A Fundamental Approach for Providing Service-Level Guarantees for Wide-Area Networks

by

Jeremy Bogle

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2019

Certified by .....
Manya Ghobadi
Assistant Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Commitee

A Fundamental Approach for Providing Service-Level Guarantees for Wide-Area Networks

by

Jeremy Bogle

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

To keep up with the continuous growth in demand, cloud providers spend millions of dollars augmenting the capacity of their wide-area backbones and devote significant effort to efficiently utilizing WAN capacity. A key challenge is striking a good balance between network *utilization* and *availability*, as these are inherently at odds; a highly utilized network might not be able to withstand unexpected traffic shifts resulting from link/node failures.

I motivate this problem using real data from a large service provider and propose a solution called TeaVaR (Traffic Engineering Applying Value at Risk), which draws on financial risk theory to realize a risk management approach to traffic engineering (TE). I leverage empirical data to generate a probabilistic model of network failures, and formulate a Linear Program (LP) that maximizes bandwidth allocation to network users subject to a service level agreement (SLA). I prove TeaVaR's correctness, and then compare it to state-of-the-art TE solutions with extensive simulations across many network topologies, failure scenarios, and real-world traffic patterns. The results show that with TeaVaR, operators can support up to twice as much throughput as other TE schemes, at the same level of availability.

I also construct a simulation tool that builds on my implementation of TeaVaR and simulates its usage in the data plane. This tool can be useful not only for testing TE schemes but also for capacity planning, as it allows network operators to see how their network is performing, where the bottlenecks are, and what kind of demand loads it can handle.

Thesis Supervisor: Manya Ghobadi
Title: Assistant Professor

Acknowledgments

I would like to thank various people for their contributions and guidance on this project: Ishai Menache and Michael Schapira for helping with the proofs in the optimization, Nikhil Bhatia, for helping with the evaluations, and Asaf Valadarsky for doing preliminary work computing failure probabilities. Special thanks should be given to my thesis advisor, Manya Ghobadi, for her professional guidance, project vision, and continued support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and motivation

Traffic engineering (TE), the dynamic adjustment of traffic splitting across network paths, is fundamental to networking and has received extensive attention in a broad variety of contexts [21, 23, 13, 7, 3, 2, 17, 19, 24, 39, 30, 27]. Given the high cost of wide-area backbone networks (WANs), large service providers (e.g., Amazon, Facebook, Google, Microsoft) are investing heavily in optimizing their WAN TE, leveraging Software-Defined Networking (SDN) to globally optimize routing and bandwidth allocation to users [17, 19, 30, 26, 29]. In addition, they have to purchase or lease fiber to support wide-area connectivity between distant data center locations, and they need to know how and when to purchase this fiber. The process of deciding which fibers to buy and how to manage them is called capacity planning.

A crucial challenge faced by WAN operators is striking a good balance between network *utilization* and *availability* in the presence of node/link failures [15, 33, 30, 5, 18]. These two objectives are inherently at odds; providing high availability requires keeping network utilization sufficiently low to absorb shifts in traffic when failures occur. To attain high availability, today's backbone networks are typically operated at fairly low utilization so as to meet user traffic demands while providing high availability (e.g., 99%+ [18]) in the presence of failures.

In this chapter, I review some previous work in the field of WAN failure analysis and explain the motivations driving this work. Chapter 2 explains a similar problem in financial risk modeling and portfolio optimization. Chapter 3 offers a solution to

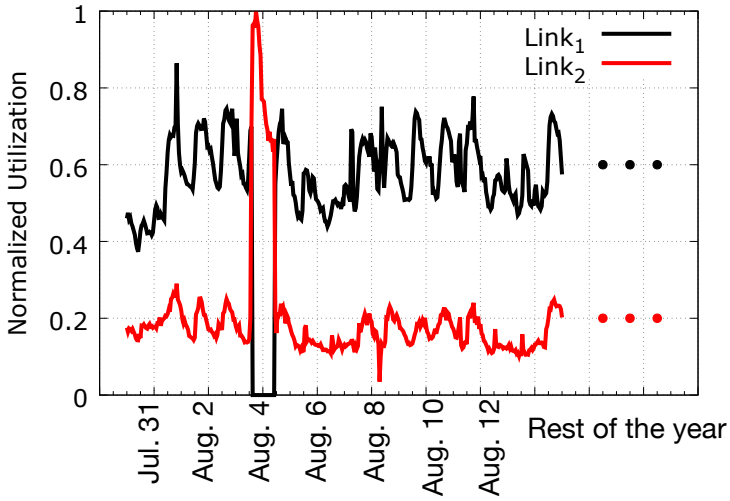


Figure 1-1: $Link_2$'s utilization is kept low to sustain the traffic shift when failures happen.

this problem where the loss in the network is cast as a linear program and minimized as a TE optimization. Chapter 4 describes the evaluation setup and performance of the solution proposed in Chapter 3. Chapter 5 covers the simulation and visualization of results. In this chapter I also explain how this same TE approach can be used with the simulation for capacity planning in an SDN.

1.1 Effects of failures in the WAN

Figure 1-1 plots the link utilization of two IP links in a backbone network in North America with the same source location but different destinations. The utilization of each link is normalized by the maximum achieved link utilization; hence, the actual link utilization is lower than plotted. On August 4^{th}, $Link_1$ failed, and its utilization dropped to zero. This, in turn, increased the utilization of $Link_2$. Importantly, however, under normal conditions, the normalized utilization of $Link_2$ is only around 20%, making $Link_2$ underutilized almost all the time.

This example illustrates a common problem with today's WANs. In Chapter 4, I show how state-of-the-art TE schemes fail to maximize the traffic load that can be supported by the WAN for the desired level of availability. Under these schemes, the allocated bandwidth falls significantly below the available capacity, resulting in

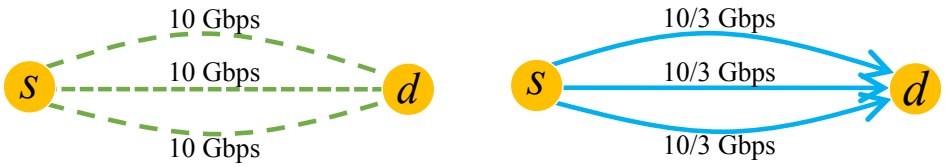
needlessly low network utilization. A key insight of my research is that operators should *explicitly* optimize network utilization subject to target availability thresholds. Instead of explicitly considering availability, today's TE schemes use the number of concurrent link/node failures the TE configuration can withstand (e.g., by sending traffic on link-disjoint network paths) as a proxy for availability. However, the failure probability of a single link can greatly differ across links, sometimes by three orders of magnitude [14]. Consequently, some failure scenarios involving two links might be more probable than others involving a single link. Alternatively, some failure scenarios might have only negligible probability; thus, lowering network utilization to accommodate them is wasteful and has no meaningful bearing on availability.

Fortunately, operators have high visibility into failure patterns and dynamics. For example, it has been shown that link failures are more probable during working hours [15] and can be predicted by sudden drops in optical signal quality, "with a 50% chance of an outage within an hour of a drop event and a 70% chance of an outage within one day" [14]. I propose that this wealth of timely empirical data on node/link failures in the WAN should be exploited to probe the probability of different failure scenarios when optimizing TE.

1.2 Motivation for this work

In this section, I provide a more concrete example of my work motivation. As mentioned above, the number of concurrent node/link failures a TE configuration can withstand is used as a proxy for availability. This can be manifested, e.g., in sending user traffic on multiple network paths (tunnels) that do not share any, or share only a few, links or in splitting traffic across paths in a manner that is resilient to a certain number of concurrent link failures [30]. In what follows, I explain why reasoning about availability in terms of the number of concurrent failures that can be tolerated is often not enough, using a demonstration of the recently proposed Forward Fault Correction (FFC) TE scheme [30] to substantiate my explanation.

FFC example. FFC maximizes bandwidth allocation to be robust against up to k



(a) (b)

Figure 1-2: (a) A network of three links each with 10Gbps bandwidth; (b) Under conventional TE schemes, such as FFC [30], the total admissible traffic is *always 10Gbps*, split equally between available paths.

concurrent link failures, for a configurable value k. To accomplish this, FFC optimization formulation sets a cap on the maximum bandwidth b_i each network flow i (identified by source/destination pair) can utilize and generates routing (and rerouting) rules such that the network can simultaneously support bandwidth b_i for each flow i in any failure scenario involving at most k failures.

Figure 1-2 shows an example of FFC, where the source node s is connected to the destination node d via three links, each with a capacity of 10Gbps. Suppose that the objective is to support the maximum total amount of traffic from s to d in a manner that is resilient to, at most, two concurrent link failures. Figure 1-2(b) presents the optimal solution under FFC: rate-limiting the (s, d)-flow to send at 10Gbps and *always* splitting its traffic equally between all links that are intact; e.g., when no link failures occur, traffic is sent at $\frac{10}{3}$Gbps on each link, when a single link failure occurs, each of the two surviving links carries 5Gbps, and with two link failures, all traffic is sent on the single surviving link. Thus, this solution guarantees the flow reserved bandwidth of 10Gbps without exceeding link capacities under any failure scenario that involves at most two failed links. Observe, however, that this comes at the cost of keeping each link underutilized (one third utilization) when no failures occur.

Striking the right balance. The question is whether high availability can be achieved without such drastic over-provisioning. Approaches such as FFC are compelling in that they provide strong availability guarantees; in Figure 1-2(b), the (s, d) flow is guaranteed a total bandwidth of 10Gbps even if two links become *permanently* unavailable. Suppose, however, that the availability, i.e., the fraction of time a link is up, is consistently 99.9% for each of the three links. In this scenario, the network can

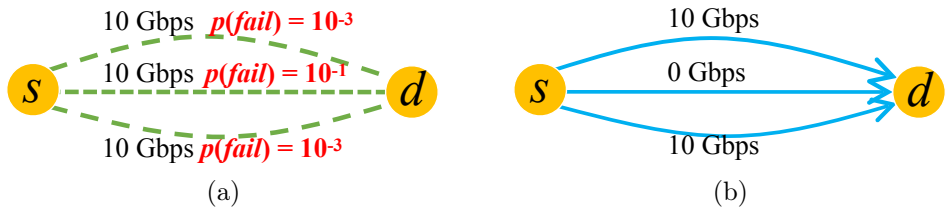


Figure 1-3: (a) The same network as in Fig. 1-2(a), with added information about link failure probabilities; (b) A possible flow allocations under TEAVAR with total admissible traffic of 20Gbps 99.8% of time.

easily support 30Gbps throughput (3× improvement over FFC) around 99.9% of the time simply by utilizing the full bandwidth of each link and never rerouting traffic.

This example captures the limitation of *failure-probability-oblivious* approaches to TE, such as FFC, namely that they ignore the underlying link availability (and the derived probability of failure). As discussed elsewhere [15, 14], link availability greatly varies across different links. Consequently, probability-oblivious TE solutions might lead to low network efficiency under prevailing conditions to accommodate potentially highly unlikely failure scenarios (i.e., with little bearing on availability). However, not only might a probability-oblivious approach overemphasize *unlikely* failure scenarios, it might even disregard *likely* failure scenarios. Consider a scenario where three links in a large network have low availability (say, 99% each), and all other links have extremely high availability (say, 99.999%). When the operator's objective is to withstand two concurrent link failures, the scenario where the three less available links might be simultaneously unavailable will not be considered, but much less likely scenarios in which two of the highly available links fail simultaneously will be considered.

To explain the motivation for a risk-management approach, let us revisit the example in Figure 1-2. Now, suppose the probability of a link being up is as described in the figure, and the link failure probabilities are uncorrelated (I will discuss correlated failures in Chapter 3). In this case, the probability of different failure scenarios can be expressed in terms of individual links' failure probabilities (e.g., the probability of all three links failing simultaneously is 10^{-7}). Under these failure probabilities, the network can support 30Gbps traffic almost 90% of the time simply by utilizing the full bandwidth of each link and not rerouting traffic in the event of failures.

FFC's solution, shown in Figure 1-2(b), can be regarded as corresponding to the objective of maximizing the throughput for a level of availability in the order of 7 nines (99.99999%), as the scenario of all links failing concurrently occurs with probability 10^{-7}. Observe that the bandwidth assignment in Figure 1-3(b) guarantees a total throughput of 20Gps at a level of availability of nearly 3 nines (99.8%).[1] Thus, the network administrator can trade network utilization for availability to reflect the operational objectives and strike a balance between the two.

1.3 My work: TeaVaR (Traffic engineering applying value at risk)

Instead of reasoning about availability indirectly in terms of the maximum number of tolerable failures [30], network operators could generate a probabilistic failure model from empirical data (e.g., encompassing uncorrelated/correlated link failures, node failures, signal decay, etc.) and optimize TE with respect to an availability bound.

My collaborators and I implemented a TE optimization framework called TeaVaR. This framework allows operators to harness the information required to fine-tune the tradeoff between network utilization and availability, thus striking a balance that best suits their goals. TeaVaR is the first formal TE framework that enables operators to jointly optimize network utilization and availability. In the next chapter, I discuss the idea for this framework; I explain how it relates to portfolio optimization and financial risk modeling and show how this mindset can be applied to TE. In Chapter 3, I explain the Linear Program of TeaVaR in more detail.

Note that this approach to risk-aware TE is orthogonal and complementary to the challenge of capacity planning. While capacity planning focuses on determining in what manner WAN capacity should be augmented to provide high availability, my goal is to optimize the utilization of available network capacity with respect to *real-time* information about traffic demands and expected failures. I explain how this approach can be applied to capacity planning in Chapter 5.

[1] The probability of the upper and lower links both being up, regardless of the middle link, is $(1-10^{-3})^2 = 0.998$.

Chapter 2

Probabilistic approach to risk management

In this chapter, I relate the central concept of Value at Risk (VaR) in finance to resource allocation in networks and, more specifically, to TE. I then highlight the main challenges of and ideas underlying TeaVaR—a CVaR-based TE solution. A full description of TeaVaR appears in Chapter 3.

2.1 Probabilistic risk management in finance

In many financial contexts, the goal of an investor is to manage a collection of assets (e.g., stocks), also called a portfolio, so as to maximize the expected return on the investment *while* considering the probability of possible market changes that could result in losses or smaller-than-expected gains.

Consider a setting in which an investor must decide how many of each type of stock to acquire by quantifying the return from the different investment possibilities. Let $x = (x_1, \ldots, x_n)$ be a vector representing an investment, where x_i represents the amount of stock i acquired, and let $y = (y_1, \ldots, y_n)$ be a vector randomly generated from a probability distribution reflecting market statistics, where y_i represents the return on investing in stock i. In financial risk literature, vector x is termed *the control* and vector y is termed *the uncertainty vector*. The loss function $L(x, y)$

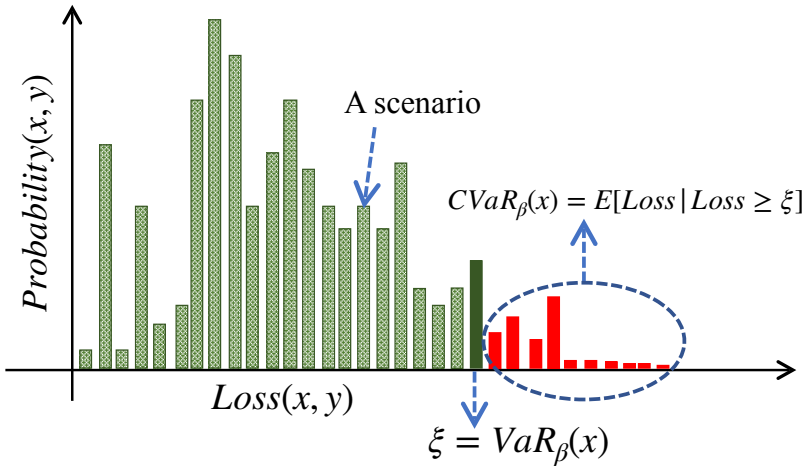


Figure 2-1: An illustration of $VaR_\beta(x)$ and $CVaR_\beta(x)$.

captures the return on investment x under y and is simply $L(x, y) = -\Sigma_{i=1}^{n} x_i y_i$, i.e., the negative of the gain.

Investors wish to provide customers with bounds on the loss they might incur, such as "the loss will be less than \$100 with probability 0.95," or "the loss will be less than \$500 with probability 0.99." Value at Risk (VaR) [22] captures these bounds. Given a probability threshold β (say $\beta = 0.99$), VaR_β provides a probabilistic upper bound on the loss: the loss is less than VaR_β with probability β.

Figure 2-1 gives a graphical illustration of the concepts of VaR_β and $CVaR_\beta$ (described below). For a given control vector x and probability distribution on the uncertainty vector y, the figure plots the probability mass function of individual scenarios (x, y), sorted according to the loss associated with each scenario. Assuming all possible scenarios are considered, the total area under the curve amounts to 1. At the point on the x-axis marked by $\xi = VaR_\beta(x)$, the area under the curve is greater than or equal to β. Given a probability threshold β (say $\beta = 0.99$) and a fixed control x, $VaR_\beta(x)$ provides a probabilistic upper bound on the loss: the loss is less than $VaR_\beta(x)$ with probability β. Equivalently, $VaR_\beta(x)$ is the β-percentile of the loss given x. Value at Risk (VaR_β) is obtained by minimizing $VaR_\beta(x)$ (or ξ) over all possible control vectors x, for a given a probability threshold β. The VaR notion has been applied to various contexts, including hedge fund investments [36], energy markets [10], credit risk [4], and even cancer treatment [31].

Note that VaR_β does not necessarily minimize the loss at the *tail* (colored in red in Figure 2-1), i.e., the worst scenarios in terms of probability, which have total probability mass of at most $1-\beta$. A closely related risk measure that does minimize the loss at the tail is termed *β-Conditional Value at Risk* ($CVaR_\beta$)[35]; $CVaR_\beta$ is defined as the expected loss at the tail, or, equivalently, as the expected loss of all scenarios with loss greater or equal to VaR_β. VaR minimization is typically intractable. In contrast, minimizing CVaR can be cast as a convex optimization problem under mild assumptions [35]. Further, minimizing CVaR can be a good proxy for minimizing VaR.

2.2 Probabilistic risk management in networks

Optimizing traffic flow in a network also entails contending with loss, which in this context reflects the possibility of failing to satisfy user demands when traffic shifts as link/node failures congest the network. This section presents a high-level overview of how the VaR and CVaR can be applied to this context. See Chapter 3 for the formal presentation of TeaVaR.

We can model the WAN as a network graph, in which nodes represent switches, edges represent links, and each link is associated with a capacity. Links (or, more broadly, shared risk groups) also have failure probabilities. As in prior studies [17, 30, 19], in each time epoch a set of source-destination switch-pairs ("commodities" or "flows") wish to communicate, where each such pair f is associated with a demand d_f and a fixed set of possible routes (or tunnels) T_f on which its traffic can be routed.

Intuitively, under the formulation of TE optimization as a risk-management challenge, the control vector $a_{f,t}$ captures how much bandwidth is allocated to each flow on each of its tunnels, and the uncertainty vector y specifies, for each tunnel, whether the tunnel is available (i.e., whether all of its links are up). Note that y is stochastic, and its probability distribution is derived from the probabilities of the underlying failure events (e.g., link/node failures). The aim is to maximize the bandwidth assigned to users subject to a desired, operator-specified, availability threshold β.

However, applying $CVaR_\beta$ to network resource allocation incurs three nontrivial challenges.

Achieving fairness across network users. Avoiding starvation and achieving fairness are arguably less pivotal in stock markets, because money is money no matter where it comes from, but they are essential in network resource allocation. In particular, TE involves multiple network users, and a crucial requirement is that high bandwidth and availability guarantees for some users not come at the expense of unacceptable bandwidth or availability for others. In the formulation, this translates into carefully choosing the loss function $L(x, y)$ so that minimizing the chosen notion of loss implies that such undesirable phenomena do not occur. This also means we must represent the control vector as 2-dimensional so that we can have a control vector associated with every flow. I refer to the control vector as $a_{f,t}$ where f refers to the flows and t represents a tunnel for that flow. I show how these modifications are accomplished in Chapter 3.

Capturing fast rerouting of traffic in the data plane. Unlike the above formulation of stock management, in the TE context the consequences of the realization of the uncertainty vector cannot be captured by a simple loss function, such as $L(a, y) = -\Sigma_{t=1}^{n} a_{f,t} y_t$ because the CVaR-based optimization formalism must take into account that the unavailability of a certain tunnel might imply more traffic having to traverse *other* tunnels.

Providing high availability in WAN TE cannot rely on the online global recomputation of tunnels as this can be too time consuming and adversely impact availability [30, 38]. To quickly recover from failures, TeaVaR is required to readjust the traffic splitting ratios on surviving tunnels via re-hashing mechanisms implemented in the data plane [30, 38]. Thus, the realization of the uncertainty vector, which corresponds to a specification of which tunnels are up, impact the control, capturing how much is sent on each tunnel.

Achieving computational tractability. A naive formulation of the CVaR-minimizing TE machinery yields a non-convex optimization problem. Hence, the first challenge is to transform the basic formulation into an equivalent convex program. We are,

in fact, able to formulate the TE optimization as a linear program through careful reformulation using auxiliary variables. In addition, because the number of all possible failure scenarios increases exponentially with the network size, solving this LP becomes intractable for realistic network sizes. To address this additional challenge, I introduce an efficient pruning process that allows us to consider fewer scenarios. I explain scenario pruning in §3.3 and we show how it substantially improves the runtime with little effect on accuracy in Chapter 4.

Chapter 3

Optimization framework

In this chapter, I describe the TeaVaR optimization framework in detail. I formalize the model and delineate the goals of WAN TE [19, 17, 27, 30] (§3.1). Then, I introduce TeaVaR's novel approach to TE, showing that it enables probabilistic guarantees on network throughput (§3.2).

3.1 Overview of WAN TE

In this section, I give a brief overview of a typical approach to WAN TE. Table 3.1 shows these inputs, along with a the additional inputs and outputs of TeaVaR. The table also shows the linear program for TeaVaR, which is described in more detail later in this chapter.

Input. As in other WAN TE studies, I model the WAN as a directed graph $G = (V, E)$, where the vertex set V represents switches and edge set E represents links between switches. Link capacities are given by $C = (c_1, \ldots, c_{|E|})$ (e.g., in bps) and as in any TE formulation, the total flow on each link should not exceed its capacity. TE decisions are made at fixed time intervals (say, every 5 minutes [17]), based on the estimated user traffic demands for that interval. In each time epoch, there is a set of source-destination switch-pairs ("commodities" or "flows"); each such pair f is associated with a demand d_f and a fixed set of paths (or "tunnels") $T_f \in T$ on which its traffic should be routed. TeaVaR assumes the tunnels are part of the input.

TE Input	$G(V, E)$	Network graph with switches V and links E.
	$c_e \in C$	Bandwidth capacity of link $e \in E$.
	$d_f \in D$	Bandwidth demand of flow $f \in F$.
	$T_f \in T$	Set of tunnels for flow $f \in F$.
Additional TeaVaR Input	β	Target availability level (e.g.,99.9%).
	$s \in S$	Network state corresponding to failure scenarios of shared risk groups.
	p_q	Probability of network state q.
Auxiliary Variables	u_s	Total loss in scenario s.
	$t_{s,f}$	Loss on flow f in scenario s
	$Y_{s,t}$	1 if tunnel t is available in scenario s, 0 otherwise
	$X_{t,e}$	1 if tunnel t uses edge e, 0 otherwise
TE Output	b_f	Total bandwidth for flow f.
	$a_{f,t}$	Allocated traffic on tunnel $t \in T_f$ for flow $f \in F$.
Additional TeaVaR Output	α	"Loss" (a.k.a the Value at Risk (VaR)).

$$\begin{aligned}
\text{minimize} \quad & \alpha + \tfrac{1}{1-\beta} \sum_{s \in S} p_s u_s \\
\text{subject to} \quad & \sum_{f \in F, t \in T_f} a_{f,t} X_{t,e} \leq c_e \qquad \forall e \\
& u_s \geq t_{f,s} - \alpha \qquad \forall f, s \\
\text{where} \quad & t_{s,f} = 1 - \frac{\sum_{t \in t_f} a_{f,t} Y_{s,f}}{d_f} \qquad \forall f, s
\end{aligned}$$

Table 3.1: Key notations in the TeaVaR formulation. The original optimization problem is minimizing (3.3) subject to (3.1) – (3.2). Here, we show the LP formulation; see §3.2 for its derivation.

In Chapter 4, I evaluate the impact of the tunnel selection scheme (e.g., k-shortest paths, edge-disjoint paths, oblivious-routing) on performance. My evaluation shows the TeaVaR optimization improves the achievable utilization-availability balance for all considered tunnel-selection schemes.

Output. The output of TeaVaR, consists of two parts (see Table 3.1): (1) the *total* bandwidth b_f that flow (source-destination pair) f is permitted to utilize (across all of its tunnels in T_f); (2) a specification for each flow f of how its allocated bandwidth b_f is split across its tunnels T_f. The bandwidth allocated on tunnel t is denoted by

$a_{f,t}$.

Optimization goal. Previous studies of TE considered optimization goals such as maximizing total concurrent flow [37, 17, 6, 30], max-min fairness [34, 19, 11], minimizing link over-utilization [27], minimizing hop count [28], and accounting for hierarchical bandwidth allocations [26]. As formalized below, an appropriate choice for the present context is selecting the $a_{f,t}$s (per-tunnel bandwidth assignments) in a manner that maximizes the well-studied maximum-concurrent-flow objective [37]. This choice of objective will enable us to maximize network throughput while achieving some notion of fairness in terms of availability across network users. In §3.2, I discuss ways to extend the framework to other optimization objectives.

Under maximum-concurrent-flow, the goal is to maximize the value $\delta \in [0, 1]$ such that at least an δ-fraction of each flow f's demand is satisfied across all flows. For example, $\delta = 1$ implies all demands are fully satisfied by the resulting bandwidth allocation, while $\delta = \frac{1}{3}$ implies at least a third of each flow's demand is satisfied.

3.2 TeaVaR: deriving the LP

TeaVaR's additional inputs and outputs are listed in Table 3.1. Given a target availability level β, the goal is to cast TE optimization as a CVaR-minimization problem whose output is a bandwidth allocation to flows that can be materialized with probability of at least β. Doing so requires careful specification of (i) the "control" and "uncertainty" vectors, as described in Chapter 2, and (ii) a "loss function" that provides fairness and avoids starvation across network flows. The formulation is shown in Table 3.1. In this section, I describe in detail how this formula is derived.

3.2.1 Probabilistic failure model

I consider a general failure model, consisting of a set of *failure events* Z. A failure event $z \in Z$ represents a set of SRGs becoming unavailable (the construction of a set of SRGs is described elsewhere [38, 25], and §4.1). Importantly, while failure events in my formulation are *uncorrelated*, this does not preclude modeling *correlated link*

failures. Consider a failure event z representing a technical failure in a certain link l and another failure event z' representing a technical failure in a switch or a power outage that will cause multiple links, including l, to become unavailable concurrently. Note that even though link l is inactive, whether z or z' is realized, z and z', which capture failures of different components, are independent events.

Each failure event z occurs with probability p_z. As described earlier, the failure probabilities are obtained from historical data (see Chapter 4 for more details on estimation techniques, as well as sensitivity analysis for inaccuracies in these estimations). I denote by $s = (s_1, \ldots, s_{|Z|})$ a network *state*, where each element s_z is a binary random variable, indicating whether failure event z occurred ($s_z = 1$) or not. For example, for a network with 15 SRGs, the possible set of events (Z) is a vector with 15 elements, where each element indicates whether the corresponding SRG has failed or not. For example, $\hat{s} = (0, \ldots, 0, 1)$ captures the network state in which only SRG_15 is down. More formally, let S be the set of all possible states, and let $p_{\hat{s}}$ denote the probability of state $\hat{s} = (\hat{s}_1, \ldots \hat{s}_{|Z|}) \in S$. Hence, the probability of network state s can be obtained using the following equation,

$$\hat{p}_s = P(s_1 = \hat{s}_1, \ldots, s_{|Z|} = \hat{s}_{|Z|}) = \Pi_z\big(\hat{s}_z p_z + (1 - \hat{s}_z)(1 - p_z)\big).$$

The uncertainty vector specifies which tunnels are up. I define y as a vector of size $|T|$, where each vector element y_t is a binary random variable that captures whether tunnel t is available ($y_t = 1$) or not ($y_t = 0$). This random variable depends on realizations of relevant failure events. For example, y_t will equal 0 if one of the links or switches on the tunnel is down. Since each random variable y_t is a function of the random network state s, we often use $y_t(s)$, and $y(s)$ to denote the resulting vector of random variables. In the LP formulation, I pre-compute these values to provide fast look-ups and store them in the matrix $Y_{s,t}$, where $Y_{s,t}$ is a 1 if tunnel t is available in scenario s or 0 if it is not.

The control vector specifies how bandwidth is assigned to tunnels. Recall that the output a in my WAN TE formulation captures the traffic allocation for each

flow on each of its tunnels. This is the control vector for my CVaR-minimization. As in TE schemes, such per tunnel bandwidth assignment has to ensure the edge capacities are respected, i.e., satisfy the following constraint:

$$\sum_{f \in F, t \in T_f} a_{f,t} X_{t,e} \leq c_e, \quad \forall e \in E. \tag{3.1}$$

Here, this constraint must sum all overlapping flows on any given edge. The matrix $X_{t,e}$ is a binary variable that represents the edges for each tunnel as a 1 if tunnel t uses a certain edge e, and 0 otherwise. This matrix is pre-computed to allow fast look-ups. To account for potential failures, I allow the total allocated bandwidth per user a, $\sum_{t \in T_f} a_{f,t}$, to exceed its demand d_f.

3.2.2 Loss function

I define the loss function in two steps. First, I define a loss function for each flow, called the flow-level loss. Then, I define a total loss as a function of the flow-level losses for all flows, known as the network-level loss.

Flow-level loss. Recall that in the TE formulation, the optimization objective is to assign the control variables $a_{f,t}$ (per-tunnel bandwidth allocations) in a manner that maximizes the concurrent-flow, i.e., maximizes the value δ for which each flow can send at least a δ-fraction of its demand. To achieve this, loss in the framework is measured in terms of the fraction of demand *not* satisfied (i.e., $1 - \delta$). My goal thus translates into generating the per-tunnel bandwidth assignments that minimize the fraction of demand not satisfied for a specified level of availability β.

In my formulation, the maximal satisfied demand for flow f is given by the sum of the allocations on the tunnels for that flow, $\sum_{t \in T_f} a_{f,t} Y_{s,f}$. Thus, the loss for each flow f with respect to its demand d_f is captured by $\left[1 - \frac{\Sigma_{t \in T_f} a_{f,t} Y_{s,f}}{d_f}\right]^+$, where $[z]^+ = \max\{z, 0\}$; note that the $[+]$ operator ensures that the loss is not negative (hence, the optimization will not gain by sending more traffic than the actual demand). This notion of per-flow loss captures the loss of assigned bandwidth for a given network state s.

Network-level loss function. To achieve fairness, in terms of availability, we define the global loss function as the maximum loss across all flows, i.e.,

$$L(a, y) = \max_i \left[1 - \frac{\Sigma_{t \in T_f} a_{f,t} y_r}{d_i}\right]^+. \tag{3.2}$$

While this loss function is nonlinear, I am able to transform into a linear constraint and thus turn the optimization problem into a linear program (LP) in the next section.

3.2.3 Objective function

To formulate the optimization objective, I introduce the mathematical definitions of VaR_β and $CVaR_\beta$. For a given loss function L, the $VaR_\beta(x)$ is defined as $V_\beta(a) = \min\{\xi \mid \psi(a, \xi) \geq \beta\}$, where $\psi(a, \xi) = P(s \mid L(a, y(s)) \leq \xi)$, and $P(q \mid L(a, y(s)) \leq \xi)$ denotes the cumulative probability mass of all network states satisfying the condition $L(a, y(s)) \leq \xi$. $CVaR_\beta$ is simply the mean of the β-tail distribution of $L(a, y)$, or put formally:

$$CVaR_\beta(a) = \frac{1}{1-\beta} \sum_{L(a,y(s)) \geq V_\beta(a)} p_q L(a, y(s)).$$

Note that the definition of $CVaR_\beta$ utilizes the definition of VaR_β. To minimize $CVaR_\beta$, I define the following potential function

$$\begin{aligned} F_\beta(a, \alpha) &= \alpha + \frac{1}{1-\beta} E[[L(a, y) - \alpha]^+] \\ &= \alpha + \frac{1}{1-\beta} \sum_s p_s [L(a, y(s)) - \alpha]^+. \end{aligned} \tag{3.3}$$

The optimization goal is to minimize $F_\beta(x, \alpha)$ over $X, \mathcal{R}$, subject to (3.1) – (3.2). I leverage the following theorem, which states that by minimizing the potential function, the optimal $CVaR_\beta$ and (approximately) also the corresponding VaR_β are obtained.

Theorem 1 *[36] If (a^*, α^*) minimizes F_β, then not only does a^* minimize the $CVaR_\beta$*

C_β over X, but also

$$C_\beta(a^*, \alpha^*) = F_\beta(a^*, \alpha^*), \tag{3.4}$$

$$V_\beta(a^*) \approx \alpha^*. \tag{3.5}$$

The beauty of this theorem is that although the definition of $CVaR_\beta$ uses the definition of VaR_β, we do not need to work directly with the VaR_β function $V_\beta(a)$ to minimize $CVaR_\beta$. This is significant since, as mentioned above, $V_\beta(a)$ is a non-smooth function which is hard to deal with mathematically. The statement of the theorem uses the notation $\approx$ to denote that with high probability, α^* is equal to $V_\beta(a^*)$. When this is not so, α^* constitutes an upper bound on the VaR_β. The actual VaR_β can be easily obtained from α^*, as discussed in Appendix A.1.

3.2.4 Linearizing the loss function

I am interested in minimizing

$$F_\beta(a, \alpha) = \alpha + \frac{1}{1-\beta} E[\max\{0, L(x, y) - \alpha\}] \tag{3.6}$$

$$= \alpha + \frac{1}{1-\beta} \Sigma_q p_y [L(x, y(q)) - \alpha]^+, \tag{3.7}$$

In what follows, I "linearize" the objective function by adding additional (linear) constraints. I introduce a new set of variables $u = \{u_s\}$, where u_s represents the total loss of a given network state. I rewrite the objective function as

$$\tilde{F}_\beta(s, \alpha) = \alpha + \frac{1}{1-\beta} \sum_s p_s u_s, \tag{3.8}$$

and add the following constraints

$$u_s \geq L(a, y(s)) - \alpha \quad \forall s \tag{3.9}$$

$$u_s \geq 0. \quad \forall s \tag{3.10}$$

Observe that minimizing F w.r.t. a and α is equivalent to minimizing $\tilde{F}$ w.r.t. s, a, α. I have removed the initial max operator and + operator in (3.6). However, $L(\cdot,\cdot)$ still involves a max operator. We must rewrite (3.9) as $u_s + \alpha \geq L(a, y(s))$; now we can materialize the max operator through the following inequalities

$$u_s + \alpha \geq 0, \quad \forall s \tag{3.11}$$

$$u_s + \alpha \geq t_{s,f} \quad \forall s, f, \tag{3.12}$$

where

$$t_{s,f} = 1 - \frac{\Sigma_{t \in T_f} a_{f,t} y_r(s)}{d_f}. \tag{3.13}$$

We end up with an LP with decision variables $a, \alpha, u_s, t_{s,f}$ (u_s and $t_{s,f}$ can be viewed as auxiliary variables); the objective of the LP is minimizing (3.8) subject to (3.1), (3.10)–(3.13).

3.2.5 Routing (and re-routing) in TEAVAR

Bandwidth allocations. The total bandwidth that flow f is permitted to utilize (across all tunnels in T_f) is given by b_f. Clearly, b_f should not exceed flow f's demand d_f to avoid needlessly wasting capacity. However, I do not add an explicit constraint for this requirement. Instead, I embed it implicitly in the loss function (3.2). Once the solution is computed, each flow f is given: (i) the total allowed bandwidth $(1 - V_\beta(a^*))\%$ of its demand; i.e., $b_f = (1 - V_\beta(a^*))d_f$; and (ii) a weight assignment $w_{f,t}$, where $w_{f,t} = \frac{a^*_{f,t}}{\sum_{t \in T_f} a^*_{f,t}}$.

Tunnel Weights As in other TE solutions (e.g., [30, 38]), I use a simple rule for flow re-assignment in the event of network failures: the traffic of flow f is split between all surviving tunnels in T_f so that each tunnel t carries traffic proportionally to $w_{f,t}$. Put formally, let $\bar{T}_f \subseteq T_f$ be the subset of f's tunnels that are available. Then, each tunnel $t \in \bar{T}_f$ carries $\frac{w_{f,t}}{\sum_{t \in \bar{T}_f} w_{f,t}} b_f$ traffic (i.e., its proportional share of b_f). I henceforth refer to this allocation rule as the *proportional assignment* rule.

Proportional assignment does not require global coordination/computation upon

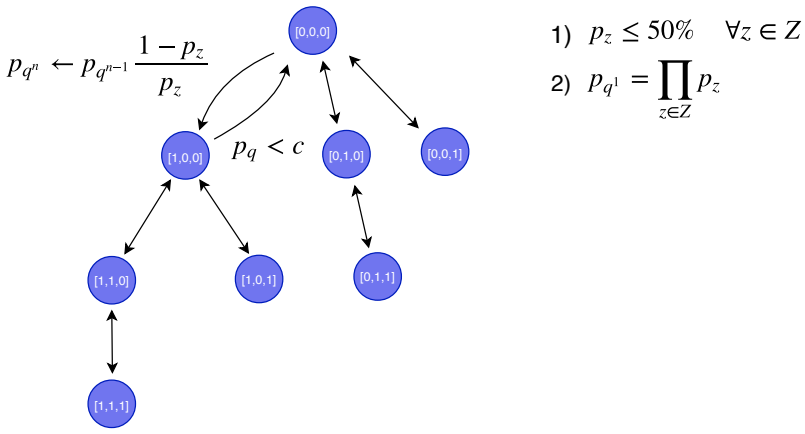


Figure 3-1: A diagram of the scenario space pruning algorithm with assumptions 1) and 2). The algorithm recursively searches the tree depth first, updating p_q at each level until $p_q < c$ upon which it regresses one level.

failures and can be easily implemented in the data plane via (re-)hashing. Because the proportional assignment rule is not directly encoded in the CVaR minimization framework, traffic re-assignment might result in congestion, i.e., violating constraint (3.1). Nevertheless, it turns out that this rather simple rule guarantees such violation occurs with very low probability (upper-bounded by $1 - \beta$). Formally,

Theorem 2 *Under* TeaVaR*, each flow i is allocated bandwidth $(1 - V_\beta(a^*))d_f$ and no link capacity is exceeded with probability of at least β.*

See Appendix A.2 for the proof.

3.3 Scenario pruning algorithm

As discussed earlier, applying TeaVaR to a large network is challenging because the number of network states, representing combinations of link failures, increases exponentially with the network size. To deal with this, I devise a scenario pruning algorithm to efficiently filter out scenarios that occur with negligible probability. The main idea behind the algorithm is to use a tree representation of the different scenarios, where the scenario probability decreases with movement away from the root. The tree is traversed recursively with a stopping condition of reaching a scenario whose

probability is below a *cutoff threshold*. The pruned scenarios are accounted for in the optimization as follows. To give an upper-bound on the $CVaR_\beta$ (equivalently, a lower-bound on the throughput), I collapse all the pruned scenarios into a single scenario with probability equal to the sum of probabilities of the pruned scenarios. I then associate a maximal loss of 1 with that scenario. In §4.7, I evaluate the impact of the scenario pruning algorithm on both running time and accuracy.

Recall that a scenario is represented as a set of failure events. I introduce the idea of a scenario cutoff c, and prune out all scenarios that occur with probability $p < c$. Generally, I refer to failure events as independent events that together can make up a network state. A network state can be modeled as a bitmap where each bit represents whether or not an edge can carry traffic. I call this bitmap a *scenario*. Each failure event, such as a fiber cut, a node outage, or a correlated link failure, can lead to a scenario in which one or more bits in a network state is flipped.

The scenario pruning algorithm (see Figure 3-1 for an illustration), uses a tree representation of the different scenarios, where the root note is the scenario where no failure event occurs $[0, 0, \ldots, 0]$, and every child node differs from its parent by one bit. The tree is constructed such that each flipped bit must be to the right of the previously flipped bit to prevent revisiting any previously visited states. Assuming each failure event occurs with probability less than 0.5, the scenario probability decreases as we traverse away from the root. Accordingly, we traverse the tree in a depth-first search until the condition $p_q < c$ is met, at which point no further scenarios down that path need to be searched. It is important to efficiently calculate the scenario probabilities while traversing the tree. To do so, I update the probability of a child scenario incrementally from its parent via $p_q^n \leftarrow p_q^{n-1} \frac{1-p_z}{p_z}$; this update rule follows from immediately from the formula $\hat{p}_q = P(q_1 = \hat{q}_1, \ldots, q_{|Z|} = \hat{q}_{|Z|}) = \Pi_z \big(\hat{q}_z p_z + (1 - \hat{q}_z)(1 - p_z)\big)$. Note that the tree is not balanced, because if we consider failures in order, taking the leftmost path in the tree, we do not have to revisit any scenarios where failure z has previously occurred.

Chapter 4

Evaluating TeaVaR

In this section, I present the evaluation of TeaVaR. I begin by describing the experimental framework and evaluation methodology (§4.1). The experimental results focus on the following elements:

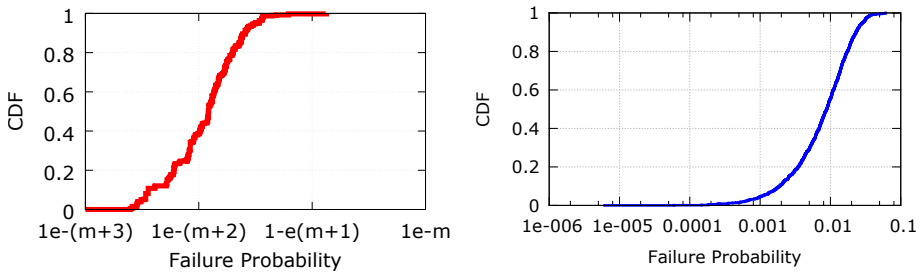
1. Benchmarking TeaVaR's performance against the state-of-the-art TE schemes (§4.2).

2. Examining TeaVaR's robustness to noisy estimates of failure probabilities (§4.6).

3. Quantifying the effect of scenario pruning on the running time and the quality of the solution (§4.7).

Topology Name	#Nodes	#Edges
B4	12	38
IBM	18	48
ATT	25	112
X-Net	≈ 30	≈ 100

Table 4.1: Network topologies used in the evaluations. For confidentiality reasons, I do not report exact numbers for the X-Net topology.



(a) Empirical data from X-Net (b) Weibull distribution

Figure 4-1: CDF of failure probabilities used in our experiments. The exact value of m in (a) is not shown due to confidentiality reasons. The shape and scale parameters in (b) are 0.8 and 10^{-4}, respectively.

4.1 Experimental setting

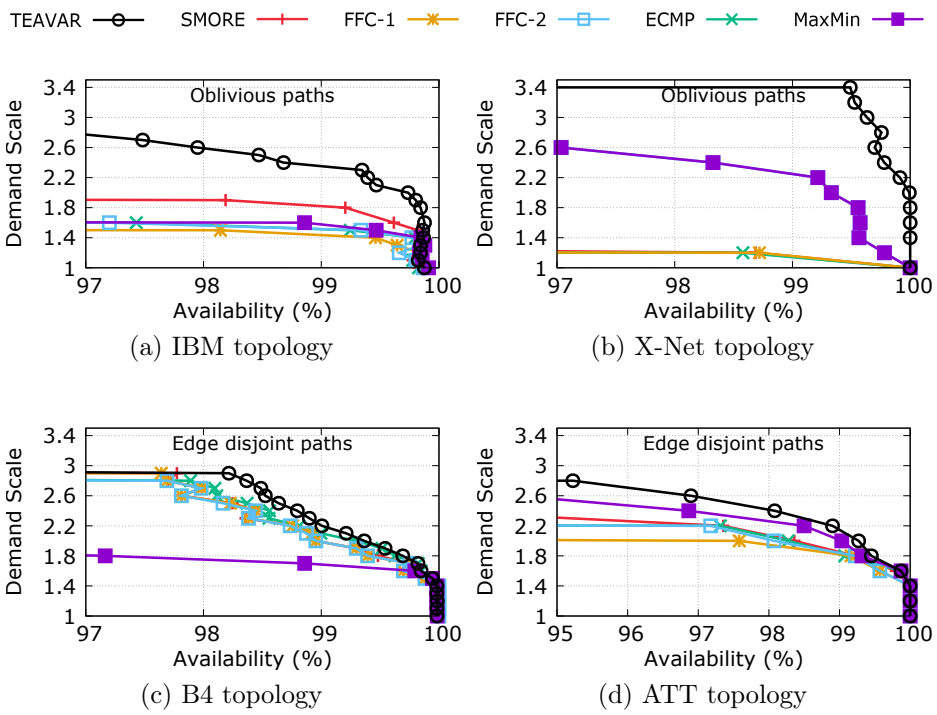
Topologies. I evaluate TeaVaR on four network topologies: B4, IBM, ATT, and X-Net. The first three topologies (and their traffic matrices) were obtained from the authors of SMORE [27]. X-Net is the network topology of a large cloud provider in North America. See Table 4.1 for a specification of network sizes. The empirical data from the X-Net network consists of the following: the capacity of all links (in Gbps) and the traffic matrices (source, destination, amount of data in Mbps) over four months at a resolution of one sample per hour. Data on failure events include the up/down state of each link at 15-minute granularity over the course of a year, as well as a list of possible shared risk groups. Data on ATT, B4, and IBM topologies include a set of at least 24 demand matrices and link capacities, but per-link failure probabilities are missing. Hence, I use a Weibull distribution derived from the X-Net measurements and change its parameters over the course of the simulations.

Tunnel selection. TE schemes [24, 20, 17, 30] often use link-disjoint tunnels for each source-destination pair. However, recent work shows performance improvement by using oblivious tunnels (interchangeably also referred to as oblivious paths) [27]. Because TeaVaR's optimization framework is orthogonal to tunnel selection, I run simulations with a variety of tunnel-selection schemes, including oblivious paths, link disjoint paths, and k-shortest paths. As I show later in the section, TeaVaR achieves higher throughput regardless of the tunnel selection algorithm. We also study the

impact of tunnel selection on TeaVaR and find that combining TeaVaR with the tunnel selection of oblivious-routing [27] leads to better performance (§4.2).

Deriving failure probability distributions. For each link e, I examine historical data and track whether e was up or down in a measured time epoch. Each epoch is a 15-minute period. I obtain a sequence of the form $(\psi_1, \psi_2, \ldots)$ such that‘ each ψ_t specifies whether the link was up ($\psi_t = 1$) or down ($\psi_t = 0$) during the tth measured time epoch. From this sequence, another sequence $(\delta_1, \delta_2, \ldots, \delta_M)$ is derived such that δ_j is the number of consecutive time epochs the link was up prior to the jth time it failed. For example, from the sequence $(\psi_1, \psi_2, \ldots, \psi_{12}) = (1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0)$, I derive the sequence $(\delta_1, \delta_2, \delta_3) = (2, 3, 1)$ (the link was up for 2 time epochs before the first failure, 3 before the second failure, and 1 before the last failure). An unbiased estimator of the mean *uptime* is given by $U = \frac{\Sigma_{j=1}^{M} \delta_j}{M}$. I make a simplified assumption that the link up-time is drawn from a geometric distribution (i.e., the failure probability is fixed and consistent across time epochs). Then, the failure probability p_e of link e is simply the inverse of the mean uptime, that is, $p_e = \frac{1}{U}$. Note that we can use this exact analysis for other shared-risk groups, such as switches.

Figure 4-1(a) plots the cumulative distribution function (CDF) for the failure probability across the network links, derived by applying the above analysis methodology to the empirical availability traces of the X-Net network. The x-axis on the plot represents the failure probability, parametrized by m. The exact value of m is not disclosed for reasons of confidentiality. Nonetheless, the important takeaway from this figure is that the failure probabilities of different links might differ by orders of magnitude. To accommodate the reproducibility of results, I obtain a Weibull probability distribution which fits the shape of the empirical data. The Weibull distribution, which has been used in prior study of failures in large backbones [32], is used here to model failures over time for topologies for which I do not have empirical failure data. I denote the Weibull distribution with shape parameter λ and scale parameter k by $W(\lambda, k)$. In Figure 4-1(b), I plot the Weibull distribution used in the evaluation, as well as the parameters needed to generate it. Throughout the experiments, I change the shape and scale parameters of the Weibull distribution and study the impact of



(a) IBM topology (b) X-Net topology

(c) B4 topology (d) ATT topology

Figure 4-2: Comparison of TeaVaR against various TE schemes under different tunnel selection algorithms. All schemes in (a) and (b) use oblivious paths, and all schemes in (c) and (d) use edge disjoint paths. The term *availability* refers to the percentage of scenarios that meet the 100% demand-satisfaction requirement.

probability distribution on performance.

Optimization. The optimization framework uses the Gurobi LP solver [16] and is implemented using the Julia optimization language [8].

4.2 Throughput *vs.* availability

I examine the performance of different TE schemes with respect to both throughput and availability.

Setup. I benchmark TeaVaR against several approaches: SMORE [27], FFC [30], MaxMin (in particular, the algorithm used in B4 [19, 11]), and ECMP [12]. SMORE minimizes the maximum link utilization without explicit guarantees on availability, FFC maximizes the throughput while explicitly considering failures, MaxMin maximizes minimum bandwidth per user [11], and TeaVaR minimizes the CVaR for an input probability. When link failures occur, traffic is redistributed across tunnels ac-

cording to the proportional assignment mechanism (see §3.2) without re-optimizing weights. In our evaluations, I am concerned with *both* the granted bandwidth *and* the probabilistic availability guarantee it comes with. In TeaVaR, the probability is explicit (controlled by the β parameter in the formulation as shown in Eq. 3.3). In FFC, the per-user bandwidth is granted with 100% availability for scenarios with up to k-link failures. However, to fairly compare the ability of the algorithm to accommodate scaled-up demands, I allow all algorithms to send the entire demand (at the expense of potential degradation in availability).

Availability *vs.* demand scale. I first seek to analyze the availability achieved by various TE schemes as demand is scaled up. Given that current networks are designed with traditional worst-case assumptions about failures, all topologies are over-provisioned. Hence, I begin with the input demands, compute the availability achieved when satisfying them for different schemes, and then scale the demands by introducing a (uniform) demand scale-up factor $s \geq 1$ which multiplies each entry in the demand matrix.

Availability is calculated by running a post-processing simulation in which I induce failure scenarios according to their probability of occurrence, and attempt to send the entirety of the demand through the network. For each scenario, I record the amount of unsatisfied demand (loss) for each flow, as well as the probability associated with that scenario. The sum of the probabilities for scenarios where demand is *fully satisfied* reflects the availability in that experiment. For example, if a TE scheme's bandwidth allocation is unable to fully satisfy demand in 0.1% of scenarios, it has an availability of 99.9%. I then scale the demand matrix and repeat the above analysis for at least 24 demand matrices per topology. In Figure 4-2, I summarize the results by depicting the demand-scale vs. the corresponding availability.

The results show a consistent trend: TeaVaR achieves higher demand scale for a given availability level. In particular, for any target availability level, TeaVaR achieves up to twice the demand scale-up compared to other approaches. Notably, in X-Net topology and with oblivious paths, TeaVaR is able to scale the demand by a factor of 3.4, while MaxMin achieves a scale up of 2.6. The remaining approaches

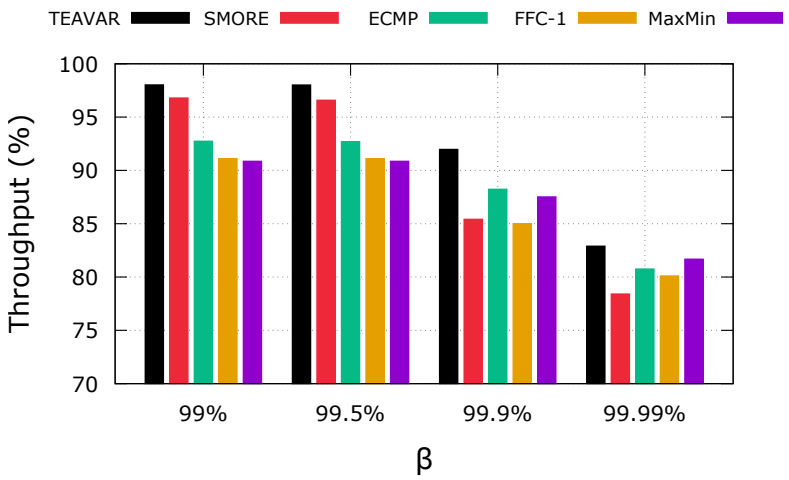


Figure 4-3: Averaged throughput guarantees for different β values on ATT, B4, and IBM topologies. TeaVaR is the only scheme that can explicitly optimize for a given β. For all other schemes, the value on the x-axis is computed based on achieved throughput.

cannot scale beyond 1.4x the original demand.

4.3 Achieved throughput

I next demonstrate the tradeoff between a target availability threshold and the achieved throughput without scaling the demand. In the previous set of experiments, availability was measured as the probability mass of scenarios in which demand is fully satisfied ("all-or-nothing" requirement). In contrast, in this set of experiments, I measure the *fraction* of the total demand that can be guaranteed for a given availability target (or the throughput). This fraction is optimized explicitly in TeaVaR for a given value of β. For other TE schemes, I obtain the fraction of demand using a similar post-processing method as before: for each failure scenario, I simulate the outcome of sending the entire demand through the network, sort the scenarios in increasing loss values and report the demand fraction at the β-percentile (i.e., the throughput is greater or equal to that value for β percent of scenarios). The range of availability values is chosen according to typical availability targets [18].

Figure 4-3 plots the average throughput for ATT, B4, and IBM topologies. For each TE scheme, I report the results under the tunnel selection algorithm which has

	TeaVaR		FFC_1		FFC_2	
Probability	Avg_b	Min_b	Avg_b	Min_b	Avg_b	Min_b
90%	100%	100%	88.32%	4.62%	29.76%	0%
95%	99.9%	99.9%	88.32%	4.62%	29.76%	0%
99%	95.87%	95.87%	88.32%	4.62%	29.76%	0%
99.9%	92.53%	92.53%	88.32%	4.62%	29.76%	0%
99.99%	82.78%	82.78%	0%	0%	29.76%	0%

Table 4.2: The mathematical bandwidth guarantees of TeaVaR and FFC averaged across 3 topologies, and 10 demand matrices. Avg_b represents the average bandwidth for all flows, guaranteed with probability p% of the time. Min_b represents the minimum bandwidth of all flows.

performed the best (SMORE, TeaVaR and MaxMin with oblivious, and FFC with link-disjoint paths). The results demonstrate that TeaVaR is able to achieve higher throughput for each of the target availability values. This is because it can optimize throughput for an explicit availability target within a probabilistic model of failures.

4.4 Mathematical guarantees with tunable β.

The previous two experiments used a post processing simulation is used to calculate availability and throughput. This simulation allows sending all of the demand in the network to compare fairly across all TE schemes instead of adhering to the mathematical guarantees from each scheme. FFC and TeaVaR (explained in §3.2.5) both give permitted bandwidth amounts for each flow (b_i) to ensure their mathematical guarantees hold. FFC guarantees that the amount sent will be successfully received in all k-link failure scenarios. TeaVaR guarantees that the amount sent will be received β% of the time. In Table 4.2, instead of sending all the demand, I assume the permitted bandwidth amounts (b_i) and report the guaranteed average and minimum across all flows. Results are averaged across multiple topologies and demand matrices. While FFC_k gives a single bandwidth amount that only holds up to the probability mass of the k-failure scenarios, we see TeaVaR can achieve significant bandwidth improvements by optimizing explicitly for a given probability. We also

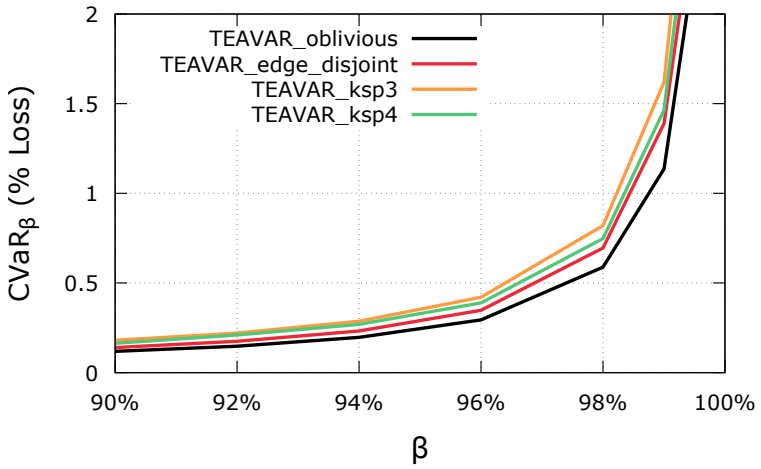


Figure 4-4: The effect of tunnel selection scheme on TeaVaR's performance, quantified by the resulting $CVaR_\beta$ (or loss) for different values of β. TeaVaR using oblivious paths has better performance (lower loss).

see that TeaVaR uses max-min bandwidth allocation so that the average is also the minimum allowed per flow. In contrast, FFC starves certain flows to achieve a slightly higher average bandwidth. The paper on FFC [30] mentions an iterative version of FFC that can avoid starvation, but the LP is not inherently built for fairness or flexible guarantees like TeaVaR.

4.5 Impact of tunnel selection.

So far, TeaVaR has been simulated with either link-disjoint or oblivious tunnels [27] schemes. I now analyze the effect of other tunnel selections on TeaVaR's performance. I demonstrate that while path selection is an important aspect of any TE scheme, no specific choice is needed for TeaVaR's success. In Figure 4-4, I plot the obtained $CVaR_\beta$ as a function of beta for various tunnel selection schemes: k-shortest paths with 3 or 4 paths, FFC's link disjoint paths, and SMORE's oblivious routing. TeaVaR performs comparably well regardless of the tunnel selection scheme. However, the figure shows that oblivious paths are still superior. For example, the obtained $CVaR_\beta$ value is at least 20% better for $\beta = 0.99$ than in any other tunnel selection scheme. These results indicate that an oblivious-routing tunnel selection is a good choice to complement TeaVaR. Oblivious routing is intended to avoid link

over-utilization through diverse and low-stretch path selection. Intuitively, these path properties are useful for TeaVaR, providing the optimization framework with a set of tunnels that can provide high availability.

4.6 Robustness to probability estimates

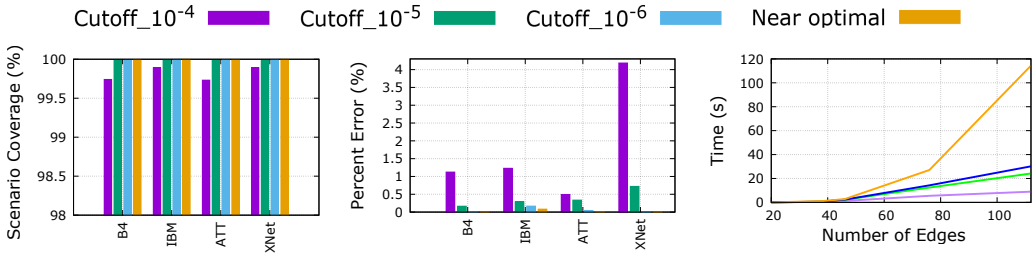
TeaVaR uses a probabilistic model of network failures. In particular, the optimization framework explicitly accounts for the probabilities of failure events, such as link failures. These probabilities are estimated by analyzing the historical time-series of up/down status and are inherently prone to estimation errors, regardless of the technique used (I provide a basic technique here, but more sophisticated techniques, e.g., based on machine-learning, can be applied).

To examine the effects of such inaccuracies, I use the following methodology. I assume there is a set of probabilities that is the ground-truth, and the actual performance of any TE solution should be evaluated against these probabilities. In particular, I evaluate two versions of TeaVaR: (i) TeaVaR using the ground-truth probabilities; (ii) TeaVaR using a *perturbed* version of the ground-truth probabilities (reflecting estimation errors). The generation of (ii) requires a magnitude of noise n. The probabilities that are given to TeaVaR as input are generated via $\tilde{p}_z = p_z + p_z nr$, where r is random noise, distributed uniformly on $[-1, 1]$.

For each noise level in Table 4.3, I report the percent error in average throughput across all scenarios compared to the error when optimized using the ground-truth

Noise in probability estimations	% error in throughput
1%	1.43%
5%	2.95%
10%	3.07%
15%	3.95%
20%	6.73%

Table 4.3: The effect of inaccurate probability estimations on TeaVaR's performance. The decrease in throughout is caused by running TeaVaR with the ground-truth probabilities $\{p_q\}$.



(a) Scenario coverage (b) Cutoff error (c) Optimizer times

Figure 4-5: Impact of scenario pruning on accuracy and runtime.I use 10^{-4} as the cutoff threshold for ATT and 10^{-5} for all other topologies. (a) The cutoff thresholds cover more than 99.5% of all possible scenarios. (b) The error incurred by pruning scenarios is less than 5% in all cases. (c) The cutoffs applied lead to manageable running times.

probabilities. The noise has a relatively small effect on the solution quality. This is because fine-grained distribution modeling pays off less than intuition might lead us to believe. For example, when the perturbed probabilities are within 10% of the ground truth, TeaVaR's throughput is within 3% of the solution obtained with the actual probabilities.

4.7 Sensitivity to scenario pruning

In §3.2 I described an efficient algorithm for pruning scenarios. I now elaborate on this algorithm and evaluate its impact on performance and accuracy. The first step is to see what percentage of the scenario space was being pruned by various cutoff thresholds. The cutoff threshold is defined in §3.3. Figure 4-5(a) shows the probability mass of all the scenarios remaining in the optimization after a given cutoff. Modest cutoffs leave a large portion, over 95% of the total space. Figure 4-5(b) shows the effect of pruning on accuracy. Specifically, I compare the case of the CVaR value with pruning to the case where we consider 100% of the scenario space ("optimal") using a standard error formula, $\frac{|CVaR_{\beta,cutoff} - CVaR_{\beta,optimal}|}{CVaR_{\beta,optimal}}$. With a cutoff similar to that used in the bulk of my experiments (10^{-4} for ATT and 10^{-5} for all other topologies), we achieve high scenario coverage with less than 5% error. It is also important to note that the speed benefits of the cutoff are substantial, as shown in Figure 4-5(c). Using cutoff values reduces the computation time from minutes (in the near-optimal

case) to a few tens of seconds, a plausible compute overhead for typical TE periods of 5-15 minutes. Note that all time complexity benchmarks were performed on a fairly standard processor (4-core, 2.60 GHz processor with 32 GB RAM).

Chapter 5

Simulating and visualizing results

Optical backbones are million dollar assets, with fiber comprising their most expensive component. Companies like Google, Microsoft and Facebook purchase or lease fiber to support wide-area connectivity between distant data center locations, and they need to know how and when to purchase this fiber. The process of deciding which fibers to buy and how to manage them is capacity planning. While TeaVaR is cast as a TE scheme throughout this thesis, it can also be used for capacity planning.

In this chapter, I build out a simulation tool, called TE-VIS. The tool communicates with my implementation of TeaVaR, and simulates its usage in the data plane. I explain how TE-VIS can not only be useful for debugging any TE scheme but also for capacity planning. The simulation allows network operators to see how their network is performing, where the bottlenecks are, and what kind of demand loads it can handle. It is a user interface that allows a user to select various inputs to a TE scheme and take the output of that scheme, in this case TeaVaR, and use it to simulate traffic through any chosen network and any given failure scenario. TE-VIS can be used as a first step to implementing TeaVaR in a Software-Defined Network (SDN) or long-term capacity planning.

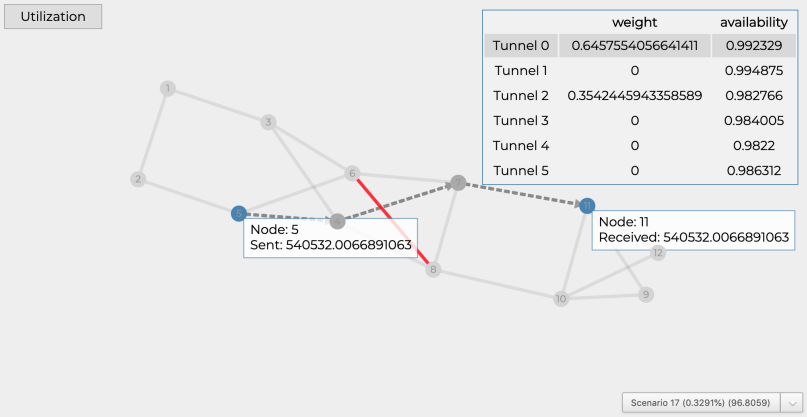


Figure 5-1: An image showing TE-VIS when two nodes are selected. The tunnels between the two nodes along with their weights are displayed on the top right, and the dashed line is the current selected path.

5.1 Developing an API for the optimization

For TE-VIS to be up and running, it must be able to communicate with the optimizer of TeaVaR. To do this, I turn TeaVaR into an API with a single endpoint that takes as input, a a network topology, demand matrix, path selection algorithm, β value, and scenario cutoff. I use the Julia HTTP package to accept HTTP requests with proper headers, and the JSON package to decode these inputs in the JSON data payload. The optimizer then runs, encodes its results, and send back a JSON response.

5.2 Visualizing the output

To visualize the output, I need an interactive, fast, flexible data visualization library. I build out the front-end using React [1] and use D3 [9] to display the network as a 2D force-graph. Using a D3 force graph, I make the network interactive, and add labels, click, and drag events accordingly. Most importantly, I make the force-graph reactive to changes in the state of the overall application, so the user can manually change the failure scenario, toggle link utilization labels, and select various nodes to view more information about them. In addition, the front-end has a form with inputs for the the optimization and makes HTTP requests to get the TE output. The

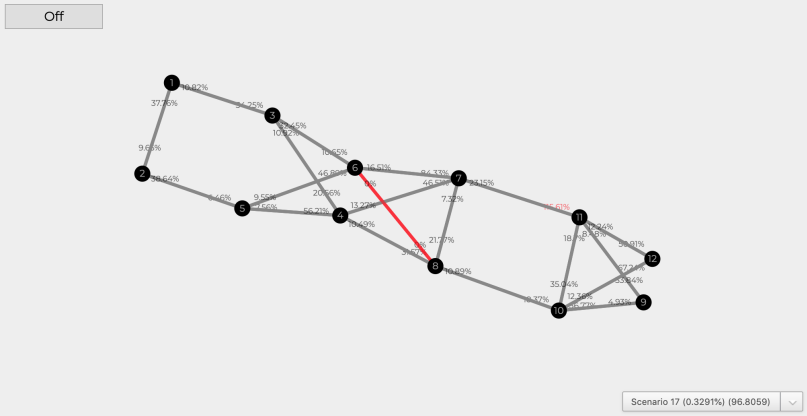


Figure 5-2: This image shows the utilization on each link. Utilization is defined as the amount of traffic divided by that links total capacity. The labels in red show where congestion would occur and traffic would be dropped because of the link breaking capacity.

output is then mapped onto the graph, and traffic is routed and re-routed according to the application state using the routing mechanisms described earlier. This allows a real-time look at utilization and node routing information. When a single node is selected, incoming and outgoing traffic are displayed. When two nodes are selected, the demand between them is displayed, and the paths and their weights can be cycled through using the arrow keys. All paths from source to destination are highlighted, and the current selected path is marked by a dashed line.

5.3 Applications to capacity planning

While TE-VIS was built to demonstrate and explain the results of TeaVaR, I think it has promise as a general tool for capacity planning. From what I have seen, it is often hard to compare various TE algorithms, and TE-VIS may give a realistic visualization of how traffic would actually be routed and suggest the effect of failures for any TE scheme. Figure 5-2 shows an image from TE-VIS with the amount of traffic flowing on each link. We can see that one link in red is down because of a failure. We can also see one link with its utilization number in red because it is greater than 100%. This link would become congested and cause traffic loss in this scenario. In this case, network operators could choose to augment this link's

capacity to avoid congestion there. With TE-VIS, it can easily be determined where the bottleneck links are and how to avoid them with proper capacity planning. In addition, users can manually increase the capacity on given links and see the response in the network. They can also simulate high traffic hours or large data transfers and view the immediate affects all across the network.

Chapter 6

Contributions

The main contributions of this thesis is its introduction of a novel TE paradigm, TeaVaR, to explicitly account for the likelihood of different failure events with the goal of minimizing a formal notion of risk to a level deemed acceptable by network operators. I design and evaluate this scheme with an optimization framework that allows operators to optimize bandwidth assignment subject to meeting a desired availability bar (e.g., providing 99.9% availability). I address algorithmic challenges related to the tractability of risk minimization in this context, as well as operational challenges. I explore various objective goals of network operators, and apply TeaVaR to real-world data from the inter-datacenter backbone of a large service provider in North America. I find TeaVaR can support up to twice as much traffic as today's state-of-the-art TE schemes at the same level of availability. TeaVaR illustrates the usefulness of adopting financial risk theory's notion of Conditional Value at Risk to network resource allocation challenges. The approach may find other important applications in the networking domain.

Another important contribution is the visualization tool that was originally built for TeaVaR, as it can be extended to any other TE or capacity planning scheme. This tool is extremely useful for visualizing complex network simulations and various network events. TeaVaR takes an important step in the development of failure-aware TE schemes by modeling the reality of failures directly in the algorithm, and it provides the real world bandwidth guarantees that network operators seek to achieve.

Appendix A

Appendix

A.1 Calculating VaR_β

Here, I describe a post-processing procedure to calculate the VaR_β. Note that the procedure is generic, and applicable to any setting with a discrete number of states (recall that in the present case, each network state corresponds to a different combination of links, switches, etc. that are up or are down). The procedure is the following: fixing x, we sort the states in increasing order of their loss. With some abuse of notations, we enumerate the states according to the sorted order; let the corresponding state losses be $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_K$. Let K_β be the unique index such that $\Sigma_{k=1}^{K_\beta} P_k \geq \beta > \Sigma_{k=1}^{K_\beta - 1} P_k$, where P_k is the probability of state k. Then the VaR_β is given by $V_\beta(x) = \ell_{K_\beta}$. See Proposition 8 in [36] for a proof.

Computing VaR_β in the rare case when $V_\beta(x^*) < \alpha^*$ (see Theorem 1). We get this inequality when $\Sigma_{k=1}^{K_\beta} P_k = \beta$. This is unlikely to occur because the probabilities correspond to empirical measures of up and down time (hence, the numbers often have several decimal positions more than β). But even if this case occurs, we have $V_\beta(x) = \ell_{K_\beta}$ as described above.

Determining the VaR_β for individual user SLAs. Unlike the current formulation, the per-user SLA (allowed demand and the corresponding probabilistic guarantee) is not explicitly obtained as output of the optimization framework. In such cases, we apply the above procedure on each individual user as follows. The key observation

is that the tunnel allocation per user is given from the optimization. We also know the demand d_f. For each user f, we first reduce the dimension of the state-space to include events that correspond only to links, switches, etc., which belong to one or more of its routs. We sort the states and obtain the VaR_β as described above; by using this procedure, the network provider can actually give different probabilistic guarantees for different users, e.g., by fixing a different β for each user.

A.2 Proof of theorem 2

Since the theorem's guarantee is with probability greater than or equal to β, by definition of VaR_β, we may restrict our attention to failure states whose loss is less than or equal to $V_\beta(a^*)$. Consider any such state; let $\bar{T_f} \subseteq T_f$ be the subset of f's tunnels that are available in that state. Since $V_\beta(a^*)$ is an upper bound for each loss scenario up to the β percentile, it follows from the definition of the loss function (3.2) that

$$\Sigma_{t\in\bar{T_f}} a^*_{f,t} \geq \big(1 - V_\beta(a^*)\big)d_f.$$

Using this inequality together with the proportional assignment rule, we obtain that each active tunnel $r \in \bar{R}_i$ carries flow f_r, satisfying

$$f_r = \frac{w_{f,t}}{\Sigma_{t'\in\bar{T_f}} w'_{f,t}} b_f \tag{A.1}$$

$$= \frac{a^*_{f,t}}{\Sigma_{t'\in\bar{T_f}} a^*_{f,t'}} \big(1 - V_\beta(a^*)\big)d_f \tag{A.2}$$

$$\leq \frac{a^*_{f,t}}{\big(1 - V_\beta(a^*)d_f\big)} \big(1 - V_\beta(a^*)\big)d_f = a^*_{f,t}. \tag{A.3}$$

The theorem then follows immediately by recalling that each feasible solution satisfies the capacity constraint (3.1).

Bibliography

[1] Todd Abel. *ReactJS: Become a Professional in Web App Development*. CreateSpace Independent Publishing Platform, USA, 2016.

[2] Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. A roadmap for traffic engineering in sdn-openflow networks. *Comput. Netw.*, 71:1–30, October 2014.

[3] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. Conga: Distributed congestion-aware load balancing for datacenters. *SIGCOMM Comput. Commun. Rev.*, 44(4):503–514, August 2014.

[4] Fredrik Andersson, Helmut Mausser, Dan Rosen, and Stanislav Uryasev. Credit risk optimization with conditional value-at-risk criterion. *Mathematical Programming*, 89(2):273–291, 2001.

[5] Ajay Kumar Bangla, Alireza Ghaffarkhah, Ben Preskill, Bikash Koley, Christoph Albrecht, Emilie Danna, Joe Jiang, and Xiaoxue Zhao. Capacity planning for the google backbone network. In *ISMP 2015 (International Symposium on Mathematical Programming)*, 2015.

[6] Cynthia Barnhart, Niranjan Krishnan, and Pamela H. Vance. Multicommodity flow problems. In *Encyclopedia of Optimization*, pages 2354–2362. Springer, 2009.

[7] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. MicroTE: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 8. ACM, 2011.

[8] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[9] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, December 2011.

[10] Antonio J Conejo, Miguel Carrión, Juan M Morales, et al. *Decision making under uncertainty in electricity markets*, volume 1. Springer, 2010.

[11] E. Danna, S. Mandal, and A. Singh. A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering. In *2012 Proceedings IEEE INFOCOM*, pages 846–854, March 2012.

[12] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine*, 40(10):118–124, Oct 2002.

[13] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In *INFOCOM 2000. Nineteenth annual joint conference of the IEEE computer and communications societies. Proceedings. IEEE*, volume 2, pages 519–528. IEEE, 2000.

[14] Monia Ghobadi and Ratul Mahajan. Optical layer failures in a large backbone. In *Proceedings of the 2016 ACM on Internet Measurement Conference*, pages 461–467. ACM, 2016.

[15] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 58–72, New York, NY, USA, 2016. ACM.

[16] Zonghao Gu, Edward Rothberg, and Robert Bixby. Gurobi Optimizer Reference Manual, Version 5.0. *Gurobi Optimization Inc., Houston, USA*, 2012.

[17] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013*, pages 15–26, 2013.

[18] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. *SIGCOMM '18*, pages 74–87, 2018.

[19] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a globally-deployed software defined wan. *SIGCOMM Comput. Commun. Rev.*, 43(4):3–14, August 2013.

[20] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, and Ishai Menache. Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 73–86. ACM, 2016.

[21] Wenjie Jiang, Rui Zhang-Shen, Jennifer Rexford, and Mung Chiang. Cooperative content distribution and traffic engineering in an isp network. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 239–250. ACM, 2009.

[22] P. Jorion. *Value at Risk: The New Benchmark for Managing Financial Risk.* MacGraw-Hill international editions: Finance series. McGraw-Hill, 2001.

[23] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 253–264. ACM, 2005.

[24] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. Calendaring for wide area networks. In Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy, editors, *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, pages 515–526. ACM, 2014.

[25] F. A. Kuipers. An overview of algorithms for network survivability. *CN*, 2012:24:24–24:24, January 2012.

[26] Alok Kumar, Sushant Jain, Uday Naik, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Sigcomm '15*, 2015.

[27] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, Renton, WA, 2018. USENIX Association.

[28] Youngseok Lee, Yongho Seok, Yanghee Choi, and Changhoon Kim. A constrained multipath traffic engineering scheme for MPLS networks. In *ICC*, pages 2431–2436. IEEE, 2002.

[29] George Leopold. Building Express Backbone: Facebook's new long-haul network. http://code.facebook.com/posts/1782709872057497/, 2017.

[30] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. Traffic engineering with forward fault correction. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, pages 527–538, 2014.

[31] Houra Mahmoudzadeh. *Robust Optimization Methods for Breast Cancer Radiation Therapy.* PhD thesis, University of Toronto, 11 2015.

[32] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. N. Chuah, Y. Ganjali, and C. Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Transactions on Networking*, 16(4):749–762, Aug 2008.

[33] Jeffrey C Mogul, Rebecca Isaacs, and Brent Welch. Thinking about availability in large service infrastructures. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 12–17. ACM, 2017.

[34] Dritan Nace and Michal Pióro. Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial. *IEEE Communications Surveys and Tutorials*, 10(1-4):5–17, 2008.

[35] R Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[36] R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

[37] Farhad Shahrokhi and David W. Matula. The maximum concurrent flow problem. *J. ACM*, 37:318–334, 1990.

[38] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. Network architecture for joint failure recovery and traffic engineering. *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, 2011.

[39] Hong Zhang, Kai Chen, Wei Bai, Dongsu Han, Chen Tian, Hao Wang, Haibing Guan, and Ming Zhang. Guaranteeing deadlines for inter-data center transfers. *IEEE/ACM Transactions on Networking (TON)*, 25(1):579–595, 2017.