (3.1) has many more unknowns than equations. The traffic demands are uniquely determined in rare instances only. One such example is when the network is fully meshed and traffic is routed on the single-hop path connecting the communicating node pair. In general, however, networks are far from fully meshed. Since the number of links tends to grow linearly while the number of node pairs grows quadratically, the traffic estimation problem becomes even more under-constrained as the size of the network grows.

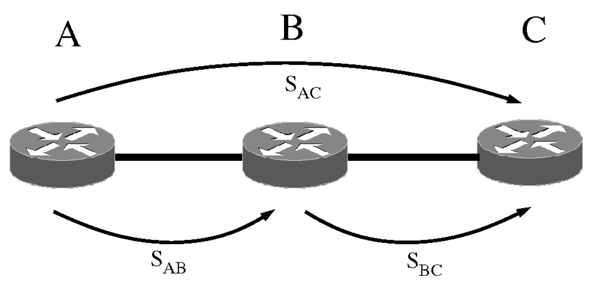


Figure 3.7: A simple network with three nodes and three traffic demands

Figure 3.7 illustrates the difficulties in the traffic matrix estimation problem with a simple example. The figure shows a small network with three nodes and three source-destination traffic demands. From the picture it is clear that looking at the link loads alone, it is impossible to observe an increase in s_{AC} if s_{AB} and s_{BC} decrease at the same time. For the example, the link load equation (3.1) becomes

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} s_{AB} \\ s_{AC} \\ s_{BC} \end{pmatrix} = \begin{pmatrix} t_{AB} \\ t_{BC} \end{pmatrix}.$$

The rows in the routing matrix describe the flows routed across link (A,B) and link (B,C), respectively. The columns represent the paths π_{AB}, π_{AC} and π_{BC}. The rank of the routing matrix is less than the number of unknown traffic demands and the null space of the matrix is spanned by the vector $(1, -1, 1)^T$.

To make the estimation problem well-posed, more information about the traffic must be added. This can be a prior guess $s^{(p)}$ of the traffic situation, or a model of the traffic (e.g. that the traffic matrix is a sample from a given probability distribution). One could then try to find the traffic matrix closest to the prior guess that explains the observed link loads as illustrated in Figure 3.8. This can be formulated as the optimization problem

$$\begin{array}{ll} \text{minimize} & D(\hat{s}, s^{(p)}) \\ \text{subject to} & R\hat{s} = t \\ & \hat{s} \succeq 0 \end{array} \tag{3.11}$$

where $\hat{s}$ denotes an estimate of s and $D(\hat{s}, s^{(p)})$ the distance (in an appropriate measure) between $\hat{s}$ and $s^{(p)}$.

In many cases, however, it makes sense to sacrifice some accuracy in explaining the link loads in order to have a better match with the prior guess. One then solves the problem

$$\begin{array}{ll} \text{minimize} & D(\hat{s}, s^{(p)}) + \epsilon \|R\hat{s} - t\| \\ \text{subject to} & \hat{s} \succeq 0 \end{array} \tag{3.12}$$

This formulation is sometimes referred to as *regularization* (cf. [7]). The non-negative weight ϵ is called the regularization parameter, and allows to emphasize good reconstruction of the observed link loads or good accordance with the prior guess. One advantage with this approach is that it allows for inconsistent values in the vector of observed link loads. Inconsistent measurements do occur in practice, for example when some of the measurement data is lost during transmission or when different measurement points are poorly synchronized.

For this formulation, the traffic matrix estimation problem now breaks down to picking the prior guess $s^{(p)}$, the appropriate distance measure $D(\cdot, \cdot)$, and the regularization parameter ϵ. Many traffic matrix estimation algorithms can be seen as variations of the basic regularization approaches. This includes the celebrated

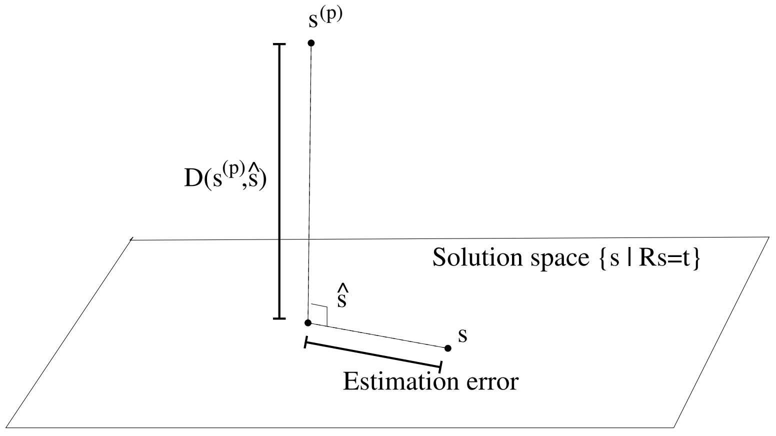


Figure 3.8: The relation between prior guess and estimated traffic demands and real traffic demands

tomogravity approach [77] where the gravity model is used to determine the prior and the Kullback-Leibler divergence is used as distance measure. Also the estimation procedure due to Vardi [67] is related. Paper A details these links.

Even if the traffic demands are fluctuating over time it is sometimes assumed that that the fanout factors (3.4) remain constant. From this assumption it is possible to deduce a slightly different approach to the estimation of the traffic demands. Given a time series of K link load measurements the link load equation (3.1) assume the form

$$RS[k]\alpha = t[k], \qquad\qquad k = 1, \ldots, K,$$

where $S[k]$ is a diagonal scaling matrix such that $s[k] = S[k]\alpha[k]$. Although R does not have full rank, as K grows the system of equations quickly become overdetermined. The fanout factors can be found by solving the quadratic (and hence convex) optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{K} \|RS[k]\alpha - t[k]\|_2^2 \\ \text{subject to} & \sum_{d=1}^{N} \alpha_{sd} = 1, \qquad s = 1, \ldots, |N|. \end{array}$$

Figure 3.9 shows fluctuations of the four largest outgoing point-to-point traffic demands from the four largest sender nodes in a large IP backbone during a seven day period. Figure 3.10 shows their corresponding fanout factors. We observe that even though fanout factors display a smaller amount of variability than their corresponding traffic demands. For many demands the variability is still substantial making estimation based on stability of fanout factors difficult.

In Paper A in this thesis we evaluate a wide selection of regularized methods as well as estimation based on fanout factors. For the evaluation we have access to

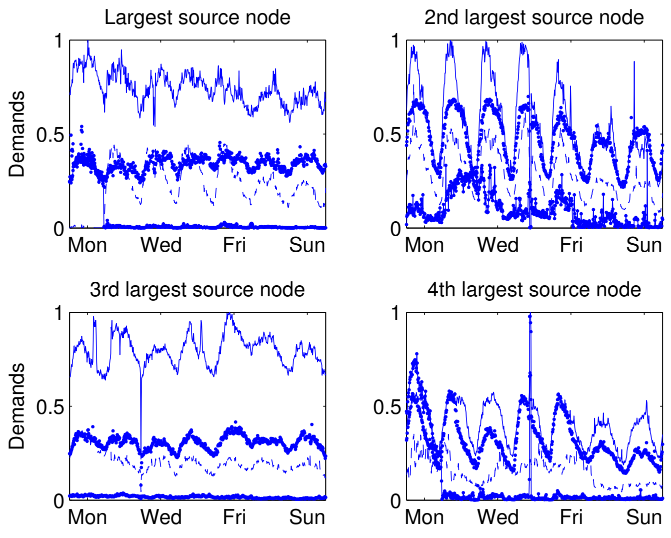


Figure 3.9: The four largest outgoing traffic demands from the four largest sources in a large IP backbone

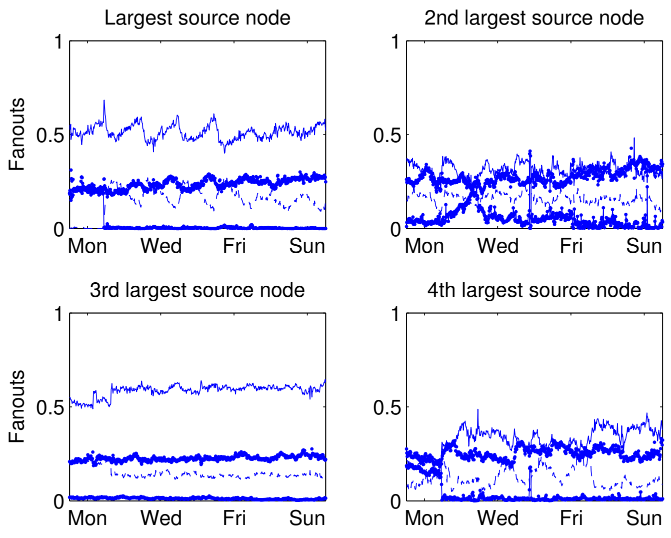


Figure 3.10: The associated fanout factors for the four largest sources in a large IP backbone

a unique data set of complete traffic matrices from an operational Tier-1 IP backbone. Furthermore, applications of estimated traffic matrices are studied in Papers B and C.

3.3 Proactive traffic engineering

Most methods for optimizing routing settings assume that the traffic matrix is given. However, as we have seen in Section 3.1, it can be hard to determine which conditions to optimize routing for, and it can also be hard to estimate the traffic matrix at a given point in time based on available data. Thus, the traffic matrix that we give as input to the optimization routine is almost certainly different than the actual traffic situation. Since most optimization techniques are sensitive to variations in input data, and suffer from severe performance degradations when this data is inaccurate, it is important to look for techniques that are robust to typical modeling errors and traffic shifts.

In general terms, a system can be said to be *robust* if it is able to gracefully handle variations from normal operating conditions. In a networking context this entails the ability to sustain acceptable performance despite foreseeable traffic variations and component failures. To realize this, we have identified two approaches in Section 1.1. However, our focus is on proactive traffic engineering. Since the Internet is a network of networks it is often difficult to accurately predict the outcome of a change in the routing system in one network. A change can lead to unanticipated changes in traffic patterns in the network where the changes take place as well as in adjacent networks. Hence, network operators are reluctant to change the routing configuration too often. Thus, a proactive approach is better in-line with current practice in network management.

Performance metrics for traffic engineering

To optimize the routing setting, we also need a performance measure with which we can measure and compare different settings. Delay is used by e.g. Gallager [24] as performance metric for a distributed algorithm to minimize the sum of delay on the links in the network. Fortz and Thorup [21] use a piecewise linear cost function which attempts to resemble the delay on a link. Here the cost is low until utilization approaches full link utilization where the cost increases rapidly as delay increases when packets are queued before being sent on the link. In this thesis, we primarily use maximum link utilization as performance metric. Formally, the maximum link utilization is defined as follows

$$u_{\max} = \max_{l \in E} \{c^{-1} r_l^T s\} \tag{3.13}$$

and the performance objective is to minimize $u_{\max}$. This performance metric is widely used in the research literature and it is easy to comprehend and analyze. However, it also has some drawbacks. For instance, it focuses only on the most loaded links, and decreasing the traffic on the most loaded link, traffic is typically rerouted along alternative, often longer, paths that increase the total delay for many flows. One way to alleviate such effects is to also include a small penalty on delay. However, for sake of clarity, we will omit such considerations in this thesis.

Proactive traffic engineering for MPLS networks

When routing is not confined to single paths, routing optimization is typically performed using network flow optimization techniques. There are many formulations of the basic network flow problem (see, e.g. [53]) but the following linear programming formulation will be useful in our work

$$\begin{array}{lll} \text{minimize} & u_{\max} & \\ \text{subject to} & r_l^T s \leq c_l u_{\max} & \forall l \in E \\ & A_p r = b_p & \forall p \in P \\ & r_{lp} \geq 0 & \forall l \in E,\, p \in P. \end{array} \tag{3.14}$$

The first set of inequalities bound the maximum link utilization. The second set of constraints describe flow-conservation at each node such that

$$\sum_{l \in \mathcal{O}(n)} r_{lp} - \sum_{l \in \mathcal{I}(n)} r_{lp} = \begin{cases} 1 & \text{if } n = o(p) \\ -1 & \text{if } n = d(p) \\ 0 & \text{otherwise.} \end{cases}$$

That is, traffic is required to follow a physical path from source to destination. The vector of optimization variables

$$r = \begin{pmatrix} r_1^T, & r_2^T, & \dots & r_{|E|}^T \end{pmatrix}^T$$

define the routing matrix R. We will shortly see that this formulation is readily extended to a robust setting. The formulation above is similar to the well known *node-link* formulation where optimization variables describe the amount of source destination traffic demands routed over each link (cf. [53]). A drawback with this formulation (and node-link), is that the number of variables quickly becomes intractable since there are P variables for each link in the network. A more compact representation can be derived from a *link-path* formulation [53]. In this setting, the optimization problem becomes

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \displaystyle\sum_p \sum_{\pi \in \Pi_p} \rho_{l\pi} \alpha_{\pi p} s_p \leq c_l u_{\max} \, \forall l \in E \\ & \displaystyle\sum_{\pi \in \Pi_p} \alpha_{\pi p} = 1, \qquad \alpha_{\pi p} \geq 0. \end{array} \tag{3.15}$$

The optimization variables $\alpha_{\pi p}$ determine the fraction of source destination traffic demand s_p routed on path π. The first set of constraints are link load constraints and the second set of constraints state that all traffic needs to be routed on some path. To find an optimal solution to (3.15) we need to consider every possible path in the network. This would be computationally demanding and create an intractable number of variables. However, in the vast majority of instances of this problem the optimal solution only includes a small fraction of the possible paths in the network. To this end, we start with a small set of paths generated by e.g., SPF routing and sequentially add new paths. The optimization problem is solved again with the new set of paths to improve the objective. New paths can be generated by solving the dual to (3.15) and use the dual variables to the link constraints in (3.15) as link weights, see e.g. [53] for details. The characteristics of the dual problem allows us to interpret the dual variables as link weights. Using these link weights will result in an algorithm with finite convergence [6]. The procedure where one starts with a small subset of variables and add new variables to improve performance is called *column generation* [43]. More details in the context of path generation in communication networks can be found in [53].

To extend the optimization problems above to a robust setting we define a traffic uncertainty or variability set $\mathcal{S}$ representing the set of traffic scenarios we want to take into account in the optimization as described in Section 3.1. For the link-path formulation the optimization problem becomes

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \sum_p \sum_{\pi \in \Pi_p} \rho_{l\pi} \alpha_{\pi p} s_p \leq c_l u_{\max} \, \forall l \in E, \forall s \in \mathcal{S} \\ & \sum_{\pi \in \Pi_p} \alpha_{\pi p} = 1, \qquad \alpha_{\pi p} \geq 0 \end{array} \tag{3.16}$$

and analogously for the node-link formulation (3.14). As explained earlier, we will consider both polyhedral and ellipsoidal representations of the uncertainty set $\mathcal{S}$. At first glance, this formulation can appear intractable as the constraints have to hold for all $s \in \mathcal{S}$. However, for the case of polyhedral uncertainty, it is sufficient to enforce the constraints for all vertexes of $\mathcal{S}$,

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \sum_p \sum_{\pi \in \Pi_p} \rho_{l\pi} \alpha_{\pi p} s_p^{(v)} \leq c_l u_{\max} \, \forall l \in E, v = 1, \ldots, V \\ & \sum_{\pi \in \Pi_p} \alpha_{\pi p} = 1, \qquad \alpha_{\pi p} \geq 0. \end{array} \tag{3.17}$$

Thus we have one set of link load constraints for each traffic scenario $s^{(v)}$. Nevertheless, this formulation has some problems associated with it. First, the set of traffic scenarios can be very large, e.g. it could be a time-series of estimated traffic matrices over each 15 minute interval over the last week or even month. Second, even if the uncertainty set is polyhedral, it might not be given as the convex hull of extreme points, but rather as the solution of a set of linear inequalities (e.g. all non-negative s satisfying Rs=t). Converting this representation to vertex form is computationally hard and should be avoided if possible. It turns out that both these issues can be addressed in a way similar to column generation. We start with a single traffic scenario and solve the routing problem (3.17). From the solution we identify the worst case traffic scenarios denoted $s^{(wc)}$ in $\mathcal{S}$, add $s^{(wc)}$ to the set of traffic scenarios included in the optimization. Then solve (3.17) again for the extended set of traffic scenarios. The worst case traffic scenario is found by solving for each link $l \in E$, the (convex) optimization problem

$$\begin{array}{ll} \text{maximize} & c_l^{-1} r_l^{(+)} s \\ \text{subject to} & s \in \mathcal{S} \end{array} \tag{3.18}$$

and take the traffic scenario that yields the highest link utilization as $s^{(wc)}$. The procedure is terminated when a worst case traffic scenario with higher $u_{\max}$ no longer can be found. The combination of column and constraint generation can be shown to converge in a finite number of steps [6]. Our experiments show that only a handful of iterations are needed before the algorithm terminates. As shown in Paper C in this thesis, a robust routing setting with performance close to optimal, i.e., a routing optimized for the real traffic demands can often be found using column and constraint generation.

One important type of traffic uncertainty that can be modeled by (3.9) is the one caused by interdomain reroutes as illustrated in Figure 2.4. This kind of traffic uncertainty has been shown to cause large traffic disruptions in multi-homed IP networks in series of papers by Teixeira *et al.* [63, 64, 65]. To describe this uncertainty on the form (3.9) we introduce utility variables δ_{pd} indicating the fraction of traffic to prefix p leaving the network at egress router d. The traffic uncertainty set becomes

$$\begin{aligned} \mathcal{S} = \{ s_{sd} = & \sum_{p \in P(d)} t_{sp} \delta_{pd} \mid \\ & \sum_{d \in E(p)} \delta_{pd} = 1,\ \delta_{pd} \geq 0 \text{ and } \delta_{pd} = 0 \text{ for } d \notin E(p) \} \end{aligned} \tag{3.19}$$

where t_{sp} denote the amount of traffic to prefix p injected in the network at s and $P(d)$ is the set of prefixes announced by egress router d. Both t_{sp} and $P(d)$ are assumed to be known quantities. A routing setting optimized for this traffic uncertainty will be insensitive to changes in the interdomain routing system. Note that although the variables δ_{pd} are continuous variables between zero and one, for the worst case traffic scenario these variables will assume either zero or one. The worst case occurs when all traffic leaves the network at one egress router and other routers receive no traffic to that prefix. However, in its basic form the model described in (3.19) will contain one variable for each prefix and egress router pair.

Papers D and E discuss methods for reducing the number of variables in the model to make it computationally tractable. To illustrate performance of robust routing under interdomain reroutes we calculate the average sending rate to each prefix and source router in a large IP backbone during one week. From the resulting average sending rates to the prefixes, we determine a routing setting for a nominal traffic scenario and a routing setting robust to interdomain reroutes. Further, the routing settings are used to calculate worst case traffic scenarios for each measurement period the following week. Figure 3.11 illustrates performance for the worst case traffic scenarios for nominal and robust routing for the second week. The figure shows that large performance gains can be made with robust routing. Nevertheless, for some instances the performance gain is small due to fluctuations and unanticipated disruptions in traffic behavior.

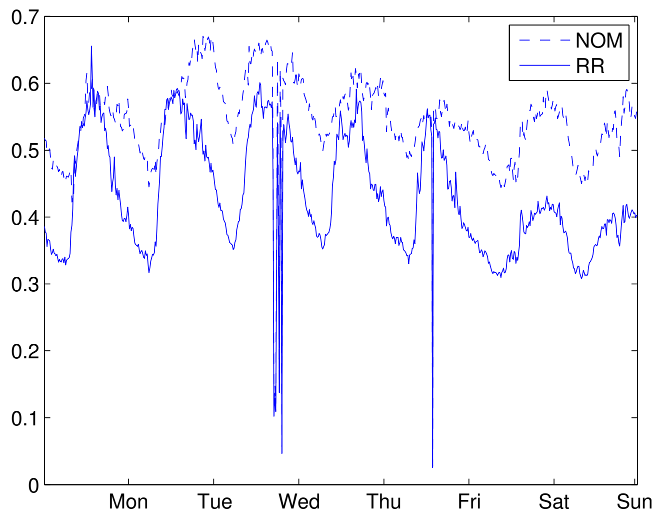


Figure 3.11: Comparison of worst case performance for traffic uncertainty caused by interdomain reroutes between routing optimized for normal operation (dashed) and routing optimized for interdomain reroutes (solid)

Although polytopic uncertainties are difficult to incorporate in the node-link representation described in (3.14), ellipsoidal models are more suitable for this representation. To this end we transform the representation of the uncertainty ellipsoid in (3.10) to

$$\mathcal{S} = \{s = \lambda + L^T u \mid \|u\|_2 \leq 1\}$$

where $M = L^T L$ is the Cholesky decomposition of M. Using simple manipulations detailed in Paper D, the robust routing problem can now be written as

$$\begin{array}{lll} \text{minimize} & u_{\max} & \\ \text{subject to} & r_l^T\lambda + \|Lr_l\|_2 \le c_l u_{\max} & \forall l \in E \\ & A_p r = b_p & \forall p \in P \\ & r_{lp} \ge 0 & \forall l \in E,\, p \in P. \end{array} \tag{3.20}$$

This is a second order cone programming (SOCP) problem, for which efficient polynomial-time algorithms exists (e.g. [39]). In other words, when we consider ellipsoidal uncertainty sets, we can solve the robust routing problem in a single optimization of polynomial time complexity. Nevertheless, also in this formulation, the number of variables grows quickly with problem size and solution times increase substantially for large networks. Paper D in thesis details robust routing under ellipsoidal traffic uncertainty and presents an alternative suboptimal column-and constraint-generation technique that tends to work well in practice.

Traffic engineering in SPF routing

The network flow formulation assumes that traffic can be split arbitrary at nodes and does not impose any constraints on the number of paths that can be used to route traffic between each source and destination. In SPF networks, on the other hand, the routing must follow a single path, and the set of routing paths in the network must be consistent with a set of link weights used for shortest path calculations. The *SPF weight setting problem* consists of assigning positive integer weights to links in order to achieve better network performance when the demands are routed according to the rules of SPF routing.

The restrictions of single path routing protocols make the weight setting problem NP-hard [20]. Not only is the problem hard in theory, but it is also hard in practice. Current integer programming formulations do not even scale to moderate-size networks, if we require finding provably optimal solutions. By giving up optimality and using search heuristics rather than extensive branch and bound procedures, the time for finding good weight settings can be improved substantially. The heuristic attempts to improve an objective function by evaluating different weight settings. The heuristic generates a sequence of new weights using a local search. Each set of link weights is viewed as a point in a high-dimensional search space. A neighbor to a point is another set of weights produced by changing the value of one (or sometimes more) weights. The different neighbors are evaluated with respect to the overall performance objective. The neighbor with the best objective is the one that is used in the next iteration of the algorithm. Typically the algorithm is terminated either when no improvement is detected or after a specified number of iterations.

The following example illustrates how a heuristic works as well as the restrictions of SPF routing. The network shown in Figure 3.12 has four nodes and four bidirectional links with a capacity of 10 units of traffic in each direction. There are two traffic demands, s_{AC} transmits 7.5 units of traffic between nodes A and C, and

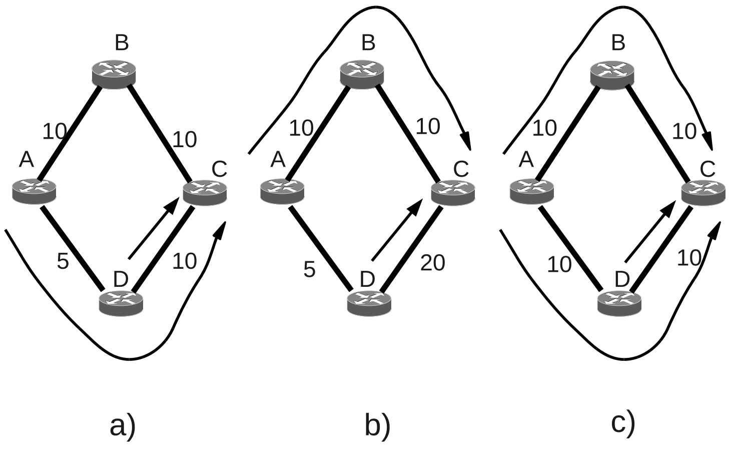


Figure 3.12: A simple example of search heuristics for finding a weight setting

s_{DC} transmits 2.5 units of traffic between nodes D and C. In case a) s_{AC} is routed on path A→D→C and s_{DC} is routed on the path D→C leading to 100% utilization of link (D,C). Many search heuristics attempt to alleviate congestion by deviating traffic from the link with highest utilization. In our example the weight of link (D,C) is increased in b) deviating s_{AC} to path A→B→C. Congestion is lowered to 75% on link (A,B) and (B,C). Finally, some search heuristics attempt to balance load on equal cost paths (ECMP). In Figure 3.12 c) demand s_{AC} is split between paths A→B→C and A→D→C leading to maximum link utilization of 63% on link (D,C). The example in Figure 3.12 highlights the limitations of SPF and ECMP forwarding. The best solution would be to split demand s_{AC} with 2/3 of the traffic on path A→B→C and 1/3 of traffic on path A→D→C reducing highest link utilization to 50%. However, this split ratio is not possible with SPF or ECMP forwarding.

A robust version of the weight setting problem could be to search for link weights with guaranteed performance under all foreseeable traffic variations. The corresponding search heuristic would be to determine a worst-case traffic scenario $s^{(wc)} \in \mathcal{S}$ by solving (3.18) for each weight setting under evaluation, and then execute the weight change that guarantees the lowest worst-case performance. Unfortunately, in a robust setting a number of weight changes need to be executed before progress can be detected. During iterations without progress the search has no information about which weights to change to decrease worst case performance. To illustrate this we return to the simple example network in Figure 2.2 and extend it to a hot-potato routing setting analogous to Figure 2.4. In this example nodes

A and E inject five units of traffic that may leave the network either at node C or D depending on the choice of preferred route by the interdomain routing system. The uncertainty of the traffic can be summarized in the equations

$$s_{\mathrm{AC}} + s_{\mathrm{AD}} = 5$$
$$s_{\mathrm{EC}} + s_{\mathrm{ED}} = 5.$$

Figure 3.13 shows the initial setting of link weights and routes taken by the traffic demands. Assuming all links have the capacity of ten units of traffic, the worst case link load is 100 % utilization on the link between E and D as well as the link between D and C. However, as shown in Figure 3.14 with careful tuning of the link weights we are able to decrease worst case utilization to 50%. Note that we need to tune two weights to accomplish this. After the first weight change worst case performance remains at 100% giving the heuristic little information about what weight change to execute. From Figure 3.14 we observe another important property of robust routing in SPF networks. By routing known total traffic together we are able to control the load on the links in the network. In this example traffic demand s_{AC} and s_{AD} are routed together and traffic demand s_{EC} and s_{ED} are routed together but on separate links to decrease worst case performance.

Based on these observations we are able to design a *hint* function that is added to the original objective in order to guide the heuristic when the search is not able to make progress in the original objective. Details about the hint function and evaluation can be found in Paper E in this thesis. Furthermore, two well known search heuristics are evaluated for estimated traffic demands from link load measurements in Paper B.

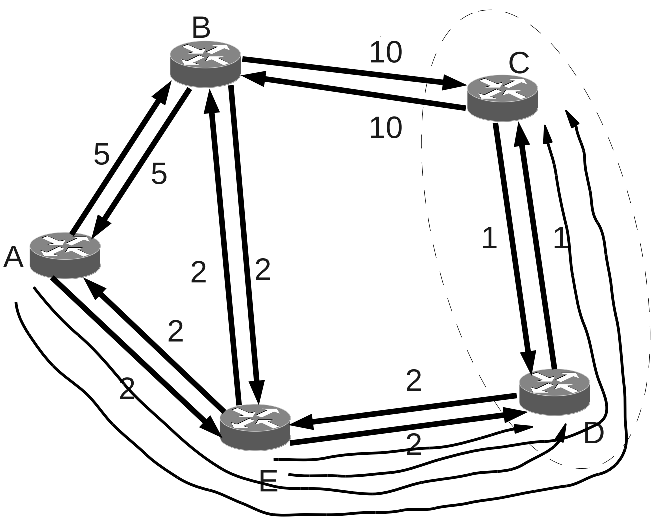


Figure 3.13: A simple example of link state routing and traffic uncertainty

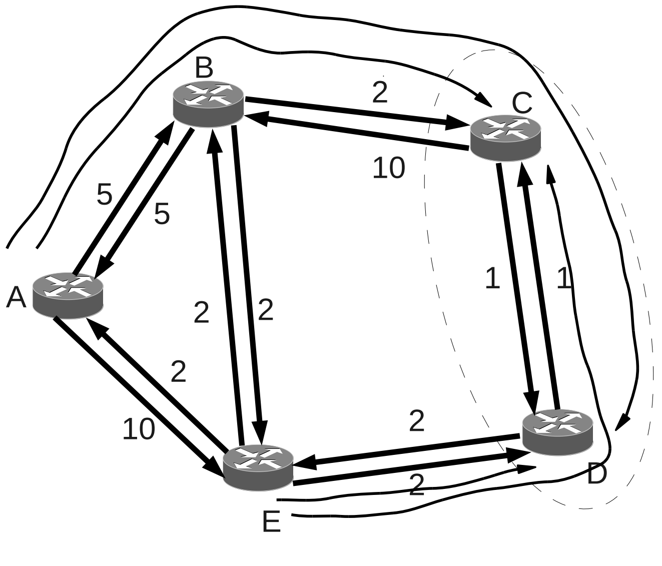


Figure 3.14: Optimal weight setting under interdomain routing traffic uncertainty

Chapter 4

Literature Survey

This chapter surveys research in traffic engineering in IP networks during recent years. We start with methods for deriving the traffic situation in the network followed by a presentation of methods to balance load by optimization of the routing.

4.1 Methods for obtaining the traffic matrix

The methods for deriving the traffic matrix can be divided into three classes. The first class is estimation based where the traffic demands are estimated from incomplete data. The second class of methods are measurement based and rely on flow measurements performed in routers. The third class is a combination of measurements and estimation.

Estimation based methods

The origin-destination estimation problem for telephone traffic is a well-studied problem in the telecom world. For instance, already in 1937, Kruithof [35] suggested an iterative method for estimation of point-to-point traffic demands in a telephone network based on a prior traffic matrix and measurements of incoming and outgoing traffic. Kruithof's method was first analyzed by Krupp [36], who showed that the approach can be interpreted from an information theoretic point-of-view: it minimizes the Kullback-Leibler divergence between the prior guess of the traffic matrix and the estimate under load constraints. Further, Krupp showed that the extended iterative method converges to the unique optimal solution. It is interesting to note that Kruithof's method appears to be the first iterative scaling method in statistics, and that these methods are closely related to the EM-algorithm [14].

However, it appears that it was not until 1996 that the problem was addressed specifically for IP networks. To handle the difficulties of an under-constrained problem, Vardi [67] assumed a Poisson model for the traffic demands. Using the

Poisson model the sample average and sample covariance of the link loads are calculated for a sequence of measurements. The samples are used as additional constraints. The traffic demands are estimated by Maximum Likelihood estimation. Related to Vardi's approach is Cao *et al.* [8] who propose to use a more general scaling law between means and variances of demands. The Poisson model is also used by Tebaldi and West [62], but rather than using ML estimation, they use a Bayesian approach. Since posterior distributions are hard to calculate, the authors use a Markov Chain Monte Carlo simulation to simulate the posterior distribution. The Bayesian approach is refined by Vaton *et al.* [68], who propose an iterative method to improve the prior distribution of the traffic matrix elements. The estimated traffic matrix from one measurement of link loads is used in the next estimation using new measurements of link loads. The process is repeated until no significant change is made in the estimated traffic matrix. An evaluation of the methods in [62, 67] together with a linear programming model is performed by Medina *et al.* [42]. A novel approach based on choice models is also suggested in the article. The choice model tries to estimate the probability of an origin node to send a packet to a destination node in the network and is similar to the fanout model described in Section 3.1. The gravity model is introduced by Zhang *et al.* [76]. As described in Section 3.1, in its simplest form the gravity model assumes a proportionality relation between the traffic entering the network at node i and destined to node j and the total amount of traffic entering at node i and the total amount of traffic leaving the network at node j. The authors of the paper use additional information about the structure and configuration of the network such as peering agreements and customer agreements to improve performance of the method. An information-theoretic approach is used by Zhang *et al.* [77] to estimate the traffic demands. Here, the the mutual information is minimized between source and destination. In all papers mentioned above, the routing is considered to be constant. In a paper by Nucci *et al.*[46] routing is changed and shifting of link load is used to infer the traffic demands.

Measurement based methods

An alternative method to estimation for finding the traffic demands in a network is to use the measurement facilities present in routers, e.g. Cisco's Netflow. Feldmann *et al.* [19] collect flow measurements from routers using Netflow to derive point-to-multipoint traffic demands using routing information from inter and intradomain routing protocols. Choi and Bhattacharyya [12] investigate the accuracy of sampled Netflow. The authors of the paper find that accuracy is satisfactory but care should be taken when Netflow is used in backbone routers since measurement overhead grows linearly with the number of active flows passing the router. An approach to control the measurement overhead is developed by Duffield *et al.* [15]. A scaling factor is recalculated in order to control sampling rate and number of flow records dynamically. In addition, the method is designed to minimize variance in the estimator. Estan *et al.* [18] discuss improvements to

Netflow but the changes are aimed to facilitate traffic flow analysis and are not directed towards traffic matrix measurements.

Combined traffic matrix derivation

A more recent approach is to combine measurement based traffic matrix derivation with estimation-based methods. Papagiannaki *et al.* [50] use Netflow measurements over a 24 hour period to calibrate parameters of a fanout model. The fanout model assumes that the fraction of traffic destined to each other node in the network stays stable even though the corresponding traffic demands fluctuates over time. Each router in the network performs the necessary measurements to calibrate the fanout factors that are sent to the Network Operations Center (NOC). Link-count measurements performed by SNMP are used by the NOC to derive the traffic matrix. The authors devise a heuristic to check the parameters of the fanout model in order to monitor the accuracy of the measurements. If the measured values differ significantly from the parameter values the model is re-calibrated. In addition to summarizing several years of research on traffic matrix estimation, Soule *et al.* [60] introduce two methods that combine flow measurements with estimation. The PCA method is based on principal component analysis first introduced by Lakhina *et al.* [37] and attempts to find a low dimension representation of the traffic demands. Lakhina *et al.* [37] observe that a traffic matrix is dominated by a limited number of flows. By concentrating the analysis of the traffic matrix to these eigenflows the problem is reduced to a well-posed estimation problem. The second method tracks traffic demands using Kalman filters. The Kalman filter performs a prediction on the traffic demands in the next time interval. Measurements of link loads are used to recalibrate the parameters of the prediction at the end of the measurement period. Since the result of the prediction is dependent of the state of the Kalman filter there is an inertia in how the prediction adapts to changes in traffic patterns. To ensure the model is consistent to changes in the underlying model recurring flow measurements are used to calibrate filter parameters.

4.2 Traffic engineering

The aim of traffic engineering is to optimize the usage of network resources under traffic constraints. However, the traffic situation in the network may change over time, e.g. due to changing user behavior, new applications or changes in the routing system. To handle the changes there are basically two approaches as explained in Section 1.1. In this section we elaborate on related work for reactive and proactive traffic engineering.

Proactive traffic engineering

In link state routing the link weights are the parameters the operator can adjust to balance load in the network. One of the earliest and most referenced papers on link weight optimization is due to Fortz and Thorup [20]. The authors use a search heuristic which is shown to be very efficient in finding a suitable weight setting to a given traffic situation. The search heuristic is extended to find a weight setting for a wider range of traffic situations in [21]. Balon and Leduc [5] use ECMP to handle traffic disruptions caused by hot-potato routing. Ramakrishnan and Rodrigues [54] use a different heuristic that increases a link weight until one of the paths traversing the link finds a shorter path to the destination and is deviated. If the change leads to lower maximum link utilization the change is executed and another link is selected. A genetic algorithm is used for optimization of link weights by Ericsson *et al.* [17]. The algorithm uses two basic genetic operators, crossover to combine two weight settings and mutations similar to local search. The selection of parent weight settings is probabilistic to avoid local minimum. Holmberg and Yuan [29] develop a decomposition scheme to reduce the computational effort of finding link weights. The optimization problem is decomposed into a number of smaller optimization problems that are solved in sequence. To further reduce the computational burden the authors device a heuristic based on simulated annealing. Both schemes are approximations of the original link weight optimization problem, but it is demonstrated in the paper that large performance gains can be made with the proposed methods. A somewhat different approach to weight setting is introduced by Pettersson *et al.* [52] where constraint programming is used to address the problem. During the search a subset of the variables are set and constraints are used to prune inconsistent solutions. The search is enhanced with LP relaxations to improve performance. The solution produced is a set of paths that are realizable by SPF routing. Given the set of paths it is possible to find the link weights by solving a LP problem. Constraint programming is also used by Viet *et al.* [69]. The search technique is implemented in COMET and take fast reroute alternatives into consideration. COMET is an object oriented programming language suited for combinatorial problem solving. The related problem of reconfiguring link weights to new optimal values with a minimum of disruption is studied by Francois *et al.* [22] and Raza *et al.* [55].

Traffic engineering using search heuristics with estimated traffic matrices is explored by Roughan *et al.* [57]. Wang *et al.* [72] compute the link weights from the solution of the dual problem of a multi commodity flow problem. Variables in the dual problem can be interpreted as cost of utilizing the resource associated with the dual variable; in our case a link in the network. A slightly different approach is taken by Sridharan *et al.* [61]. Instead of calculating the link weights the authors use a heuristic to allocate routing prefixes to equal-cost multi-paths.

Xu *et al.* [74] introduce DEFT where traffic can be sent over non shortest paths using exponential penalty on longer paths. DEFT can be integrated in OSPF/IS-IS routing with minor changes only.

Applegate and Cohen [4] show that it is possible to find an efficient routing setting with fairly limited knowledge of the traffic demands. Furthermore, the authors give a lower bound on performance for the routing for all possible traffic situations. Column generation is used by Ben-Ameur and Kerivin [6] to find a routing that is optimal for a set of different traffic matrices. The authors describe an algorithm which starts from a small set of paths in the network and set of traffic scenarios and continue to add paths and traffic scenarios until no further improvement is observed for the objective. It is shown that the algorithm terminates in a finite number of steps to an optimal solution. In a paper by Wang *et al.* [71] propose Common-case Optimization with Penalty Envelope (COPE) which computes a routing setting that optimizes for a set of traffic matrices which constitute common case traffic scenarios. Furthermore, COPE gives an upper bound of performance of a larger set of admissible traffic scenarios called a traffic envelope.

Abrahamsson *et al.* [1] use a two step cost function which strives to keep load in the network below a given utilization set by the network operator. The method combines properties of cost functions that minimize link utilization with cost functions that minimize bandwidth usage in the network. This cost function is also used as an performance objective for search heuristics for weight setting by Abrahamsson and Björkman [2].

The approaches to robust load balancing by Zhang-Shen and McKeown [78] and Kodialam *et al.* [34] are both based on the mechanism first introduced by Valiant [66] for load balancing in parallel processor systems. In Valiant load balancing the network is assumed to be fully meshed and the routing is calculated in two steps. In the first step traffic is sent to an intermediate router according to precalculated proportions while in the second step traffic is routed towards the destination. Using Valiant load balancing it is possible to deduce performance bounds under a variety of different traffic variations and component failures.

Reactive traffic engineering

One of the earliest papers on reactive traffic engineering is Gallager's classical paper on minimum delay routing [24] where the author gives sufficient conditions for minimum delay routing and develops a distributed algorithm to calculate the minimum delay routing. The distributed algorithm depends on a global traffic parameter for convergence which makes the algorithm impractical for implementation. Vutukury and Garcia-Luna-Aceves [70] devise an algorithm that approximates the results of Gallager's distributed algorithm. Although Gallager's minimum delay routing algorithm is not used in operational IP networks it serves as a benchmark for other algorithms and the paper is still read and referenced over thirty years after it was first published. A rare achievement in this line of research.

In the papers mentioned above calculations of paths in the network is incorporated in the load balancing. This will enforce optimal performance but complicates implementation of the routing protocol. Hence, in many cases it is desirable to separate path calculations from load balancing and distribute the traffic over precalculated paths set up by e.g. MPLS. Reactive traffic engineering with MPLS has been the subject of a number of research papers during recent years. Elwalid *et al.* [16] introduce a routing algorithm based on optimization. A distributed method called TeXCP for MPLS traffic engineering is introduced by Kandula *et al.* [31]. Load balancing is performed over a set of precomputed MPLS paths between source and destination based on feedback about the traffic situation from the network. The authors prove stability and convergence as well as optimality of the method.

Casas *et al.* [10] attempt to combine proactive and reactive traffic engineering. The proposed method is called Reactive Robust routing (RRR) where a robust routing setting is calculated for a set of anticipated traffic scenarios. An anomaly detection mechanism is used to detect deviations from the anticipated traffic scenarios and trigger a recalculation of the routing to meet the new traffic situation.

Recent research study how overlay routing interacts with load balancing in the network layer. Liu *et al.* [38] demonstrate both analytically and experimentally that overlay routing can lead to severe performance loss as well as routing instability. Interaction between congestion control performed at network edges and traffic engineering performed in the network is investigated by He *et al.* [27]. Based on experiences from their experiments the authors develop an algorithm that unifies congestion control and traffic engineering. The algorithm, Distributed Adaptive Traffic Engineering (DATE), is shown to have faster convergence and is more robust to sudden changes in the traffic situation than current techniques. A number of different cost functions are evaluated by He *et al.* [28]. The authors argue that optimization of user utility will leave performance sensitive to sudden traffic bursts. Instead a combination of user utility and network usability is suggested to balance performance and robustness. Experiences from the evaluation of different cost functions and solution methods are combined to define TRaffic Management Using Multi-path Protocol (TRUMP). TRUMP is a heuristic algorithm without guarantees on convergence or optimality. However, simulations indicate that the algorithm converges to a stable and optimal point for a number of different network topologies and traffic scenarios.

Chapter 5

Summary of Included Papers and their Contributions

This thesis is composed of five papers. Paper A,B and C have been published in international conferences and workshops with peer review. Also, Paper A was awarded best student paper at the conference. Paper D is a journal publication, based on two conference papers published in international peer-reviewed conferences. Furthermore, a shorter version of Paper E has been published at a conference with peer review.

Paper A: Traffic Matrix Estimation on a Global IP Backbone - A Comparison on Real Data

A. Gunnar, M. Johansson and T. Telkamp. Traffic Matrix Estimation on a Global IP Backbone - A Comparison on Real Data. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina Italy, October 2004.

Summary: In this paper we consider the problem of estimating the point-to-point traffic matrix in an operational IP backbone. The analysis is based on complete traffic matrices from a global IP network measured over five-minute intervals. The paper describes the data collection infrastructure, present spatial and temporal demand distributions, investigate the stability of fanout factors, and analyze the mean-variance relationships between demands. We evaluate existing and novel methods for traffic matrix estimation, including recursive fanout estimation, regularized estimation techniques, and methods that rely on mean-variance relationships. We discuss weaknesses and strengths of the various methods. We highlight differences in traffic patterns on different continents and show how this affect the estimation. This paper was awarded the best student paper award at the conference.

A preliminary version of this paper can be found in “Traffic Matrix Estimation for a Global IP Network” at the Nordic teletraffic seminar, Oslo Norway August 2004.

Contribution of this paper: The contribution of this work is a balanced evaluation of traffic matrix estimation methods using a unique data set of complete traffic matrices from an operational IP backbone.

My contribution: I implemented the methods in close cooperation with Mikael Johansson and performed a large part of the analysis of the data set and the experimental evaluation.

Paper B: Performance of Traffic Engineering in Operational IP-Networks - An Experimental Study

A. Gunnar, H. Abrahamsson and M. Söderqvist. Performance of Traffic Engineering in Operational IP-Networks - An Experimental Study, In T. Magedanz, E.R.M. Madeira and P. Dini Editors: *IPOM 2005, LNCS 3751*, pp 202-211 Springer Verlag.

Summary: The paper analyzes two different weight setting methods and compare performance with the optimal solution given by a multi-commodity flow optimization problem. Further, the robustness in terms of how well they cope with estimated traffic matrix data is investigated. The evaluation is performed using network topology and traffic data from an operational IP network.

Parts of this work can be found in “Performance of Traffic Engineering using Estimated Traffic Matrices”. In proceedings of Radio Sciences and Communication RVK 05, June 2005, Linköping Sweden.

Contribution of this paper: The contribution of this work is an evaluation of two search heuristics for weight setting in OSPF/IS-IS using complete traffic data from a Tier-1 IP network operator.

My contribution: I performed the analysis in the paper and performed most of the writing of the paper. The implementation was performed by Mattias Söderqvist but I made some adjustments to the code in order to fit the experiments in the paper.

Paper C: Data-driven traffic engineering: techniques, experiences and challenges

M. Johansson and A. Gunnar. Data-driven traffic engineering: techniques, experiences and challenges, In *Proceedings of the Third International Conference on Broadband Communications, Networks and Systems Broadnets 2006*, San Jose, California, USA, October 2006.

Summary: In this paper we study how routing settings robust to traffic uncertainties can be found. In particular, we study the interplay between traffic matrix estimation and routing optimization. To solve the optimization problem under traffic uncertainty we describe iterative algorithms based on column and constraint generation and demonstrate performance of our approach using a unique data set from an operational IP backbone. Furthermore, we present a technique to compute a sparse MPLS mesh where paths with a small contribution to the optimization are pruned. The sparse mesh approximates the optimal robust routing setting with a specified level of sub-optimality. A scheme to tune OSPF/IS-IS link weights is also introduced in the paper. In addition, the paper discusses a number of challenges later addressed in other papers included in this thesis.

Contribution of this paper: This paper demonstrates that it is possible to find robust routing settings under traffic matrix uncertainty. The performance loss compared to complete knowledge of traffic demands is small.

My contribution: The algorithms and analysis in this paper were developed in cooperation between Mikael Johansson and me. The paper was jointly written with Mikael Johansson.

Paper D: Robust load balancing under traffic uncertainty-tractable models and efficient algorithms

A. Gunnar and M. Johansson. Robust load balancing under traffic uncertainty-tractable models and efficient algorithms, Telecommunication Systems Journal, In Press.

Summary: Routing settings are often sensitive to variations in traffic load leading to severe performance degradation when the traffic situation deviates from the nominal case. In this paper we identify a number of different sources of traffic uncertainty and demonstrate how they can be modeled, incorporated and efficiently solved as an optimization problem. We use linear equations for polyhedral traffic uncertainty and positive definite matrices for elliptic uncertainty. For elliptic uncertainty we solve the problem as a Second Order Conic Programming (SOCP) problem while iterative methods based on column and constraint generation is used for polyhedral traffic uncertainty. Performance of the algorithms is highlighted using network topology and detailed traffic data from an operational IP network.

Parts of the results in this paper can be found in "Robust Routing Under BGP Reroutes" presented at Globecom 2007 and "Robust load-balancing under statistical uncertainty: models and polynomial-time algorithms" presented at NGI 2009.

Contribution of this paper: The paper describes how many traffic uncertainties arising in IP networks can be characterized in a mathematical optimization problem. In particular, for elliptic traffic uncertainty we show that a robust routing setting can be calculated in polynomial time.

My contribution: I formulated the problems addressed and developed the solutions presented in paper in close cooperation with Mikael Johansson. The experimental evaluation and analysis was performed by me and I wrote the paper under the supervision of Mikael Johansson.

Paper E: Cautious Weight Tuning for Link State Routing Protocols

A. Gunnar and M. Johansson. Cautious Weight Tuning for Link State Routing Protocols, SICS Technical Report T2011:01, ISSN 1100-3154 , January 2011.

Summary: In this paper we address the problem of computing suitable link weights in link state routing when complete knowledge of the traffic demands is not available. Experiences from previous research is used to improve cautious weight tuning first introduced in Paper C. We define a novel objective function that favor weight changes with desirable properties for a robust routing setting. The new objective function is used as a hint to the search heuristic to help the search move in a direction where it is likely to find robust weight settings. Performance is demonstrated using an operational IP backbone with real traffic data. In addition, the paper also presents an explanation of why optimization of link weights is robust to link load traffic uncertainty.

A shorter version of this paper was presented as a poster at the 6th international conference on network and service management (CNSM) in Niagara Falls Canada, October 2010.

Contribution of this paper: We show that robust weight settings with performance close to optimal robust routing without the constraints of link-state routing exists. In addition, using our enhanced objective function it is possible to find such weight settings with well known search heuristics.

My contribution: The use of the hint function to guide cautions weight tuning was suggested by me. Furthermore, the algorithms were implemented and tested by me and I wrote the paper under the supervision of Mikael Johansson.

5.1 Other publications by the author

This section contains a list of peer reviewed publications authored or co-authored by the author of this thesis. The author changed family name from Andersson to Gunnar in August 2003.

- A. Gunnar and M. Johansson, Cautious Weight Tuning for OSPF/IS-IS Routing. In *Proc. Conference of Network and Service Management 2010*, 25-29 October 2010, Niagara Falls, Canada.
- A. Gunnar and M. Johansson, Robust load-balancing under statistical uncertainty: models and polynomial-time algorithms. In *Proc. NGI 2009*, 1-3 July 2009, Aveiro, Portugal.
- A. Gunnar and M. Johansson, Robust Routing Under BGP Reroutes. In *Proc. Globecom 2007*, 26-30 November 2007, Washington DC, USA.
- A. Gunnar, B. Ahlgren, O. Blume, L. Burness, P. Eardley, E. Hepworth, J. Sachs and A. Surtees, Access and Path Selection in Ambient Networks. In *Proc. IST Mobile Summit 2007*, 1-5 July 2007, Budapest, Hungary.

- A. Gunnar, Identifying Critical Traffic Demands in an IP Backbone. In *Proc. Swedish National Computer Networking Workshop, SNCNW 2006*, 26-27 Oct 2006, Luleå, Sweden.
- M. Brunner, A. Galis, L. Cheng, J. Colas, B. Ahlgren, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, S. Schuetz, A. Gonzalez, R. Stadler and G. Molnar, Towards Ambient Networks Management. In *Proc. IEEE MATA 2005 Second International Workshop on Mobility Aware Technologies and Applications*, November 2005, Montreal, Canada.
- M. Söderqvist and A. Gunnar, Performance of Traffic Engineering using Estimated Traffic Matrices. In *Proc. Radio Sciences and Communication RVK'05*, June 2005, Linköping, Sweden.
- H. Abrahamsson and A. Gunnar, Traffic Engineering in Ambient Networks : Challenges and Approaches. In *Proc. Swedish National Computer Networking Workshop, SNCNW 2004*, November 2004, Karlstad, Sweden.
- M. Brunner, A. Galis, J. Colas, Jorge, A. Gunnar, B. Ahlgren, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, A. Gonzalez, R. Stadler, G. Molnar and L. Cheng, Ambient Networks Management Challenges and Approaches. In *Proc. IEEE MATA 2004 1st International Workshop on Mobility Aware Technologies and Applications*, November 2004, Florianopolis, Brazil.
- A. Gunnar, M. Johansson and T. Telkamp, Traffic Matrix Estimation for a Global IP Network. In *Proc. 17th Nordic Teletraffic Seminar*, August 2004, Oslo, Norway.
- H. Abrahamsson, B. Ahlgren, J. Alonso, A. Andersson and P. Kreuger, A Multi Path Routing Algorithm for IP Networks Based on Flow Optimization. In *Proc. QofIS'02*, October 2002, Zurich, Switzerland.
- B. Ahlgren, A. Andersson, O. Hagsand, and I. Marsh, Dimensioning links for IP telephony, In *Proc. 2nd IP-Telephony Workshop (IPtel 2001)*, April 2001, New York City, New York, USA.

Chapter 6

Conclusions and Future Work

Internet traffic at the backbone level is highly predictable and planning the management and upgrading of the network is possible. On the other hand, network traffic can behave in an unpredictable manner in case of, for instance, router or link failure [63]. This calls for methods for monitoring the traffic but also optimization of the routing function where variations in the traffic is accounted for to deliver a reliable communication service. Furthermore, even if average utilization in the network is low [48], it is a well known fact that traffic in the Internet is far from uniformly distributed (cf. [19, 49]) leading to a large fraction of the network being underutilized while a small number of links obtain higher load. Balancing load on these critical links can lead to significant performance gains in the network and delay costly upgrades of the network.

In this thesis we have shown that it is possible to efficiently estimate the traffic situation, and tune the routing such that estimation errors are compensated for in the optimization. Furthermore, by taking traffic variability and uncertainty into account we are able to find routing settings that are able to guarantee performance under these circumstances. We have identified sources of traffic uncertainty and formulated mathematical models for the uncertainties. Furthermore, we have shown how the models can be incorporated into the optimization of the routing to find a routing setting which is able to handle all traffic scenarios included in the model without the need to reconfigure the routing setting. Our experiments indicate that the presence of correlations is important to obtain performance gains for proactive traffic engineering. Nevertheless, proactive traffic engineering can be useful to give probabilistic service guarantees for uncorrelated traffic.

In the early days of the Internet routing could be adapted to the traffic situation [58]. However, this was soon abandoned due to oscillatory behavior of the routing. Nowadays the routing configuration is set to a static value and is rarely changed. Reactive traffic engineering on the other hand, requires new functionality to be installed in routers in the network. Our aim in this thesis has been to optimize legacy functionality in the network as much as possible. However, new software or hardware may be needed in order to split flows arbitrarily between several paths between source and destination. This can be achieved by adopting the solutions described in [1, 9].

To take full advantage of our approach it is beneficial to adopt a full separation of the control and forwarding plane in a network. There are a number of proposals in this direction [23, 40, 75]. A separated control plane could calculate the routing setting at a centralized server and disseminate the result in the network. This fits well with our approach since solving a robust optimization of the routing is inherently centralized. However, this is in stark contrast to how networks are organized today. Advantages and disadvantages of such an approach need more investigation. In addition, Figure 3.11 indicates that a detection mechanism to determine when to recalculate the routing would be beneficial in order to better balance load in the network. This would make proactive traffic engineering adaptive to changes in traffic patterns and thus, combine proactive and reactive traffic engineering.

Bibliography

[1] H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimisation. In B. Stiller, M. Smirnow, M. Karsten, and P. Reichl, editors, *From QoS Provisioning to QoS Charging – Third COST 263 International Workshop on Quality of Future Internet Services, QoFIS 2002 and Second Interntational Workshop on Internet Charging and QoS Technologies, ICQT 2002*, pages 135–144, Zürich, Switzerland, October 2002. Springer. LNCS 2511.

[2] H. Abrahamsson and M. Björkman. Robust traffic engineering using L-balanced weight-settings in OSPF/IS-IS. In *Broadnets*, Madrid, Spain, September 2009.

[3] R. K. Ahuja, T. L. Magnati, and J.B. Orlin. *Network Flows*. Prentice Hall, 1993.

[4] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc. ACM SIGCOMM*, pages 313–324, Karlsruhe, Germany, August 2003.

[5] S. Balon and G. Leduc. Combined intra- and inter-domain traffic engineering using hot-potato aware link weights optimization. In *Proc. ACM SIGMETRICS*, pages 441–442, Jun 2008.

[6] W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. *Optimization and Engineering*, 6(3):283–313, 2005.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[8] J. Cao, D. Davis, S. Vander Wiel, and B. Yu. Time-varying network tomography: router link data. *Journal of Americal Statistical Association*, 95:1063–1075, 2000.

[9] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proc. IEEE INFOCOM*, pages 332–341, Tel Aviv, Israel, March 2000.

[10] P. Casas, L. Fillatre, and S Vaton. Robust and reactive traffic engineering for dynamic traffic demands. In *Proc. NGI 2008*, pages 69–76, Krakow, Poland, April 2008.

[11] V. Chew. Confidence, prediction and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*, 61(315):605–617, 1966.

[12] B. Choi and S. Bhattacharyya. Observations on cisco sampled netflow. *SIGMETRICS Perform. Eval. Rev.*, 33(3):18–23, 2005.

[13] M. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE /ACM Transactions on Networking*, 5(6):835–846, 1997.

[14] I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics and Decisions, Suppl. 1*, Supplement Issue No. 1:205–237, 1984.

[15] N. Duffield, C. Lund, and M. Thorup. Learn more sample less: control of volume and variance in network measurement. *IEEE Transactions on Information Theory*, 51(5):1756 – 1775, May 2005.

[16] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proc. IEEE INFOCOM*, pages 1300–1309, Anchorage, Alaska, USA, May 2001.

[17] M. Ericsson, M. G. C. Resende, and P. M. Pardalos. A genetic algorithm for the weight setting problem in ospf routing. *Journal of Combinatorial Optimization*, 6(3):299–333, 2002.

[18] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better netflow. In *Proc. ACM SIGCOMM*, pages 245–256, Portland, Oregon, USA, 2004.

[19] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proc. ACM SIGCOMM*, pages 257–270, Stockholm, Sweden, August 2000.

[20] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. IEEE INFOCOM*, pages 519–528, Tel Aviv, Israel, March 2000.

[21] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[22] P. Francois, M. Shand, and O. Bonaventure. Disruption-free topology reconfiguration in ospf networks. In *IEEE INFOCOM*, Anchorage, USA, May 2007. INFOCOM 2007 Best Paper Award.

[23] J. Fu, P. Sjödin, and G. Karlsson. Intra-domain routing convergence with centralized control. *Computer Networks*, 53:2985–2996, December 2009.

[24] R. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25(1):73–85, 1977.

[25] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a global IP backbone - a comparison on real data. In *Proc. ACM Internet Measurement Conference*, pages 149–160, Taormina, Sicily, Italy, October 2004.

[26] S. Halabi and D. McPherson. *Internet Routing Archtectures*. Cisco Press, 2001.

[27] J. He, M. Bresler, M. Chiang, and J. Rexford. Towards robust multi-layer traffic engineering: Optimization of congestion control and routing. *IEEE Journal on Selected Areas in Communications*, 25(5):868–880, June 2007.

[28] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang. Rethinking internet traffic management: from multiple decompositions to a practical protocol. In *Proc. CoNEXT 2007*, pages 1–12, New York, New York, USA, December 2007.

[29] K. Holmberg and D. Yuan. Optimization of internet protocol network design and routing. *Networks*, 43(1):39–53, January 2004.

[30] I. Juva, R. Susitaival, M. Peuhkuri, and S. Aalto. Effects of spatial aggregation on the characteristics of origin-destination pair traffic in funet. In *Proc. NEW2AN*, St Petersburg, Russia, September 2007.

[31] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, pages 253–264, Philadelphia, Pennsylvania, USA, August 2005.

[32] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):374–395, 1984.

[33] S. Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997.

[34] M. Kodialam, T. V. Lakshaman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *HotNets III*, San Diego, California, USA, November 2004.

[35] J. Kruithof. Telefoonverkeersrekening. *De Ingenieur*, 52(8):E15–E25, 1937.

[36] R. S. Krupp. Properties of Kruithof's projection method. *The Bell System Technical Journal*, 58(2):517–538, February 1979.

[37] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proc. ACM SIGMETRICS*, New York, New York, USA, June 2004.

[38] Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the interaction between overlay routing and underlay routing. In *Proc. IEEE INFOCOM*, pages 2543–2553, Miami, Florida, USA, March 2005.

[39] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebert. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[40] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[41] A. Medina, K. Salamatian, N. Taft, I. Matta, Y. Tsang, and C. Diot. On the convergence of statistical techniques for inferring network traffic demands. Technical Report BUCS-2003-003, Boston University, Computer Science, USA, February 2003.

[42] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. ACM SIGCOMM*, pages 161–174, Pittsburgh, Pennsylvania, USA, August 2002.

[43] M. Minoux. *Mathematical Programming: Theory and Algrithms*. John Wiley and Sons, 1986.

[44] J. Moy. *OSPF Anatomy of an Internet Routing Protocol*. Addison Wesley, 1998.

[45] S. Nash and A. Sofer. *Linear and nonlinear programming*. McGraw-Hill, 2004.

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

[47] A. Nucci and D. Papagiannaki. *Design, Measurement and Management of Large-Scale IP Networks*. Cambridge University Press, 2009.

[48] A. Odlyzko. Networks are lightly utilized, and will stay that way. *Review of Network Economics*, 2(3):210–237, September 2003.

[49] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot. A pragmatic definition of elephants in Internet backbone traffic. In *Proc. ACM Internet Measurment Workshop*, Marseille, France, November 2002. ACM Press.

[50] K. Papagiannaki, N. Taft, and A. Lakhina. A distributed approach to measure IP traffic matrices. In *Proc. ACM Internet Measurement Conference*, pages 161–174, Taormina, Sicily, Italy, October 2004.

[51] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[52] M. P. Pettersson, R. Szymanek, and K. Kuchcinski. A CP-LP approach to network management in OSPF routing. In *Proc. IEEE SAC*, Seoul, Korea, March 2007.

[53] M. Pioro and D. Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[54] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest-path data networks. *Bell Labs Technical Journal*, 6(1):117–138, 2001.

[55] S. Raza, Y. Zhu, and C. Chuah. Graceful network operations. In *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[56] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in modeling backbone traffic variability: models, metrics, measurements and meaning. In *Proc. ACM Internet Measurement Workshop*, Marseille, France, November 2002.

[57] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *Proc. ACM Internet Measurement Conference*, pages 248–258, Miami Beach, Florida, USA, October 2003.

[58] A. Shaikh, J. Rexford, and K. G. Shin. Load-sensitive routing of long-lived IP flows. In *Proc. ACM SIGCOMM*, pages 215–226, Cambridge, Massachusetts, USA, 1999.

[59] Y. Shavitt and N. Zilberman. A structural approach for pop geo-location. In *Proc. NetSciCom*, San Diego, California, USA, March 2010.

[60] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot. Traffic matrices: Balancing measurements, inference and modeling. In *Proc. ACM SIGMETRICS*, Banff, Canada, June 2005.

[61] A. Sridarhan, R. Guerin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. In *Proc. IEEE INFOCOM*, San Francisco, California, USA , November 2003.

[62] C. Tebaldi and M. West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442):557–576, June 1998.

[63] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan. Traffic matrix reloaded: The impact of routing changes. In *Proc. Passive Active Measurements*, Boston, Massachusetts, USA, April 2005.

[64] R. Teixeira, T. Griffin, G. Voelker, and A. Shaikh. Network sensitivity to hot potato disruptions. In *Proc. ACM SIGCOMM*, pages 231–244, Portland, Oregon , USA, August 2004.

[65] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, pages 307–319, New York, USA, June 2004.

[66] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11:350–361, 1982.

[67] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the Americal Statistical Association*, 91(433):365–377, March 1996.

[68] S. Vaton and A. Gravey. Network tomography : an iterative bayesian analysis. In *Proc. ITC 18*, Berlin, Germany, August 2003.

[69] H. T. Viet, P. Francois, Y. Deville, and O. Bonaventure. Implementation of a traffic engineering technique that preserves ip fast reroute in comet. In *Rencontres Francophones sur les Aspects Algorithmiques des Telecommunications, Algotel (2009)*, 2009.

[70] S. Vutukury and J. J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *Proc. ACM SIGCOMM*, pages 227–238, Cambridge, Massachusetts, USA, August 1999.

[71] H. Wang, H. Xie, L. Qiu, Y. Yang, Y. Zhang, and A. Greenberg. COPE: traffic engineering in dynamic networks. In *Proc. ACM SIGCOMM*, pages 99–110, Pisa, Italy, 2006.

[72] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proc. IEEE INFOCOM*, pages 565–571, Anchorage, Alaska, USA, May 2001.

[73] L. Wolsey. *Integer programming*. John Wiley and Sons, 1998.

[74] D. Xu, M. Chiang, and J. Rexford. Deft: Distributed exponentially-weighted flow splitting. In *Proc. IEEE INFOCOM*, pages 71–79, Anchorage, Alaska, USA, May 2007.

[75] H. Yan, D. Maltz, E. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4d network control plane. In *Proc. USENIX NSDI07*, pages 369–382, Cambridge, Massachusetts, USA, April 2007.

[76] 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*, pages 206–217, San Diego, California, USA, June 2003.

[77] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information theoretic approach to traffic matrix estimation. In *Proc. ACM SIGCOMM*, pages 301–312, Karlsruhe, Germany, August 2003.

[78] R. Zhang-Shen and N. McKeown. Designing a predictable internet backbone network. In *HotNets III*, San Diego, California, USA, November 2004.

Part II

Included Papers

Chapter 7

Paper A: Traffic Matrix Estimation on a Large IP Backbone - a Comparison on Real Data

Anders Gunnar, Mikael Johansson, and Thomas Telkamp. In *Proceedings of the Third ACM SIGCOMM Conference on Internet Measurements IMC 2004*, Taormina, Sicily, Italy, October 2004.

Abstract

This paper considers the problem of estimating the point-to-point traffic matrix in an operational IP backbone. Contrary to previous studies, that have used a partial traffic matrix or demands estimated from aggregated Netflow traces, we use a unique data set of complete traffic matrices from a global IP network measured over five-minute intervals. This allows us to do an accurate data analysis on the time-scale of typical link-load measurements and enables us to make a balanced evaluation of different traffic matrix estimation techniques. We describe the data collection infrastructure, present spatial and temporal demand distributions, investigate the stability of fan-out factors, and analyze the mean-variance relationships between demands. We perform a critical evaluation of existing and novel methods for traffic matrix estimation, including recursive fanout estimation, worst-case bounds, regularized estimation techniques, and methods that rely on mean-variance relationships. We discuss the weaknesses and strengths of the various methods, and highlight differences in the results for the European and American subnetworks.

7.1 Introduction

Many of the decisions that IP network operators make depend on how the traffic flows in their network. A *traffic matrix* describes the amount of data traffic transmitted between every pair of ingress and egress points in a network. When used together with routing information, the traffic matrix gives the network operator valuable information about the current network state and is instrumental in traffic engineering, network management and provisioning (see, e.g., [1, 18, 2, 17]).

Despite the importance of knowing the traffic matrix, the support in routers for measuring traffic matrices is poor and operators are often forced to estimate the traffic matrix from other available data, typically link load measurements and routing configurations. In its simplest form, the estimation problem then reduces to finding a non-negative vector s that satisfies $Rs = t$, where R is a matrix reflecting the routing, t is a vector of measured link loads and s is a vectorized version of the (unknown) traffic matrix. The link loads are readily obtained using the Simple Network Management Protocol (SNMP). This approach leads to an under-constrained problem since the number of links in a network is typically much smaller than the number of node pairs. Some sort of side information or assumptions must then be added to make the estimation problem well-posed.

To evaluate how well different approaches to traffic matrix estimation will work in an operational IP network, and how reasonable various assumptions are, one needs access to a measured traffic matrix on the time-scale of standard link-load measurements. Previous studies have used NetFlow data to measure the traffic matrix in 5-minute increments on a single router [3] or one-hour traffic matrices on a partial network [23]. However, since NetFlow data is unable to capture traffic variability within flows, this is not very accurate for validating estimation methods that use a time-series of link-load measurements. Our study provides new results in the sense that it uses a complete network traffic matrix, based on direct measurements at 5-minute intervals. The data set is collected from Global Crossing's global backbone and consists of routing configuration and the number of bytes transfered in MPLS-tunnels during 5-minute intervals over a 24-hour period.

To make the analysis more transparent, we extract traffic matrices and routing information for the American and European subnetworks. We present temporal and spatial demand distributions and analyze some statistical properties of the demands. In particular, we find that there is a surprisingly strong relationship between the mean and variance of demands, and that fanout factors tend to be relatively more stable over time compared to the demand themselves. We then evaluate a selection of existing methods for traffic matrix estimation, including gravity models, regularized methods (such as Bayesian and maximum entropy approaches), and methods that exploit mean-variance relationships. In addition, we investigate the use of worst-case bounds and estimation of fanout factors based on a time-series of link load measurements. We find that the regularized methods work very well provided that we choose the regularization parameter, *i.e.*, the tradeoff between prior information and link measurement, appropriately. Somewhat surprisingly, we fail to achieve good results using methods that exploit mean-variance relationship. We argue that the failure stems from the problem of accurately estimating the covariance matrix of link loads, and present a study on synthetic data to support our claim.

One can note that many classes of traffic matrices occur in the literature (see [11] for a thorough classification). In this paper, we only study the performance of the estimation methods on PoP-to-PoP traffic matrices. This choice is solely based on properties of the data we have obtained, and we make no statement on which class of traffic matrices is more important than the other.

The remaining parts of this paper are organized as follows. In Section 7.2, we present related work in this area. Section 7.3 introduces the problem and notation. The estimation methods that we evaluate are introduced in Section 7.4, while data collection, data analysis and benchmarking of the methods is presented in Section 7.5. Finally, some concluding remarks are collected in Section 7.6.

7.2 Related work

The origin-destination estimation problem for telephone traffic is a well-studied problem in the telecom world. For instance, already in 1937, Kruithof [9] suggested a method for estimation of point-to-point traffic demands in a telephone network based on a prior traffic matrix and measurements of incoming and outgoing traffic. However, it appears that it was not until 1996 that the problem was addressed specifically for IP networks. In order to handle the difficulties of an under-constrained problem, Vardi [20] assumes a Poisson model for the traffic demands and covariances of the link loads is used as additional constraints. The traffic demands are estimated by Maximum Likelihood estimation. Related to Vardi's approach is Cao *et al.* [3] that propose to use a more general scaling law between means and variances of demands. The Poisson model is also used by Tebaldi and West [19], but rather than using ML estimation, they use a Bayesian approach. Since posterior distributions are hard to calculate, the authors use a Markov Chain Monte Carlo simulation to simulate the posterior distribution. The Bayesian approach is refined by Vaton *et al.* [21], who propose an iterative method to improve the prior distribution of the traffic matrix elements. The estimated traffic matrix from one measurement of link loads is used in the next estimation using new measurements of link loads. The process is repeated until no significant change is made in the estimated traffic matrix. An evaluation of the methods in [19, 20] together with a linear programming model is performed by Medina *et al.* [13]. A novel approach based on choice models is also suggested in the article. The choice model tries to estimate the probability of an origin node to send a packet to a destination node in the network. Similar to the choice model is the gravity model introduced by Zhang *et al.* [23]. In its simplest form the gravity model assumes a proportionality relation between the traffic entering the network at node i and destined to node j and the total amount of traffic entering at node i and the total amount of traffic leaving the network at node j. The authors of the paper use additional information about the structure and configuration of the network such as peering agreements and customer agreements to improve performance of the method. An information-theoretic approach is used by Zhang *et al.* in [24] to estimate the traffic demands. Here, the Kullback-Leibler distance is used to minimize the mutual information between source and destination. In all papers mentioned above, the routing is considered to be constant. In a paper by Nucci *et al.* [15] the routing is changed and shifting of link load is used to infer the traffic demands. Feldmann *et al.* [7] uses a somewhat different approach to calculate the traffic demands. Instead of estimating from link counts they collect flow measurements from routers using Cisco's NetFlow tool and derive point-to-multipoint traffic demands using routing information from inter- and intra-domain routing protocols.

7.3 Preliminaries

Notation and Problem Statement

We consider a network with N nodes and L directed links. Such a network has $P = N(N-1)$ pair of distinct nodes that may communicate with each other. The aggregate communication rate (in bits/second) between any pair (n, m) of nodes is called the *point-to-point demand* between the nodes, and we will use s_{nm} to denote the rate of the aggregate data traffic that enters the network at node n and exits the network at node m. The matrix $S = [s_{nm}]$ is called the *traffic matrix*. It is usually more convenient to represent the traffic matrix in vector form. We then enumerate all P source-destination pairs, and let s_p denote the point-to-point demand of node pair p.

For simplicity, we will assume that each point-to-point demand is routed on a single path. The paths are represented by a *routing matrix* $R \in \mathbb{R}^{L \times P}$ whose entries r_{lp} are defined as

$$r_{lp} = \begin{cases} 1 & \text{if the demand of node pair } p \text{ is routed across link } l \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

Note that the routing matrix may easily be transformed to reflect a situation where traffic demands are routed on more than one path from source to destination by allowing fractional values in the routing matrix. Let t_l denote the aggregate data rate on link l, $t = [t_l] \in \mathbb{R}^L$ be the vector of link rates, and $s \in \mathbb{R}^P$ be the vector of demands for all source-destination pairs. Then, s and t are related via

$$Rs = t \tag{7.2}$$

The *traffic matrix estimation problem* is simply the one of estimating the non-negative vector s based on knowledge of R and t.

The challenge in this problem comes from the fact that this system of equations tends to be highly underdetermined: there are typically many more source-destination pairs than links in a network, and (7.2) has many more unknowns than equations.

It is important to note that in an IP network setting, not all links are interior links connecting the core routers in the network: some of the links are access and peering links that supply data to and receive data from the edge nodes. To make this more explicit, we introduce the notation $e(n)$ for the link over which demand enters at node n, and $x(m)$ for the link over which demand exits at node m. For ease of notation, we assume that each edge node is either an access or a peering point (if this is not the case we can always introduce artificial nodes in our network representation so that this holds). Under these assumptions, $t_{e(n)}$ is the total traffic entering the network at node n and $t_{x(m)}$ is the total traffic exiting the network at node m. Finally, we let $\mathcal{A}$ be the set of nodes acting as access points, and $\mathcal{P}$ the set of nodes acting as peering points.

Alternative formulations of traffic estimation problems

The traffic matrix as a demand distribution

Since demands are non-negative, it is natural to normalize s with the total network traffic

$$s_{\text{tot}} = \sum_i \sum_j s_{ij} = \sum_n t_{e(n)}$$

and view $\tilde{s} = s/s_{\text{tot}}$ as a probability distribution. We may then interpret $\tilde{s}_p$ as the probability that a random packet in the network is sent between node pair p. Introducing $\tilde{t} = t/s_{\text{tot}}$, we can re-write (7.2) as

$$\begin{cases} R\tilde{s} = \tilde{t} \\ \mathbf{1}^T\tilde{s} = 1, \quad \tilde{s} \succeq 0 \end{cases} \tag{7.3}$$

The traffic estimation problem then becomes the one of estimating a vector $\tilde{s}$ that satisfies (7.3) based on knowledge of R and $\tilde{t}$ (cf. [9, 10]).

Fanout formulations

Another alternative is to normalize the demands by the total aggregate traffic entering the source node, i.e., to write

$$s_{nm} = \alpha_{nm} \sum_m s_{nm} = \alpha_{nm} t_{e(n)} \qquad \sum_m \alpha_{nm} = 1 \tag{7.4}$$

Rather than estimating s_{nm}, one can now focus on estimating the *fanouts* $\alpha_{nm} = s_{nm}/t_{e(n)}$. Also the fanouts can be interpreted as probability distributions: α_{nm} is the probability that a random packet entering the network at node n will exit the network at node m (cf. [13, 12]).

7.4 Methods for Traffic Matrix Estimation

Gravity Models

A simple method for estimating the traffic matrix is to use a so-called *gravity model*. Although these models have a long history in the social sciences [25] and in telephony networks [8], the first application to demand estimation in IP networks appears to be [16]. In our notation, the basic version of the gravity model predicts the demand between node n and node m as

$$s_{nm}^{(p)} = Ct_{e(n)}t_{x(m)} \tag{7.5}$$

where C is a normalization constant that makes the sum of estimated demands equal to the measured total network traffic. With the choice $C = 1/\sum_m t_{x(m)}$, the gravity model reduces to

$$s_{nm}^{(p)} = \frac{t_{x(m)}}{\sum_m t_{x(m)}} t_{e(n)}$$

and a comparison with (7.4) reveals that this is equivalent to the fanout model

$$\alpha_{nm} = \frac{t_{x(m)}}{\sum_m t_{x(m)}}$$

i.e., that the amount of data that node n sends to node m is proportional to the fraction of the total network traffic that exits at node m. Such a model makes sense if the user populations served by different nodes are relatively uniform. However, as pointed out in [23], traffic transit between peering networks behaves very differently. This has led to the generalized gravity model, where traffic between peers is forced to be zero, i.e.,

$$s_{nm}^{(p)} = \begin{cases} 0 & \text{if } n \in \mathcal{P} \text{ and } m \in \mathcal{P} \\ Ct_{e(n)}t_{x(m)} & \text{otherwise} \end{cases}$$

Once again, C is a normalization constant that makes, for example, the estimated total traffic equal to the measured total network traffic. In this study, however, we focus on the simple gravity model and leave the generalized gravity model without further reference. It should be noted that the gravity model does not use any information about the traffic on links interior to the network, and that the estimates are typically not consistent with the link load measurements (in fact, the model may not even produce consistent estimates of the total traffic exiting each node). Thus, gravity models are often not used in isolation, but in combination with some statistical approach that accounts for measured link loads. Such methods will be described next.

Statistical Approaches

Kruithof's Projection Method

One of the oldest methods for estimating traffic matrices is the iterative method due to Kruithof [9]. The original formulation considers the problem of estimating point-to-point traffic in a telephony network based on a known prior traffic matrix and measurements of total incoming and total outgoing traffic to each node in the network. Thus, Kruithof's method can, for example, be used to adjust the gravity model estimate to be consistent with measurement of total incoming and outgoing traffic at edge nodes.

Kruithof's method was first analyzed by Krupp [10], who showed that that the approach can be interpreted from an information theoretic point-of-view: it minimizes the Kullback-Leibler distance from the prior traffic matrix $\left[s_{ij}^{(p)}\right]$ (interpreted as a demand distribution). Krupp also extended Kruithof's basic method to general linear constraints,

$$\begin{array}{ll} \text{minimize} & D(s\|s^{(p)}) \\ \text{subject to} & Rs = t, \qquad s \succeq 0 \end{array}$$

and showed that the extended iterative method convergences to the unique optimal solution. It is interesting to note that Kruithof's method appears to be the first iterative scaling method in statistics, and that these methods are closely related to the celebrated EM-algorithm [6].

Recently [24], Zhang *et al.* have suggested to use the related criterion

$$\begin{array}{ll} \text{minimize} & \|Rs - t\|_2^2 + \sigma^{-2} D(s\|s^{(p)}) \\ \text{subject to} & s \succeq 0 \end{array} \tag{7.6}$$

for estimating traffic matrices for backbone IP traffic. The practical advantage of this formulation, which we will refer to as the Entropy approach, is that the optimization problem admits a solution even if the system of linear constraints is inconsistent. We will comment on possible choices of prior matrices at the end of this section.

Estimation under Poissonian and Generalized Linear Modeling Assumptions

Vardi [20] suggested to use a Poissonian model for the traffic, i.e., to assume that

$$s_p \sim \text{Poisson}(\lambda_p)$$

and showed that the mean and covariance matrix of the link loads are given by

$$\mathbf{E}\{t\} = R\lambda \qquad\qquad \mathbf{Cov}\{t\} = R\,\text{diag}(\lambda)\,R^T$$

A key observation is that the Poissonian model provides an explicit link between the mean and covariance matrix of the traffic. Based on a time-series of link load measurements, we compute the sample mean and covariance,

$$\hat{t} = \frac{1}{K}\sum_{k=1}^{K} t[k] \qquad\qquad \hat{\Sigma} = \frac{1}{K}\sum_{k=1}^{K}(t[k]-\hat{t})(t[k]-\hat{t})^T$$

and then match the measured moments with the theoretical, *i.e.*, solve

$$R\lambda = \hat{t} \qquad\qquad R\,\mathrm{diag}(\lambda)\,R^T = \hat{\Sigma}$$

for the vector λ of mean traffic rates. By accounting for the model of the covariance matrix we get $L(L+1)/2$ additional relations, and Vardi proves that the combined information makes the vector λ statistically identifiable. In practice, however, there will typically be no vector λ that attains equality in the moment matching conditions (this may for example be due to lack of data, outliers, or violated modeling assumptions). Vardi suggests to use the EM algorithm to minimize the Kullback-Leibler distance between the observed sample moments and their theoretical values. However, as pointed out in [5], when the observed values are not guaranteed to be non-negative, it is more reasonable to use a least squares fit. To this end, we find the estimate λ by solving the non-negative least-squares problem

$$\begin{array}{ll} \text{minimize} & \|R\lambda-\hat{t}\|_2^2+\sigma^{-2}\|R\mathrm{diag}(\lambda)R-\hat{\Sigma}\|_2^2 \\ \text{subject to} & \lambda \succeq 0 \end{array}$$

The parameter $\sigma^{-2} \in [0,1]$ reflects our faith in the Poissonian modeling assumption (compare [20, Section 4]): if σ^{-2} tends to zero, then we base our estimate solely on the first moments, while $\sigma^{-2}=1$ is natural if we believe in the Poisson assumption.

Cao *et al.* [3] have extended the Vardi's approach by considering a generalized linear modeling assumption

$$s_p \sim \mathcal{N}\left(\lambda_p, \phi\lambda_p^c\right)$$

and assumes that all source-destination flows are independent. The additional scaling parameters ϕ and c give somewhat more freedom than the strict Poissonian assumption. However, even for fixed scaling constants ϕ and c, the estimation procedure is more complex (the associated optimization problem is non-convex), and Cao *et al.* propose a pseudo-EM method for estimation under fixed value of c. An interesting aspect of the paper by Cao *et al.* is that they also try to account for time-variations in the OD flows in order to use more measurements than the 12 link count vectors logged during a busy hour.

Regularized and Bayesian Methods

A related class of methods can be motivated from Bayesian statistics [19]. For example, by modeling our prior knowledge of the traffic matrix as

$$s \sim \mathcal{N}(s^{(p)}, \sigma^2 I)$$

and assuming that the traffic measurements are subject to white noise with unit variance, i.e.

$$t = Rs + v$$

with $\mathbf{E}\{v\} = 0$, $\mathbf{Cov}\{v\} = I$, the maximum a posteriori (MAP) estimate is found by solving

$$\text{minimize} \quad \|Rs - t\|_2^2 + \sigma^{-2}\|s - s^{(p)}\|_2^2 \tag{7.7}$$

Once again, the optimal estimate can be computed by minimizing a weighted distance of the errors between theoretical and observed means and the distance between the estimated demands and a prior "guesstimate". The variance σ^2 in the prior model is typically used as a tuning parameter to weigh the relative importance that we should put on the two criteria. The formulation (7.7) has been used in, for example, [23], where the prior is computed using a gravity model.

Fanout Estimation

Although fanout estimation does not simplify the estimation problem if we only use a single snapshot of the link loads, it can be useful when we have a time-series of link load measurements. As discussed in Section 7.3, the fanout formulation of (7.2) is the one of finding a non-negative vector $\alpha[k]$ such as

$$RS[k]\alpha[k] = t[k], \qquad \sum_m \alpha_{nm}[k] = 1, \quad n = 1, \dots, N$$

where $S[k]$ is a diagonal scaling matrix such that $s[k] = S[k]\alpha[k]$.

Given a time series of link load measurements, we may assume that the fanouts are constant (i.e., that all link load fluctuations are due to changes in the total traffic generated by each node) and try to find $\alpha \succeq 0$ satisfying

$$\begin{aligned} RS[k]\alpha &= t[k], & k &= 1, \dots, K, \\ \sum_m \alpha_{nm} &= 1, & n &= 1, \dots, N \end{aligned}$$

Even if the routing matrix itself does not have full rank, the above system of equations will quickly become overdetermined, and there is a unique vector α that minimizes the errors (in a given norm) between the observed link counts and the ones predicted by the constant-fanout model. These can be found by solving the optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{K} \|RS[k]\alpha - t[k]\|_2^2 \\ \text{subject to} & \sum_{n=1}^{N} \alpha_{nm} = 1, \qquad m = 1, \dots, N \end{array}$$

which is simply an equality-constrained quadratic programming problem.

Deterministic Approaches

Worst-case bounds on demands

In addition to statistical estimates, it is also interesting to find upper and lower bounds on the demands. Making no underlying statistical assumptions on the demands, we note that a single measurement $t[k]$ of the link loads could be generated by the set of possible communication rates,

$$\mathcal{S} = \{s \succeq 0 \mid Rs = t[k]\}$$

Thus, an upper bound on demand p can be computed by solving the linear programming problem

$$\begin{array}{ll} \text{maximize} & s_p \\ \text{subject to} & Rs = t[k], \qquad s \succeq 0 \end{array}$$

The associated lower bound is found by minimizing s_p subject to the constraints. Obviously, this approach is only interesting when it finds an upper bound smaller than the trivial $\max_{l \in \mathcal{L}(p)} t_l[k]$ and a lower bound greater than zero. Also note that the method is computationally expensive, as it requires solving two linear programs for each point-to-point demand.

7.5 Benchmarking the Methods on Real Data

A major contribution of this paper is to study the traffic in the backbone of a commercial Internet operator, and to benchmark the existing traffic matrix estimation methods on this data. A complete traffic matrix is measured using the operator's MPLS-enabled network.

Previous work also validated estimation methods on real data, but they instead used NetFlow data to measure the traffic matrix on single router or on a partial network. [23] validates the tomogravity method with NetFlow measurements of 2/3 of a tier-1 IP backbone, using hourly traffic matrices. In [3] NetFlow data from a single router is used to create traffic matrices in 5 minute increments, for validating time-varying network tomography.

NetFlow exports flow information from the routers to a collector system. The exported information contains the start and end time of every flow, and the number of bytes transmitted during that interval. The collector calculates the average rate during the lifetime of the flow, and adds that to the traffic matrix. For validating time-varying tomography, this is not a very accurate methodology. The variability within a flow is lost because of the NetFlow aggregation. This might affect the variance-mean relationship this method is based on.

Our study provides new results in the sense that it uses a *full* network traffic matrix, based on the *direct measurements* (rather than analysis of NetFlow traces) of all demands at *5 minute intervals*.

In the remaining parts of this section, we describe how a complete traffic matrix is measured using Global Crossing's MPLS-enabled network, investigate some basic properties of the demands, and evaluate the existing methods for traffic matrix estimation on the data.

Data Collection and Evaluation Data Set

Network

Global Crossing is using MPLS for Traffic Engineering on its global IP backbone. A mesh of Label Switched Paths (LSPs, a.k.a. "tunnels") has been established between all the core routers in the network. Every LSP has a bandwidth value associated with it, and the core router originating the LSP (head-end) will use a constraint based routing algorithm (CSPF) to find the shortest path that has the required bandwidth available. RSVP is then used to setup the actual path across the network. This architecture is described in detail in [22].

By measuring the utilization of every LSP in 5 minute intervals using SNMP, we can create a full and accurate traffic matrix of the network. This is an additional, but important, benefit of running an MPLS-enabled network.

Data Collection

To collect SNMP data from the network, a geographically distributed system of "pollers" has been set up. Each poller retrieves SNMP information from a dedicated set of routers in its area, and also functions as a backup for neighboring pollers. SNMP uses the unreliable UDP protocol for communications between the routers and monitoring systems, and hence there is the risk of losing data during transmission. A distributed system with the pollers located close to the routers being monitored increases the reliability in the case of network performance issues or outages, and keeps the load per poller manageable.

The link and LSP utilizations are collected every 5 minutes, at fixed timestamps (e.g. 9:00:00, 9:05:00, 9:10:00, etc.). There will be some variation in the exact polling time, as it is impossible to query every router and interface at exactly the same time. The exact response time of the routers is recorded, and the corresponding utilization rate data is adjusted for the length of the real measurement interval (e.g. 5 minutes and 3 seconds). The impact of this on the measurements is only minimal, and it provides uniform time series of link and LSP utilization data.

The pollers transfer their data to a central database at fixed intervals, using a reliable transport protocol (TCP).

Routing Matrix

The routing matrix in the form described by equations (7.1) and (7.2) is created using a simulation of the network. Although the routing of the LSPs in the network could be retrieved from the routers, it proves to be more practical to simulate the constraint based routing protocol (CSPF) as used by the routers, using the same constraints data (i.e. LSP bandwidth values).

We use the tool MATE from Cariden [4] to perform this routing simulation, and export this information in a text file. The data is then converted to a routing matrix according to equation (7.1). Although the routing in the network is often in a state of flux because of link and/or equipment outages, this is not of much relevance to our study. These routing changes only have a minor effect on the point-to-point demands (i.e., the traffic matrix).

Evaluation Data Set

In order to perform a scientific evaluation of the estimation methods, we need the measurements of routing, traffic matrix elements and link loads to be consistent. By consistent we mean measurements which satisfies equation (7.2).

By using equation (7.2) we are able to compute the link loads needed as input to the estimation methods, from the measured point-to-point demands and the simulated routing matrix. The above mentioned procedure enable us to evaluate the accuracy of the methods on real data without the errors incurred by errors in the measurement of the link loads.

From Global Crossing's network, we have extracted routing information and traffic matrices for the European and American subnetworks. The reason for this is that we wanted to study networks of manageable size that still accommodate large traffic demands. It also allows us to study if there are any significant differences in the demand patterns on the two continents. To create these separate traffic and routing matrices, we simply exclude all links and demands that do not have both source and destination inside the specific region.

Further, core routers located in the same city were aggregated to form a point of presence (PoP), and we study the PoP-to-PoP traffic matrix. Many PoPs contain routers who only transit traffic. We have in this study included links between these transit routers since we focus on estimation in real networks where transit routers are present. Because not necessarily all the original demands between two PoPs were following the same path, we decided to route the aggregated demand according to the routing of the largest original demand. In practice though, most parallel demands already followed the same path.

Using this approach, the European network has 12 PoPs (thus 132 point-to-point demands) and 72 links, while the American network has 25 PoPs (600 demands) and 284 links.

Since the precise details of the traffic are considered proprietary, we scale all plots by the maximum value of the total traffic during the measurement period. It might, however, be interesting to know that the largest traffic demands are on the order of 1200 Mbps.

Preliminary Data Analysis

Busy hours and demand distributions

Figure 7.1 shows how the normalized total traffic in the two subnetworks vary with time. The solid and dashed lines represent the European and American networks, respectively. There is a clear diurnal cycle, and both subnetworks have a pronounced busy periods. The busy periods overlap partly around 18:00 GMT, and the time period shaded in Figure 7.1. We will focus our data analysis to this interval.

Figure 7.2 shows the cumulative demand distribution for the subnetworks. The figure shows that the top 20 percent of demands account for approximately 80 percent of the traffic in both networks. A similar insight can be obtained from the spatial traffic distributions illustrated in Figure 7.3, where we see that a limited subset of nodes account for the majority of network traffic.

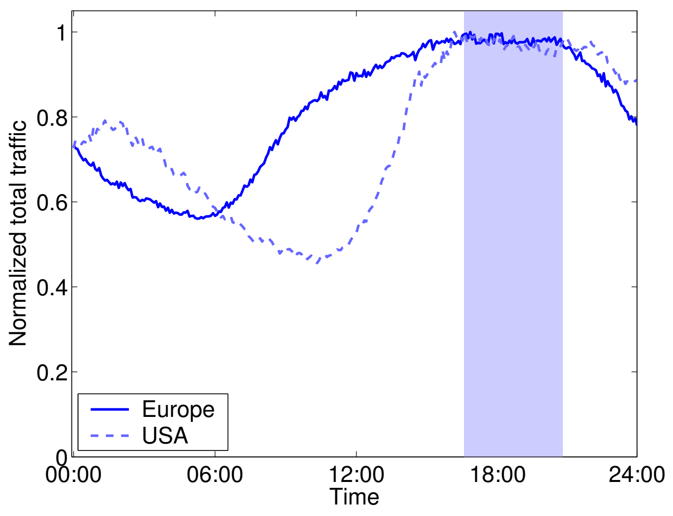


Figure 7.1: Total network traffic over time. The solid line represents the European network, while the dashed line represents the American subnetwork.

On the stability of fanout factors

As we have seen in Section 7.3, there are several possible formulations of the traffic estimation problem: we may estimate the demands directly, focus on the relative demands (viewing the traffic matrix as a demand distribution) or the fanout factors. While the total network traffic changes with the number of active users, one may conjecture that fanouts would be stable as long as the average user behavior does not change. In this section, we investigate whether fanouts are more stable

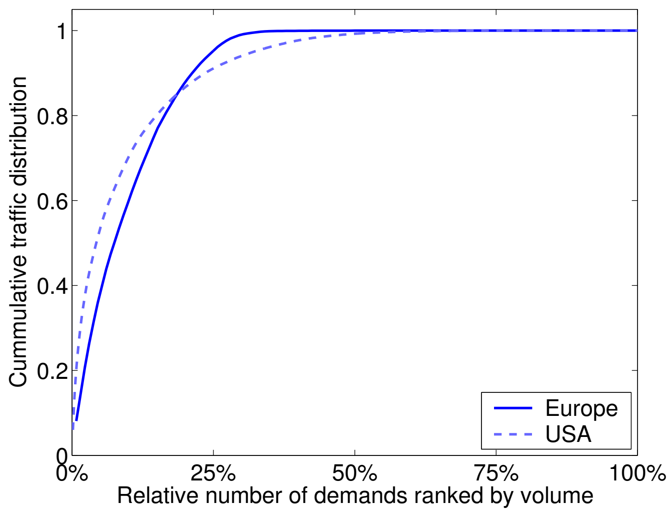


Figure 7.2: Cumulative demand distributions for the European network (solid) and the American subnetwork (dashed).

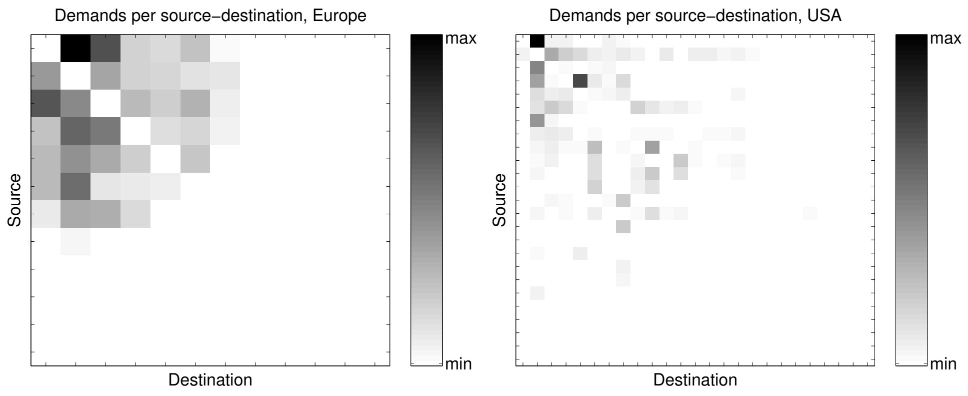


Figure 7.3: Spatial distribution of traffic in the two subnetworks.

over time than demands themselves. If this is the case, fanout estimation may be easier than demand estimation since we do not have to rely on data logged only during the stationary busy hour. Furthermore, if fanouts are stable, it is a worthwhile idea to develop models for fanout factors based on node characteristics (cf. [12]).

Figure 7.4 shows how the demands from the four largest PoPs in the American network fluctuate over the 24-hour measurement period, while Figure 7.5 shows the associated fanouts. We can see that the fanouts are much more stable than the demand themselves during this measurement period. The same qualitative relationship holds for all large demands in the network; for the smaller demands, however, the fanouts sometimes fluctuate more than the demands themselves.

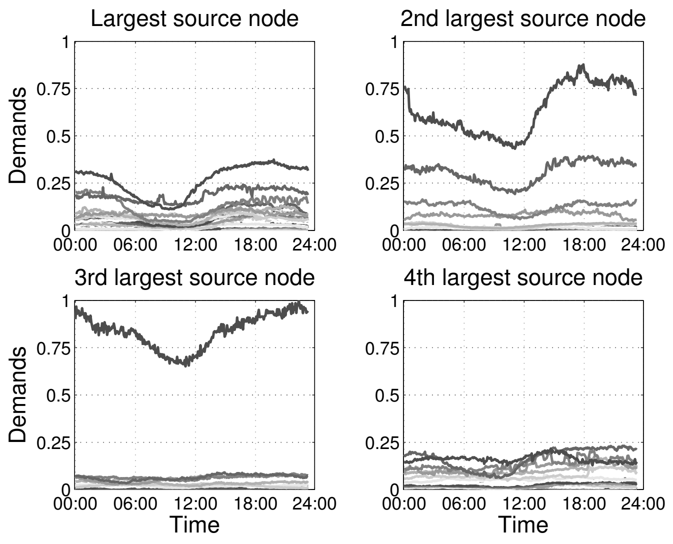


Figure 7.4: The four largest outgoing demands from the four largest PoPs in the American network.

On the Poissonian Modeling Assumption

The assumption that demands are Poissonian, or that they follow a generalized scaling law, provides an explicit link between mean and covariances of link load measurements. Such a link allows us, at least in theory, to statistically identify the demands based on a time series of link load measurements. It is therefore interesting to investigate how well our data satisfies the generalized scaling law [3]

$$\mathbf{Var}\left\{s_p\right\} = \phi\lambda_p^c$$

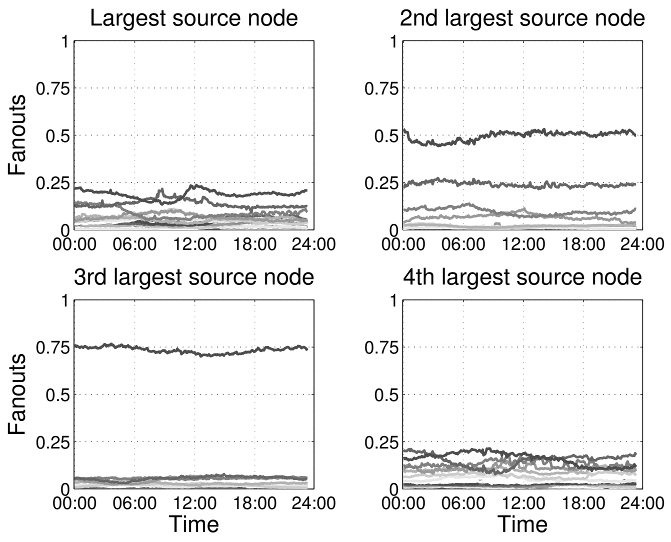


Figure 7.5: The associated fanouts for the four largest outgoing demands from the four largest PoPs in the American network.

In particular, if the traffic is Poissonian, then $\phi = c = 1$. Figure 7.6 shows the relationship between the 5-minute averages of mean and variance for the demands in our subnetworks during busy hour. The plots show a remarkably strong relation between mean and variance and that the generalized scaling law is able to capture the mean-variance relationship for the demands in both subnetworks. The parameters $\phi = 0.82, c = 1.6$ gives the best fit for the European demands, and $\phi = 2.44, c = 1.5$ results in the best fit for the American network.

Similar mean-variance relationships have been established for web-traffic in [14] and for IP traffic demands in [3, 13]. Our observations are consistent with the measurements on a single LAN router in [3] (which suggest that $c = 2$ is more reasonable than the Poissonian assumption $c = 1$), but differs from the measurements on the Sprint backbone reported in [13] (which finds that c varies uniformly over the interval $[0.5, 4.0]$). This difference could be explained from the fact that [13] calculates the 1-second mean-variance relationship *per demand* over 400 intervals of 100 seconds each. The variation of the per demand mean over these 400 intervals (a little more than 11 hours) is not going to be very large. In our analysis, we use the 5-minute mean-variance numbers from all demands during a single interval, like the busy hour for which we want to estimate the traffic matrix. This way we fit the data over an average demand range of 6 magnitudes or more, based on the same measurement intervals that will be used for the estimation procedures.

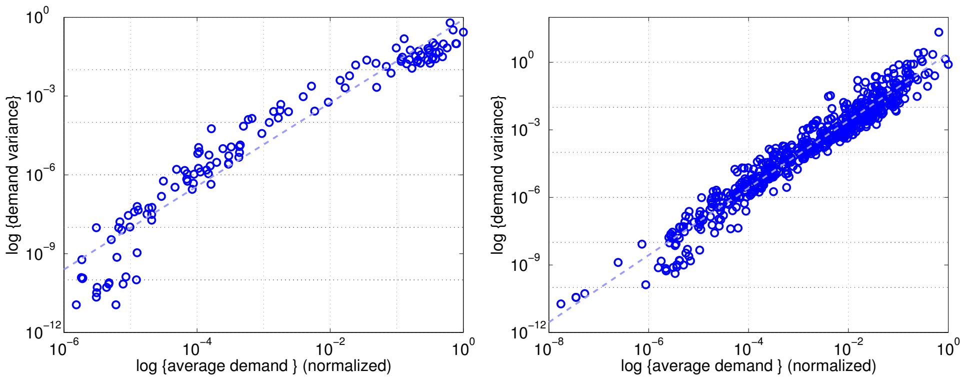


Figure 7.6: Relation between mean and variance for the demands in the European (left) and American (right) subnetworks.

On the Gravity model assumption

Finally, we investigate to what extent the gravity model provides a good estimate of the demands. We focus our analysis on the simple gravity model although the generalized gravity model potentially yield more accurate results since the latter model requires information we do not have access to. Figure 7.7 shows the actual traffic matrix elements against the gravity model estimates. While the gravity model is reasonably accurate for the European network, it significantly underestimates the large demands in the American network. With our knowledge about the spatial distribution of demands shown in Figure 7.3 we could have foreseen this result. Contrary to the gravity model assumption that all PoPs send the same fraction of their total traffic to each destination, PoPs tend to have a few dominating destinations that differ from PoP to PoP.

Evaluation of Traffic Matrix Estimation Methods

In this section, we evaluate the methods for traffic matrix estimation described in Section 7.4. Since fanout estimation and the Vardi approach both use a time-series of measurements rather than a snapshot, they are analyzed separately from the other methods.

Performance Metrics

To evaluate the methods, we must first determine an appropriate performance measure. Although many aspects could potentially be included in the evaluation, we focus on the potential impact of performance errors on traffic engineering tasks such as load balancing or failure analysis. For these applications, it is most important to have accurate estimation of the largest demands since the small demands

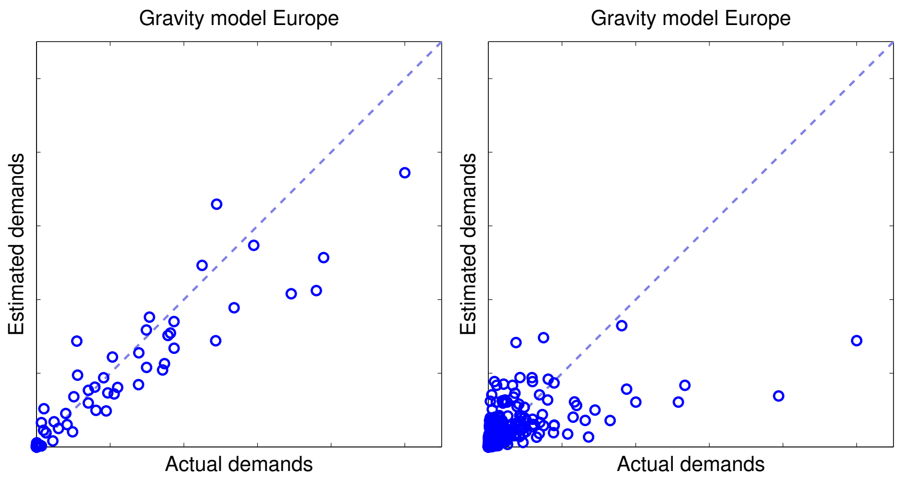


Figure 7.7: Real demands vs. gravity model estimates for European (left) and American (right) subnetworks.

have little influence on the link utilizations in the backbone. We will thus focus our performance analysis on how well the methods are able to estimate the large demands. In order to quantify performance of the estimation and compare results from different estimation methods we introduce the mean relative error (MRE):

$$MRE = \frac{1}{N_T} \sum_{i:s_i > s_T} \left| \frac{\hat{s}_i - s_i}{s_i} \right| \tag{7.8}$$

Here, s_i denotes the true traffic matrix element and $\hat{s}_i$ denotes the corresponding estimate. The sum is taken over the elements in s larger than s_T and N_T is the number of elements in s larger than the threshold. In our analysis, we have chosen the threshold so that the demands under consideration carry approximately 90% of the total traffic. This corresponds to including the 29 largest demands in the European subnetwork, and the 155 largest demands in the American network.

Evaluation of Worst-Case Bounds

To get a feel for how difficult it is to estimate different demands, it is useful to compute worst-case bounds for the demands using the approach described in Section 7.4. The resulting bounds for are shown in Figure 7.8.

Although most bounds are non-trivial, they tend to be relatively loose and only very few bounds can be measured exactly. Still, as shown in Figure 7.9 the average of the upper and lower bound for each flow gives a relatively accurate estimate of the demands. We can observe that many of the largest demands in the European

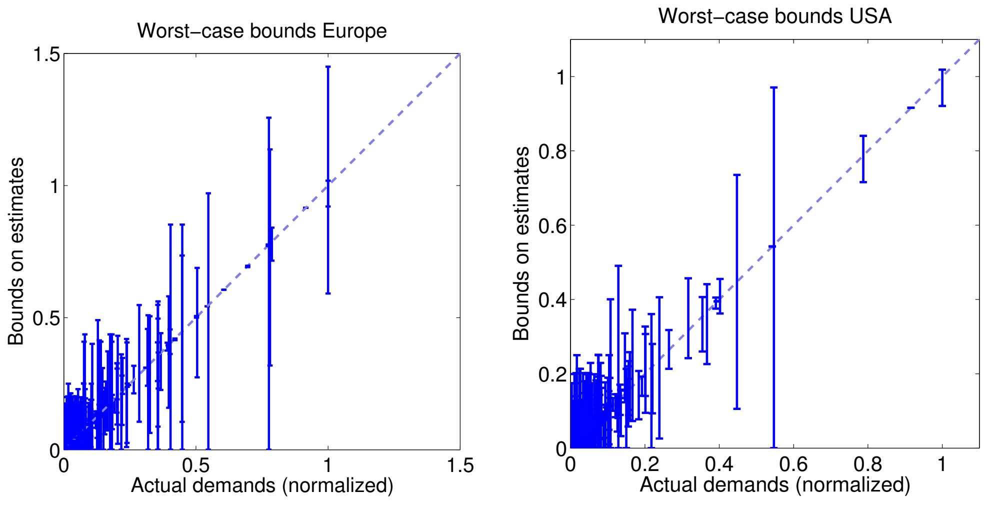


Figure 7.8: Worst-case bounds on demands in European (left) and American (right) subnetworks.

subnetwork have relatively large worst-case bounds, indicating potentially large uncertainty in the estimates.

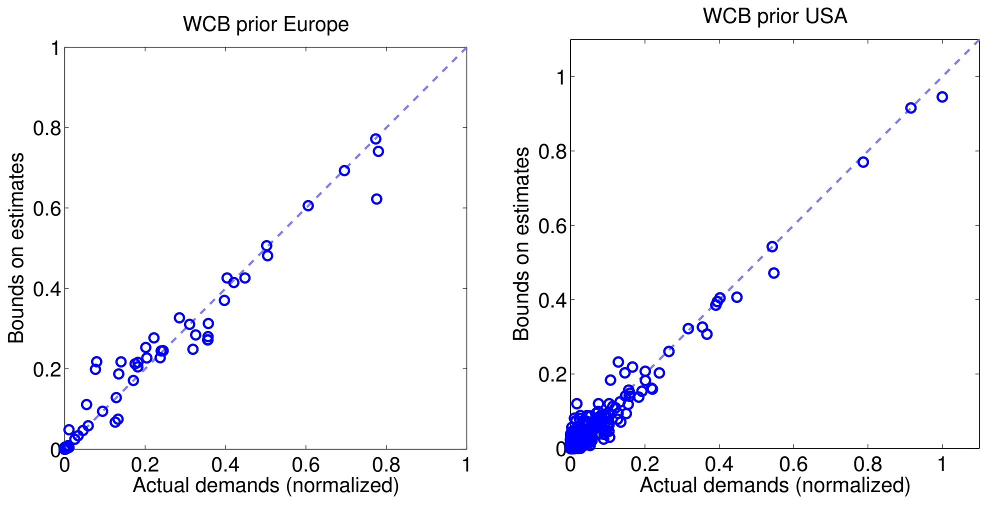


Figure 7.9: Priors obtained from worst-case bounds.

Evaluation of Fanout Estimation

Figure 7.10 shows the results of the fanout-based estimation scheme on the American subnetwork. Since the approach uses a time-series of link load measurements, we have the average demands over the time window on the x-axis against the estimated average demand on the y-axis. Although the system of equations becomes overdetermined already for a window length of 3, the actual performance only improves marginally as we include more data.

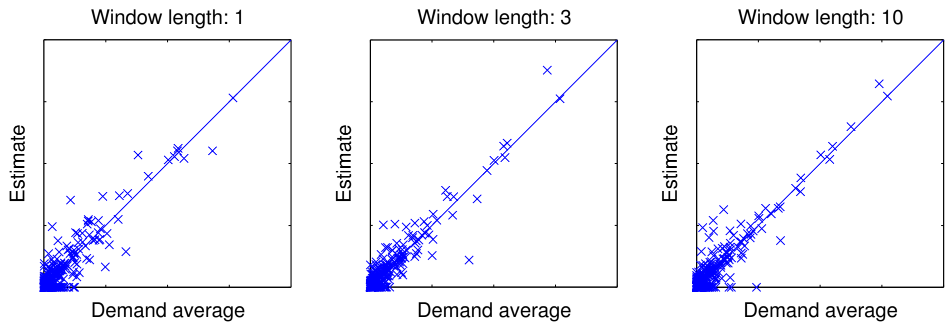


Figure 7.10: Average demands over time window vs. estimates for the fanout estimation procedure using actual data from American subnetwork.

To quantify the error we plot the MRE as a function of the window length as shown in Figure 7.11. The figure shows that the error decreases for short time-series of measurements, but levels out for larger window sizes.

Evaluation of the Vardi approach

In the analysis of the Vardi approach, we apply the method on the busy period of respective network (*i.e.*, the shaded interval in Figure 7.1). The busy period is 250 minutes, or 50 samples long, and we use the sample mean of the traffic demands over the busy period as the reference value in the MRE calculations.

Table 7.1 shows MRE for $\sigma^{-2} = 0.01$ and $\sigma^{-2} = 1$. The value $\sigma^{-2} = 1$, which corresponds to strong faith in the Poisson assumption, gives unacceptable performance; some estimates are several orders of magnitude larger than the true demands while other elements are set to zero despite that the corresponding demand is non-zero. Smaller values of σ give better performance, but are still not very convincing. We believe there are two reasons for the poor performance. First, although there is a strong mean-variance relationship, the analysis in Section 7.5 has shown that the demands are not Poissonian. Second, the convergence of the covariance matrix estimation is slow and one needs a large set of samples to have an accurate estimate. To support this argument, we calculate the mean of the elements of the traffic matrix over the busy period and generate a time-series of

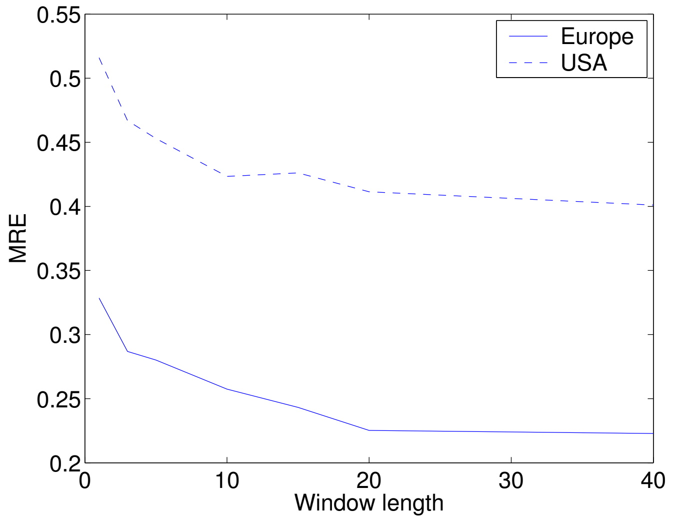


Figure 7.11: MRE as a function of window length.

	Europe	America
$\sigma^{-2} = 0.01$	0.47	0.98
$\sigma^{-2} = 1$	302	1183

Table 7.1: MRE for the Vardi approach, $K = 50$

synthetic traffic matrices with Poisson distributed elements with the calculated mean. Figure 7.12 shows MRE as a function of window size for synthetically generated traffic matrices. The solid line shows the error for the European network and the dashed line the error for the American network. To have errors in the estimation less than 20% we need a window size of 100 for the American network. Hence, even when the Poisson assumption is valid, a large window size is needed in order to achieve an acceptable level of the estimation error.

Comparison of Bayesian and Entropy models

In this section, we evaluate the methods that use a single snapshot measurement from the network. We use the simple gravity model as prior. As before, the threshold value of the MRE method is adjusted so that approximately 90% of the total traffic in the network is included in the study.

Relying on regularization, the results of both the Bayesian (7.7) and the Entropy (7.6) approach depend on the choice of regularization parameter. For a small values of σ we make little use of the measurement and focus on finding a solution that is close to the prior. For very large values of σ, on the other hand, we put a

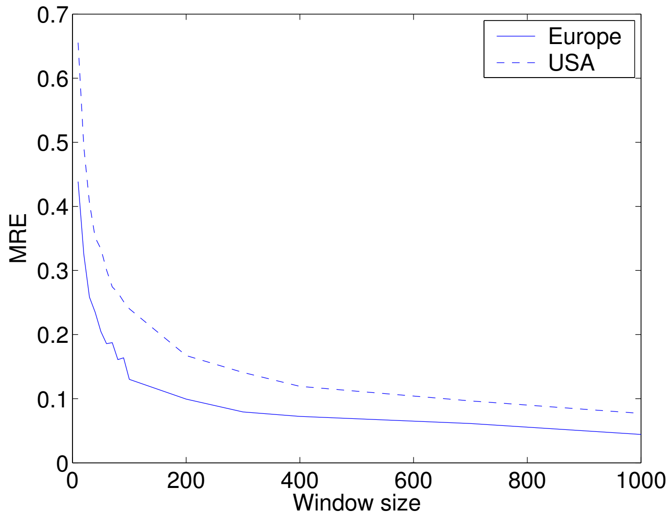


Figure 7.12: MRE as a function of window size for a synthetic traffic matrix, $\sigma^{-2} = 1$

strong emphasis on the measurements, and only use prior to select the most plausible solutions of the demand estimates that satisfy $Rs = t$. This is clearly shown in Figure 7.13, where we have computed the MRE values for both methods as function of the regularization parameter. The leftmost values should be compared with the MRE of the gravity prior, which is 0.26 in European and 0.8 in the American subnetwork. As the plots show, we get the best results for large values of the regularization parameter. We can also see that there is no single best method; the Bayesian performs better in Europe while the Entropy approach works better in the American subnetwork.

To gain intuition about the performance of the estimation we have plotted the actual traffic matrix elements against the estimated for the American network. Figure 7.14 shows the plot for Bayesian (left) and Entropy (right) estimation. The regularization parameter was set to 1000 producing the best possible estimation for both Bayesian and Entropy estimation. The plots show that the estimation manage to capture the traffic demands for the whole spectrum of traffic demands.

Finally, we have demonstrated that using the mean of the upper and lower worst-case bound for each demand resulted in an estimate which is significantly better than the gravity model, and is thus natural to use this as an alternative prior in the regularized approaches. Figure 7.15 shows the MRE for the Bayesian approach as function of regularization parameter for the gravity and worst-case bound prior on the European (left) subnetwork and the American (right) subnetwork. We can see that the worst-case bound prior gives significantly better results

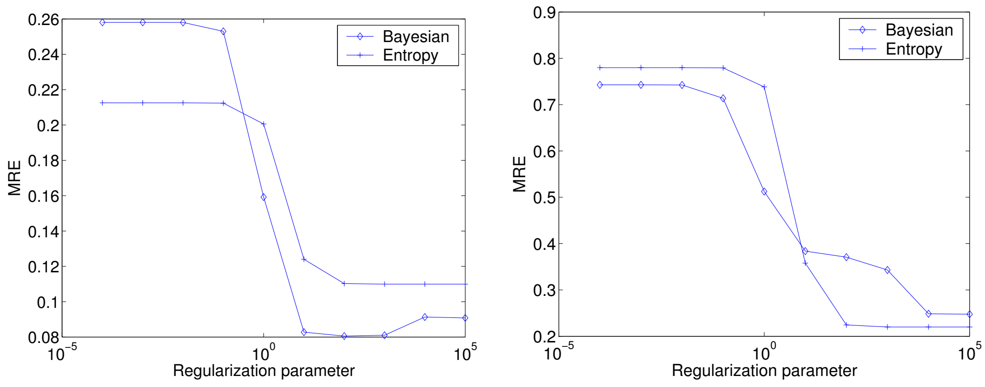


Figure 7.13: Mean relative error (MRE) as a function of the regularization parameter for the European network (left) and the American network (right).

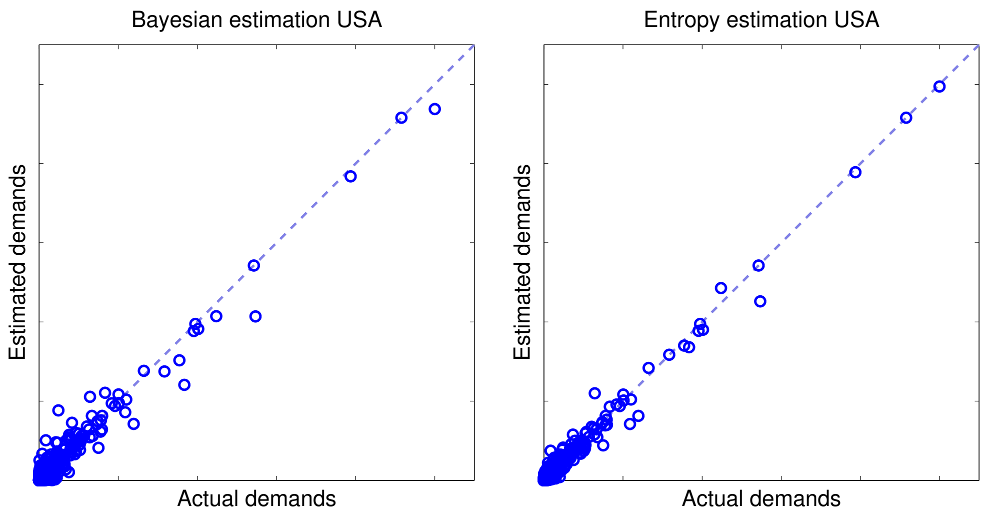


Figure 7.14: Real vs. estimated traffic demands for the American subnetwork using the Bayesian approach (left) and Entropy estimation (right).

for small values of the regularization constant (*i.e.* when we put large emphasis on the prior). For large values of the regularization parameter, however, the performance of the two priors is practically equal.

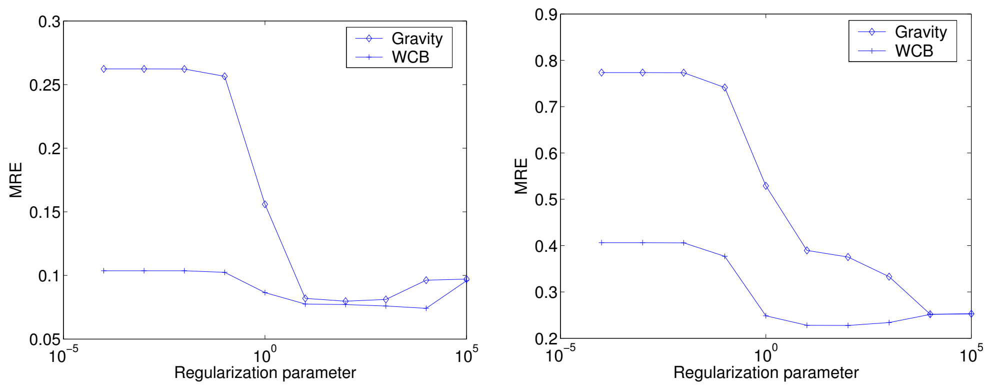


Figure 7.15: Mean relative error (MRE) as a function of the regularization parameter for the European (left) network and the American (right) network using gravity and worst-case bound priors.

Combining Tomography with Direct Measurements

As a final exercise, we investigate the usefulness of combining traffic matrix estimation based on link-loads with direct measurements of specific demands. To get correspondence with the rest of this paper, we focus on the problem of adding measurements that allow us to decrease the MRE of the Entropy method.

Figure 7.16 shows how the MRE for the Entropy approach decreases with the number of measured demands for the European subnetwork. We can see that it is sufficient to measure six demands in order for the MRE to drop from the initial 11% to below 1%. For the American network, on the other hand, we need to measure 17 demands for the MRE to decrease from the initial 23% to below 10%. These results are generated by finding, by exhaustive search in each step, the demand that when measured gives the largest decrease in MRE. They indicate that significant performance improvements can be achieved by measuring only a handful demands.

In practice, however, one would also need an approach for choosing the best demand. Comparing Figures 7.16 and 7.1, one is easily led to believe that they are nothing but each others' inverses, and it would be sufficient to measure the largest demands. In passing, we note that most estimation methods are very accurate in ranking the size of demands, so identifying the largest demands and measuring them is indeed a viable practical approach. However, the MRE measures the relative error, and in our data set, it is not the largest demands that have the largest relative estimation errors. In Europe, one would need to measure the 19 largest demands to have a MRE less than 1%, and in the American network, one would need to measure 74 demands to force the MRE below 10%.

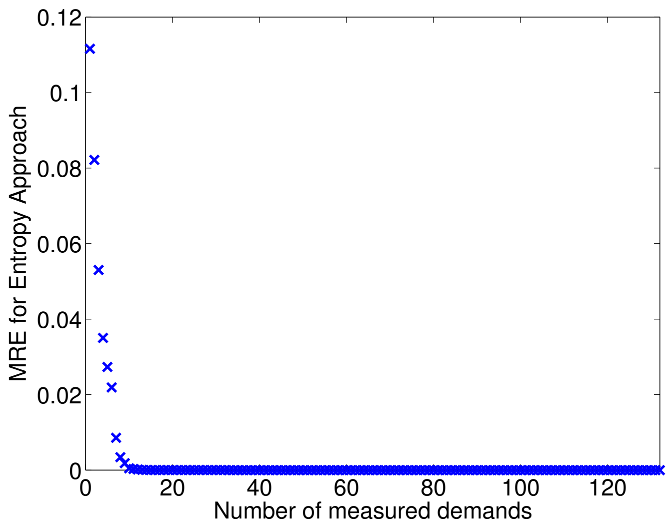


Figure 7.16: The MRE versus number of demands that we measure exactly in the European network.

Evaluation in summary

To allow an easy performance comparison of the methods, Table 7.2 summarizes the best MRE values that we have been able to achieve for the different approaches. The table demonstrates that the Bayesian and Entropy methods gave the best performance, followed by the fanout and Vardi approaches. The worst-case bounds provide a better prior than the simple gravity model on our dataset, and both methods provide better MRE values than the Vardi approach. Note, however, that the fanout and Vardi approaches use, and are evaluated on, a sequence of link load measurements.

Since our experiences of other aspects of the methods, such as ease-of-use and computational complexity, are not easily summarized in a single numbers, we have omitted a direct comparison and refer to the discussions above.

7.6 Conclusion and Future Work

This paper has presented an evaluation of traffic matrix estimation techniques on data from a large IP backbone. In contrast to previous studies that used partial traffic matrices or demands estimated from aggregated NetFlow traces, we have used a unique data set of complete traffic matrices measured over five-minute intervals. The data set has allowed us to do accurate data analysis on the time-scale of standard link-load measurements and enabled us to evaluate both methods that

	Europe	America
Worst-case bound prior	0.10	0.39
Simple gravity prior	0.26	0.78
Entropy w. gravity prior	0.11	0.22
Bayes w. gravity prior	0.08	0.25
Bayes w. WCB prior	0.07	0.23
Fanout	0.22	0.40
Vardi	0.47	0.98

Table 7.2: Performance comparison of the various methods. The table shows the best MRE values that we have been able to achieve for the various methods on the two subnetworks.

use a time-series of link-loads and methods that rely on snapshot measurements.

We have shown that the demands in our data set have a remarkably strong mean-variance relationship, yet we have been unable to achieve good estimation performance using methods that try to exploit this fact. We have argued that this failure is due the problem of accurate estimation of covariance matrices and presented a study on synthetic data to support this claim.

Based on our observation that fanout factors tend to be much more stable over time than the demands themselves, we have proposed a novel method for estimating fanouts based on a time-series of link load measurements. We have also proposed to estimate worst-case bounds on the demands. Although these bounds are not always very tight, they turned out to be useful for constructing a prior for use in other estimation schemes. We have illustrated that the gravity model fails to construct a good prior in one of our subnetworks due to violations of underlying assumptions in the traffic patterns. The regularized methods, such as Bayesian and Entropy approaches, were found to be simple and provide the best results, if the regularization parameter was chosen appropriately. Finally, we noted that by measuring only a handful of demands directly, it was possible to obtain significant decreases in the MRE of the Entropy approach.

This study has focused on analyzing key properties of the demand data set and evaluating the performance of traffic matrix estimation techniques in terms of their estimation error. Although we have covered most methods from the literature, we have not implemented and evaluated the approach by Cao *et al.* [3]. Clearly, a more complete evaluation should include also this method. It would also be useful to complement the evaluation by a more rigorous theoretical analysis to bring a better understanding of our observations. Our study also leaves many important issues unexplored. For example, our data set does not contain measurement errors or component failures and we have not evaluated the effect of such events on the estimation. Furthermore, we have not considered how sensitive traffic engineering tasks are to estimation errors in different demands, and how such information could be incorporated in the estimation procedures. Another interesting topic for future work would be to understand the nature of the worst-case bounds, and see if they could be exploited in other ways.

Acknowledgments

This work is supported in part by the Swedish Foundation for Strategic Research (SSF), the Swedish Research Council (VR), the Swedish Agency for Innovation Systems (VINNOVA) and the European Commission.

Bibliography

[1] H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimisation. In B. Stiller, M. Smirnow, M. Karsten, and P. Reichl, editors, *From QoS Provisioning to QoS Charging – Third COST 263 International Workshop on Quality of Future Internet Services, QoFIS 2002 and Second Interntational Workshop on Internet Charging and QoS Technologies, ICQT 2002*, pages 135–144, Zürich, Switzerland, October 2002. Springer. LNCS 2511.

[2] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[3] J. Cao, D. Davis, S. Vander Wiel, and B. Yu. Time-varying network tomography: router link data. *Journal of Americal Statistical Association*, 95:1063–1075, 2000.

[4] Cariden, Inc., Mountain View, CA. *MATE*, 2004. http://www.cariden.com.

[5] I. Csiszár. Why least squares and maximum entropy? – an axiomatic approach to inverse problems. *The Annals of Statistics*, 19:2033–2066, December 1991.

[6] I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics and Decisions, Suppl. 1*, Supplement Issue No. 1:205–237, 1984.

[7] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[8] J. Kowalski and B. Warfield. Modeling traffic demand between nodes in a telecommunications network. In *Australian Telecommunications and Networks Conference*, Sydney, Australia, December 1995.

[9] J. Kruithof. Telefoonverkeersrekening. *De Ingenieur*, 52(8):E15–E25, 1937.

[10] R. S. Krupp. Properties of Kruithof's projection method. *The Bell System Technical Journal*, 58(2):517–538, February 1979.

[11] 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, August 2002.

[12] A. Medina, K. Salamatian, N. Taft, I. Matta, Y. Tsang, and C. Diot. On the convergence of statistical techniques for inferring network traffic demands. Technical Report BUCS-2003-003, Boston University, Computer Science, USA, February 2003.

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

[14] R. Morris and D. Lin. Variance of aggregated web traffic. In *Proc. IEEE INFOCOM*, pages 360–366, Tel Aviv, Israel, March 2000.

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

[16] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in modeling backbone traffic variability: models, metrics, measurements and meaning. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.

[17] M. Roughan, Mikkel Thorup, and Yin Zhang. Traffic engineering with estimated traffic matrices. In *Proc. ACM Internet Measurement Conference*, Miami Beach, Florida, USA, October 2003.

[18] A. Sridarhan, R. Guerin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. In *Proc. of IEEE INFOCOM 2003*, San Francisco, USA , November 2003.

[19] C. Tebaldi and M. West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442):557–576, June 1998.

[20] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the Americal Statistical Association*, 91(433):365–377, March 1996.

[21] S. Vaton and A. Gravey. Network tomography : an iterative bayesian analysis. In *Proc. ITC 18*, Berlin, Germany, August 2003.

[22] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni. Traffic engineering with MPLS in the internet. *IEEE Network*, 14(2):28–33, March–April 2000.

[23] 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*, San Diego, CA, June 2003.

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

[25] G. K. Zipf. Somde determinants of the circulation of information. *American Journal of Psychology*, 59:401–421, 1946.

Chapter 8

Paper B: Performance of Traffic Engineering in Operational IP-networks - an Experimental Study

Anders Gunnar, Henrik Abrahamsson, and Mattias Söderqvist. In *Proceedings of the 5th IEEE International Workshop on IP Operations and Management IPOM 2005*, Barcelona, Spain.

Abstract

Today, the main alternative for intra-domain traffic engineering in IP networks is to use different methods for setting the weights (and so decide upon the shortest-paths) in the routing protocols OSPF and IS-IS. In this paper we study how traffic engineering performs in real networks. We analyse different weight-setting methods and compare performance with the optimal solution given by a multi-commodity flow optimization problem. Further, we investigate their robustness in terms of how well they manage to cope with estimated traffic matrix data. For the evaluation we have access to network topology and traffic data from an operational IP network.

8.1 Introduction

For a network operator it is important to tune the network in order to accommodate more traffic and meet service level agreements (SLAs) made with their customers. In addition, as new bandwidth demanding and also delay and loss sensitive services are introduced it will be even more important for the operator to manage the traffic situation in the network. This process of managing the traffic is often referred to as *traffic engineering*. The aim is to use the network resources as efficiently as possible and to avoid congestion; *i.e.* deviate traffic from highly utilized links to less utilized links.

In this paper we investigate performance of traffic engineering in operational networks. To make the investigation more balanced we use two traffic engineering methods. The results are compared to the optimal routing obtained from multi commodity flow optimization and the inverse capacity weight setting recommended by Cisco. In order to optimize the routing an estimate of the traffic situation in the network is needed. The traffic situation can be captured in a traffic matrix. The entries in the traffic matrix represent the amount of traffic sent between each source destination pair in the network. However, since routers often lack functionality to measure the traffic matrix directly operators are forced to estimate it from other available data. We use two well known traffic matrix estimation methods to investigate how traffic engineering performs when subjected to estimated traffic matrices.

For the evaluation we have access to a full traffic matrix as well as network topology obtained from direct measurements in a commercial IP network. Previous work have shown that traffic engineering enables the network operator to accommodate substantially more traffic in the network. However, the evaluations have been performed on synthetic data or only partial traffic matrices obtained from an operational IP network.

Our focus is not on the actual methods we use in the study but in how they perform in a real network with real traffic demands. In addition, we study the interplay between traffic estimation and an application of the estimate, *i.e.* traffic engineering. Hence, we only give a brief description of the methods we use and the interested reader should consult the references for further details.

The rest of the paper is organized as follows. Section 8.2 gives a short description of traffic engineering in IP networks. We also discuss related work on the subject. The experiments together with a short description of traffic matrix estimation is given in Section 8.3. The evaluation is described in Section 8.4. Finally we make some concluding remarks about our findings and discuss future work.

8.2 Traffic Engineering in IP Networks

Traffic engineering encompasses performance evaluation and performance optimization of operational networks. An important goal is to avoid congestion in the network and to make better use of available network resources by adapting the routing to the current traffic situation.

The two most common intradomain routing protocols today are OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System). They are both link-state protocols and the routing decisions are based on link costs and a shortest (least-cost) path calculation. With the equal-cost multi-path (ECMP) extension to the routing protocols the traffic can also be distributed over several paths that have the same cost.

These routing protocols are designed to be simple and robust rather than to optimize the resource usage. They do not by themselves consider network utilization and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exist alternative paths. It is up to the operator to find a set of link costs (weights) that is best suited for the current traffic situation and avoids congestion in the network.

The traffic engineering process is illustrated in Figure 8.1. The first step is to collect the necessary information about network topology and the current traffic situation. Most traffic engineering methods need as input a traffic matrix describing the demand between each pair of nodes in the network. But today the support in routers for measuring the traffic matrix is limited. Instead, an often suggested approach is to estimate the traffic matrix from link loads and routing information [5, 6, 13]. Link loads are readily obtained using the Simple Network Management Protocol (SNMP) and routing information is available from OSPF or IS-IS link-state updates.

The traffic matrix is then used as input to the routing optimization step, and the optimized parameters are finally used to update the current routing. In this study this means that the traffic matrix is used together with heuristic search methods to find the best set of links weights.

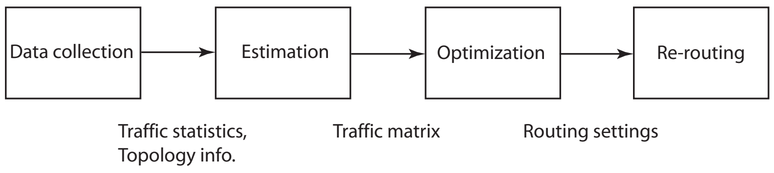


Figure 8.1: The traffic engineering process

Optimal Routing

The general problem of finding the best way to route traffic through a network can be mathematically formulated as a multi-commodity flow (MCF) optimization problem (see, e.g., [1, 3, 7]). The network is then modeled as a graph. The problem consists of routing the traffic, given by a demand matrix, in the graph with given link capacities while minimizing a cost function. This can be formulated and solved as a linear program.

How the traffic is distributed in the network very much depends on the objectives expressed in the cost function. Since one of the main purposes with traffic engineering is to avoid congestion a reasonable objective would be to minimize the maximum link utilization in the network. Another often proposed objective function is described by Fortz and Thorup [3]. Here the sum of the cost over all links is considered and a piece-wise linear increasing cost function is applied to the flow on each link. The basic idea is that the cost should depend on the utilization of a link and that it should be cheap to use a link with small utilization while using a link that approaches 100% utilization should be heavily penalized. The characteristics of the cost function is shown in Figure 8.2.

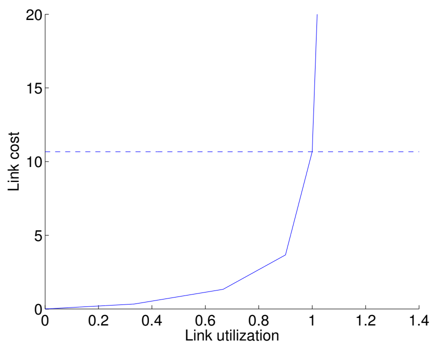


Figure 8.2: Cost function for load on a link

Though the solution given by the linear program is the optimal routing, the method is in general not used directly for routing in operational IP networks. First, the method is inherently centralized. And also, since the solution requires flows to be arbitrary split among several paths towards the destination it would require modifications to the forwarding mechanisms that is used today [1]. In this paper we focus on the legacy routing protocols OSPF and IS-IS. The optimal routing will be used for comparison only since it constitutes a lower bound for the performance of legacy routing mechanisms.

Unfortunately, when taking the restrictions of shortest-paths or equal-cost multipaths in the OSPF and IS-IS protocols into consideration, the problem of finding the optimal routing becomes much harder. The problem of finding weights that optimizes the routing is NP-hard [3, 7]. This means that one usually has to rely on heuristic methods to find the set of weights.

Heuristic Search Methods

An often proposed method to determine the best set of link weights is to use local search heuristics [3, 4, 8]. Given network topology, link capacities and the demand matrix the heuristics evaluate points in a search space, where a point is represented by a set of weights. A neighbor to a point is another set of weights produced by changing the value of one or more weights from the first point. In the heuristics, different neighbors are produced and the cost of each one is calculated using a cost function. From each heuristic the neighbor with the best cost is the one that will be the output. In this study we have selected two heuristics:

- Local search (Fortz and Thorup [3])
- Strictly descending search (Ramakrishnan and Rodrigues [8])

Both heuristics have been studied for random topologies and synthetic traffic demands by Söderqvist [10]. As objective function we use the cost function by Fortz and Thorup [3] mentioned in section 8.2.

Related Work

With the prospect of better utilizing available network resources and optimizing traffic performance, a lot of research has been done in the area of traffic engineering. The general principles and requirements for traffic engineering are described in RFC 3272 [2] produced by the IETF Internet Traffic Engineering working group.

Many researchers use multi-commodity flow models for traffic engineering. The book by Pióro and Medhi [7] gives a comprehensive description of design models and optimizations methods for communication networks, including networks with shortest-path routing.

The performance of weight-setting methods using search heuristics has been investigated with real network topologies and synthetic data or partial traffic matrices in [3, 4, 9, 11]. Fortz and Thorup [3] evaluate their search heuristic using a proposed AT&T backbone network and demands projected from measurements. Sridharan *et al.* [11] use a heuristic to allocate routing prefixes to equal-cost multipaths and evaluate this using data from the Sprint backbone network. An alternative approach is to use the dual of a linear program to find a weight setting [12].

Roughan *et al.* [9] investigate the performance of traffic engineering methods with estimated traffic matrices. However, the authors use partial traffic matrices and one weight setting method only.

8.3 Methodology

This section describes the methodology in this study. First we describe the performance metrics for the experiments followed by a discussion on how the experiments are conducted. Finally we give a short introduction to the traffic matrix estimation methods used in this paper.

Evaluation Metrics

Since the main objective of traffic engineering is to avoid congestion one natural metric of performance is maximum link utilization in the network. The utilization u_a of link a is defined as :

$$u_a = \frac{l_a}{c_a} \tag{8.1}$$

where the load on link a is denoted l_a and c_a is the capacity of the link. However, maximum link utilization only reflect one link in the network. To quantify performance for the routing where the whole network is taken into account we define the normalized cost:

$$\Phi^* = \frac{\Phi}{\Phi_{norm}}. \tag{8.2}$$

Here Φ is the cost function from Section 8.2 and Φ_{norm} is a normalization factor such that the normalized cost is comparable between different network topologies. Further details about the normalized cost can be found in [10].

Experimental Setup

In this paper we use an experimental approach to address the problem. We use a unique data set of complete traffic matrices and topology from a commercial IP network operator to simulate the effects of different weight settings for OSPF/IS-IS routing.

A measured traffic matrix, *i.e.* a traffic matrix without errors, together with the network topology is provided as input to the weight optimization. The output from the optimization is a new set of weights which we use to calculate the new routing. Finally, the measured traffic matrix is applied to the new routing in order to determine link utilization and calculate the normalized cost.

To obtain an estimated traffic matrix we simulate the routing with inverse capacity routing. The link loads obtained by applying the measured traffic matrix is then used to find an estimate of the traffic matrix. The estimated traffic matrix is used as input to the weight optimization algorithm. Finally, the optimized links weights are used to calculate the links loads by applying the original measured traffic matrix.

The optimal solution to the routing problem discussed in section 8.2 will serve as a benchmark for our experiments with the search heuristics. In addition, the routing from the inverse capacity weight setting is also included for comparison as it is often used by network operators and is the recommended weight setting by Cisco [3].

Traffic Matrix Estimation

The traffic matrix estimation problem has been addressed by many researchers before (*e.g.* [5, 6, 13]). In this study we focus on two estimation methods.

- Simple Gravity method
- Entropy method

The simple gravity method is based on the assumption that traffic between source node s and destination node d is proportional to the total amount of traffic sent by s and total amount of traffic destined to d. The strength of this method lies in its simplicity. However, the method is also known to be unaccurate in some situations [5].

A different approach to obtain the traffic matrix is to estimate it from link loads and routing information [6, 13]. Link loads are readily obtained using SNMP and routing is available from OSPF or IS-IS link-state updates. This approach often leads to an ill-posed estimation problem since operational IP networks typically have many more node pairs (entries in the traffic matrix) than links. In order to add more constraints to the problem additional information must be added. This information is usually in the form of some assumption made about the traffic matrix. The entropy method [13] minimizes the Kullback-Leibler distance between the estimate and a prior guess of the traffic demands. With the entropy method it is possible to obtain an accurate estimate of the traffic matrix (cf. [5, 13]). In our experiments we use the gravity method to produce a prior.

By choosing one accurate and one less accurate method we are able to make a more balanced evaluation of how estimation errors influence the performance of traffic engineering subjected to estimated traffic demands.

8.4 Results

In this section we present the results obtained from our experiments. For the evaluation we used network topologies and traffic matrices obtained from a global MPLS-enabled IP network. From the data we isolated the European and the American subnetworks in order to obtain networks of manageable size but still carry large traffic demands. In addition, we obtain two networks with slightly different characteristics. More details about the networks and traffic demands can be found in Gunnar *et al.* [5]. However, it might be interesting to mention that the European network has 12 nodes and 40 links and the American network has 25 nodes and 112 links.

As previously mentioned we use two measures of performance, the normalized cost function introduced by Fortz and Thorup [3] and maximum link utilization in the network. The results are plotted for the following methods:

- **Opt**, the optimal solution to the general routing problem. Included for comparison since it is a lower bound for the other methods.

- **InvCap**, sets the weight inversely proportional to the capacity of the link. Like Opt this method is included as a benchmark as it is the default setting recommended by Cisco.
- **FT**, the search heuristic proposed by Fortz and Thorup [3] starting from a random weight setting.
- **RR**, the search heuristic proposed by Ramakrishnan and Rodrigues [8].

For each topology the algorithms were run with different scaling on the traffic demands. The scalings were obtained by multiplying the traffic demand matrix with a scalar. All algorithms except InvCap use different weight settings for different scalings. In the results both cost and max utilization are presented for each topology and for each scaling. For all algorithms except OPT the cost and the max utilization is computed using the same weight setting. But for OPT the cost and the max utilization are computed independently, using different objective functions.

Experiments with Measured Traffic Matrices

Figure 8.3 shows the normalized cost and maximum link utilization for the American network and for both search heuristics. The plot shows that both search heuristics are close to the optimal routing given by the linear programming model. However, in the European network the results are somewhat different as Figure 8.4 reveals. Both heuristics improve performance compared to inverse capacity weight setting but neither of them are close to the optimal routing.

In comparison with previous studies we see that our findings confirm the results of Fortz and Thorup [3] who use a real network topology and a partial traffic matrix derived from Netflow measurements as well as synthetic data. Söderqvist [10] use synthetic topologies and traffic demands with power-law properties to show that optimizing weights improve network performance considerably compared to inverse capacity weight setting.

Optimizing Weights Using Estimated Traffic Demands

A somewhat controversial assumption made in the previous section is that an exact measure of the traffic matrix is available. In this section we investigate how the search heuristics perform when they are subjected to estimated traffic demands.

We focus on two well known estimation methods. The gravity method and the entropy method. Both methods have been evaluated on the data set we use in this study [5]. The simple gravity method was shown to give a surprisingly accurate estimate in the European network despite its simplicity. In the American network, on the other hand, the gravity methods failed to give an accurate estimate of the traffic demands due to violation of the gravity assumption. The more sophisticated entropy method produced an accurate estimate for both the European and the American networks.

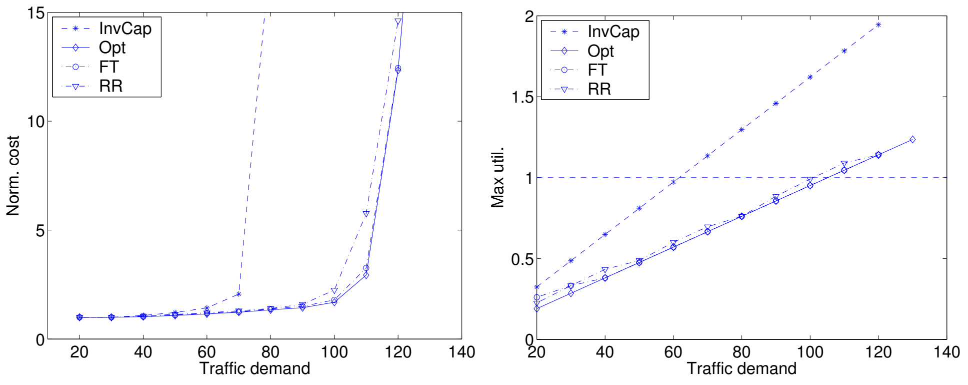


Figure 8.3: Normalized cost (left) and maximum link utilization (right) for the American network

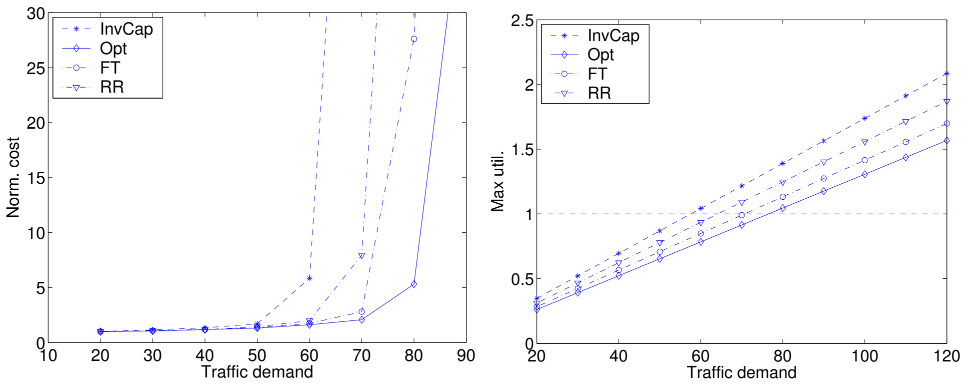
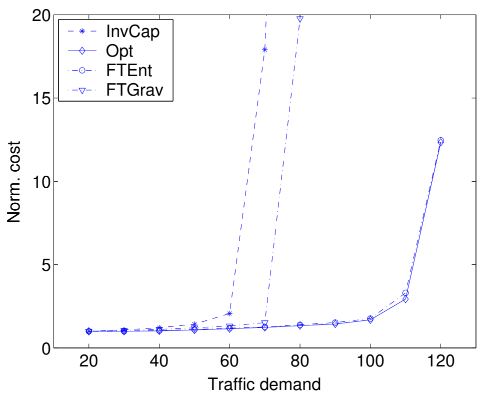


Figure 8.4: Normalized cost (left) and maximum link utilization (right) for the European network

In the plot to the left in Figure 8.5 we have plotted normalized cost as a function of traffic demands for the local search heuristic in the American network. The plot indicates that estimation using the more advanced entropy method has a negligible effect on performance. However, when the optimization is based on the less accurate gravity model performance is degraded considerably. In the European network (Figure 8.5 right), where the gravity model is more accurate, the heuristics have similar performance. The same experiment has been conducted using descending search producing similar results as local search. But we have omitted the plots for descending search due to space limitations.



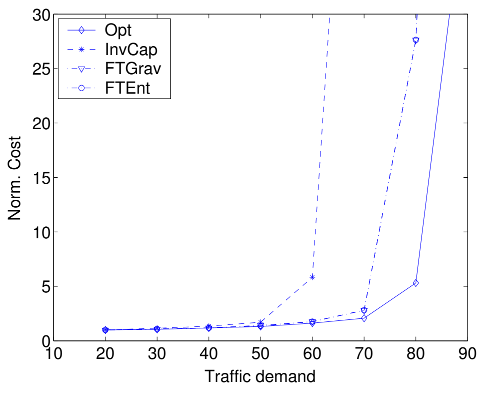


Figure 8.5: Normalized cost for the American network (left) and the European network (right) using estimated traffic matrices

8.5 Conclusions and Future Work

This paper has investigated how traffic engineering performs in a real network with real traffic demands from a commercial IP network operator. The traffic engineering methods are based on search heuristics for weight settings in link state routing. From the study we concluded that both search heuristics are able to find weight settings which are able to accommodate substantially more traffic in the network than the default inverse capacity weight setting and come close in performance to the optimal solution of the routing problem. In addition, we study the performance of the traffic engineering using estimated traffic demands. We investigate two traffic matrix estimation methods. One simple and one which is more sophisticated and accurate. Our observations indicate that when the optimized weight setting using the estimated traffic matrix from the accurate entropy method is applied to the real traffic demands performance is only degraded marginally. But for the less accurate gravity model performance was degraded significantly in some cases. However, still an improvement compared to the inverse capacity weight setting recommended by Cisco.

Our findings confirm the results form previous studies using partial traffic demands derived from flow measurements or synthetic data [3, 9, 10].

This study has focused on a static traffic matrix. In the future we intend to investigate how the weight setting can be designed to be robust in order to cope with a changing traffic situation in the network.

Acknowledgments

This paper describes work undertaken in the context of the Ambient Networks - Information Society Technologies project, which is partially funded by the Commission of the European Union. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the Ambient Networks Project.

Bibliography

[1] H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimisation. In *Proc. Third COST 263 International Workshop on Quality of Future Internet Services, QoFIS 2002*, pages 135–144, Zürich, Switzerland, October 2002. Springer. LNCS 2511.

[2] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet Traffic Engineering. Internet RFC 3272, May 2002.

[3] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. IEEE INFOCOM*, pages 519–528, Tel-Aviv, Israel, March 2000.

[4] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[5] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a global IP backbone - a comparison on real data. In *Proc. ACM SIGCOMM Internet Measurement Conference*, pages 149–160, Taormina, Italy, October 2004.

[6] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. ACM SIGCOMM*, pages 161–174, Pittsburgh, Pennsylvania, USA, August 2002.

[7] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Commmunication and Computer Networks*. Morgan Kaufmann, 2004.

[8] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 2001.

[9] M. Roughan, Mikkel Thorup, and Yin Zhang. Traffic engineering with estimated traffic matrices. In *Proc. ACM Internet Measurement Conference*, pages 248–258, Miami Beach, Florida, USA, October 2003.

[10] M. Söderqvist. Search Heuristics for Load Balancing in IP-networks. Technical Report T2005:04, SICS – Swedish Institute of Computer Science, March 2005.

[11] A. Sridarhan, R. Guerin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. In *Proc. IEEE INFOCOM*, San Francisco, California, USA , November 2003.

[12] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proc. IEEE INFOCOM*, pages 565–571, Anchorage, Alaska, USA, May 2001.

[13] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information theoretic approach to traffic matrix estimation. In *Proc. ACM SIGCOMM*, pages 301–312, Karlsruhe, Germany, August 2003.

Chapter 9

Paper C: Data-driven traffic engineering: techniques, experiences and challenges

Mikael Johansson, Anders Gunnar. In *Proceedings of the Third International Conference on Broadband Communications, Networks and Systems Broadnets 2006*, San Jose, California, USA, October 2006.

Abstract

This paper presents a global view of measurement-driven traffic engineering, explores the interplay between traffic matrix estimation and routing optimization and demonstrates how demand uncertainties can be accounted for in the optimization step to guarantee a robust and reliable result. Based on a unique data set of complete measured traffic matrices, we quantify the demand uncertainties in an operational IP network and demonstrate how a number of robust optimization schemes allow to find fixed MPLS configurations that are close to the performance limits given by time-varying routing under full demand knowledge. We present a novel scheme for computing a sparse MPLS mesh to complement a baseline routing, and explore how the performance depends on the size of the partial mesh. Corresponding methods for robust OSPF optimization are discussed and a number of challenges are detailed.

9.1 Introduction

Many of the decisions that IP network operators make depend on how the traffic flows in their network. A traffic matrix describes the amount of data transmitted between every pair of ingress and egress nodes. However, in many networks it is hard to directly measure the traffic matrix, and operators are often forced to estimate the point-to-point demands from other available data, typically link load measurements and routing configurations. Estimating traffic matrices in this way is non-trivial, as the problem is heavily under-constrained: there are many more elements in the traffic matrix than there are links in the network. The traffic matrix estimation problem has received considerable attention in the research community during the last couple of years, and a wide variety of methods have been developed [10, 20, 8]. However, the more refined methods make various (model-based) assumptions, and may produce large estimation errors even when the underlying models are reasonably accurate. It is thus important to understand what effects such estimation errors can have on the traffic engineering process. Our investigation will focus on optimizing the maximum link-utilization under no-failure scenario. Earlier studies in this direction [11, 6] have indicated that OSPF weight tuning procedures *tend* to work-well, although a complete understanding of why is still lacking. In the same study, MPLS-optimization via multi-commodity network flows was demonstrated to be very sensitive towards the estimation errors present in typical traffic matrix estimation methods.

The aims of this contribution is to shred new light on the interplay between estimation and optimization, and to evaluate the potential benefits of robust routing solutions on operational IP networks. Making no assumptions on the nature of the underlying traffic demands, we try to find the optimal routing with respect to all demands that are consistent with the link-load observations. We argue for the use of cautionary OSPF-weight tuning procedures that attempt to *guarantee* performance improvements, or at least provide bounds (under the polyhedral traffic model) on the performance after each weight-change. However, when applied to data from an operational IP backbone, we note that a prototypical OSPF-tuning scheme often fails to find weight-changes that guarantee performance improvements, and that the performance-bounds can be very loose. For MPLS-optimization, on the other hand, the situation is very different. We illustrate how recent advances in robust optimization [1, 2] allow us to find routings that are astonishingly robust towards uncertainties in the traffic matrix estimation. In practice, however, an MPLS-enabled network allows detailed information of the traffic matrix, so although the observation is of intellectual interest its practical relevance can be questioned. The key demand uncertainty is then its time-variations. We demonstrate how the robust techniques find fixed MPLS settings that are close to the performance limits (given by time-varying routing under full demand knowledge) for our data set, and compare the robust performance with the rule-of-thumb of performing routing optimization with the busy-hour traffic matrix. Finally, we present a novel scheme for computing a sparse MPLS mesh to complement a baseline routing, and explore how the performance depends on the size of the partial mesh. The paper is concluded by a set of challenges for data-driven traffic engineering.

9.2 Preliminaries

Point-to-point demands and the traffic matrix

We consider a network with N nodes and L directed links. Such a network has $P = N(N-1)$ pair of distinct nodes that may communicate with each other. The aggregate communication rate (in bits/second) between any pair (n, m) of nodes is called the *point-to-point demand* between the nodes, and we will use s_{nm} to denote the rate of the aggregate data traffic that enters the network at node n and exits the network at node m. The matrix $S = [s_{nm}]$ is called the *traffic matrix*. For computational tasks, it is often convenient to represent the traffic matrix in vector form. We then enumerate all P source-destination pairs, and let s_p denote the point-to-point demand of node pair p.

Traffic matrix estimation

For simplicity, assume that each point-to-point demand is routed on a single path. The paths can be represented by a *routing matrix* $R \in \mathbb{R}^{L\times P}$ whose entries r_{lp} are defined as

$$r_{lp} = \begin{cases} 1 & \text{if the demand of node pair } p \text{ is routed across link } l \\ 0 & \text{otherwise} \end{cases}$$

Note that the routing matrix may easily be transformed to reflect a situation where traffic demands are routed on more than one path from source to destination by allowing real values in the routing matrix. Let t_l denote the aggregate data rate on link l, $t = [t_l] \in \mathbb{R}^L$ be the vector of link rates, and $s \in \mathbb{R}^P$ be the vector of demands for all source-destination pairs. Then, s and t are related via

$$Rs = t \tag{9.1}$$

The *traffic matrix estimation problem* is simply the one of estimating the non-negative vector s based on knowledge of R and t. The challenge in this problem comes from the fact that this system of equations tends to be highly underdetermined: there are typically many more source-destination pairs than links in a network, and (9.1) has many more unknowns than equations.

It is important to note that in an IP network setting, not all links are interior links connecting the core routers in the network: some of the links are access and peering links that supply data to and receive data from the edge nodes. To make this more explicit, we introduce the notation $e(n)$ for the link over which demand enters at node n, and $x(m)$ for the link over which demand exits at node m. For ease of notation, we assume that each edge node is either an access or a peering point (if this is not the case we can always introduce artificial nodes in our network representation so that this holds). Under these assumptions, $t_{e(n)}$ is the total traffic entering the network at node n and $t_{x(m)}$ is the total traffic exiting the network at node m.

Routing optimization

By routing optimization, we refer to the process of finding the routing configuration that results in the optimal network performance. We will focus on minimizing the maximum link utilization within an autonomous system. This is by no means a universal performance measure, but it is common in the literature and reasonable in our network. The optimization problem depends on the routing protocol employed within the autonomous system. We will consider both OSPF and MPLS-networks.

In OSPF, packets are forwarded along shortest paths based on their destination only. Associated to each link is a weight, and routers exchange link state information that allow nodes to compute the shortest paths to every other node in the network. In this case, routing optimization amounts to finding the link weights that results in the optimal network performance. The metric optimization problem is NP-hard. For small to moderate-sized networks, the optimal link weights can be computed via mixed-integer-linear programming [3], while larger networks have to be addressed using heuristic search procedures (see, *e.g.*, [5, 13]). We note that there are many additional issues in metric optimization, including exploiting the equal-cost multipath (ECMP) option, and a desire to keep the number of re-routed demands low. Such issues are out of the scope of this paper.

MPLS allows to specify explicit routes (so-called label-switch paths, LSPs), through the network. At the ingress nodes, incoming packets are classified and labels are attached to their headers. Within the autonomous systems, forwarding decisions are based solely on the labels. In the ideal case, where there are no limits on the number of LSPs and ingress nodes can balance the load arbitrarily across the LSPs, the optimal routing can be computed via a classical multicommodity network flow optimization. When including practical constraints, such as limits on the number of LSPs or restricted load balancing across paths, the problem becomes a mixed integer-linear programming problems and only small networks can be configured with guaranteed optimality.

9.3 Data from a global IP backbone

The main source of data for this study is a time-series of complete network traffic matrices based on direct measurements on 5-minute intervals on Global Crossing's IP backbone. The same data set has been used in a previous study to evaluate traffic matrix estimation techniques in isolation [8]. While that paper contains detailed information about the performance of a wide range of estimation approaches, the data set description in this paper focuses only on aspects that will be critical for the data-driven routing optimization.

The data set

Global Crossing uses MPLS for traffic engineering on its backbone. A mesh of label-switched paths has been established between all core routers in the network. Eery LSP has a bandwidth value associated with it, and the core router originating the LSP (head-end) will use a constraint-based routing algorithm (CSPF) to find the shortest path that has the required bandwidth available. The actual path across the network is then set up using RSVP. A detailed description of this architecture can be found in [18]. By measuring the utilization of each LSP in 5 minute intervals using SNMP, we can create a full and accurate traffic matrix of the network [8].

The routing matrix and the associated link loads are created using a shortest-path simulation of the network using the MATE tool [4]. From the global backbone, we extract the European subnetwork on the level of point-of-presence (PoP) to point-of-presence, where each PoP can contain many access and core routers. The demands are given by the internal traffic matrix (*i.e.* no peering or other transit traffic is included). Details about the American subnetwork can be found in [8].

Properties of the demand data set

Figure 9.1 shows how the (normalized) total traffic in the European subnetwork varies with time. There is a clear diurnal cycle and a pronounced busy period (shaded in the Figure). Figure 9.2 shows the spatial demand distribution in the network. The plot demonstrates that a limited subset of nodes account for the majority of the network traffic, and that the traffic distribution between these nodes is relatively uniform. Although it cannot be seen in this picture, about 20% of the demands account for over 80% of the total traffic.

Figure 9.3 indicates how the individual demands vary with time. The plot shows the demands originating in the four PoPs that source the most traffic. Although there are some variations, we see that the main source of time variation is the diurnal cycle related to the usage pattern of the network customers. This observation is taken further in [8] where it is shown how the relative sourced traffic (the "fanouts") is more stable than the demands themselves.

9.4 Traffic matrix estimation

The gravity model

A simple method for estimating the traffic matrix is to use a so-called *gravity model* [15]. In our notation, the basic version of the gravity model predicts the

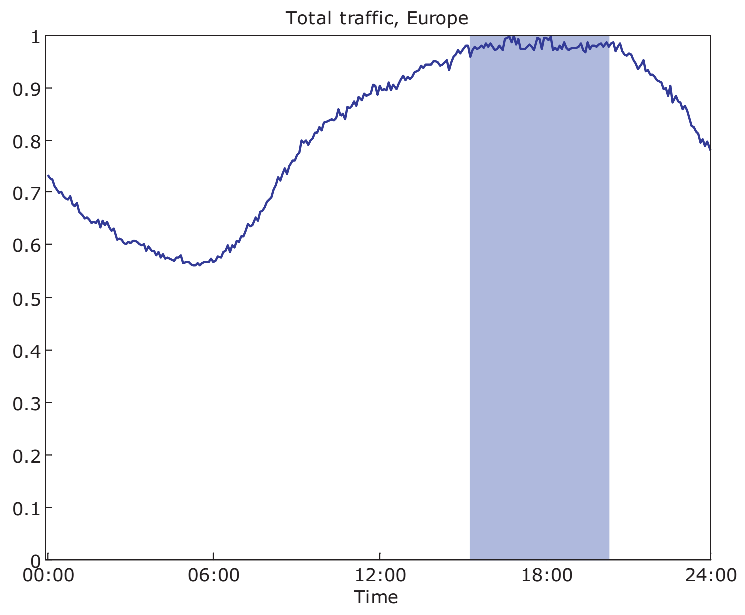


Figure 9.1: The total traffic follows a diurnal cycle and displays a pronounced busy period (shaded time interval).

demand between node n and node m as

$$s_{nm}^{(p)} = C \sum_i s_{im} \sum_j s_{jm} = C t_{e(n)} t_{x(m)} \tag{9.2}$$

where $t_{e(n)}$ and $t_{x(m)}$ denote the total traffic entering the network at PoP n and existing the network at PoP m, respectively, while C is a normalization constant that makes the sum of estimated demands equal to the measured total network traffic. With the choice $C = 1/\sum_m t_{x(m)}$, the gravity model reduces to

$$s_{nm}^{(p)} = \frac{t_{x(m)}}{\sum_m t_{x(m)}} t_{e(n)}$$

i.e., the amount of data that node n sends to node m is assumed to be proportional to the fraction of the total network traffic that exits at node m. Such a model makes sense if the user populations served by different nodes are relatively uniform. As we have seen from the spatial demand distribution, this assumption appears to be reasonable for the European subnetwork. The evaluation of the gravity model shown in Figure 9.4 demonstrates that it is quite inaccurate, but captures the general trend.

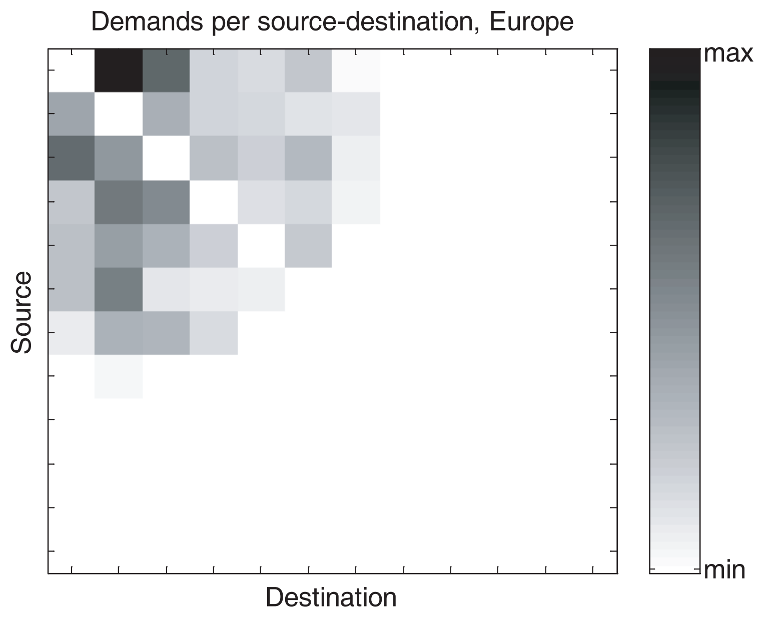


Figure 9.2: The spatial demand distribution: the color of element (i, j) indicates the average busy hour traffic from PoP i to PoP j.

The tomogravity approach

The gravity model is typically not used in isolation, but combined with approaches that utilize also the individual link loads in the network. Although several methods exist, we will focus on the so-called tomogravity approach which strikes a nice balance between simplicity and accuracy (see *e.g.*, [8]). The basic idea of the method is to find the estimate "closest" to the gravity model which satisfies the link load relations $Rs = t$, see Figure 9.5. This estimation problem can be written as

$$\begin{array}{ll} \text{minimize} & D(s, s^{(p)}) \\ \text{subject to} & Rs = t, \qquad s \succeq 0 \end{array} \tag{9.3}$$

where $D(s, s^{(p)})$ denotes some distance measure. Typically, D is either taken to be the Euclidean distance (squared) or the Kullback-Leibler divergence [20]. In practice, routing matrices and link load data can contain errors and it may be impossible to satisfy the link-load relations exactly. It is then more convenient to use

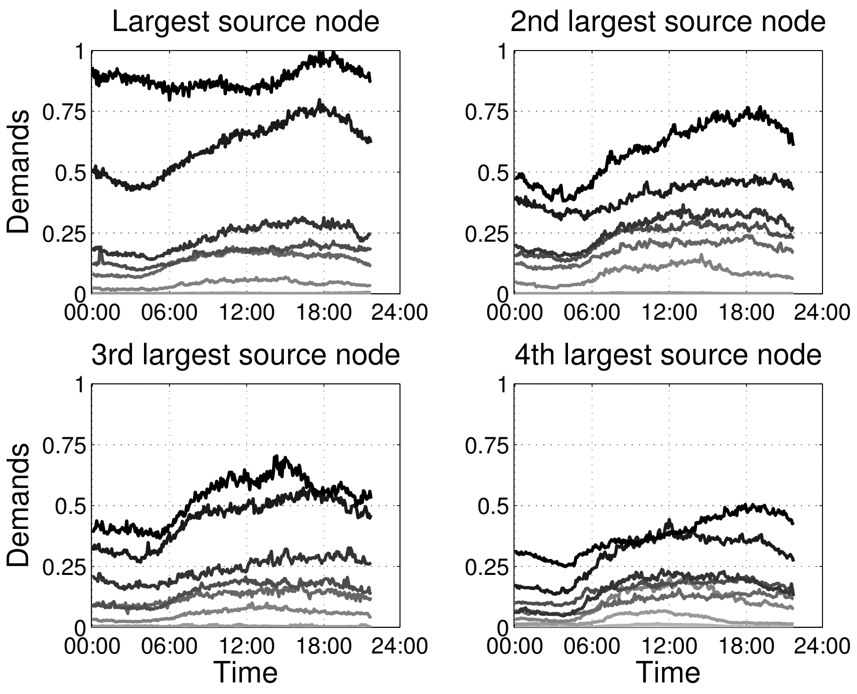


Figure 9.3: The demands from the four PoPs that source the most traffic.

the alternative formulation

$$\begin{array}{ll} \text{minimize} & D(s, s^{(p)}) + \sigma^2 \|Rs - t\|_2^2 \\ \text{subject to} & s \succeq 0 \end{array}$$

The parameter σ specifies how important it is that the estimate satisfies the link load relations relative to how important it is that it stays close to the gravity model estimate.

An evaluation of the formulation (9.3) with the distance function taken as the Euclidean distance is shown in Figure 9.6. We can see how including the link load information results in a drastically improved estimate, while certain elements of the traffic matrix estimate remain inaccurate.

Worst-case bounds

As discussed above, the traffic matrix problem is typically heavily under-constrained: a single snapshot t of the link-loads could be generated by any traffic matrix in the set

$$\mathcal{S} = \{s \succeq 0 \mid Rs = t\}$$

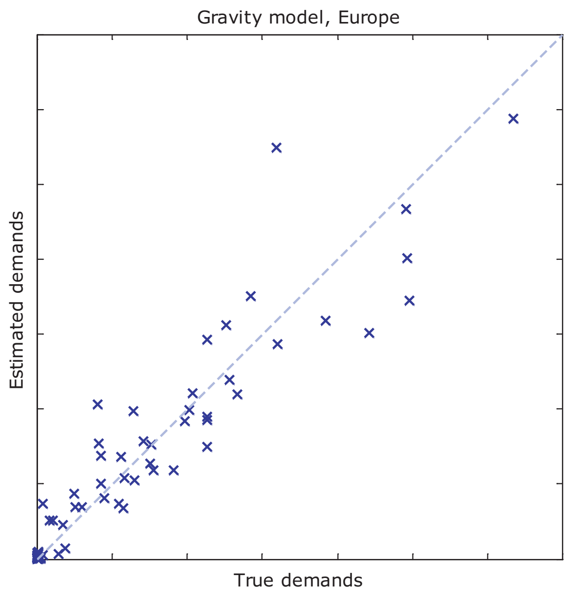


Figure 9.4: The gravity model applied to the Europan subnetwork. The demands predicted by the gravity model vs the true demands.

Conversely, if we do not introduce any other (*e.g.* model-based) assumptions on the underlying demands, the only thing we know with certainty is that the true demand lies in the set $\mathcal{S}$. Thus, an upper bound on demand p can be computed by solving the linear programming problem

$$\begin{array}{ll} \text{maximize} & s_p \\ \text{subject to} & Rs = t, \quad s \succeq 0 \end{array}$$

and the associated lower bound is found by minimizing s_p subject to the constraints. Note that this method is computationally expensive, as it may require solving two linear programs to bound each demand.

Figure 9.7 shows the worst-case bounds for the demands in the European subnetwork. Although most bounds are non-trivial, they tend to be relatively loose and only very few demands can be guaranteed to be estimated exactly. We can observe that the largest demands in the European subnetwork have relatively large worst-case bounds, indicating potentially large uncertainty in estimates.

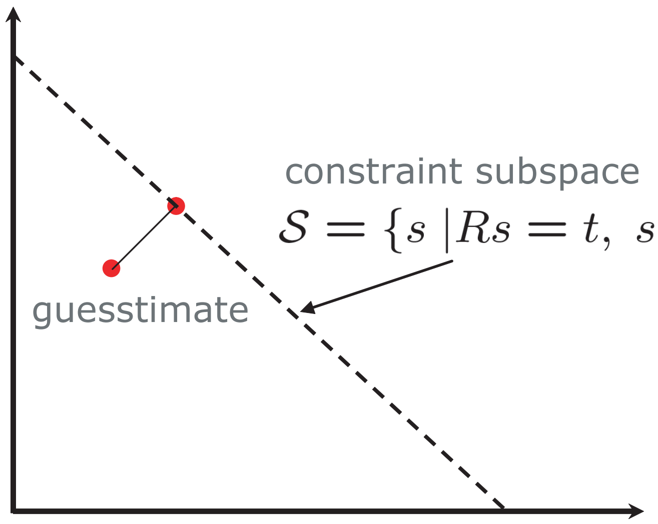


Figure 9.5: The basic idea of the tomogravity method: find the estimate closest to the gravity model prediction that satisfies the link load relations.

9.5 Robust routing

Robustness, referring to the ability to cope with variations from the nominal operating conditions, is a key property of any engineering system. In this spirit, a robust network should be able to sustain acceptable performance despite foreseeable traffic variations and component failures. A common optimization objective in robust networking is to minimize the worst-case link loads, where worst-case should be understood as over all potential load variations or component failures. Our focus is on demand variations.

It is sometimes claimed that while the demands themselves are hard to estimate, the worst-case link loads are easy. Such claims are often motivated from studies such as the one shown in Figure 9.8 where true worst-case link loads are compared to estimated link loads under all single-link failures. The accuracy is astonishing, but its relevance to routing optimization is limited. In this scenario, the routing is fixed and the link loads are computed for all link failures. Since the worst-case link load typically occurs when all demands on a failing link are rerouted over the same secondary path, the additional load on the secondary path is perfectly known, and so are the resulting link loads. The situation is drastically different when a routing optimization is performed and only a subset of demands are re-routed. In this case, the amount of re-routed data is typically hard to estimate with and the link loads after the routing change can be very different from the predictions.

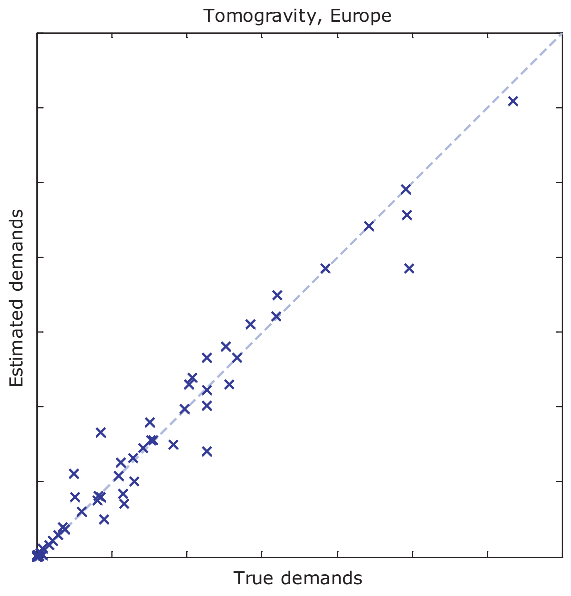


Figure 9.6: True vs. estimated demands for the tomogravity model. The approach improves substantially over the gravity model, but certain elements of the traffic matrix estimate remain inaccurate.

Robust MPLS routing

We will start by evaluating the potential of robust MPLS routing. Several methods for robust routing have been proposed recently [1, 16, 2]. We will base our developments on the approach by Ben-Ameur and Kerivin [2] as we find it the most transparent. The method starts out from a standard arc-path formulation of multicommodity network flows

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \displaystyle\sum_k \sum_{p \in \mathcal{P}_k} r_{lp} \alpha_{pk} s_k \leq c_l u_{\max} \, \forall l \\ & \displaystyle\sum_{p \in \mathcal{P}_k} \alpha_{pk} = 1, \qquad \alpha_{pk} \geq 0 \end{array} \tag{9.4}$$

Here, s_k is the aggregate traffic between source-destination pair k, $\mathcal{P}_k$ is the set of all paths between source-destination pair k and r_{lp} is an indicator variable taking the value one if path p traverses link l and zero otherwise. The optimization variables α_{pk} determine what fraction of the traffic between source destination pair k that is routed across path p. The first set of constraints state that the total traffic

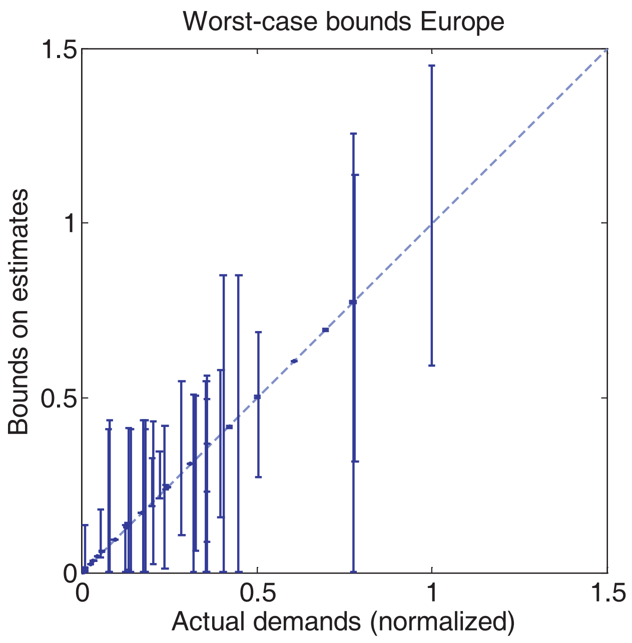


Figure 9.7: True demands vs. their worst-case bounds. Most bounds are relatively loose, indicating large demand uncertainties.

across each link l is bounded by the link capacity times the maximal link utilization, while the second constraint states that all traffic must be routed across some path. The classical way of solving (9.4) is by column generation. Rather than explicitly enumerating all paths in the network, one starts out with a small subset of paths (*e.g.*, the shortest-hop routing) and then sequentially adds new paths to the problem to improve the optimization objective, see *e.g.*, [12].

The robust multicommodity network flow problem is to find the routing that guarantees the smallest link utilization for all feasible traffic scenarios (that is, for all $s \in \mathcal{S}$). We can formulate the problem as

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \displaystyle\sum_k \sum_{p \in \mathcal{P}_k} r_{lp} \alpha_{pk} s_k \le c_l u_{\max} \, \forall l, \forall s \in \mathcal{S} \\ & \displaystyle\sum_{p \in \mathcal{P}_k} \alpha_{pk} = 1, \qquad \alpha_{pk} \ge 0 \end{array} \tag{9.5}$$

Depending on the nature of the traffic uncertainty set $\mathcal{S}$, this problem may or may not admit an efficient solution. If the traffic uncertainty is polyhedral $\mathcal{S} =$

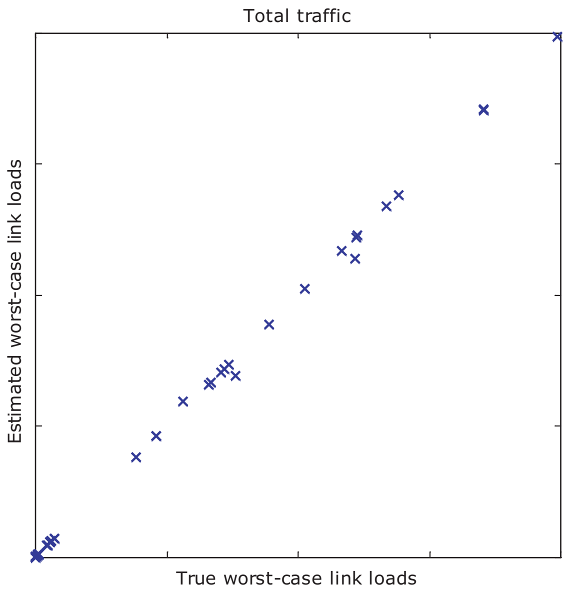


Figure 9.8: True and estimated link worst-case link loads over all link failures.

co$\{s^{(1)}, \cdots, s^{(V)}\}$, then (9.5) can be equivalently expressed as

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \sum_{k} \sum_{p \in \mathcal{P}_k} r_{lp} \alpha_{pk} s_k^{(v)} \leq c_l u_{\max} \; \forall l, v \\ & \sum_{p \in \mathcal{P}_k} \alpha_{pk} = 1, \qquad \alpha_{pk} \geq 0 \end{array}$$

There are at least two problems with this formulation. First, the traffic uncertainty sets are typically not given in vertex form, but as the set of solutions to a system of linear inequalities (cf. the demand uncertainty set $\mathcal{S}$ in Section 9.4). Secondly, the uncertainty set may have many vertices, so that explicit enumeration is computationally unattractive. These two issues can be addressed similarly to the way column generation is used to avoid explicit enumeration of all paths in the nominal formulation: one starts out with a single traffic scenario in the uncertainty set, solves the routing problem, and then verifies whether the computed routing satisfies the link constraints for all feasible traffic loads. If this is not the case, one adds the traffic matrix that violates the constraints the most to the vertex description of the uncertainty set and repeats. The resulting method is a combined column- and constraint generation scheme, and is readily shown to have finite convergence (*e.g.* [2]).

Evaluation

Coping with traffic estimation uncertainty The first investigation considers the ability of robust routing to cope with traffic estimation uncertainties. We consider a single snapshot scenario and assume that we only know the current routing configuration and link loads. Based on this data we compare two approaches: the first is to estimate a single traffic matrix using the tomogravity approach and then perform MPLS-routing based on this traffic matrix; the second one is robust MPLS-routing with the uncertainty set

$$\mathcal{S} = \{s \succeq 0 \mid Rs = t\}$$

These two schemes are compared with the ideal situation of MPLS optimization using the true traffic matrix. To arrive at a judged evaluation, we evaluate the performance of the different schemes for each five-minute interval in our data set. The results are shown in Figure 9.9. While routing optimization using the estimated traffic matrix is very fragile and results in a large performance loss (the maximum link load is on average 29% higher than the ideal case) the robust approach shows remarkably good performance as it consistently comes within a few percent of the ideal performance (the average increase in actual link-utilization is less than 2%).

Although this investigation could be potentially useful for MPLS migration, it is fair to argue that its relevance is intellectual rather than practical. It does show that robust routing can be performed at a very low opportunity cost, despite the large demand uncertainties. However, in an operational MPLS network, the traffic matrix is known and the key uncertainties are the time variations in the demands. We will investigate these uncertainties next.

Coping with time-varying demands Since our dataset contains internal traffic only, the dominating variation is the diurnal usage pattern; we have not been able to discern any effects that could be associated with BGP reroutes, flashcrowds, or other common sources for demand variations.

Classical rule-of-thumbs for routing time-varying demands include using the busy-hour traffic matrix or a traffic matrix with the maximal demands over the relevant time period. The robust optimization framework allows for an alternative: it can be used to compute the single routing that minimizes the worst-case link utilization over the full time-series of (predicted) traffic matrices. To investigate the potential of this approach, we compare three schemes: MPLS routing with the busy-hour traffic matrix, robust routing assuming that the complete time-series of traffic matrices is known in advance, and an ideal case where the routing is re-computed at each sample using the current traffic matrix. The results are shown in Figure 9.10.

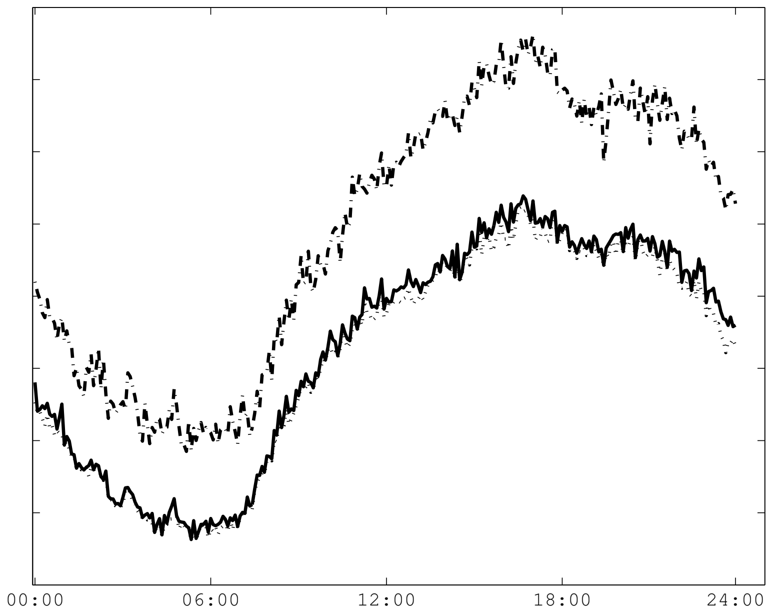


Figure 9.9: Routing with an estimated traffic matrix (dash-dotted line) results in large increases in the maximum link load, while the robust approach (full line) comes very close to the performance of the ideal scenario (dotted). Although the actual link utilizations are not shown, routing with estimated demands results in an average performance loss of about 29%, while robust routing can be performed with less than 2% performance loss.

In this case, the benefits are much harder to discern: although the robust routing improves over the busy-hour traffic matrix approach, the benefits are minimal. One reason for the disappointing performance can be found by inspecting the correlations between the traffic matrix elements. In our dataset, 95% of the demands are positively correlated (compare Figure 9.3). Thus, when one demand increases the others tend to increase at the same time, and there is little room for "statistical multiplexing" on the granularity of aggregated end-to-end demands.

An extension: sparse robust routing

The robust routing algorithm has finite convergence, but may generate a very large number of explicit paths. Operating an MPLS network with many explicit paths (in the extreme, a fully meshed network) increases the burden on the network operator and the increased management effort can quickly outweigh the benefits of having a network with reliable link utilizations. Thus, from a practical perspective it is very interesting to investigate means of reducing the number of paths while

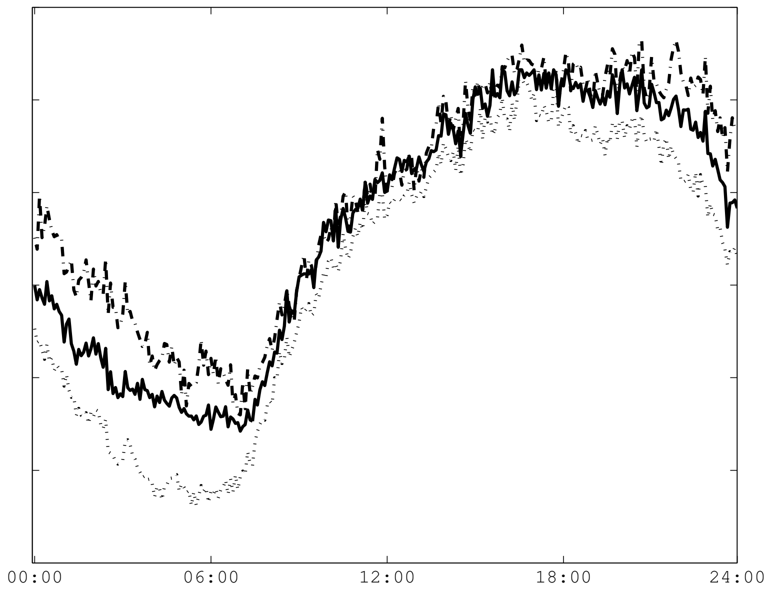


Figure 9.10: Time-variations

maintaining a robust network performance. We have approached this problem by first solving the robust MPLS routing problem (9.5), and then entering a "sparsification phase", by solving the mixed-integer linear programming problem

$$\begin{array}{ll}\text{minimize} & \sum_p x_p \\ \text{subject to} & \alpha_p \leq x_p, \quad x_p \in \{0,1\} \\ & \displaystyle\sum_k \sum_{p \in \mathcal{P}_k} r_{lp} \alpha_{pk} s_k^{(v)} \leq c_l u_{tgt} \, \forall l, v \\ & \displaystyle\sum_{p \in \mathcal{P}_k} \alpha_{pk} = 1, \qquad \alpha_{pk} \geq 0 \end{array}$$

Here $s^{(v)}$ are the extremal traffic matrices generated while solving (9.5) and the binary variables x_p indicate whether or not path p is needed for maintaining the optimal performance. The objective is to minimize the total number of active paths. In this formulation, the target link utilization u_{tgt} is fixed and ideally equal to the value $u_{\max}$ computed via (9.5). However, to reduce the number of paths in the solution even further, it can be useful to introduce some additional slack by letting $u_{tgt} = (1+\varepsilon)u_{\max}$ for some small $\varepsilon \geq 0$. Note that the computed routing needs to be validated against the uncertain traffic demands: it may happen that a new traffic scenario is the worst-case for the reduced routing. In this case, this scenario needs to be added to the formulation followed by a re-optimization.

To evaluate the sparse robust MPLS routing, we return to the estimation uncertainty scenario used in Section 9.5. In the sparsification step, we do not attempt to remove the initial paths obtained by the shortest-hop routing. The purpose with this is to look for routing composed of a baseline metric routing complemented by a partial MPLS mesh. The results of this study is shown in Figure 9.11. By varying the parameter ε, we can compute the minimal number of tunnels needed for a range of worst-case link utilizations. Note, in particular, that by introducing an additional 10 tunnels, the performance is around 5% worse than optimal.

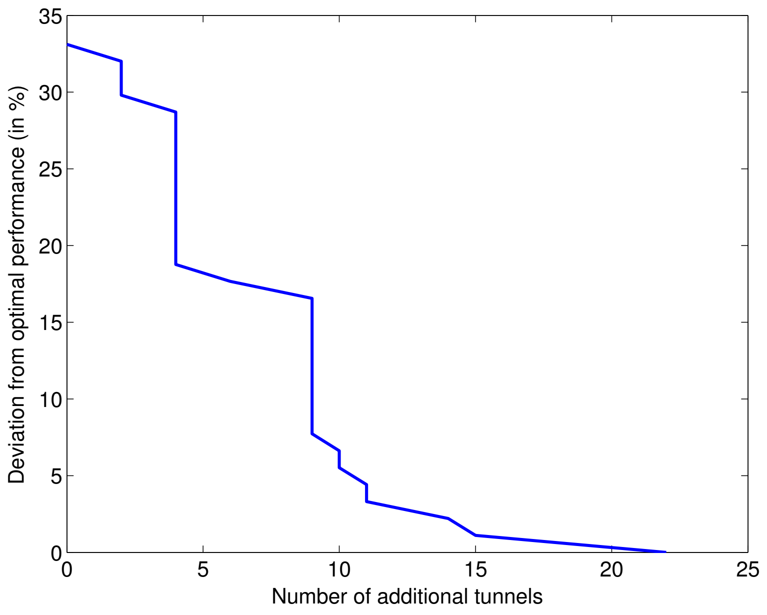


Figure 9.11: Performance loss vs. number of additional tunnels (over the baseline shortest path routing): with only 10 additional tunnels, the performance is about 5% from the optimal MPLS routing.

Cautionary OSPF-tuning

Following the successful development of robust MPLS-routing optimization, it is natural to look for similar developments also for OSPF networks.

Many heuristics have been proposed for tuning of OSPF weights under the assumption of known traffic matrix (see, *e.g.*, [5, 13, 12]). We will use the method of Ramakrishnan and Rodrigues, since it is transparent and intuitive while comparative studies (*e.g.* [7]) have indicated performance close to the more refined methods. The method is a greedy local search from a given set of initial OSPF link weights, trying in each step to find the weight change that gives the largest performance improvement. A key insight behind the algorithm is the following:

as a single link weight is increased, OD flows will be diverted from that link in the order of their secondary path lengths. The secondary paths can be computed by temporarily removing the associated link from the network topology and routing the OD pairs according to the shortest path weights in the reduced topology. The OD pair with shortest secondary path length will be diverted first, and the re-routing occurs as soon as the link weight is increased more than the difference between the secondary and primary path lengths. The influence of the weight-change on the performance can be evaluated using a simple routing simulation, *i.e.* if R^+ be the routing matrix for the new weight settings, then the associated link loads are given by $t^+ = R^+ s$. To find which link weight to change, the procedure is applied to all links in the network. The algorithm terminates when no single link weight change can be found that improves the performance.

Although algorithms of this kind have been found to *tend* to work well also on estimated traffic matrices, the method could potentially be misled by estimation errors and suggest link changes that result in worse performance than the initial routing. A cautionary approach to OSPF tuning can be found by combing the worst-case estimation with the heuristic tuning procedure, trying to find the weight change with largest *guaranteed* performance improvement, or at least provide bounds on the performance after each suggested weight change. Let $(r_l^+)^T$ be the l^{th} row of the routing matrix corresponding to the new link weight settings. Then, the worst-case load on link l, consistent with the observations, can be computed via the linear program

$$\begin{array}{ll} \text{maximize} & (r_l^+)^T s \\ \text{subject to} & Rs = t, \quad s \succeq 0 \end{array}$$

The performance of the cautionary OSPF tuning procedure are shown in Figure 9.12. For our network and dataset, there is no single weight change that can guarantee strictly improved routing performance. However, the cautionary approach can guarantee that the new routing will not be worse than the current, and it would be safe to attempt executing the changes. However, the situation is not always this good. To illustrate a worse situation, we consider the NSFnet topology and traffic matrix from [14]. The results of a run of the tuning procedure is shown in Figure 9.12(bottom), where we see that the algorithm terminates after six weight changes, decreasing the expected maximum link utilization from an initial 97.99% down to 69.38%. However, a worst-case analysis of the routing indicates that the maximum link load with the new routing could be up to 76.75% worse than the initial. In this case, there is a substantial uncertainty in the final result and executing the link-weight changes requires a considerable leap of faith.

9.6 Conclusions and challenges

This paper has advocated a global view of the data-driven traffic engineering process and argued that demand uncertainties should be accounted for in the routing

```
-----------------------------------------------------
Iteration   Estimated max-utilization   Worst-case bound
-----------------------------------------------------
        0                u_nom          u_nom
        1                0.93u_nom      u_nom
        2                0.91u_nom      u_nom
        6                0.91u_nom  <no changes detected>
-----------------------------------------------------

-----------------------------------------------------
Iteration   Estimated max-utilization   Worst-case bound
-----------------------------------------------------
        0                   97.99%       97.99%
        1                   86.27%      111.38%
        2                   74.04%      141.97%
        3                   70.37%      141.97%
        4                   69.76%      141.97%
        5                   69.55%      176.75%
        6                   69.55%  <no changes detected>
-----------------------------------------------------
```

Figure 9.12: OSPF weight tuning for the Global Crossing network (top) and for the NSFnet topology, starting from unity link weights (bottom). For the NSFnet, the maximum link utilization is estimated to 69.3% but not guaranteed to be below 176.75%.

optimization so as to guarantee a robust and reliable overall result. Based on a unique data set of complete measured traffic matrices, we have quantified the demand uncertainties in an operational IP network and demonstrated how robust optimization schemes allow to find fixed MPLS configurations that are close to the performance limits given by time-varying routing under full demand knowledge. We have presented a novel scheme for computing a sparse MPLS mesh to complement a baseline routing, and explored how the performance depends on the size of the partial mesh. A corresponding development for OSPF networks resulted in a proposal for cautionary weight tuning that attempts to guarantee performance improvements, or at least provide bounds (under the polyhedral traffic model) on the performance after each weight-change. However, when applied to data from an operational IP backbone, we noted that a prototypical OSPF-tuning scheme often fails to find weight-changes that guarantee performance improvements, and that the bounds can be very loose.

There are many natural extensions of this work as many challenges remain.

First of all, a better understanding of the fundamental limitations of robust routing needs to be developed, along with a better insight into what features of the robust MPLS solution that enable its strong performance. The investigations in this paper are largely based on a single dataset. Although the data comes from a commercial and operational IP backbone and is clearly relevant, it does not include peering traffic and some important sources for demand variations are thus not present. Other data sets have recently become public, including those from the Geant and Abiline networks [19, 17]. While the traffic patterns in the experimental Abiline network are very different from our dataset, the traffic situation in the Geant network is an interesting intermediate between the two. However, the maximum link load is a poor performance objective in the Geant network, since the most loaded links are low-capacity links that carry data for which there is no alternative path. By the appropriate modification of the objective functions in this paper, the investigations could be extended to include data from the Geant network. Finally, it is not clear whether time-varying demands should be addressed using proactive (e.g. robust) or reactive (dynamic, adaptive) methods such as [9]. The answer to this problem involves many additional issues, but a better understanding could have far-reaching design implications for future generation networks.

Bibliography

[1] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental issues. In *ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[2] W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. *Optimization and Engineering*, 6(3):283–313, 2005.

[3] A. Bley, M. Grötschel, and R. Wessäly. *Robust communication networks: interconnection and survivability*, volume 53 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, chapter Design of broadband virtual private networks: model and heuristics for the B-WiN, pages 1–16. AMS, 1998.

[4] Cariden, Inc. Mate. http://www.cariden.com, 2005.

[5] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *IEEE Infocom*, Tel-Aviv, Israel, 2000.

[6] B. Fortz and M. Thorup. Optimizing ospf/is-is weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[7] A. Gunnar, H. Abrahamsson, and M. Söderqvist. Performance of traffic engineering in operationl IP networks – an experimental study. In *IEEE International Workshop on IP Operations & Management*, Barcelona, Spain, October 2005.

[8] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large IP backbone – a comparison on real data. In *ACM Internet Measurement Conference*, Taormina, Italy, October 2004.

[9] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *ACM SIGCOMM*, Philadelphia, PA, August 2005.

[10] A. Lakhina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[11] M. Thorup M. Roughan and Y. Zhang. Traffic engineering with estimated traffic matrices. In *ACM Internet Measurement Conference*, Miami, FL, 2003.

[12] M. Pioro and D. Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[13] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest-path data networks. *Bell Labs Technical Journal*, 6(1):117–138, 2001.

[14] R. Ramaswami and S. N. Kumar. Design of logical topologies for wavelength-routed optical networks. *IEEE Journal on Selected Areas in Communications*, 14:840–851, June 1996.

[15] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in modeling backbone traffic variability: models, metrics, measurements and meaning. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.

[16] A. Sridharan, R. Guérin, C. Diot, and S. Bhattacharyya. The impact of traffic granuarity of robustness of traffic aware routing. Technical report, University of Pennsylvania, 2004. Available via http://einstein.seas.upenn.edu/mnlab.

[17] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86, January 2006.

[18] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni. Traffic engineering with MPLS in the Internet. *IEEE Network*, 14(2):28–33, March–April 2000.

[19] Y. Zhang. Abilene traffic matrix data set. Available via http://www.cs.utexas.edu/˜yzhang/, 2005.

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

Chapter 10

Paper D: Robust load balancing under traffic uncertainty-tractable models and efficient algorithms

Anders Gunnar, Mikael Johansson. In *Telecommunication Systems Journal*, In Press.

Abstract

Routing configurations that have been optimized for a nominal traffic scenario often display significant performance degradation when they are subjected to real network traffic. These degradations are due to the inherent sensitivity of classical optimization techniques to changes in model parameters combined with the significant traffic variations caused by demand fluctuations, component failures and network reconfigurations. In this paper, we review important sources for traffic variations in data networks and describe tractable models for capturing the associated traffic uncertainty. We demonstrate how robust routing settings with guaranteed performance for all foreseen traffic variations can be effectively computed via memory efficient iterative techniques and polynomial-time algorithms. The techniques are illustrated on real data from operational IP networks.

10.1 Introduction

Appropriate load balancing in a data network allows the service provider to accommodate larger traffic volumes or to provide more stable and predictable performance without the need for additional capacity. For a given traffic situation, the problem of finding a routing setting that maintains a desired service level using a minimum amount of network resources can be formulated and solved as a mathematical programming problem [26]. The traffic situation is usually estimated based on a traffic model in combination with measurements [16, 28, 38]. Due to the stochastic nature of the traffic, estimates of the traffic matrix from a finite number of data will always contain inaccuracies and the network traffic will be different from, and sometimes significantly different from, the nominal case it was optimized for. In addition to statistical fluctuations, large traffic shifts are caused by component failures and routing parameter or policy reconfigurations. Hence, it is important that the routing setting is insensitive, or *robust*, to deviations from the nominal case. As shown by multiple authors, the classical routing optimization based on linear programming is very sensitive to errors in the estimated traffic matrix [27] (this follows also from the broader observation that linear programming problems are sensitive to uncertainty in the problem data [5]). With robust routing it is possible to optimize the routing setting not only for a nominal traffic situation but for a *set* of traffic situations. As long as the actual traffic stays within the bounds assumed in the optimization, the actual performance is guaranteed to be better or equal to the optimization objective.

In this paper we revisit, summarize and expand our experiences reported in conference publications [14, 15, 18]. We assume that the network can be monitored and controlled by a single entity and that the internal network topology remains stable, i.e. we disregard node and link failures and focus on variations in the traffic. We discuss sources of traffic variations, ranging from time-variations in demands to large traffic shifts due to interdomain routing changes, and outline how the variations can be modelled in a way that allows us to efficiently optimize the routing settings. We describe an iterative linear programming method which combines column and constraint generation techniques to find a robust routing setting in a finite number of optimization steps. We then discuss how statistical uncertainties arising from the stochastic behavior of data traffic can be dealt with and how routing settings with guaranteed performance can be found in polynomial time. The effectiveness of the methods are demonstrated on real data from operational networks and insight gained in our studies is discussed.

The paper is organized as follows: we begin with a short review of routing and load balancing in the Internet, followed by an introduction of the mathematical notation used in the paper. Section 10.4 identifies a number of important sources for traffic variations in data networks and introduces set-based models for capturing the uncertainty. Efficient optimization algorithms for solving the associated robust routing problems are developed in Section 10.5. Our methodology is evaluated in numerical examples with real data from operational networks in

Section 10.6. Finally, related work is discussed in Section 10.7 and conclusions are drawn in Section 10.8.

10.2 Routing and load balancing in the Internet

Since the Internet is managed by different organizations known as Internet Service Providers (ISPs), the network is partitioned into subnetworks called Autonomous Systems (AS:es) and the routing is divided between intradomain routing inside an AS and interdomain routing between AS:es.

The routing inside an AS is managed by an Interior Gateway Protocol (IGP). Typically, the IGP is a link state routing protocol like Intermediate System Intermediate System (IS-IS) or Open Shortest Path First (OSPF). In link state routing, the network is modeled as a graph where nodes represent routers and arcs represent links connecting the routers. Associated to each link is a weight reflecting the cost of sending traffic over the link. Nodes in the network advertise which nodes they connect to and the associated link costs in Links State Advertisements (LSA) that are flooded in the network. Each node collects information in the LSA:s and builds a map of the network topology. The shortest path to each destination node in the network can then be calculated using Dijkstra's algorithm. The traffic is controlled by adjusting the link costs to deviate traffic from congested links to under-utilized ones. A variant of shortest path routing is Equal Cost Multi-Path (ECMP) where traffic is split evenly over paths with the same cost to the destination. ECMP relaxes some of the constraints imposed by shortest path routing only to a limited extent. A difficulty with link-state routing as it is implemented today is that it is difficult to measure the end-to-end traffic demands. Operators often have to resort to estimating the traffic demands from incomplete data or deriving the traffic demands from sampled flow measurements.

Although forwarding traffic along shortest paths is simple and easy to implement, it is rather coarse. The forwarding is based on the destination address and all traffic follows the same path to the destination. A more fine-grained forwarding can be implemented with Multi Protocol Label Switching (MPLS). With MPLS, label switched paths are set up between ingress and egress node pairs. The ingress router selects a label for an incoming packet based on criteria such as destination, source/destination combination, or traffic class. Packets following the same paths are grouped into a Forwarding Equivalence Class (FEC), and are forwarded along the path based on the label until they reach the egress router where the label is removed. Since MPLS allows traffic to be forwarded arbitrarily in the network, it is easier to optimize than shortest-path routing. One common approach for computing label switched paths is to use Constrained Shortest Path First (CSPF). In CSPF links in the network that do not meet a given criteria are removed from the routing calculations. The shortest paths are then calculated on the reduced topology in the same manner as in shortest path routing. More sophisticated routing optimizations can also be done, for example using the framework of Multi Commodity

Network Flows (MCNF) [2, 26]. The advantage of MCNF is that the resulting routing setting can be optimized for a given objective. However, the optimal routing is sometimes inconvenient to implement since the optimal traffic split often introduces many paths between each pair of ingress and egress routers. Another advantage with deploying MPLS is that it easy to measure the end-to-end traffic demands by simply monitoring the traffic on each label switched path.

In order to connect the AS:es, an External Gateway Protocol is used. ISPs usually want to apply policies reflecting the business relation between neighboring AS:es when exchanging routing information. Typical business relations are customer, provider and peer relations. A customer AS pays a provider AS for connectivity to the rest of the Internet. However, AS:es that exchange large amounts of traffic set up peering links between themselves to avoid going through a provider AS. Today, only a small group of ISPs are not a customer of another ISP. This group of ISPs, known as Tier-1 operators, peer with each other in order to obtain connectivity to the entire Internet. The interdomain routing protocol currently in use is called Border Gateway Protocol (BGP). In BGP, an AS sends announcements to its neighboring AS:es to share the network prefixes for which it has a route. The prefixes consist of the network address together with a mask hiding the least significant bits. In addition, the routing announcements have a variety of attributes to express policies associated with announced networks. Many multi-homed AS:es receive route announcements for the same prefix from more than one neighboring AS. The selection of next hop for a prefix follows a specified procedure where the BGP-specific attributes are examined followed by a comparison of distance to next-hop according to the intradomain routing protocol. In many cases, the intradomain routing decides exit point for the prefix, as the operator typically tries to select the closest exit point according to the intradomain routing cost (a policy often referred to as hot potato routing [32]). A more detailed description of BGP4 can be found in [17].

From the perspective of the technology overview given above, the aim of this paper is to identify sources for uncertainty in the aggregate traffic demands between routers within an autonomous system, introduce models for capturing these variations, and develop efficient algorithms for computing MPLS routing settings that are robust to the foreseen traffic variations.

10.3 Notation

We will now describe the notation used in this paper. The network topology is represented as a graph $G = (N, E)$, where N denotes the set of nodes (routers) and E is the set of edges (links) connecting the nodes. We let $O(n)$ and $I(n)$ be the set of links outgoing and incoming to node n, respectively. For each $l \in E$ we associate a number c_l representing the capacity of the link l.

The traffic demand between origin node o and destination node d is denoted s_{od}, and the set of all demands is called the *traffic matrix*. It is sometimes convenient to organize the elements of the traffic matrix into a vector $s = [s_p]$. We then use $o(p)$ and $d(p)$ to denote the origin and destination node of origin-destination (OD) pair p and let P be the set of all OD pairs. Note that for a network with $|N|$ nodes, $|P| = |N|(|N| - 1)$.

A routing setting is characterized by an $|E| \times |P|$ routing matrix $R = [r_{lp}]$ whose elements r_{lp} represent the fraction of traffic demand s_p routed over link l. In this notation, the total traffic t_l across link l is given by

$$t_l = \sum_{p \in P} r_{lp} s_p = r_l^T s \tag{10.1}$$

where r_l^T is the lth row of the routing matrix R. The vector $t = [t_l] \in \mathbb{R}^{|E|}$ of total traffic is given by

$$t = Rs \tag{10.2}$$

and the utilization u_l of link l is defined as

$$u_l = t_l / c_l = c_l^{-1} r_l^T s.$$

10.4 Traffic variations: sources and models

The first step towards computing robust routing settings is to identify important sources of traffic variations and develop models for the associated uncertainties. An important requirement is that the models should enable for a tractable routing optimization. To this end, we will consider set-based uncertainty models which will allow the optimal routing settings to be found using algorithms which have finite convergence and, in some cases, even polynomial-time complexity.

Sources of traffic variations

It is well-known that data traffic varies according to a diurnal pattern with distinct busy periods and times with relatively low load. This diurnal pattern tends to shift over the days in the week, creating a weekly traffic pattern which changes slowly with time (see Fig. 10). Traditionally, routing decisions in well-provisioned networks have been taken with the respect to the maximal perceived demands or the busy hour traffic situation. However, in today's global backbones, busy periods do not concur across continents and traffic from business and residential users follow different patterns. It is becoming increasingly important to understand and explore such traffic variations in routing decisions.

Most networks have some type of support for measuring the traffic demands. One approach is to conduct high-precision flow measurements with e.g. Netflow and process the flow data with the routing configuration to derive the traffic matrix [10]. This approach has several drawbacks: performing flow measurements are resource consuming for a router both in terms of CPU usage and memory consumption. Sampling can be used to reduce the computational burden but at the expense of measurement accuracy; furthermore, flow measurements produce large amounts of data that needs to be sent to a central unit for processing.

An alternative is to estimate the traffic matrix from link load measurements [16, 24, 30, 35, 38]. Link loads are readily obtained by SNMP and represented by a small amount of data: one counter for each link in the network. The traffic matrix problem is then to estimate the traffic matrix from the measured link load and the current routing matrix, cf. Equation 10.2. The challenge of this problem is that the system of equations (10.2) tends to be highly under-determined: there are typically many more point-to-point demands than there are links in the network. Additional information must be added to the problem in order to find an accurate estimate of the traffic, but numerous studies on real data have shown that the estimation errors still tend to be significant [10, 16].

There is always a component of randomness in the traffic due to unpredictable user and application behavior. Thus, a time-series of observations of traffic demands can be seen as samples from a probability distribution. Depending on where in the network (e.g. within a local area network or in a Tier-1 backbone) and on what time scale the traffic is observed the characteristics differ significantly. When it comes to traffic matrix estimation most work has assumed Poisson [35] and the Gaussian distributions [7]. The theoretically most elegant results have been achieved for Poisson traffic, partly since the mean and variance of the distribution coincide which simplifies the estimation process. However, it has been demonstrated that the Poisson distribution does not give a satisfactory description of the behavior of the traffic at the granularity and time scale we are interested in [16]. Much better results are achieved by a Gaussian distribution with a parameterized power-law relationship between mean and variance [16, 19]. Under this assumption, one can additionally characterize the estimation errors incurred by maximum likelihood estimation of the traffic matrix based on link-counts [6].

Many large traffic shifts that occur in a network are due to changes in the interdomain routing or interactions between inter and intradomain routing [31, 32, 33]. On the one hand changes in interdomain routing parameters influence the selection of next hop by BGP, as explained in Section 10.2. On the other hand, changes in OSPF/IS-IS link weights influence the way BGP selects next hop for prefixes announced by more than one peering point. Changes in neighboring AS:es due to reconfigurations or component failures also affect the set of possible exit points and, hence, how the traffic flows in the network.

The discussion above indicates that it can be dangerous to optimize the routing for a single traffic scenario. In what follows, we will introduce models that describe *set of traffic scenarios* and how to optimize the routing setting to achieve guaranteed performance for all traffic realizations in this set.

Polytopic models

An efficient way to represent a full set of traffic situations is to identify a limited number “extreme" scenarios and consider all realizations in their convex hull:

$$\mathcal{S}_v = \{s = \sum_{v=1}^{V} \theta^{(v)} s^{(v)} \mid \theta^{(v)} \geq 0, \sum_{v=1}^{V} \theta^{(v)} = 1\} \tag{10.3}$$

Geometrically, the set $\mathcal{S}$ is a polytope with the extreme scenarios $s^{(v)}$ as vertices (extreme points).

In some cases, polytopic traffic uncertainty is not given on the form (10.3), but as the solution set to a system of linear inequalities

$$\mathcal{S}_h = \{s \mid a_i^T s \leq b_i, i = 1, \ldots, m\} \tag{10.4}$$

Geometrically, we can think of this representation as the intersection of m halfspaces. Although one could potentially write the set $\mathcal{S}_h$ on the vertex form (10.3) by first finding its extreme points (this procedure is called vertex enumeration), this operation is computationally demanding, partially since a given polytope can have a large number of vertices. Our algorithms will be able to deal with both representations.

Modelling time-varying demands. As a first example of polytopic models, consider the problem of finding a fixed routing that gives good performance at all times despite weekly traffic variations. From a prediction $\{\hat{s}(1), \ldots, \hat{s}(T)\}$ of traffic over the coming week, we can construct an associated polytopic uncertainty $\mathcal{S}_v$ by using the predicted traffic matrices $\hat{s}(t)$ as vertices. By finding a routing setting with guaranteed performance for all $s \in \mathcal{S}_v$ we will not only guarantee this performance for the predicted demands, but also for all realizations that can be written as a convex combination of the predictions. Although methodologies for traffic prediction is outside the scope of this paper, we note that methods can range from simply considering all traffic matrices that have been measured in the past to advanced time-series prediction techniques [25].

Modelling estimation uncertainty. The half-space representation of demand uncertainty $\mathcal{S}_h$ appears naturally when we want to optimize the routing setting but have to estimate the traffic matrix from link load measurements. Then, even if the routing matrix R is known, s could potentially be any realization in the polytope

$$\mathcal{S}_{est} = \{s \mid Rs = t,\ s \succeq 0\}$$

In this case, it is is natural to try to find a routing setting which is insensitive to the estimation uncertainty and guarantees good performance for all possible demands. The following example illustrates this situation. Consider a simple line network with three nodes a, b and c, and two unidirectional links $a \to b$ and

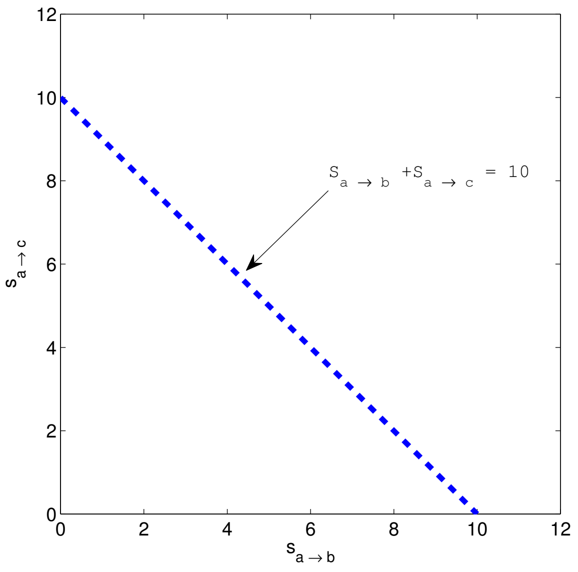


Figure 10.1: Dashed line represent possible values of the traffic demands $s_{a\to b}$ and $s_{a\to c}$ estimated from link load.

$b \to c$. There are three flows, $s_{a\to b}$, $s_{b\to c}$ and $s_{a\to c}$ and $s = \begin{pmatrix} s_{a\to b} & s_{b\to c} & s_{a\to c}\end{pmatrix}$. The sending rate of each traffic demand is set to five units of traffic. Figure 10.1 shows possible values of estimated traffic demands $s_{a\to b}$ and $s_{b\to c}$ constrained by $t_{a\to b} = 10$.

Although $\mathcal{S}_{est}$ is not on the form (10.4), it is readily put into the standard form by rewriting the linear equalities $r_l^T s = t_l$ as the double inequalities $r_l^T s \le t_l$ and $r_l^T s \ge t_l$.

Modelling traffic-shifts due to BGP reroutes. Polytopic uncertainties also appear in route selection for interdomain routing. Several studies indicate that the selection of egress point for prefixes announced to BGP by neighboring AS:es influence a large fraction of the traffic in the network [14, 31, 33]. It thus makes sense to find intradomain routing settings that are insensitive to these traffic shifts [14]. To model the associated traffic uncertainty, let t_{op} denote the traffic to prefix p originating at node o. Introduce $E(p)$ as the set of egress nodes for prefix p (i.e., the set of peering routers that could potentially announce prefix p) and, conversely, $P(d)$ as the set of prefixes that can be announced by peers connected to egress node d.

Now consider the set

$$\mathcal{S}_{BGP} = \{s_{od} = \sum_{p \in P(d)} t_{op}\delta_{pd} \mid$$
$$\sum_{d \in E(p)} \delta_{pd} = 1,\ \delta_{pd} \geq 0 \text{ and } \delta_{pd} = 0 \text{ for } d \notin E(p)\}$$

In this formulation δ_{pd} can be interpreted as the fraction of traffic demand destined to prefix p that exits the AS via egress node d. The set $\mathcal{S}_{BGP}$ contains all traffic matrices that can result from BGP-reroutes, as long as each prefix is announced by at least one peer. A routing setting with guaranteed performance for all $s \in \mathcal{S}_{BGP}$ will hence be insensitive to BGP changes.

Note that $\mathcal{S}_{BGP}$ is the convex hull of all possible traffic matrices under BGP reroutes and should not be seen as a model of BGP as such. For example, a peering AS can only decide whether or not to announce a certain prefix and not the fraction of demand for a specific prefix that it is willing to route.

Ellipsoidal models

Although polytopic models capture many important traffic uncertainties, it is sometimes natural to consider ellipsoidal uncertainty descriptions,

$$\mathcal{S}_e = \{s \mid (s - \bar{s})^T \Sigma^{-1} (s - \bar{s}) \leq 1\} \tag{10.5}$$

One reason for this is computational: while a polytope might have many vertices, hence be computationally complex to manipulate and use in routing optimization, ellipsoids have a compact description and a fixed number of parameters (in a given dimension). In fact, it might even be advantageous to approximate an uncertainty set which is polytopic in nature by an ellipsoid to improve computational efficiency.

Modelling stochastic demand variations. Ellipsoidal uncertainty appears naturally when we consider stochastic traffic models, e.g. when the traffic is assumed to follow a normal distribution

$$s \sim \mathcal{N}(\bar{s}, \Sigma) \tag{10.6}$$

In the literature on traffic matrix estimation, the mean and variance are often related via a power-law

$$\Sigma = \text{diag}(\phi \bar{s}^c) \tag{10.7}$$

where ϕ and c are scalar parameters. This model, introduced by Cao *et al.* [7], has been verified on many different real-world datasets [16, 19].

When the traffic is stochastic, it is not possible to find a routing with guaranteed performance for all possible realizations. However, we can focus on the most likely traffic scenarios using the concept of *likelihood regions*. Specifically, when the traffic is Gaussian, the most likely outcomes of s are ellipsoids

$$\mathcal{E}_\alpha = \{s \mid (s-\bar{s})^T\Sigma^{-1}(s-\bar{s}) \leq \alpha^2\} \tag{10.8}$$

To make $\mathcal{E}_\alpha$ an $(100\gamma)\%$ confidence region for s, we have to set $\alpha^2 = \chi^2(1-\gamma, |P|)$, *i.e.*, the upper $(100\gamma)\%$-point of the Chi-square distribution with $|P|$ degrees of freedom. As shown in [8], many variations of the setup, including when $\bar{s}$ and Σ are unknown and estimated from a sequence of realizations of s also yield ellipsoidal likelihood regions.

Clearly, if we can ensure that the linear inequality

$$r_l^T s \leq c_l u_{\max}$$

holds for all $s \in \mathcal{E}_\alpha$, then the probability that the inequality holds exceeds $(100\gamma)\%$. To have more precise control of the probability of constraint violation, we can also consider probabilistic requirements on the form

$$\text{Prob}\{r_l^T s \leq c_l u_{max}\} \geq 1-\epsilon \tag{10.9}$$

as explained in Sect. 10.5.

Stochastic models of estimation uncertainty An interesting variation arises when the traffic cannot be measured and has to be estimated from measurements of the link loads t_l and the routing matrix R, *e.g.* from relations on the form (10.2). In [20], it was shown that the maximum-likelihood estimator of the model parameters $(\bar{s}, \phi, c)$ under the traffic model (10.6,10.7) ensures that the estimated traffic $\hat{s}$ is asymptotically normal

$$\hat{s} \sim \mathcal{N}(\bar{s}, \mathcal{I}^{-1}) \tag{10.10}$$

where $\mathcal{I}^{-1}$ is the inverse (of the relevant part) of the Fischer information matrix. Analytical expressions for $\mathcal{I}$ are derived in [6] and are omitted here.

To illustrate the stochastic traffic models, consider the set-up from Example 10.4 and let $s = \begin{pmatrix} s_{a\to b} & s_{b\to c} & s_{a\to c}\end{pmatrix}$ follows the traffic model (10.7) with parameters $\bar{s} = \begin{pmatrix} 5 & 5 & 5 \end{pmatrix}$, $c = 1.5$ and $\phi = 0.01$. Figure 10.2 shows 1000 realizations of the first and third component of the traffic itself (left) and the estimated traffic (right). We note again that the uncertainty is significantly larger when the traffic has to be estimated, and that the uncertainty is limited in the direction $s_{a\to b}+s_{a\to c}$ since this quantity is the load over the first link which we assume can be measured without errors. Since the traffic itself is stochastic, there is a small variation also in this direction. Finally, Figure 10.3 shows examples of the confidence ellipsoids for both the traffic and the estimated traffic.

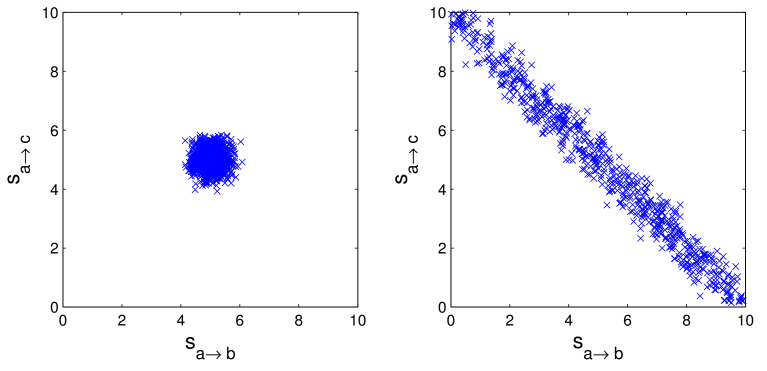


Figure 10.2: Traffic realizations (left)and estimated traffic (right) of $s_{a\to b}$ and $s_{a\to c}$.

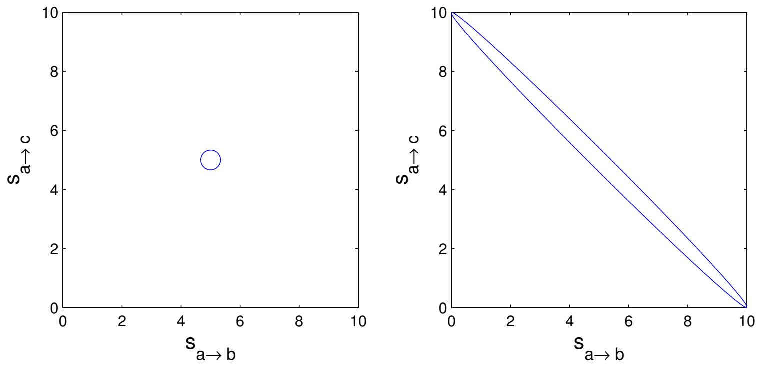


Figure 10.3: Confidence ellipsoids for real traffic (left) and estimated (right).

10.5 Robust optimization

Robustness, referring to the ability to cope with variations from the nominal operating conditions, is a key property of any engineering system. In this spirit, a robust network should be able to sustain acceptable performance despite foreseeable traffic variations and component failures. Our focus in this paper is robustness against traffic variations, and we will understand the term *robust load balancing* as the problem of finding a fixed MPLS routing setting that gives good performance for all traffic matrices in a given scenario set. While many performance measures are possible, we will focus on minimizing the worst-case link load.

Robust load balancing under polytopic uncertainty

Several methods for robust routing under polytopic uncertainty have been proposed recently [3, 4, 18, 29]. Our developments are based on the approach by Ben-Ameur and Kerivin [4]. The method starts our from a standard arc-path formulation of a multi commodity network flow problem

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \sum_p \sum_{\pi\in\Pi_p} \rho_{l\pi}\alpha_{\pi p}s_p \le c_l u_{\max}\,\forall l \\ & \sum_{\pi\in\Pi_p} \alpha_{\pi p} = 1, \qquad \alpha_{\pi p}\ge 0 \end{array} \tag{10.11}$$

Here, s_p is the aggregate traffic between the p^{th} source-destination, Π_p is the set of all paths between source-destination pair p and $\rho_{l\pi}$ is an indicator variable taking the value one if path π traverses link l and zero otherwise. The optimization variables $\alpha_{\pi p}$ determine what fraction of the traffic between source destination pair p that is routed across path π. The first set of constraints state that the total traffic across each link l is bounded by the link capacity multiplied by the maximal link utilization, while the second constraint states that all traffic must be routed across some path.

The classical way of solving (10.11) is by column generation: rather than explicitly enumerating all paths in the network, one starts out with a small subset of paths (*e.g.*, the shortest-hop routes) and then sequentially adds new paths to the problem to improve the optimization objective. The new paths are found by solving the dual linear programming problem associated with (10.11) and interpreting the optimal dual variables of the link capacity constraints as link weights. Solving the shortest path problem with these link weights and adding the paths to the path set results in an algorithm with guaranteed convergence, see *e.g.*, [26].

The robust load balancing problem is now to find a routing which minimizes the maximum link utilization that can be guaranteed for all feasible traffic scenarios:

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \sum_p \sum_{\pi\in\Pi_p} \rho_{l\pi}\alpha_{\pi p}s_p \le c_l u_{\max}\,\forall l, \forall s\in\mathcal{S} \\ & \sum_{\pi\in\Pi_p} \alpha_{\pi p} = 1, \qquad \alpha_{\pi p}\ge 0 \end{array} \tag{10.12}$$

Note that we now optimize for *all* traffic matrices in the scenario set $\mathcal{S}$ rather than for a single traffic scenario. Depending on the nature of the traffic uncertainty set $\mathcal{S}$, this problem may or may not admit an efficient solution.

If the traffic uncertainty is a polytope given on the form (10.3), then (10.12) can be equivalently expressed as

$$\begin{array}{ll} \text{minimize} & u_{\max} \\ \text{subject to} & \sum_{p} \sum_{\pi \in \Pi_p} \rho_{l\pi} \alpha_{\pi p} s_p^{(v)} \leq c_l u_{\max} \, \forall l, v \\ & \sum_{\pi \in \Pi_p} \alpha_{\pi p} = 1, \qquad \alpha_{\pi p} \geq 0 \end{array} \tag{10.13}$$

There are at least two problems with this formulation. First, in many cases traffic uncertainty sets are not given in vertex form, but as the solution set to a system of linear inequalities (compare our models for traffic matrix estimation uncertainty and BGP reroutes detailed in Section 10.4). Secondly, the uncertainty set may have many vertices and explicit enumeration is computationally unattractive. These two issues can be addressed similarly to the way column generation is used to avoid explicit enumeration of all paths in the nominal formulation: we start out with a single traffic scenario in the uncertainty set, solve the routing problem, and then check if the computed routing satisfies the link constraints for all feasible traffic loads. If this is not the case, we add the traffic matrix that violates the constraints the most to the vertex description of the uncertainty set and repeat the procedure. The resulting method is a combined column- and constraint generation scheme, which can be summarized as follows:

1. Initialize the set Π of paths, e.g. with the paths found by applying Dijkstras algorithm to the administrative link weights from OSPF/IS-IS routing, and initialize $\mathcal{S}_v$ with a single scenario from the traffic uncertainty set $\mathcal{S}$.

2. Solve the linear programming problem (10.13) using column generation, resulting in an expanded path set Π and a lower bound on $u_{\max}$.

3. Fix the current routing and determine the traffic scenario $s^{(wc)} \in \mathcal{S}$ that results in the largest link load. This value is an upper bound on $u_{\max}$.

4. If the upper and lower bounds on $u_{\max}$ differ significantly, add $s^{(wc)}$ to $\mathcal{S}_v$, return to step two and repeat the procedure.

Note that step two returns a lower bound on $u_{\max}$ since not all traffic scenarios are considered in (10.13) until possibly when the algorithm terminates. Similarly, step three returns an upper bound since the routing was optimized without taking this new scenario into account. When the scenario set has a finite number of vertices, the algorithm will terminate in a finite number of optimization steps [4]. Our computational experience, reported in Section 10.6, indicates that only a handful of iterations are needed to find the optimal solution.

While the core of the algorithm remains the same, Step 3 needs to be tailored to the specific traffic uncertainty. We will now detail how this step can be performed when the uncertainty set represents time varying demands, estimation uncertainty, and BGP reroutes.

Scenario generation for time-varying demands From the time series of traffic matrices $\{s(1), ..., s(T)\}$, we form an associated polytopic uncertainty set by taking their convex hull

$$\mathcal{S}_T = \{s = \sum_{t=1}^{T} \theta^{(t)} s(t) \mid \theta^{(t)} \geq 0, \sum_{t=1}^{T} \theta^{(t)} = 1\}$$

Given the optimal variables $\alpha_{\pi p}$ of the optimization problem solved in Step 2, we compute the elements of the corresponding routing matrix

$$r_{lp} = \sum_{\pi \in \Pi_p} \rho_{l\pi} \alpha_{\pi p} \tag{10.14}$$

and identify the worst case traffic scenario as

$$\begin{aligned} s^{(wc)} &=_{s \in \mathcal{S}_T} \{\max_l c_l^{-1} \begin{pmatrix} r_{l1} & \cdots & r_{lP} r_{l1} & \cdots & r_{lP} \end{pmatrix} s\} = \\ &=_{s \in \{s(1), \cdots, s(T)\}} \{\max_l c_l^{-1} \begin{pmatrix} r_{l1} & \cdots & r_{lP} r_{l1} & \cdots & r_{lP} \end{pmatrix} s\} \end{aligned}$$

Note that the worst-case traffic matrix is found by a simple search over the T traffic scenarios.

Scenario generation for estimation uncertainty Next, we consider scenario generation for the situation where the traffic matrix has been estimated from a vector of link loads t_{est} measured under a fixed routing matrix R_{est}.

$$\mathcal{S}_{est} = \{s \mid R_{est} s = t_{est},\ s \succeq 0\}$$

Given the optimal load balancing computed in Step 2, we compute the associated routing matrix as in (10.14). The traffic scenario that causes the maximum link utilization can then be found by solving L linear programming problems, each on the form

$$\begin{array}{ll} \text{maximize} & c_l^{-1} r_l^T s \\ \text{subject to} & R_{est} s = t_{est} \\ & s \succeq 0 \end{array}$$

The problem with the highest objective value identifies the most loaded link, and the corresponding optimizer is the worst-case traffic scenario that should be added to the scenario set.

Scenario generation for BGP-reroutes For fixed internal routing, the utilization of link l can be written as

$$u_l = c_l^{-1} \sum_o \sum_d r_{lod} s_{od}$$

where r_{lod} (calculated from Equation 10.14) is the fraction of traffic from origin node o to destination node d routed over link l. To find the $s \in \mathcal{S}_{BGP}$ which maximizes the maximal link utilization, we need to solve L linear programming problems on the form

$$\begin{array}{ll} \text{maximize} & c_l^{-1} \sum_o \sum_d r_{lod} \sum_{p \in P(d)} d_{op} \delta_{pd} \\ \text{subject to} & \sum_{d \in E(p)} \delta_{pd} = 1 \\ & \delta_{pd} \geq 0,\ \delta_{pd} = 0 \text{ for } d \notin E(p) \end{array} \tag{10.15}$$

Although the number of prefixes can be very large, these problems admit an explicit solution: the worst-case situation is when prefix p is only announced at the destination node d with largest value of $r_{lod}d_{op}$ (i.e. when $\delta_{pd} = 1$ for this egress node and zero for the others). Thus, the worst-case traffic scenarios generated by adjusting the prefix distributions to maximize the worst-case link utilization will be such that each prefix is announced by a single peer only. Hence, the generated traffic scenarios are compatible with plausible BGP behavior.

Robust load-balancing under stochastic and ellipsoidal uncertainty

In this section we demonstrate how one can solve the robust routing problem under statistical and ellipsoidal uncertainty. Our key contribution is to demonstrate how techniques from robust linear programming [5, 23] can be used to develop optimization models that can be solved in polynomial time.

Consider the following network flow formulation for minimizing the maximum link utilization under a given traffic scenario

$$\begin{array}{lll} \text{minimize} & u_{max} & \\ \text{subject to} & r_l^T s \leq c_l u_{\max} & \forall l \in E \\ & A_p r = b_p & \forall p \in P \\ & r_{lp} \geq 0 & \forall l \in E,\, p \in P \end{array} \tag{10.16}$$

The first set of inequalities bound the maximum link utilization. The second set of constraints describe flow-conservation at each node, *i*.e.

$$\sum_{l \in \mathcal{O}(n)} r_{lp} - \sum_{l \in \mathcal{I}(n)} r_{lp} = \begin{cases} 1 & \text{if } n = o(p) \\ -1 & \text{if } n = d(p) \\ 0 & \text{otherwise} \end{cases}$$

for appropriately defined matrices A_p and b_p and

$$r = \begin{pmatrix} r_1^T, & r_2^T, & \dots & r_L^T \end{pmatrix}^T .$$

The robust counterpart to (10.16) is simply given by

$$\begin{array}{lll} \text{minimize} & u_{max} & \\ \text{subject to} & r_l^T s \leq c_l u_{\max} & \forall l \in E,\, \forall s \in \mathcal{S} \\ & A_p r = b_p & \forall p \in P \\ & r_{lp} \geq 0 & \forall l \in E,\, p \in P \end{array}$$

The key observation now is that when the uncertainty set is ellipsoidal, this infinite-dimensional linear programming formulation can be transformed into a finite-dimensional convex programming problem which can be solved in polynomial time. To arrive at this result, first note that we can re-parametrize the ellipsoidal uncertainty set (10.5) as

$$\mathcal{S}_e = \{s = \bar{s} + L^T u \mid \|u\|_2 \leq 1\}$$

where $\Sigma = L^T L$ is the Cholesky decomposition of Σ. In this notation, it is readily verified that

$$r_l^T s \leq c_l u_{\max} \qquad \forall s \in \mathcal{S}_e \tag{10.17}$$

if and only if

$$r_l^T \bar{s} + \|L r_l\|_2 \leq c_l u_{\max}$$

This inequality is convex in the decision vector r_l, and the re-formulated optimization problem

$$\begin{array}{lll} \text{minimize} & u_{max} & \\ \text{subject to} & r_l^T \bar{s} + \|L r_l\|_2 \leq c_l u_{\max} & \forall l \in E \\ & A_p r = b_p & \forall p \in P \\ & r_{lp} \geq 0 & \forall l \in E,\, p \in P \end{array} \tag{10.18}$$

is a second-order cone programming problem, which can be solved in polynomial-time using, *e.g.*, modern interior-point methods (see, *e.g.*, [23]).

As discussed earlier, when the underlying traffic is stochastic, constraints cannot be enforced for all possible realizations but we can consider probabilistic requirements of the form

$$\text{Prob}\{r_l^T s \leq c_l u_{max}\} \geq 1 - \epsilon \tag{10.19}$$

This specification essentially states that the target link utilization u_{max} should be respected with high probability. Following [23], we introduce $z = r_l^T s$ and let $\bar{z}$ be its expected value and σ_z its variance. Then, the requirement can be re-written as

$$\text{Prob}\left\{\frac{z - \bar{z}}{\sqrt{\sigma_z}} \leq \frac{c_l u_{\max} - \bar{z}}{\sqrt{\sigma_z}}\right\}$$

Since $(z - \bar{z})/\sqrt{\sigma_z}$ is a unit-variance normal variable, the probability above is $\Phi((c_l u_{\max} - \bar{z})/\sqrt{\sigma_z})$ where

$$\Phi(z) = \frac{1}{2\pi}\int_{-\infty}^{z} e^{-t^2/2}\, dt$$

The constraint can be re-written as

$$\frac{c_l u_{\max} - \bar{z}}{\sqrt{\sigma_z}} \geq \Phi^{-1}(1-\epsilon)$$

or, equivalently,

$$r_l^T \bar{s} + \Phi^{-1}(1-\epsilon)\|Lr_l\|_2 \leq c_l u_{\max} \tag{10.20}$$

The complete load-balancing problem now reads

$$\begin{array}{ll} \text{minimize} & u_{max} \\ \text{subject to} & r_l^T \bar{s} + \Phi^{-1}(1-\epsilon)\|Lr_l\|_2 \leq c_l u_{\max} \quad \forall l \in E \\ & A_p r = b_p \qquad\qquad\qquad\qquad\qquad\quad \forall p \in P \\ & r_{lp} \geq 0 \qquad\qquad\qquad\qquad\quad \forall l \in E,\, p \in P \end{array}$$

Note that in comparison with a constraint on the nominal traffic, *i*.e., $r_l^T \bar{s} \leq c_l u_{\max}$, the formulations in this section include an additional "protection term" to hedge against traffic variations. In the stochastic case, the size of this protection term can be tuned precisely to match the acceptable constraint violation probability.

It is important to emphasize that although the optimization problems above do not return a set of paths, MPLS settings can easily be derived from the optimal decision variables r_{lp} by tracing how the traffic is split from source to destination node of each demand p.

Constraint generation for ellipsoidal uncertainty. While the algorithms in the previous section have polynomial-time complexity, their memory complexity grows quickly as there are $|N|$ flow constraints for each OD pair. Thus, the number of constraints grow as $|N|^3$ which makes it hard to manage very large networks using this formulation. For classical network flows, it is well-known that path (column) generation methods have superior practical performance on large networks. It thus makes sense to try to apply the combined column-constraint generation scheme also to this scenario.

The main problem with this approach is that confidence ellipsoids do not admit a representation as the convex hull of a finite number of vertices, so the standard convergence results do not hold. However, the method can be applied to compute a routing setting with a prescribed degree of sub-optimality. In this case, we simply proceed as for the polytopic uncertainty, and start out with the polytopic uncertainty set given by a single scenario, *e.g.* $\mathcal{S}_v = \{\bar{s}\}$. We then compute the robust routing (which in the initial iteration coincides with the nominal routing) and compute the associated routing matrix $R = [r_{lp}]$. The worst-case link load under this routing can be computed by comparing the worst-case load

$$c_l u_l^{wc} = \max_{s \in \mathcal{E}_\alpha} r_l^T s = r_l^T \bar{s} + \alpha \|Lr_l\|_2$$

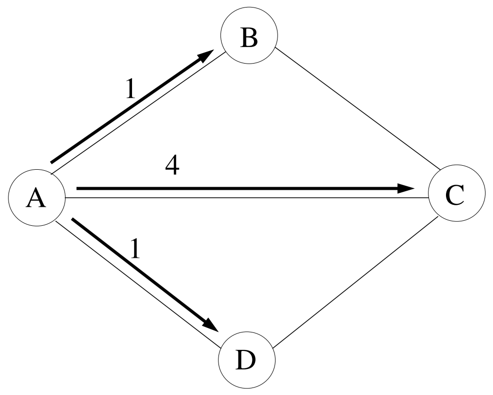


Figure 10.4: Example network with three demands.

on the $|E|$ links in the network. Let $l^{wc} =_l u_l^{wc}$. We add the traffic scenario which generates the maximum link load, *i.e.*,

$$s^{wc} = \bar{s} + \alpha L^T L r_{l^{wc}} / \|L r_{l^{wc}}\|_2 = \bar{s} + \alpha \Sigma r_{l^{wc}} / \|L r_{l^{wc}}\|_2$$

to the scenario set and continue the iterations. Since the algorithm maintains upper and lower bounds on the maximum link utilization, we can terminate the iterations when we have reached a desired solution accuracy.

10.6 Numerical examples

In this section we highlight some properties of the algorithms proposed in this paper and compare the performance of robust routing solutions with that of standard routing optimization for a single nominal scenario.

Basic features of robust routing

To demonstrate some basic features of our proposal we start out with the small network topology depicted in Figure 10.4. The network consists of four nodes (labeled A, B, C and D) and five links. There are three flows: s_{AC} has an average traffic volume of 4 units, while s_{AB} and s_{AD} each have an average traffic volume of one unit. We assume that all links have capacity of ten units and consider robust routing under normally distributed traffic developed in Section 10.5.

The solution which minimizes the maximum link utilization under the nominal traffic scenario maintains single path routing for the two shorter flows, while it splits the flow s_{AC} so that two units are sent on the direct path (A, C) while the remaining traffic is spread evenly over paths (A, B, C) and (A, D, C). The expected maximum link utilization is 20%. However, if we now assume that the flows are

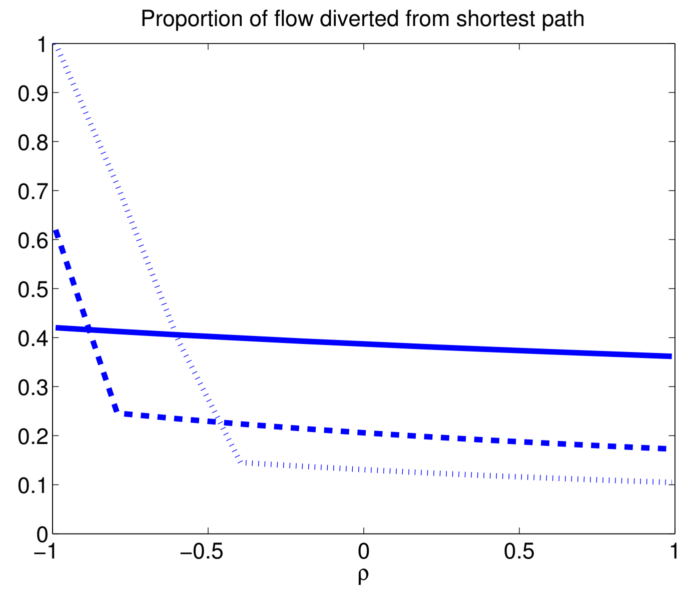


Figure 10.5: Proportion of traffic s_{AC} diverted from direct path as function of correlation coefficient between the traffic and the smaller demands. The results are shown for $\epsilon = 5$ (full) 2 (dashed) and 1% (dotted).

normally distributed with unit variance, then we can see that the variance of the load on links (A, B) and (A, D) is 17/16, while the variance of the load on (A, C) is only 1/4. Thus, in the sense of the probabilistic link load measure (10.19), too much of the flow s_{AC} has been diverted and the links (A, B) and (A, D) have been overloaded. The robust routing algorithm, on the other hand, accounts for variances and correlations in the flows to balance the flows in an optimal way. For reference, aiming at $\varepsilon = 5\%$ in (10.19), the robust routing forwards 2.45 units of data on the direct path and balances the remaining load equally over the two long paths.

The result changes if we introduce a correlation between the large flow and the two smaller ones. If s_{AC} is negatively correlated with the other flows, then a larger part of the traffic will be diverted from the direct path to explore the statistical multiplexing gains with the smaller flows; a positive correlation causes a larger part of the flow to be routed on a direct path to avoid large covariations. Figure 10.5 shows the proportion of s_{AC} which is diverted from the direct path as function of the correlation coefficient between the large flow and the two smaller ones. The larger the covariance matrix, or the smaller the parameter ε in the specifications, the more dramatic is the reaction of the robust routing to demand correlations.

Next, consider the slightly larger topology shown in Figure 10.7. We assume

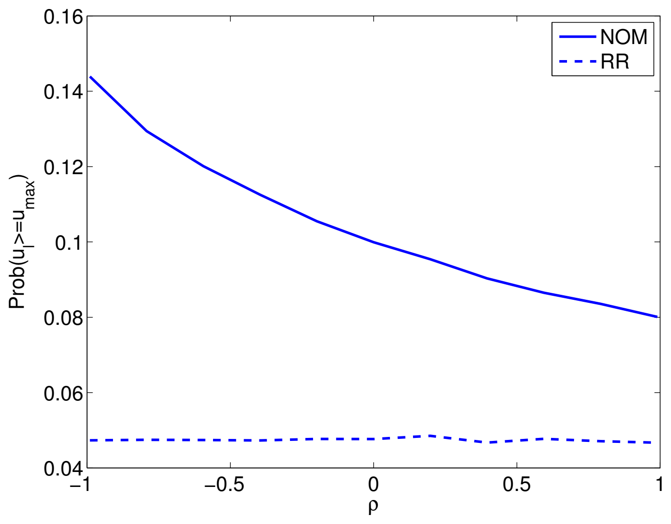


Figure 10.6: Overload probabilities for the robust routing (dashed) and nominal routing (full). Here, $u_{\max}$ refers to the maximum link utilization predicted by the nominal and robust optimization, respectively. Note how the robust routing consistently maintains the overload probability at the target value.

that all nodes have traffic to send to all other nodes, and that all traffic demands are of unit volume. We will also assume that the covariance matrix is of the form

$$\Sigma = I + \frac{\rho}{P-1}(\mathbf{1}\mathbf{1}^T - I)$$

where P is the number of demands in the network. We then vary ρ in the interval $[-1, 1]$, and compare the probability of exceeding the target link loads for robust routing (RR) and the nominal routing (NOM). As can be seen in Figure 10.6, the robust routing maintains the overload probability at the target of 5%, while the nominal routing consistently exceeds the desired overload probability.

Robust routing in an operational IP network

We will now evaluate how our models and algorithms work on real data from the operational network Abilene and GEANT. The Abilene network is a research network in North America with eleven nodes and 28 links [1]. The network topology for the Abilene network is shown in Figure 10.9. The GEANT network [13], depicted in Figure 10.8, connects national research networks in Europe and has 23 nodes and 74 links. We have access to network topology, traffic data and BGP routing information from the data sets made public by Uhlig *et al.* [34]. The measurements were conducted during a four month period and consist of 15 minute

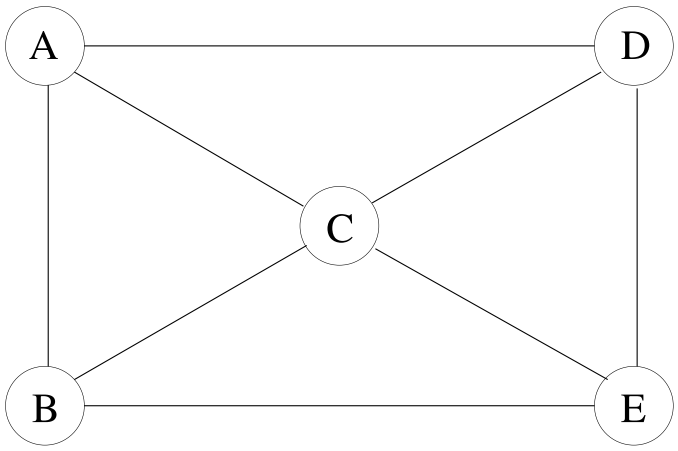


Figure 10.7: Topology for small example network with bidirectional links.

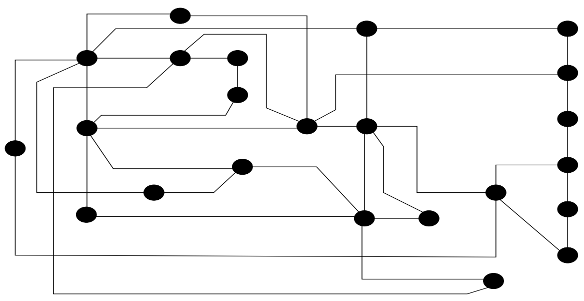


Figure 10.8: Network topology for the GEANT network.

flow exports of sampled Netflow measurements where one out of every thousand packets is sampled. Furthermore, a dump of the BGP routing information base from each day of the measurement period was conducted. Since we use maximum link utilization as objective in our optimization, we upgrade links of 155 Mbps to 2400 Mbps as these links would otherwise remain bottlenecks irrespectively of the routing.

Experiments with time-varying traffic

Figure 10.10 shows the total traffic during each 15 minute interval during one week in the GEANT network. In addition to the random traffic fluctuations, there is a clear daily pattern with distinct busy periods and slightly lower traffic during the weekend. We will focus our analysis on the daily traffic patterns as this variability is much larger than the short-term fluctuations. We want to explore if our method-

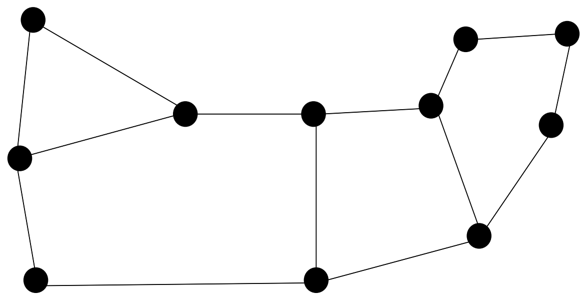


Figure 10.9: Network topology for the Abilene network.

ology is able to find a single routing setting that works well during the whole daily cycle.

For the experiments with statistical and ellipsoidal traffic uncertainty we created the traffic parameter $\bar{s}$ by estimating the average value of end-to-end demands in the data set. Assuming that the traffic is normally distributed and its mean and covariance matrix obey the power-law in Equation (10.7), we estimate the values of ϕ and c from the same data set. From the estimated mean and covariance matrix, we derive a confidence ellipsoid on the form (10.8) and set the parameter α so to include 95% of the traffic scenarios in $\mathcal{E}_\alpha$.

To evaluate the merits of robust routing, we consider several alternatives. A lower bound on the achievable performance is established by recalculating the routing for each traffic scenario. We refer to this routing as OPT. We then consider classical (non-robust) multicommodity network flow optimization for the average traffic demand $\bar{s}$ (referred to as NOM) and the peak demand (called PEAK), respectively. The robust routing (RR) and nominal routing for average and peak demands are then fixed and evaluated on fresh data for a full day. Figure 10.11 shows the maximum link utilization for the different approaches during a 24 hour period in the GEANT network. The robust routing is almost identical to the ideal case where the routing is re-optimized for every new measurement period and significantly outperforms the two classical routing optimizations. Thus, we conclude that the robust routing is able to stay close to optimal for all traffic situations occurring during the day without the need to reconfigure.

Experiments with BGP reroutes

We have analyzed our algorithm for robust routing under BGP reroutes for the traffic data obtained from the GEANT network. A preliminary data analysis outlined in [14] reveal that of the 160 000 prefix present in the routing table at the time only three percent were announced by a single router. The rest of the prefixes were announced by more than one router and can potentially contribute to traffic shifts due to BGP reroutes.

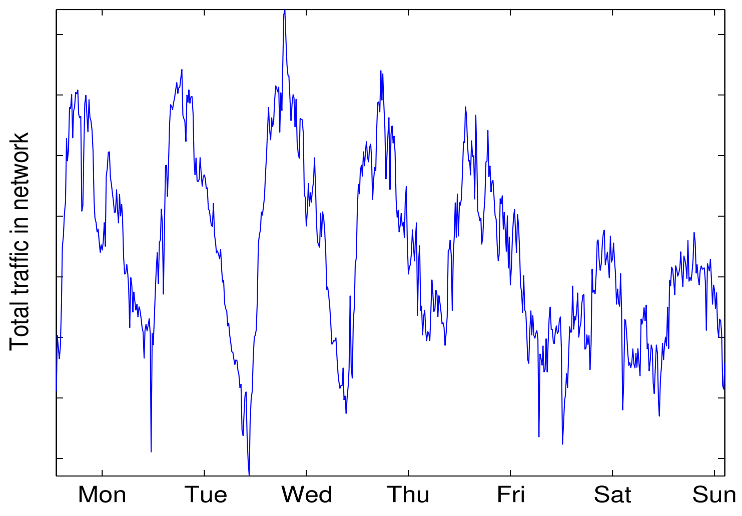


Figure 10.10: Daily traffic variations in the GEANT network during one week.

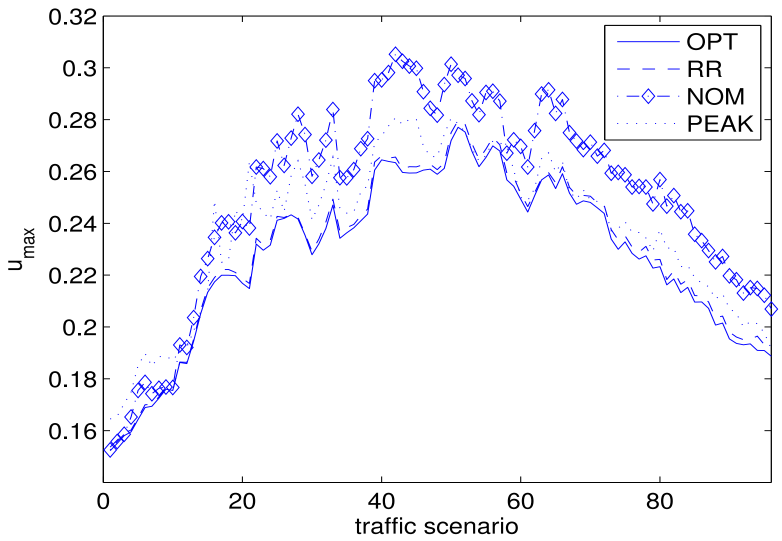


Figure 10.11: Maximum link utilization for robust(RR), optimal (OPT), nominal (NOM) and routing optimized for peak rate (PEAK) for the GEANT network during a 24 hour period.

If we were to take every prefix in the routing table into account when solving the optimization problem in Equation (10.15) this would create a very large optimization problem. However, from the preliminary data analysis in [14] we learned that only a small fraction of the prefixes account for the traffic in the network. Hence, by removing the prefixes with negligible traffic we are able to reduce the number of variables. In our experiments we selected the prefixes that account for 90% of the traffic. The number of prefixes in our equations were reduced to 3600 and the problem can be easily handled on a regular desktop computer.

In our evaluations, we use the traffic scenario where all possible routes are available and link weights are set to the original values used in the operational network as base. We have then evaluated the link loads for the following routing principles:

- ROBUST: The combined constraint/column generation method where worst-case traffic scenarios in the constraint generation phase are computed as in (10.15).
- NOM: Multi commodity network flow routing to minimize the maximum link utilization under the nominal traffic scenario.
- SPF: Shortest path first routing using the original link weights from the GEANT network.

Figure 10.12 shows the utilization for the links in the GEANT network under ROBUST, NOM and SPF routing for the nominal traffic scenario. We can see that although the NOM and ROBUST routing settings achieve the same maximum link utilization, the robust routing achieves a better balance in the overall link utilization. This is partly due to the fact that ROBUST uses more paths for load balancing and that the new paths computed using the dual variables of the link constraints in (10.13) discourages routing across highly loaded links.

In Figure 10.13 we have plotted maximum link utilization under feasible traffic shifts for SPF, NOM and ROBUST under three traffic scenarios (nominal traffic and two worst-case scenarios generated during the robust optimization). The robust routing is able to route efficiently in all three scenarios whereas the multi-commodity network flow routing optimized for the nominal traffic scenario suffers a substantial performance losses under BGP-reroutes, and performs on par with the original shortest-path routing.

Performance and scaling issues

Finally, to illustrate computational performance we conduct experiments also for the Abilene network. Table 10.1 presents memory requirements and computation times for the polynomial time algorithm described in Section 10.5. The table indicates that both memory and computational complexity grows rapidly with problem size. However, if high accuracy is required, the constraint/column generation

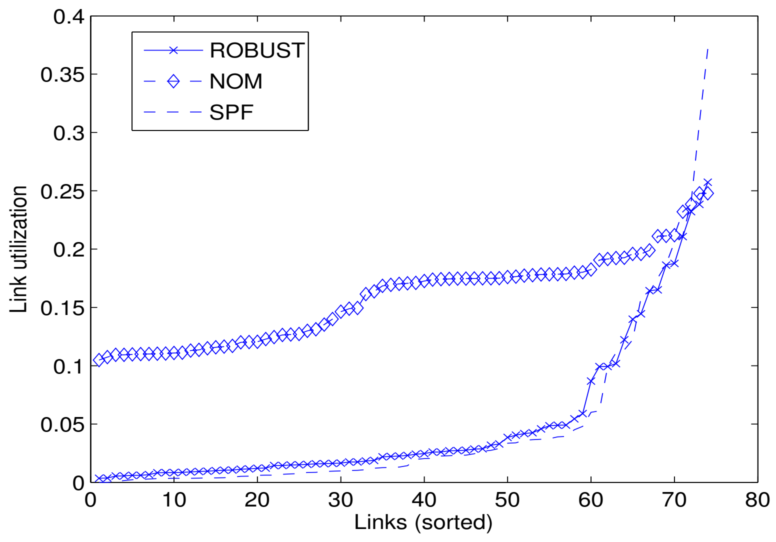


Figure 10.12: Link load for the links in the GEANT network for robust, optimal and shortest path routing using the real link weights.

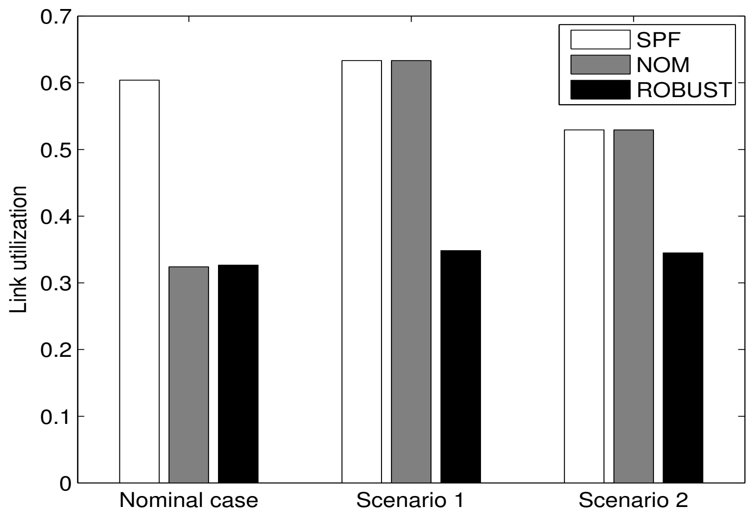


Figure 10.13: Maximum link load for SPF, Optimal routing for the nominal case and robust routing for the nominal traffic situation and two worst case scenarios.

Table 10.1: Computational and memory complexity of the polynomial time algorithm.

	Constraints	Variables	CPU time (sec.)
Abilene	4318	6189	5.9
GEANT	49156	74963	47.32

approach takes a comparable time to execute. Specifically, the iterative method took 4.9 seconds to converge for the Abilene network and 38.9 seconds to converge for the GEANT network. The iterative method was more memory efficient, since only a handful of scenarios and a limited number of paths were needed to find the optimal solution.

10.7 Related work

To handle traffic uncertainties one need to identify a set of traffic scenarios and optimize the routing for all instances of the set. This has been the topic of a number of research papers. Fortz and Thorup use a search heuristic which is shown to be efficient in finding a suitable OSPF/IS-IS weight setting to a wide range of traffic situations in [11]. The uncertainty set is constructed by forming the convex hull of a number of typical identified traffic scenarios. Applegate and Cohen [3] show that it is possible to find an efficient routing setting with fairly limited knowledge of the traffic demands. Furthermore, the authors give a lower bound on performance for the routing for all possible traffic situations. Column/constraint generation was first introduced in this context by Ben-Ameur and Kerivin [4] to balance load in Virtual Private Networks (VPN's). A recent article by Wang *et al.* [37] propose "common-case optimization with penalty envelope" (COPE) which computes a routing setting that optimizes for a set of traffic matrices which constitute common case traffic scenarios. Furthermore COPE give an upper bound of performance of a larger set of admissible traffic scenarios called a traffic envelope.

Instead of identifying a set of traffic scenarios it is possible to continuously monitor the network and adjust the routing accordingly. An early attempt is presented in a classical article on minimum delay routing by Gallagher [12]. The network detects the state of the network and adjusts the routing to minimize delay in the network. In Gallagers paper the calculations of paths in the network is incorporated in the load balancing. This will guarantee that optimal performance is attained but complicates implementation of the routing protocol. Hence, in many cases it is desirable to separate the path calculations from the load balancing and distribute the traffic over precalculated paths. An approximation of Gallagher's algorithm is introduced by Vutukury and Garcia-Luna-Aceves [36] where the traffic is balanced on a set of pre-computed paths. Elwalid *et al.* [9] introduces MPLS Adaptive Traffic Engineering (MATE). The distribution of traffic is formulated and solved as a convex optimization problem. A distributed algorithm is devised in the paper and the authors prove that it converges to an optimal solution for a given objective. The algorithm assumes that feedback is given from the network. A similar approach is made by Kandula *et al.* [21] where TeXCP is introduced. TeXCP borrows many features from eXplicit Congestion control Protocol (XCP) [22]. In TeXCP the sending agents performing the load balance rely on explicit feedback from routers about load on links on the path from source to destination. The authors use a control theoretic framework to prove stability and optimality of their distributed algorithms.

10.8 Conclusion

We have considered the problem of robust load balancing in face of traffic uncertainties. After a short review of important sources for traffic variations in data networks, we have shown how the associated uncertainties can be effectively captured by polytopic and ellipsoidal uncertainty models, and detailed efficient algorithms for solving the associated robust load-balancing problems. Quite remarkably, the use of ellipsoidal uncertainty models allows us to develop polynomial-time algorithms for computing the associated robust routing.

We have shown how the presence of correlations plays an important role in robust optimization. Even when the traffic demands themselves are uncorrelated, correlations may appear in the estimated traffic matrices. These correlations can be explored to find routing settings that are insensitive to estimation errors. Traffic uncertainties caused by interdomain reroutes display similar properties, since if a large fraction of the traffic leaves the network at one peering point, the fraction of traffic destined to the other peering points will be small. The strength of our solution is that it returns a single routing setting with guaranteed performance under all foreseeable traffic variations. However, scalability remains an issue, and questions regarding how to handle failure scenarios have not been addressed. We intend to investigate these issues in our future work.

Acknowledgments

This work was supported by the SICS Center for Networked Systems, the Swedish Research Council, and the European Commission via the EuroNF and 4WARD projects. The authors would like to thank Dr. Bo Yang for valuable comments on the manuscript.

Bibliography

[1] *Abilene*. http://abilene.internet2.edu.

[2] R. K. Ahuja, T. L. Magnati, and J.B. Orlin. *Network Flows*. Prentice Hall, 1993.

[3] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc. ACM SIGCOMM*, pages 313–324, Karlsruhe, Germany, August 2003.

[4] W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. *Optimization and Engineering*, 6(3):283–313, 2005.

[5] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[6] P. Bermolen, S. Vaton, and I. Juva. Search for optimality in traffic matrix estimation: A rational approa ch by Cramer-Rao lower bounds. In *Proc. NGI*, Valencia, Spain, April 2006.

[7] J. Cao, D. Davis, S. Vander Wiel, and B. Yu. Time-varying network tomography: router link data. *Journal of Americal Statistical Association*, 95:1063–1075, 2000.

[8] V. Chew. Confidence, prediction and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*, 61(315):605–617, 1966.

[9] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proc. IEEE INFOCOM*, pages 1300–1309, Anchorage, Alaska, USA, May 2001.

[10] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proc. ACM SIGCOMM*, pages 257–270, Stockholm, Sweden, August 2000.

[11] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[12] R. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25(1):73–85, 1977.

[13] *Geant*. http://www.geant.net.

[14] A. Gunnar and M. Johansson. Robust routing under BGP reroutes. In *Proc. IEEE GLOBECOM*, Washington DC, USA, November 2007.

[15] A. Gunnar and M. Johansson. Robust routing under statistical uncertainty: models and polynomial time algorithms. In *Proc. NGI 2009*, Aveiro, Portugal, July 2009.

[16] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a global IP backbone - a comparison on real data. In *Proc. ACM Internet Measurement Conference*, pages 149–160, Taormina, Sicily, Italy, October 2004.

[17] S. Halabi and D. McPherson. *Internet Routing Archtectures*. Cisco Press, 2001.

[18] M. Johansson and A. Gunnar. Data-driven traffic engineering: techniques, experiences and challenges. In *Broadnets*, San Jose, California, USA, October 2006.

[19] I. Juva, R. Susitaival, M. Peuhkuri, and S. Aalto. Effects of spatial aggregation on the characteristics of origin-destination pair traffic in funet. In *Proc. NEW2AN*, St Petersburg, Russia, September 2007.

[20] I. Juva, S. Vaton, and J. Virtamo. Quick traffic matrix estimation based on link count covariances. In *Proc. ICC*, Istanbul, Turkey, June 2006.

[21] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, pages 253–264, Philadelphia, Pennsylvania, USA, August 2005.

[22] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. In *Proc. of ACM SIGCOMM*, pages 89–102, Pittsburgh, Pennsylvania, USA, August 2002.

[23] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebert. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[24] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. ACM SIGCOMM*, pages 161–174, Pittsburgh, Pennsylvania, USA, August 2002.

[25] K. Papagiannaki, N. Taft, Z. Zhang, and C. Diot. Long-term forecasting of internet backbone traffic: Observations and initial models. In *Proc. IEEE INFOCOM*, pages 1178–1188, San Francisco, California, USA, April 2003.

[26] M. Pioro and D. Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[27] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *Proc. ACM Internet Measurement Conference*, pages 248–258, Miami Beach, Florida, USA, October 2003.

[28] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot. Traffic matrices: Balancing measurements, inference and modeling. In *Proc. ACM SIGMETRICS*, June 2005.

[29] A. Sridharan, R. Guérin, C. Diot, and S. Bhattacharyya. The impact of traffic granuarity of robustness of traffic aware routing. Technical report, University of Pennsylvania, 2004.

[30] C. Tebaldi and M. West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442):557–576, June 1998.

[31] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan. Traffic matrix reloaded: The impact of routing changes. In *Proc. Passive Active Measurements*, Boston, Massachusetts, USA, April 2005.

[32] R. Teixeira, T. Griffin, G. Voelker, and A. Shaikh. Network sensitivity to hot potato disruptions. In *Proc. ACM SIGCOMM*, pages 231–244, Portland, Oregon , USA, August 2004.

[33] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, pages 307–319, New York, USA, June 2004.

[34] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1), January 2006.

[35] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the Americal Statistical Association*, 91(433):365–377, March 1996.

[36] S. Vutukury and J. J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *Proc. ACM SIGCOMM*, pages 227–238, Cambridge, Massachusetts, USA, August 1999.

[37] H. Wang, H. Xie, L. Qiu, Y. Yang, Y. Zhang, and A. Greenberg. Cope: traffic engineering in dynamic networks. In *Pric. ACM SIGCOMM*, pages 99–110, Pisa, Italy, 2006.

[38] 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*, pages 206–217, San Diego, California, USA, June 2003.

Chapter 11

Paper E: Cautious Weight Tuning for Link State Routing Protocols

Anders Gunnar, Mikael Johansson. In *SICS Technical Report T2011:01 ISSN 1100-3154*, January 2011.

Abstract

Link state routing protocols are widely used for intradomain routing in the Internet. These protocols are simple to administer and automatically update paths between sources and destinations when the topology changes. However, finding link weights that optimize network performance for a given traffic scenario is computationally hard. The situation is even more complex when the traffic is uncertain or time-varying. We present an efficient heuristic for finding link settings that give uniformly good performance also under large changes in the traffic. The heuristic combines efficient search techniques with a novel objective function. The objective function combines network performance with a cost of deviating from desirable features of robust link weight settings. Furthermore, we discuss why link weight optimization is insensitive to errors in estimated traffic data from link load measurements. We assess performance of our method using traffic data from an operational IP backbone.

11.1 Introduction

Link state routing protocols are transparent, simple to administer and possess a remarkable ability to recover from component failures. After a failure, routers autonomously find a new consistent routing in a fully distributed manner. However, link state routing protocols also have disadvantages. The shortest path principle that enables each router to find a consistent routing limits the routes that can be realized using link state routing. Furthermore, it is computationally hard to find link weights that give optimal network performance even when traffic patterns are known.

In this paper we study search heuristics for parameter setting in two of the most widely used routing protocols in the Internet today, Open Shortest Path First (OSPF) and Intermediate System Intermediate System (IS-IS). Both these protocols are link state routing protocols where the network is modeled as a graph whose nodes represent routers and edges represent links connecting the routers. Each link is assigned a weight reflecting the cost of sending traffic over the link and routes are computed such that traffic flows along the shortest path from source to destination. These shortest paths are typically computed using Dijkstra's algorithm. To tune the link weights for improved network performance one needs to understand how traffic flows in the network. However, constant monitoring of traffic in a large IP backbone can be challenging and resource consuming. Hence, many operators only perform occasional measurements on selected parts of the network. These partial and occasional measurements, combined with the time varying nature of Internet traffic lead to an uncertainty of the traffic situation. In addition, the interplay between internal routing within a network and the routing between administrative domains is known to cause large shifts in traffic patterns [15]. Thus, it is desirable to find parameter settings that reduce performance degradations for a wide range of load variations. Contrary to e.g. [6] we do not optimize for the normal traffic situation but try to minimize the worst network performance that can occur under any foreseeable traffic scenario. This is novel also with respect to other relevant work: Balon and Leduc [4] address traffic uncertainty caused by the interplay between intra and interdomain routing but do not consider the worst case traffic scenario; previous work on robust routing such as COPE [17], oblivious routing [3] and the authors' previous work [7, 8], do not apply to link state routing protocols.

With our previous experience from robust routing without the constraints of link state routing ([7, 8]) it is natural to try to develop similar algorithms for optimizing weights in link state routing protocols. To evaluate our approach we focus on uncertainties arising from interdomain reroutes. We know from our earlier work [7] that this type of uncertainty can be dealt with efficiently when the routing does not need to follow shortest paths. In this paper, we leverage on our previous work to develop efficient search heuristics for finding weight settings with guaranteed performance for all foreseeable traffic scenarios. We call our approach *cautious weight tuning* since in each step of the algorithm the weight change that gives the largest guaranteed performance improvement for every possible traffic scenario is executed. Furthermore, we explain why optimization of link weights is robust to estimation errors in estimated traffic matrices from link load measurements.

The rest of the paper is organized as follows. In the next section we define cautious weight tuning and the notation used in the paper. Section 11.3 describes our approach. A use case with interdomain reroutes and a numerical example from an operational IP backbone is presented in Section 11.4. Cautious weight tuning under traffic matrix uncertainty is studied in Section 11.5 and related work is discussed in Section 11.6. Finally, we conclude our findings in Section 11.7.

11.2 Notation and problem formulation

The network is represented by a graph (N, E) where N is the set of nodes and E is the set of edges representing routers and links, respectively. The capacity of a link $l \in E$ is denoted c_l. Furthermore, the traffic demand between source-destination pair p is denoted s_p and the set of source destination pairs P has $|N|(|N| - 1)$ elements. In addition, the traffic demands are assumed not to be known with complete certainty. Instead, the traffic demands belong to an uncertainty set $\mathcal{S}$.

We assume that routing is performed by a link state routing protocol, such as OSPF or IS-IS. Associated to each link l is a weight w_l that describes the cost of sending data across that link. Data is routed over the shortest paths in this link metric. The routes for a given weight setting can be represented by an $|E| \times |P|$ routing matrix $R = [r_{lp}]$ where each element r_{lp} represents the fraction of traffic demand s_p routed over link l. With this notation, the total traffic t_l across link l is

$$t_l = \sum_{p \in P} r_{lp} s_p = r_l^T s$$

where r_l^T is the lth row of the routing matrix R. Thus the vector $t = [t_l] \in \mathbb{R}^{|E|}$ of total traffic is given by the equation

$$t = Rs. \tag{11.1}$$

When link state routing with a single path between source and destination is used, the elements in R assume values 0 or 1 depending on whether a source-destination pair is routed on the link or not. Adjusting the link weights changes the routing matrix R and alters the total traffic t across links.

Since the traffic s is uncertain, it is hard to predict the traffic load that results from a specific change in the link weights (and hence in the routing matrix). In this work, we consider *cautious weight tuning*, where we attempt to find link weights that are guaranteed to improve system performance for all traffic scenarios in the uncertainty set. In each step of the tuning, one link weight at a time is adjusted to form a new weight setting w^+ and the corresponding routing matrix R^+ is calculated. For this routing we determine the worst case traffic scenario $s_{wc} \in \mathcal{S}$. The weight setting that gives the largest performance improvement is executed and the procedure is repeated. Cautious weight tuning was first introduced in [9] for weight tuning under traffic matrix uncertainty, but the algorithm proposed in that paper often fails to find weight changes that guarantee improved performance. In this paper we generalize the applicability of cautious weight tuning and enhance the search with a novel objective function.

11.3 Cautious weight tuning

Even under the assumption that the traffic matrix is known, link weight optimization is computationally hard. One can either approach the problem formally via mixed-integer-linear programming models [11] or using search heuristics [5]. Since the computational burden of the integer programming approach quickly becomes prohibitive as network size grows, search heuristics are often preferred in practice. Search heuristics do not certify optimality for the obtained solution but experiments indicate that it is possible to find near-optimal solutions with a reasonable computational effort [1, 5].

We base our evaluation on a search technique first introduced by Fortz and Thorup [5], referred to as FT in the rest of the paper. FT is a local search algorithm where neighboring weight settings are created by changing a single weight and the new weight settings are evaluated using their predicted network performance. Our method has two distinct differences: first, we use the worst-case traffic over the full uncertainty set as the performance measure; second, we do not evaluate routing settings based on their predicted network performance only, but also account for properties of the routing that we know are more likely to give robust solutions.

A key step in our approach is to determine the worst-case traffic scenario for a given weight setting. Using link utilization ($u_l = t_l/c_l$) as performance metric the worst case traffic scenario can be found by solving, for each link $l \in E$, the optimization problem

$$\begin{array}{ll} \text{maximize} & c_l^{-1}(r_l^+)^T s_{wc} \\ \text{subject to} & s_{wc} \in \mathcal{S} \\ & s_{wc} \succeq 0 \end{array} \tag{11.2}$$

where $(r_l^+)^T$ is the l^{th} row in the routing matrix for weight setting w^+. If $\mathcal{S}$ is a convex set then s_{wc} can be found in polynomial time using modern interior-point methods. In particular, if $\mathcal{S}$ can be described as the solution set of a set of linear equations (11.2) becomes a linear programming problem.

The straightforward robust version of FT would work as follows. In each iteration, neighboring weight settings are computed and their worst-case performance are computed by solving (11.2). The weight setting that guarantees the lowest link utilization over all traffic scenarios is executed, and the procedure is repeated until the stopping criterion is fulfilled.

A drawback with this method is that it is hard to find single weight changes that actually improve worst-case performance. In other words, multiple weight settings might be needed before the worst-case performance is improved. Including multiple weight changes in the search algorithm drastically increases the search space, and hence the computational requirements on the algorithm. The key idea of our method is to stay with single weight changes in each search step but include an additional term in the search objective that discourages weight changes that are likely to be sensitive to traffic uncertainty. For example, in many cases we know that we should discourage weight changes that split up traffic of known volume into subflows of unknown volume. The following example illustrates the idea.

We consider the simple five node example network depicted in Figure 11.1. Ten units of traffic is injected in node one and two, however, the traffic can be destined to either node three or node four. This type of uncertainty appears for interdomain routing where routes to a destination outside the network domain are available in several locations in the network. Depending on how the interdomain routing protocol selects preferred route, traffic to a destination may leave the network in several locations. The uncertainty of the traffic demands can be expressed by the equations

$$\begin{aligned} s_{13} + s_{14} &= 10 \\ s_{23} + s_{24} &= 10. \end{aligned}$$

If we set the weight on each link to one both traffic demands s_{13} and s_{23} are routed on the link between node two and three. This will result in a worst case link load of 20 units of traffic. However, with careful tuning of link weights it is possible to let traffic demand s_{14} follow the path 1→5→4 and demand s_{13} path 1→5→4→3. Assuming demands s_{23} and s_{24} take the paths 2→3 and 2→3→4 respectively we are able to reduce the worst case link load to ten units of traffic. Thus, from the example we conclude that it is desirable to route flows s_{13} and s_{14} together while demands s_{23} and s_{24} are routed together on separate links from the other two traffic demands. The challenge is to obtain a weight setting that is able to realize such a routing.

Based on our observations in the example above we add a penalty function to the search objective which encourages known traffic flows to be routed together. To this end, let R be a routing matrix calculated from a given weight setting using the shortest path principle. The $|P| \times |P|$ matrix

$$Q = R^T R - \text{diag}(R^T R) \tag{11.3}$$

will then, for each matrix element $[Q]_{ij}$ where $i \neq j$, describe the number of links shared by the shortest path routes for traffic demands s_i and s_j. The diagonal elements of the matrix $R^T R$ are the lengths of the individual paths. Since we do not penalize the lengths of individual paths (nor encourage long paths) in this paper, we make sure that the diagonal elements of Q are zero. We also define

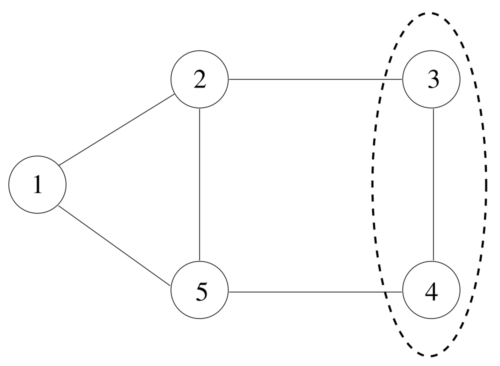


Figure 11.1: Simple example network with five nodes, one group of possible egress nodes for traffic inserted in the network at node one and two

a matrix $C \in \mathbb{R}^{|P|\times|P|}$ that encodes which traffic flows we encourage to share routes and which traffic flows we discourage from sharing routes. Specifically, the elements of C are given by

$$[C]_{ij} \begin{cases} = 1 & \text{if demand } i\text{and } j \text{ should be routed together} \\ = -1 & \text{if demand } i\text{and } j \text{ should not be routed together} \\ = 0 & \text{otherwise.} \end{cases}$$

In general, determining what traffic demands to route together have to be tailored to the traffic uncertainty at hand. This requires insight or intuition about the specific problem. We will show shortly how the elements of C can be set to generate routing settings that are robust to traffic shifts due to intradomain reroutes. Finally, we define a hint function

$$h(Q, C) = \mathrm{Tr}(CQ) \tag{11.4}$$

where $\mathrm{Tr}(\cdot)$ is the trace operator. The function $h(Q, C)$ serves as a hint to help the heuristic favor routing settings that route certain flows together to decrease uncertainty of the load on the link. The hint function gives a high value if the routing determined by R routes a high number of desired flows together and penalizes routing settings where a lower number of flows are routed together.

The objective function used by our search algorithm is thus

$$\mathrm{obj}(w) = u_{\max}(w; \mathcal{S}) - \kappa h(Q(w); C). \tag{11.5}$$

Here, $u_{\max}(w; \mathcal{S})$ is the maximum worst-case link utilization for shortest path routing with link weights w (determined by solving the optimization problem (11.2) for every link in the network) while h is the hint function defined above. The parameter κ determines how much emphasis should be given to the hint in comparison to the link utilization. If κ is set to zero, we disregard influence of the hint function.

For each iteration of the search heuristic $|E|$ different weight settings are tested, and the weight setting that gives the largest decrease in the search objective is executed. The actions taken for a weight setting can be summarized as follows:

1. Produce a neighboring weight setting w^+ and calculate the corresponding routing matrix R^+.
2. Determine the worst case traffic scenario s_{wc} for R^+.
3. Evaluate the objective function for R^+ and s_{wc}.

The iterations are repeated until a stopping criterion is satisfied. Note that a property of our objective function is that it accepts weight settings that give an increase in link utilization if the hint function is improved sufficiently much. This also means that the weight setting produced during the last iteration might not necessarily be the best (since the hint function might have caused a change). To be able to recall the best weight setting we keep track of the best $u_{\max}$ and the associated weight setting during our search.

11.4 Application of cautious weight tuning: weight setting under BGP traffic uncertainty

For resilience, network operators often connect with other operators in several different locations to exchange traffic and routing information. The routing information is encoded as prefixes representing reachable destination networks. As a result of introducing multiple connection points, many prefixes are announced (and hence available for forwarding traffic) in several locations in the network. Depending on the setting of interdomain routing protocol attributes and configuration of intradomain routing protocol parameters traffic take a selected route for each prefix. Since conditions may change due to e.g. reconfigurations, failures or withdrawn routes, there is uncertainty about how the traffic will flow in the network. It has been shown in many studies, that interdomain reroutes may cause large traffic shifts in the network (e.g. [14, 15]). These uncertainties can efficiently be handled for routing without the constraints from shortest path routing [7]. In this section we investigate if this also holds for link state routing protocols.

We assume that at every ingress router it is possible to measure the amount of traffic destined to each destination prefix in the routing table using flow measurement functionality such as Cisco's Netflow. Since flows of known volume should be routed together, traffic from a common source destined to a network prefix announced by multiple egress routers is routed together. A difficulty is that a routing table in the default free zone in the Internet today contains in the order of 200 000

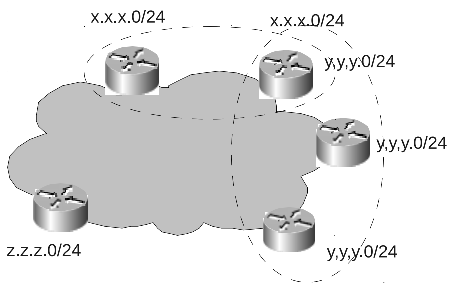


Figure 11.2: Example network with three groups of egress points for prefixes x.x.x.0/24, y.y.y.0/24 and z.z.z.0/24 respectively

prefixes. Creating a model where all prefixes are included would create an intractable model. To reduce problem size we could use the methodology described in [7] where it is observed that most of the prefixes in the routing table route negligible amounts of traffic. However, we address the scalability problem in a different fashion by observing that prefixes are typically announced by a limited number of groups of peering points. In Figure 11.2 there are three groups of peering routers announcing three prefixes. In this example we replace three prefixes with three groups of egress routers for traffic to the prefixes. However, if a group contains a large number of prefixes it is possible to obtain a large reduction of parameters to describe the traffic uncertainty by aggregating traffic from a source destined to prefixes in each group.

To formulate the model in equations we let t_{og} denote the amount of traffic sent from ingress router o to egress router group $g \in \mathcal{G}$ where $\mathcal{G}$ is the set of groups of peering routers announcing prefixes. Furthermore, $e(g)$ is the set of egress routers of group g. The variable δ_{odg} represents the fraction of traffic from source o to group g that leave the network at egress router d assuming $d \in e(g)$. If $d \notin e(g)$ then $\delta_{odg} = 0$. To identify the worst case traffic matrix for a given routing matrix R we solve the following optimization problem for each link l in the network

$$\begin{array}{ll} \text{maximize} & c_l^{-1} r_l^T s \\ \text{subject to} & \sum_{g \in \mathcal{G}} t_{og}\delta_{odg} = s_{od}, \forall o, d, o \neq d \in N \\ & \sum_{d \in e(g)} \delta_{odg} = 1, \forall g \in \mathcal{G}, \forall o \in N \\ & \delta_{odg}, s_{od} \geq 0 \end{array} \tag{11.6}$$

where δ_{odg} and s_{od} are the optimization variables and the constraints in the optimization problem constitute the traffic uncertainty set $\mathcal{S}$. The worst case is the traffic scenario that gives the highest link utilization for all $l \in E$. In our analysis we found that for the 163 000 prefixes in our routing data set we were able to identify 35 groups of routers announcing prefixes. Thus, we are able to reduce the number of variables considerably in the optimization problem.

Based on the observations made in Section 11.3 we conclude that source destination traffic demands s_{od} from a common source o and destined to a destination router d that belong to a common group $g \in \mathcal{G}$, should be routed together. To accomplish this we proceed as follows:

1. For each source router group together routers announcing the same prefixes, and set their corresponding elements in the C-matrix to 1.

2. Traffic demands destined to a router in an egress group but from different sources have their corresponding elements in C set to -1.

3. Other elements in C are set to zero.

Although this is a rather simplistic approach we will see later that it will serve our purposes well. Also note that if groups are partially overlapping, the values of involved elements in C will not be uniquely determined. However, we neglect such considerations here.

Numerical examples

For the evaluation we use data from the GEANT network provided by Uhlig *et al.* [16]. The GEANT network connects national research networks in Europe and has 23 nodes and 74 links. In our experiments we have access to network topology, traffic data and BGP routing information. The traffic measurements were performed during a four month period, and consist of 15 minute flow exports of sampled Netflow measurements where one out of every thousand packets is sampled. Furthermore, the BGP routing information base is recorded every day during the measurement period. Since we use maximum link utilization as objective in our optimization, we upgrade links of 155 Mbps to 2400 Mbps as these links would otherwise remain bottlenecks irrespectively of the routing. More details about the data set can be found in [16].

Comparison with other approaches

We compare cautious weight tuning with weight tuning for a nominal traffic scenario obtained from the original link weights used by BGP to select egress point for each prefix in the routing table (i.e. the FT heuristic for the nominal traffic scenario). For comparison we also provide results of optimal robust routing under interdomain reroutes without the constraints imposed by link state routing [7]. To demonstrate the benefit of robust routing, we also provide results where a routing optimized for the nominal case is subjected to the worst-case traffic scenario in $\mathcal{S}$.

Figure 11.3 displays $u_{\max}$ for optimization for the nominal case (nomOPT), robust optimization (robOPT) [7], weight tuning using the heuristic by Fortz and Thorup (FT) and cautious weight tuning using FT (cautiousFT). The results for nomOPT indicate that although large performance gains can be made from optimization for the nominal case, this routing setting is highly sensitive to deviations from normal operation. The worst case link utilization is almost twice as high as expected for the nominal case it was optimized for. Similar results can be observed for the weight tuning using FT. For robust optimal (robOPT) the worst case link utilization is reduced but utilization under normal operation has increased from 0.24 to 0.35 in this experiment. Furthermore, performance of cautiousFT is almost identical to optimal robust routing.

Progress

Figure 11.4 displays progress of cautiousFT for each iteration of the algorithms and best possible setting of κ. The vertical axis indicate deviation of $u_{\max}$ from optimal obtained by robOPT. The weight tuning is set to terminate after 100 iterations. However, the best weight setting is found after 53 iterations of the algorithm. We note that performance gains are made in steps after a number of iterations have been executed. In order to reduce $u_{\max}$ under traffic uncertainty a number of weight changes need to be executed before a performance gain in link utilization can be observed. During the periods when no progress in $u_{\max}$ is made, the hint function guides the heuristic in a direction where progress can be obtained. In Figure 11.4 it can be noted that at some instances slightly worse weight settings are accepted by the algorithm. In these situations the hint function outperforms the value of maximum link utilization.

Tuning the parameter κ

A critical component of cautious weight tuning is setting the parameter κ. Figure 11.5 shows deviation of $u_{\max}$ from optimal value obtained using the algorithm from [7] as a function of κ, and κ is plotted in logarithmic scale. The plot reveals that satisfactory performance is obtained for a wide variety of values of κ. Our findings indicate that weight settings that are robust to traffic shifts due to interdomain reroutes exist and with the right settings of the matrix C and the parameter κ it is possible to find these weight settings using well-established search techniques.

Computational considerations

Unfortunately we have not access to detailed traffic data from other networks than the GEANT network. Hence, we are not able to perform a detailed evaluation of

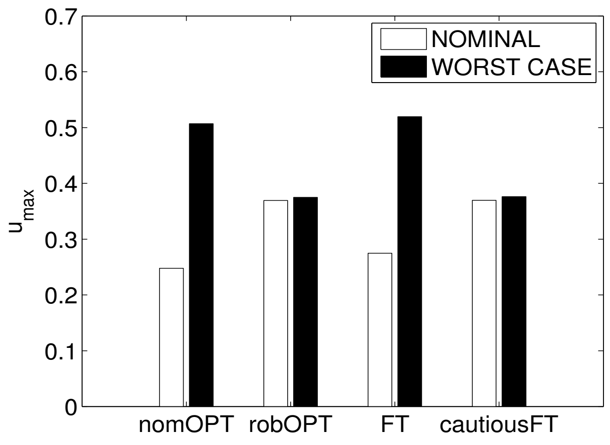


Figure 11.3: Maximum link utilization for different routing settings and traffic situations, nominal traffic scenario (White) and worst case traffic scenario (Black)

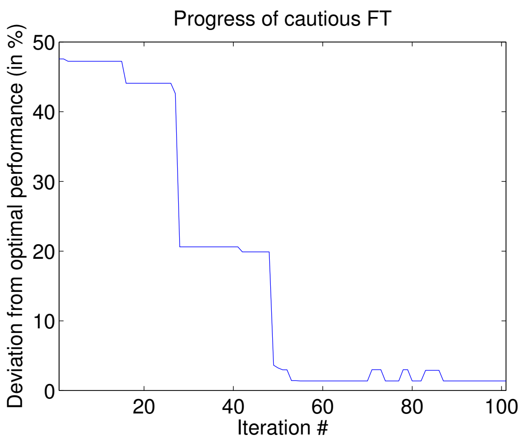


Figure 11.4: Deviation from optimal performance in $u_{\max}$ for each iteration of cautious FT

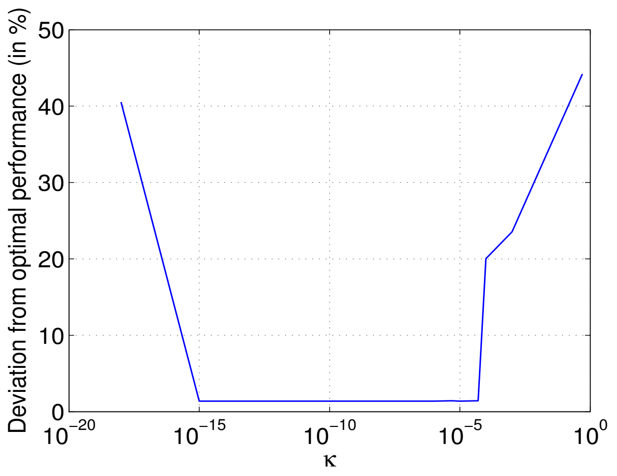


Figure 11.5: Deviation from optimal performance in $u_{\max}$ as a function of the hint multiplied by the parameter κ for cautiousFT

Table 11.1: Execution time, fraction of changed weights (from start weights) and number of identified worst case traffic scenarios for cautiousRR and cautiousFT. Parameter κ set to obtain best possible performance

	Exec. time (Sec.)	Changed weights	s_{wc} identified
cautiousFT	1972	35%	113886

the computational burden of our proposal for traffic uncertainty caused by interdomain reroutes. However, Table 11.1 summarizes some computational aspects of cautiousFT. The fraction of changed weights is the fraction between the number of changed weights and total number of link weights in the network. Furthermore, the total number of variables and equality constraints in problem (11.6) are 1826 and 1289 respectively. Note that the exact number of variables and constraints depend on the composition of the groups $g \in \mathcal{G}$ of egress routers.

11.5 Cautious weight tuning under traffic matrix uncertainty

In the light of the encouraging results on robust routing [9] and the results of optimization of OSPF/IS-IS link weights from estimated traffic demands [13], the rather disappointing results of cautious weight tuning in [9] might appear surprisings. To explain these findings we return to the determination of the worst case link traffic scenario as formulated in (11.2).

To illustrate how changes in link-weights influence the variability in worst case traffic scenarios we randomly select link weights in the GEANT network and set

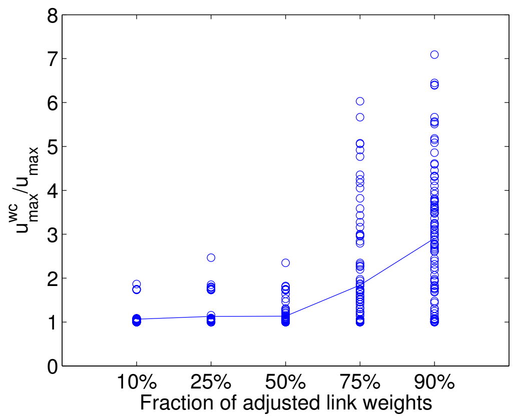


Figure 11.6: $u^{wc}_{\max}/u_{\max}$ when 10,25,50,75,90% of link weights are changed to a random integer. Average value (solid line) and samples plotted for each level

the weight to a random number between 1 and 1000. Assuming R_{est} and t_{est} are known, $u_{\max}$ for the nominal traffic scenario as well as an identified worst case traffic scenario in the solution space of $R_{\text{est}} s_{wc} = t_{\text{est}}$ is identified by solving the optimization problem

$$\begin{array}{ll} \text{maximize} & c_l^{-1}(r_l^+)^T s_{wc} \\ \text{subject to} & R_{\text{est}} s_{wc} = t_{\text{est}} \\ & s_{wc} \succeq 0 \end{array} \tag{11.7}$$

for each row r_l^+ in the adjusted routing matrix R^+. Link utilization for s_{wc} is denoted $u^{wc}_{\max}$. Figure 11.6 plots the average value of $u^{wc}_{\max}/u_{\max}$ for 100 different routing matrices for different levels of random changes of link weights. The average value is close to one up to when 50 percent of the link weights are changed. However, there is also a high degree of variability in the results. Nevertheless, the experiment indicate that maximum link utilization is rather insensitive to changes in the link weights.

To explain these observations it is instructive to consider the null space of the original routing matrix R_{est}. The solution set to $R_{\text{est}} s = t_{\text{est}}$ can be described as the sum of a particular solution s_0 satisfying the constraints and a linear combination of the vectors in the null-space of R_{est}. The solution set can be described by the equation

$$s = s_0 + (I - R^{\dagger}_{\text{est}} R_{\text{est}})z \tag{11.8}$$

where $R^{\dagger}_{\text{est}}$ is the pseudo-inverse of the matrix R_{est} and z are coordinates such that $s \succeq 0$. We replace s with (11.8) and use the property of the pseudo-inverse:

$R_{\text{est}} = R_{\text{est}} R_{\text{est}}^{\dagger} R_{\text{est}}$. After simplifications we arrive at the following optimization problem

$$\begin{array}{ll} \text{maximize} & c_l^{-1}(r_l^{+})^T (I - R_{\text{est}}^{\dagger} R_{\text{est}}) z \\ \text{subject to} & s_0 + (I - R_{\text{est}}^{\dagger} R_{\text{est}}) z \succeq 0 \end{array} \tag{11.9}$$

The optimization problem (11.9) only takes the variability due to the uncertainty into consideration. We note that if a row in the new routing matrix is unchanged, the objective will be identical to zero due to properties of the pseudo-inverse mentioned above. This helps to explain why cautious weight tuning is difficult under traffic matrix uncertainty as observed in [9]. A small change in the a link weight only makes a small change in the routing matrix. Thus, the objective function in problem (11.9) will be identical to zero for most of the links. Furthermore, maximum link utilization as objective function takes the most congested link into consideration only. Changes in other parts of the network have no influence on the objective except when another link becomes the bottleneck. Thus, the robustness of link state routing protocols to link load measurement traffic uncertainty observed [1, 13] seems to be connected to the choice of maximum link utilization as objective function. However, it is likely that other properties specific to link state routing also play an important role. For instance, since paths sharing links before a weight change will to a large extent continue to share links after a weight change due to the rules of SPF routing.

11.6 Related work

One of the earliest and also one of the most cited papers on weight setting for link state protocols is Fortz and Thorups paper on local search heuristics [5]. The local search heuristics are extended to find weight settings for a wider selection of traffic situations in [6]. Ramakrishnan and Rodrigues [12] use a descending search algorithm where in each step one flow is deviated from a link in order to decrease a cost function. Another often cited paper is Wang *et al.* [18] where the Lagrangian variables obtained from a dual optimization problem is interpreted as link weights. Abrahamsson and Björkman [2] use a two step cost function which strives to keep load in the network under a prescribed level. In addition, the search heuristic is a combination of the heuristics by Fortz and Thorup [5] and Ramakrishnan and Rodrigues [12]. To handle reroutes caused by hot potato routing, Balon and Leduc [4] design a novel cost function that attempts to compensate for these effects. Nucci *et al.* [10] design a cost function that optimizes not only for the normal network topology but also for a number of different fault scenarios that might occur where links or nodes fail.

A somewhat different approach is taken by Xu *et al.* [19] where DEFT is introduced. With DEFT traffic can be sent over non shortest paths using an exponential penalty function. However, DEFT require minor modifications of the OSPF/IS-IS protocols.

11.7 Conclusion

In this paper we develop *cautious weight tuning* for link state routing protocols such as the widely used OSPF and IS-IS routing protocols for intradomain routing in the Internet. With cautious weight tuning it is possible to optimize link weights for a set of traffic scenarios to take into account variability and uncertainty in traffic data. Our work differ from previous studies (e.g. [4, 6]) in the sense that we explicitly identify the worst case traffic scenario and optimize the routing for this case. In other words, our approach does not only optimize the network for normal operation, but attempts to find routing settings that guarantee a certain performance for all foreseeable traffic patterns. Such routing settings allow a network operator to provision a more predictable and reliable service even when the traffic changes dramatically. To guide the heuristics we augment the desired network performance objective with a hint function that captures desirable properties of a robust routing setting. We highlight performance of the augmented objective function using a well-known search heuristic. In addition, we present some evidence of why optimization of link weights is robust to errors in traffic data caused by estimation of the traffic matrix from link load measurements. However, other properties may also influence the robustness of link state routing but this requires more investigation. The robustness to estimated traffic matrices of link state routing has been observed by many researchers before (e.g. [1, 13]) but to the best of our knowledge no explanation has been presented.

Although initial results presented in this paper are promising they should be considered preliminary. Further evaluation is needed on other network topologies and traffic situations to fully assess our approach. Other types of traffic uncertainties should be considered as well as a more general method to determine the elements in the C matrix.

Acknowledgment

This work was supported in parts by the SICS Center for Networked Systems, the Swedish Research Council and the European Commission. The authors would like to thank Dr. Martin Nilsson for valuable suggestions and discussions during the development of this paper.

Bibliography

[1] A. Gunnar and H. Abrahamsson and M. Söderqvist. Performance of Traffic Engineering in Operational IP-Networks-An Experimental Study. In T. Magedanz, E. Madeira, and P. Dini, editors, *Operations and Management in IP-Based Networks*, pages 202–211, Barcelona, Spain, October 2005. Springer. LNCS 3751.

[2] H. Abrahamsson and M. Björkman. Robust traffic engineering using L-balanced weight-settings in OSPF/IS-IS. In *Broadnets*, Sept 2009.

[3] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc. ACM SIGCOMM*, pages 313–324, Karlsruhe, Germany, August 2003.

[4] S. Balon and G. Leduc. Combined intra- and inter-domain traffic engineering using hot-potato aware link weights optimization. In *Proc. ACM SIGMETRICS*, pages 441–442, Jun 2008.

[5] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. IEEE INFOCOM*, pages 519–528, Israel, March 2000.

[6] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[7] A. Gunnar and M. Johansson. Robust routing under BGP reroutes. In *Proc. IEEE GLOBECOM*, Washington DC, USA, November 2007.

[8] A. Gunnar and M. Johansson. Robust routing under statistical uncertainty: models and polynomial time algorithms. In *Proc. NGI 2009*, Aveiro, Portugal, July 2009.

[9] M. Johansson and A. Gunnar. Data-driven traffic engineering: techniques, experiences and challenges. In *Broadnets*, San Jose, California, USA, October 2006.

[10] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot. IGP link weight assignment for operational tier-1 backbones. *IEEE/ACM Trans. Netw.*, 15(4):789–802, 2007.

[11] M. Pioro and D. Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[12] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest-path data networks. *Bell Labs Technical Journal*, 6(1):117–138, 2001.

[13] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *Proc. ACM Internet Measurement Conference*, pages 248–258, Miami Beach, Florida, USA, October 2003.

[14] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan. Traffic matrix reloaded: The impact of routing changes. In *Proc. Passive Active Measurements*, Boston, Massachusetts, USA, April 2005.

[15] R. Teixeira, T. Griffin, G. Voelker, and A. Shaikh. Network sensitivity to hot potato disruptions. In *Proc. ACM SIGCOMM*, pages 231–244, Portland, Oregon , USA, August 2004.

[16] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86, January 2006.

[17] H. Wang, H. Xie, L. Qiu, Y. Yang, Y. Zhang, and A. Greenberg. Cope: traffic engineering in dynamic networks. In *Pric. ACM SIGCOMM*, pages 99–110, Pisa, Italy, 2006.

[18] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proc. IEEE INFOCOM*, pages 565–571, May 2001.

[19] D. Xu, M. Chiang, and J. Rexford. Deft: Distributed exponentially-weighted flow splitting. In *Proc. IEEE INFOCOM*, pages 71–79, May 2007.

SICS Dissertation Series

14: Sverker Jansson, AKL - - A Multiparadigm Programming Language, 1994.

15: Fredrik Orava, On the Formal Analysis of Telecommunication Protocols, 1994.

16: Torbjörn Keisu, Tree Constraints, 1994.

17: Olof Hagsand, Computer and Communication Support for Interactive Distributed Applications, 1995.

18: Björn Carlsson, Compiling and Executing Finite Domain Constraints, 1995.

19: Per Kreuger, Computational Issues in Calculi of Partial Inductive Definitions, 1995.

20: Annika Waern, Recognising Human Plans: Issues for Plan Recognition in Human-Computer Interaction, 1996.

21: Björn Gambäck, Processing Swedish Sentences: A Unification-Based Grammar and Some Applications, 1997.

22: Klas Orsvärn, Knowledge Modelling with Libraries of Task Decomposition Methods, 1996.

23: Kia Höök, A Glass Box Approach to Adaptive Hypermedia, 1996.

24: Bengt Ahlgren, Improving Computer Communication Performance by Reducing Memory Bandwidth Consumption, 1997.

25: Johan Montelius, Exploiting Fine-grain Parallelism in Concurrent Constraint Languages, 1997.

26: Jussi Karlgren, Stylistic experiments in information retrieval, 2000.

27: Ashley Saulsbury, Attacking Latency Bottlenecks in Distributed Shared Memory Systems, 1999.

28: Kristian Simsarian, Toward Human Robot Collaboration, 2000.

29: Lars-åke Fredlund, A Framework for Reasoning about Erlang Code, 2001.

30: Thiemo Voigt, Architectures for Service Differentiation in Overloaded Internet Servers, 2002.

31: Fredrik Espinoza, Individual Service Provisioning, 2003.

32: Lars Rasmusson, Network capacity sharing with QoS as a financial derivative pricing problem: algorithms and network design, 2002.

33: Martin Svensson, Defining, Designing and Evaluating Social Navigation, 2003.

34: Joe Armstrong, Making reliable distributed systems in the presence of software errors, 2003.

35: Emmanuel Frécon, DIVE on the Internet, 2004.

36: Rickard Cöster, Algorithms and Representations for Personalised Information Access, 2005.

37: Per Brand, The Design Philosophy of Distributed Programming Systems: the Mozart Experience, 2005.

38: Sameh El-Ansary, Designs and Analyses in Structured Peer-to-Peer Systems, 2005.

39: Erik Klintskog, Generic Distribution Support for Programming Systems, 2005.

40: Markus Bylund, A Design Rationale for Pervasive Computing User Experience, Contextual Change, and Technical Requirements, 2005.

41: Åsa Rudström, Co-Construction of hybrid spaces, 2005.

42: Babak Sadighi Firozabadi, Decentralised Privilege Management for Access Control, 2005.

43: Marie Sjölinder, Age-related Cognitive Decline and Navigation in Electronic Environments, 2006.

44: Magnus Sahlgren, The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-dimensional Vector Spaces, 2006.

45: Ali Ghodsi, Distributed k-ary System: Algorithms for Distributed Hash Tables, 2006.

46: Stina Nylander, Design and Implementation of Multi-Device Services, 2007

47: Adam Dunkels, Programming Memory-Constrained Networked Embedded Systems, 2007

48: Jarmo Laaksolahti, Plot, Spectacle, and Experience: Contributions to the Design and Evaluation of Interactive Storytelling, 2008

49: Daniel Gillblad, On Practical Machine Learning and Data Analysis, 2008

50: Fredrik Olsson, Bootstrapping Named Entity Annotation by Means of Active Machine Learning: a Method for Creating Corpora, 2008

51: Ian Marsh, Quality Aspects of Internet Telephony, 2009

52: Markus Bohlin, A Study of Combinatorial Optimization Problems in Industrial Computer Systems, 2009

53: Petra Sundström, Designing Affective Loop Experiences, 2010