Robust Validation of Network Designs under Uncertain Demands and Failures

Yiyang Chang, Sanjay Rao, and Mohit Tawarmalani, *Purdue University*

https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/chang

This paper is included in the Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17).

March 27–29, 2017 • Boston, MA, USA

ISBN 978-1-931971-37-9

Open access to the Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation is sponsored by USENIX.

Robust validation of network designs under uncertain demands and failures

Yiyang Chang, Sanjay Rao and Mohit Tawarmalani
Purdue University

Abstract

A key challenge confronting wide-area network architects is validating that their network designs provide assurable performance in the face of variable traffic demands and failures. Validation is hard because of the exponential, and possibly non-enumerable, set of scenarios that must be considered. Current theoretical tools provide overly conservative bounds on network performance since to remain tractable, they do not adequately model the flexible routing strategies that networks employ in practice to adapt to failures and changing traffic demands. In this paper, we develop an optimization-theoretic framework to derive the worst-case network performance across scenarios of interest by modeling flexible routing adaptation strategies. We present an approach to tackling the resulting intractable problems, which can achieve tighter bounds on network performance than current techniques. While our framework is general, we focus on bounding worst-case link utilizations, and case studies involving topology design, and MPLS tunnels, chosen both for their practical importance and to illustrate key aspects of our framework. Evaluations over real network topologies and traffic data show the promise of the approach.

1 Introduction

In designing wide-area networks for ISPs and cloud service providers, it is critical to ensure predictable performance at acceptable costs. However, achieving this goal is challenging because links fail (both owing to planned maintenance, and unplanned events such as fiber cuts and equipment failures) [38, 50, 23], and network traffic is variable [9] and constantly evolving [23].

Validating that a network can cope with a range of traffic conditions and failure scenarios is challenging because the number of scenarios to consider are typically exponentially many, and may even be non-enumerable. For instance, a common requirement is to verify that a network with N links can service demand for all combinations of f simultaneous link failures [50, 48, 37]. The number of failure scenarios to consider is $\binom{N}{f}$ for each demand. Further, the set of traffic matrices are not even enumerable, so naively considering all traffic matrices and failure scenarios is prohibitive.

There is a huge gap between practice and existing theoretical tools. Oblivious routing [40, 41, 9, 49], and more generally, robust optimization [12, 14] allow bounding worst-case performance across multiple scenarios of interest. However, to ensure tractability of the problem, these techniques make the *conservative* assumption that the network cannot adapt to changes in demands by re-routing traffic [40, 41, 9, 49], or admit limited forms of adaptation [50, 15]. In practice, networks do adapt by re-routing traffic as demands shift or failures occur, and such adaptation can make network operations much more efficient. Further, the advent of Software-Defined Networking (SDN) allows for network-wide optimization, and facilitates the deployment of flexible re-routing strategies [30, 31].

Given the large gap between theory and practice, the process of validating network designs today is ad-hoc, often requiring extensive simulations, which can be highly time consuming as well as fall short of guaranteeing provable bounds on network performance. In this paper, we take a first step towards tackling this by presenting a formal framework to provide performance bounds on a network design across a set of scenarios (demands, failures). The key novelty in our framework is that it can accommodate a richer set of adaptation mechanisms, used in practice today, for re-routing traffic on failures and changes in demands.

When flexible routing strategies are considered, providing robust performance guarantees typically requires solving intractable non-convex (and often non-linear) optimization problems. We address these difficulties by leveraging cutting-edge techniques in the non-linear optimization literature [44]. An attractive aspect of these techniques is their generality, which allows them to be

applied to a wide range of network validation problems. We show that these techniques lead to tighter bounds on the validation problem than existing state-of-the-art approaches in robust optimization, a finding that has applications beyond networking. Further. the bounds are tight in practical settings of interest - e.g., when demands are expressed as a convex combination of known historical demands [49]. Finally, we show how the techniques may be augmented with analysis of individual problem structure to substantially improve the quality of bounds.

For concreteness, we focus on link utilization, a widely accepted traffic engineering metric [9, 49, 50], which impacts application latency and throughput. We apply our framework to two contrasting, yet practical case studies to illustrate key aspects of our framework. The case studies differ in the type of uncertainty (failures and demands), and the type of adaptation. Specifically, we consider (i) multi-commodity flow (MCF) routing [21, 9, 50] which provides the most flexibility and efficiency, and (ii) MPLS-style tunneling [30, 29] which has more limited flexibility in routing.

While we focus on validation, our framework can enable the synthesis of designs with performance guarantees under uncertainty. We demonstrate this by showing how our approach can aid operators in determining the most effective ways to augment link capacities while ensuring acceptable link utilizations under failures.

We evaluate our approach using multiple real topologies [6] and public traffic data [1]. Our framework performs better than oblivious formulations for both case studies, while surprisingly matching optimal in all the experiments for the failure case study. Further, we show our framework aids in (i) identifying bad failure scenarios; (ii) determining how to best augment link capacity to handle failures; and (iii) evaluating design heuristics – e.g., we show the potential for poor performance with common tunnel selection heuristics.

2 Motivation

2.1 Robust validation applications

A network design consists of (i) *invariant parameters,* which cannot be changed (or are costly to change) across failures and/or demands; and (ii) *adaptable parameters,* which may be flexibly chosen for any scenario. Our framework ensures that the choice of invariant parameters is acceptable across a set of demands and/or failures. Below, we present motivating examples.

Topology Design. In designing network topologies, operators must determine what links to lease and how much capacity to provision. While the set of links and their capacities is difficult to change across failures and demands, the network may adapt by re-routing traffic.

MPLS Tunnel Selection. A common traffic engineering practice is to use tunnels (e.g., MPLS [42]) between each ingress and egress switch, to ensure a core network that does not need to run the BGP protocol. In such settings, a light-weight adaptation mechanism is to switch traffic across k pre-selected tunnels between each source destination pair, which only involves changing flow tables in appropriate ingress switches [29, 30]. Changing tunnels is more heavy-weight since the flow tables of internal switches also need to be modified. A good choice of pre-selected tunnels can lower the frequency of changing tunnels in response to fluctuations in demand.

Middlebox placement. Network policy may require that some of the flows traverse a set of middleboxes such as firewalls and intrusion detection systems (IDS) [39, 7]. While the placement of network middleboxes typically occurs over relatively longer time-scales, traffic may be re-routed to handle normal traffic fluctuations or failures.

In these examples, the topology itself, and the set of tunnels and placement of middleboxes as applicable are *invariant* parameters, while the fraction of traffic sent along a given tunnel is an *adaptable* parameter.

Robust validation may be performed at initial design time, as well as in a *continual* fashion as the network evolves, and new projections on demands are available. Robust validation may indicate the network is no longer able to cope with the scenarios of interest, requiring the operator to consider changes to the design (e.g., by provisioning more capacity on links). Further, it can provide information on which scenario causes the network requirements to be violated, and aid in determining design changes to address the violations.

2.2 Robust validation framework

Our framework is closely related to robust optimization. In traditional robust optimization, input parameters belong to an uncertainty set, and the objective is minimized across any parameter choice in the set [12, 14]. Further, recourse actions may be considered that depend on the specific parameter value. In the networking context, a typical recourse action involves rerouting traffic to handle a change in traffic matrix or failure. The robust optimization literature considers limited forms of recourse actions, primarily for tractability reasons, which may lead to more conservative estimates of worst-case performance (§4.4). In contrast, we model richer network adaptation, and tackle the resulting intractable problems. Prior approaches can be seen as special cases of our more general framework discussed below:

Metrics to capture performance of network design. Our framework can validate a variety of network metrics such as link utilizations, and bandwidth assigned to latency sensitive flows. For concreteness, in this paper, we focus on the utilization of the most congested link (which we will refer to as *Maximum Link Utilization (MLU)*, a widely used objective function [9, 49, 50]. Though we do

not discuss this extensively, our framework also applies to other common metrics of link utilizations (e.g., sum of penalties assigned to individual links, where penalties are convex functions of link utilizations [48, 22, 24]). We focus on utilizations given their extensive use in the traffic engineering literature, and since they reflect application performance (e.g., throughput for bandwidth sensitive applications is inversely related to utilizations).

Characterizing uncertainty in network conditions. We seek to validate that a network design performs well across demands and failure scenarios of interest. A typical set of failure scenarios to consider is all simultaneous failures of F or fewer links [50, 48]. The range of demands may be specified in multiple ways. A common model is to specify a set of historical traffic matrices, and require that all demands based on standard prediction models are considered. We formally discuss this model as well as other models in §4.3 and §5.2.

Modeling how networks adapt. Networks may respond to failures, and changes in demand by rerouting traffic in the best possible fashion to keep utilizations low. This can be achieved by determining the optimal routing (MCF) for a given scenario. This design point is becoming increasingly practical with the adoption of SDNs, given that periodic reoptimization for network state is feasible. Other models may allow adaptation, but with constraints. For instance, in the MPLS tunneling example, the network may adapt by changing how traffic is split across pre-selected tunnels between each ingress and egress pair, though the tunnels themselves do not change. This corresponds well to SDN deployments where only edge routers are SDN enabled [17]. Finally, policy constraints (e.g., a requirement that a set of middleboxes be traversed) may constrain how networks may adapt [47, 39, 7].

3 Formalizing robust validation

3.1 General problem structure

Let X denote the uncertainty set (possibly continuous and non-enumerable) of demands, or failures over which a given network design must be validated. The design includes all parameters that must remain invariant with changes in demands and failures (e.g., network topology, selection of tunnels, placement of middleboxes). For any given scenario $x \in X$, the network may adapt by routing traffic appropriately as described in §2.2.

Let y denote the parameters determined by the network when adapting to scenario x. This includes how traffic is routed – e.g., in the tunneling context, y includes parameters that capture how traffic must be split across tunnels – though there may be additional variables determined as we discuss in §3.2. Formally, the network validation problem may be written as:

$$F^* = \max_{x \in X} \min_{y \in Y(x)} F(x,y) \quad (1)$$

The *inner* minimization captures that for any given scenario $x \in X$, the network determines y in a manner that minimizes an objective function $F(x,y)$ from a set of permissible strategies $Y(x)$. For the fully flexible routing model, $Y(x)$ corresponds to strategies permitted by the standard MCF constraints [21], while for routing with middlebox policies, only strategies that ensure the desired set of middleboxes are traversed are permitted. The *outer* maximization robustly captures the worst-case performance across the set of scenarios X, assuming the network adapts in the best possible fashion for each x.

In this paper, we focus on objective functions $F(x,y)$ that minimize the MLU as discussed in §2.2. We refer to (1) as the validation problem, since it can be used to verify that a chosen design meets a desired utilization goal. For instance, when applied to topology design, $F^* > 1$ indicates the network is not sufficiently provisioned to handle all failures and demands of interest.

For any given scenario x, the inner problem is typically easy to solve (a linear program (LP)), since the network must compute y online to adapt to any failure or shift in demand. The validation problem is however challenging since exponentially many (and potentially non-enumerable) scenarios x must be considered.

3.2 Concrete validation problems

We next relate the general formulation (1) to two concrete case studies, chosen both for their practical importance and to illustrate key ideas of the framework.

- The first case study validates topology design against failures, with the most flexible network adaptation.
- The second example validates tunnel selection across variable demands, with network adaptivity constrained to splitting traffic across pre-selected tunnels.

The examples illustrate the generality of our framework in terms of its ability to consider both failures and demands (discrete and continuous uncertainty sets), and different types of adaptivity models (flexible and more constrained). However, our framework applies to a wider range of applications including simultaneously varying demands and failures, other adaptation models such as middlebox constraints, and other ways of combining adaptation models and uncertainty sets (§5).

We use the notation $x = (x^f, x^d)$ where x^f denotes a failure scenario and x^d denotes a particular demand, dropping superscripts when the context is clear. Likewise, we use $y = (r, U)$ where r denotes how traffic is routed, and U denotes utilization metrics computed as a result. Since our focus is on minimizing MLU, the inner problem may be expressed as $\min_{y \in Y(x)} U$, with constraints in $Y(x)$ which express the requirement that the

$$Y(x) = \left\{ (r,U) \;\middle|\; \begin{array}{ll} \gamma_k(x)U \geq \sum_{i\in I} \beta_{ik}(x) r_i & k \in K \\ \sum_{i\in I} \alpha_{ij}(x) r_i \geq \delta_j(x) & j \in J \\ r \geq 0 & \end{array} \right\}$$
$$F(x,r,U) = U$$

Figure 1: General structure of determining routes (r) for scenario x to minimize MLU (U).

$$\begin{array}{lll} (W) \max\limits_{x,v,\lambda} & \sum_{j\in J} \delta_j(x) v_j & \\ \text{s.t.} & \sum_{j\in J} \alpha_{ij}(x) v_j \leq \sum_{k\in K} \beta_{ik}(x) \lambda_k & i \in I \\ & \sum_{k\in K} \gamma_k(x) \lambda_k = 1 & \\ & x \in X, (v_j)_{j\in J} \geq 0, (\lambda_k)_{k\in K} \geq 0 & \end{array}$$

Figure 2: General structure of validation problem derived from Figure 1 as a single-stage formulation.

utilization of every link is at most U. We now discuss how constraints $Y(x)$ are specified for our case studies.

Fully flexible routing under uncertain failures. Let x^f_{ij} be a binary variable which is 1 if link $\langle i,j\rangle \in E$ (the set of links) has failed, and 0 otherwise. Since we do not consider variable demands in this case study, we let d_{it} denote the known demand from source i to destination t. Let r_{ijt} denote the total traffic to t carried on link $\langle i,j\rangle$. Let c_{ij} denote the capacity of link $\langle i,j\rangle$. Then, $Y(x)$ corresponds to the standard MCF constraints [21], and may be expressed as:

$$\begin{array}{l} U c_{ij}(1 - x^f_{ij}) \geq \sum_t r_{ijt} \quad \langle i,j\rangle \in E \\ \sum_j r_{ijt} - \sum_j r_{jit} = \begin{cases} d_{it} & \forall t, i \neq t \\ -\sum_j d_{jt} & \forall t, i = t \end{cases} \\ r_{ijt} \geq 0 \quad \forall i,j,t \end{array} \tag{2}$$

The first constraint ensures that (i) the utilization of link $\langle i,j\rangle$ is at most U for all non-failed links; and (ii) no traffic is carried on a failed link. The second constraint captures flow balance requirements. Specifically, the net outflow from node i to destination t is the total traffic destined to t when $i = t$, and d_{it} otherwise.

Tunnel constraints and uncertain demands. Given a set of pre-selected tunnels, let T_{ijstk} be a binary parameter that denotes whether the link $\langle i,j\rangle$ is on tunnel k for traffic from the source s to destination t. Let x^d_{st} denote the total $s-t$ traffic, and r_{stk} the subset of this traffic on tunnel k. Then, $Y(x)$ may be expressed as:

$$\begin{array}{l} U c_{ij} \geq \sum_{s,t,k} r_{stk} T_{ijstk} \quad \langle i,j\rangle \in E \\ \sum_k r_{stk} = x^d_{st} \quad \forall s,t; \quad r_{stk} \geq 0 \quad \forall s,t,k \end{array} \tag{3}$$

The first constraint ensures that the utilization of every link is bounded by U. The second constraint captures that the sum of the traffic on all tunnels k for each $s-t$ pair must add up to the total demand of that pair.

3.3 Non-linear reformulation

The validation problem in (1) has been represented in a form referred to as a *two-stage formulation* (e.g., [15]). In the two-stage problem, the optimal second-stage variables (y) depend on the first-stage (x). We simplify this problem by re-expressing it as a single-stage problem, where all the variables are determined simultaneously.

In many network validation problems, including our case studies, the inner problem $\min_{y\in Y(x)} F(x,y)$ is an LP in variable $y = (r,U)$ for a fixed scenario x. This is reasonable because online adaptations of y must be computationally efficient. Figure 1 shows the general structure of the LP. Notice that the coefficients depend on scenario x. For example, in the failure validation case study, $\alpha_{ij}(x)$, $\beta_{ik}(x)$, and $\delta_j(x)$ are constants while $\gamma_k(x)$ is a linear function of x. For a specific value of x, the inner problem is an LP.

It is well known that every LP (referred to as a primal form) involving a minimization objective may be converted into an equivalent maximization LP (referred to as a dual form) which achieves the same objective (assuming the dual is feasible) [19]. The validation problem can then be expressed as a single-stage formulation by:

1. Rewriting $\min_{y\in Y(x)} F(x,y)$ as an equivalent maximization problem using LP duality.
2. Adding the constraints $x \in X$ to the dual form to capture the set of demands or failure scenarios of interest.

Figure 2 shows the general structure of the validation problem as a single-stage formulation. Notice that variables r and U in Figure 1 have been replaced by the dual variables λ and v. Moreover, x is now a variable since the problem validates utilization over all uncertain scenarios.

Formulations **(F)** and **(V)** in Figure 3 capture the validation problem for our case studies involving failures (2) and variable demands (3) respectively. At first glance, both formulations appear non-linear – the objective in (V) involves products of x^d and u variables, while the second constraint of (F) involves a product of variables x^f_{ij} and λ_{ij}. In §4.2, we show that (F) can be written as an integer program (IP) when X is the set of scenarios involving the failure of f or fewer links simultaneously. Regardless, both (V) and (F) are hard problems (non-linear non-convex and IP respectively).

4 Making validation tractable

§3.3 has shown that the validation problems, including our case studies, are typically intractable, Given the intractable nature of the problems, we do not solve them to optimality, rather seek ways to obtain upper bounds on the true optimal of (1). Since the purpose of validation is to ensure a design is acceptable, an upper bound that satisfies the design criteria is sufficient.

We aim for a general approach to tackle a wide range

$$(F)\max_{v,\lambda,x} \quad \sum_{t,i\neq t} d_{it}(v_{it}-v_{tt})$$
$$\text{s.t.} \quad v_{it}-v_{jt}\leq \lambda_{ij} \quad \forall t,\langle i,j\rangle\in E$$
$$\sum_{\langle i,j\rangle\in E}\lambda_{ij}c_{ij}(1-x^f_{ij})=1$$
$$x^f\in X;\quad x^f_{ij}\in\{0,1\};\quad \lambda_{ij}\geq 0,\quad \langle i,j\rangle\in E$$

$$(V)\max_{v,\lambda,x} \quad \sum_{s,t} x^d_{st}v_{st}$$
$$\text{s.t.} \quad v_{st}\leq \sum_{\langle i,j\rangle\in E} T_{ijstk}\lambda_{ij} \quad \forall s,t,k$$
$$\sum_{\langle i,j\rangle\in E}\lambda_{ij}c_{ij}=1$$
$$x^d\in X;\quad \lambda_{ij}\geq 0,\quad \langle i,j\rangle\in E$$

Figure 3: Formulations of validation problems for failure case study (F), and tunnel selection case study (V).

of validation problems. In the optimization literature, problems such as (1) are referred to as robust optimization problems and have been tackled mostly for limited adaptations. Instead, we use non-linear programming techniques, and show they achieve better bounds, a finding that has applications beyond networking.

We introduce the approach in §4.1, and how it applies to our case studies involving failures and variable demands in §4.2 and §4.3 respectively. Although our framework is general, analysis of problem structure can substantially improve the quality of bounds, as we will show for the failure case study in §4.2. Finally, in §4.4, we compare our techniques with benchmarks drawn from the network management and robust optimization literature, and show that our techniques can obtain tighter bounds than these approaches.

4.1 Relaxing validation problems

Our approach works by relaxing the validation problems into more tractable LPs, and obtaining an upper bound on the worst-case link utilizations across scenarios. An optimization problem L is a relaxation of a problem N if every feasible solution in N can be mapped to a feasible solution in L, and the mapped solution's objective value in N is no better than that of its mapping in L.

Reformulation-Linearization Technique (RLT) [44] is a general approach to relax non-linear integer problems. The technique reformulates the problem by (i) adding new constraints obtained by taking products of existing constraints; and (ii) linearizing the resulting formulation by replacing monomials with new variables. For our problem (W), RLT can be constructed as long as $\alpha_{ij}(x)$, $\beta_{ik}(x)$ and $\gamma_k(x)$ are polynomial functions.

For example, consider a non-linear optimization problem where the objective is to minimize $xy-x+y$ subject to the constraints: (i) $(x-2)\geq 0$; (ii) $(3-x)\geq 0$; (iii) $(y-3)\geq 0$; and (iv) $(4-y)\geq 0$. Products of pairs of constraints are taken – e.g., the product of constraints (i) and (iii) results in a new derived constraint $(x-2)(y-3)\geq 0$, i.e., $xy-3x-2y+6\geq 0$. The product term xy is replaced by a new variable z. The objective is rewritten as $z-x+y$, and the derived constraint in the previous step expressed as $z-3x-2y+6\geq 0$. The resulting problem is linear, as it no longer has product terms. However, it is a relaxation in the sense that constraints (e.g., $z=xy$) that must be present to accurately capture the original problem are not included in the new problem.

The above represents the first step in a hierarchy of relaxations and the next steps involve multiplying more than two constraints and linearizing as discussed above. Further, the RLT hierarchy can be tightened using convex relaxations of monomials, which yield other well known hierarchies. As long as the set of inequalities in the verification problem define a bounded set, higher levels of this hierarchy of relaxations converge to the optimal value of the non-linear or integer program [27, 44]. Since the generated LPs can be large (more variables and constraints), we restrict attention to the first level of this hierarchy. Further, in practice, it often suffices to consider a subset of products even for the first level, which keeps the complexity of the resulting program manageable.

4.2 Validation across failure scenarios

Here, we discuss the RLT relaxation technique for our failure case study (formulation (F)). For concreteness, we consider all failure scenarios involving the simultaneous failure of f or fewer links. This failure model is used commonly in practice [50]. We discuss how to generalize the failure model later (§5). Incorporating this model results in replacing the constraint $x^f_{ij}\in X$ in (F) with the constraints $\sum_{\langle i,j\rangle\in E} x^f_{ij}\leq f$, and $x^f_{ij}\in\{0,1\}$.

Empirically, a simple RLT relaxation of the formulation does not yield a sufficiently tight upper bound to the validation problem. Instead, we reformulate the validation problem (F), and consequently derive constraints for the RLT relaxation, as described below:

Reformulating the validation problem. We add variables to (2), in a way that gives more flexibility in choosing solutions, but does not change the optimum. Adding variables to a primal results in additional constraints to the dual. Consequently, we derive constraints for (F) and the associated RLT relaxation LP, which are derived from the LP dual of (2), thus improving the bound on utilization. Specifically, we reformulate (2) as follows:

$$\begin{aligned}
&Uc_{ij}(1-x^f_{ij})+a_{ij}\geq \textstyle\sum_t r_{ijt} \quad \langle i,j\rangle\in E\\
&\textstyle\sum_j r_{ijt}-\sum_j r_{jit}=\begin{cases} d'_{it} & \forall t,i\neq t\\ -\sum_j d'_{jt} & \forall t,i=t\end{cases}\\
&r_{ijt},a_{ij}\geq 0 \quad \forall i,j,t\\
&d'_{it}=\begin{cases} d_{ij}+a_{ij} & \langle i,j\rangle\in E\\ d_{ij} & \langle i,t\rangle\notin E\end{cases}
\end{aligned} \tag{4}$$

We augment each link $\langle i,j\rangle$'s capacity with the extra

(variable) slack capacity a_{ij} for which we reserve the capacity along alternate paths in the network. In particular, the first constraint allows up to a_{ij} of the traffic on link $\langle i,j\rangle$ to be bypassed on the associated virtual link without counting it against the utilization of link $\langle i,j\rangle$. To compensate for this, we increase the total traffic that must be routed from i to j by a_{ij}, as indicated by the last constraint. It can be shown that (4) achieves the same optimal as (2). Further, because any feasible solution to (2) is also feasible to (4) (with slack variables a_{ij} being 0), (4) is more flexible in that it admits additional solutions.

Following the procedure outlined in Figures 1 and 2, this reformulated primal yields a reformulated validation problem (F') which consists of (F) with constraints $\lambda_{ij} \leq v_{ij} - v_{jj},\ \forall\langle i,j\rangle \in E$. Then, (F') simplifies to:

$$(G)\ \max \sum_{i,t} d_{it} v_{it}$$

$$v_{it} - v_{jt} \leq v_{ij} \quad \forall t, \langle i,j\rangle \in E \tag{5}$$

$$\sum_{\langle i,j\rangle \in E} v_{ij} c_{ij} (1 - x^f_{ij}) = 1$$

$$\sum_{\langle i,j\rangle \in E} x^f_{ij} = f \tag{6}$$

$$v_{it} \geq 0, v_{tt} = 0 \quad \forall i,t \tag{7}$$

$$x^f_{ij} \in \{0,1\}, \quad \langle i,j\rangle \in E \tag{8}$$

Proposition 1. *Reformulation (G) achieves the same optimal value as the original validation problem (F).*

The proof (see Appendix) shows that an optimal solution of (F') satisfies $v_{tt} = 0,\ \forall t$ and $v_{ij} = \lambda_{ij},\ \forall\langle i,j\rangle \in E$. The proposition then follows since (F) and (F') achieve the same optimal value having been derived respectively from primals (2) and (4) that achieve the same optimal.

Although (G) is non-linear because the product $v_{ij}x^f_{ij}$ is in the second constraint, we note that (G) has a finite objective only if the minimum cardinality edge-cut set of the topology contains more than f links, a condition that can be verified in polynomial time [18]. Moreover, we show (see Appendix) that if f failures cannot disconnect the nodes of the network, v_{ij} is bounded. Then, standard linearization of $v_{ij}x^f_{ij}$ that uses bounds on v_{ij} and $x^f_{ij} \in \{0,1\}$ reduces (G) to a mixed-integer linear program.

Relaxing the validation problem. Since the validation problem (G) is still intractable, we derive its first-level RLT relaxation as follows. First, the binary requirement $x^f_{ij} \in \{0,1\}$ is replaced by bound constraints, $x^f_{ij} \geq 0$ and $(1 - x^f_{ij}) \geq 0$. Next, the product of these bound constraints is taken with (5) and (7) and the product of (6) and (7) is taken. Finally, the nonlinear constraints are relaxed by introducing $vx^f_{iji'j'}$ to denote $v_{ij}x^f_{i'j'}$.

4.3 Validation across traffic demands

We now consider the tunnel selection case study (formulation (V)) and the problem of verifying utilization against uncertain demands. We discuss two models for specifying demands, and discuss the RLT relaxations.

Specifying demands. We consider two models:

• *Predicted demand:* This corresponds to scenarios when demands may be predicted from past history, a commonly used practice today. Consider optimizing the system for a set of known historical traffic matrices $\{d^h\}_{h\in H}$. As observed in [49], many predictors including the exponential moving average estimate the traffic matrix for a given interval as a convex combination of previously seen matrices. It may be desirable to verify the system for the convex hull of $\{d^1, d^2, \ldots d^h\}$, which ensures that all such predictors can be serviced with reasonable utilization. Specifically, this may be modeled by replacing the constraint $x^d \in X$ in (V) by the constraints $x^d = \sum_{h\in H} x_h d^h$, $x_h \geq 0$ and $\sum_{h\in H} x_h = 1$.

• *All demands that can be handled by the topology:* It may be desirable to understand the extent to which a topology must be over-provisioned if a tunneling solution is used compared to using an optimal MCF solution. This may be modeled by replacing the constraint $x^d \in X$ in (V) by the standard MCF constraints with x^d_{st} denoting demand from source s to destination t, and x^g_{ijt} a flow variable denoting traffic to t on link $\langle i,j\rangle$.

Obtaining the RLT relaxation. We obtain the RLT relaxation by taking the product of (i) inequalities involving v and λ variables with constraints of the form $x \geq 0$; (ii) inequalities involving x variables with constraints of the for $\lambda \geq 0$; (iii) inequalities involving v or λ with inequalities involving x; and (iv) equalities involving x variables with v variables.

4.4 Comparisons to alternate approaches

A key novelty of our framework is that it provides theoretical bounds on network performance across failures/demands, while allowing flexible adaptation. We can show that each RLT constraint we introduce in the problem makes the adaptations more flexible in a specific way. In contrast, prior theoretical work has focused on limited forms of adaptivity and we use them as benchmarks for our RLT relaxation approach. We show that our approach provides bounds that are at least as tight as these prior theoretical works, and later show empirically (§6) that the bounds are better in practice.

Oblivious approaches and generalizations. Oblivious routing [11, 28, 16, 9, 49] bounds utilizations across all links for a set of demands, while limiting how the network adapts to any given demand. While oblivious routing has mainly been considered in the context of MCF [9, 49], the oblivious approach applies to other networking contexts. For instance, in our tunneling case study, an *Oblivious Tunneling* formulation constrains y_{stk} (traffic on tunnel k from s to t) to be of the form $y_{stk} = \alpha_{stk} x^d_{st}$, where α_{stk} is invariant across demands.

The robust optimization literature has considered a more general form of adaptation than an oblivious approach, which can enable tighter bounds on worst-case link utilization [15]. Here, every variable y_i (e.g., each y_{stk} variable in our tunneling example) that a network determines for a given scenario x, is constrained to have the form $y_i = \alpha_{i0} + \sum_j \alpha_{ij} x_j$ where all α_{ij} coefficients must be invariant with x. Note that x_j variables capture scenario x (e.g., in our tunneling example, x is a traffic matrix, and each x_j is a cell in the matrix). In optimization terminology, y_i is an affine function of x. Note that an oblivious approach is a special case of affine policies where many of the α coefficients are zero.

We say the *linearity requirement* has been met when constraints $Y(x)$, and objective $F(x,y)$ are linear in (x,y). For example, in the tunneling case study, the constraints (3) and the objective, U, are linear in U, r, and x^d. Further, the conditions are satisfied by the original oblivious routing [9, 49], and while we do not elaborate, by other case studies such as routing with middleboxes. When network adaptation is restricted to affine policies, and the linearity requirement is met, an optimal set of α_{ij} coefficients may be computed efficiently using LP to minimize worst-case link utilizations [13]. We now state our result:

Proposition 2. *When the linearity requirement is met, an optimal affine policy can be efficiently computed. Under these circumstances, the first-level RLT relaxation for a validation problem is at least as tight as the bound from the optimal affine policy.*

The proof involves taking duals of the RLT relaxation. We do not elaborate on the technical details, and focus on the implications for validation. Further, for predicted demand (§4.3), Proposition 2 already implies that the first-level RLT can provide as tight a bound as an oblivious approach. However, we have shown a stronger result:

Proposition 3. *For the predicted demand case, the first-level RLT relaxation is an exact solution, while the oblivious solution may not always be exact.*

Some of our case studies do not satisfy the linearity requirement. In particular, the requirement is not satisfied for our case study involving failures (2) because the first constraint in (2) involves a non-linear term (product of U and x). Under these circumstances, an optimal affine policy may not be efficiently computable, and is thus not a viable benchmark. However, our framework is still applicable (as our failure case study has shown), since it only requires that the weaker condition that $Y(x)$ is linear in y variables for fixed x needs to be satisfied.

Benchmark for failure case study. R3 [50] tackles the validation problem under failures, but with the more limited goal of determining whether a network can handle all failures scenarios without congestion (i.e., whether $MLU \leq 1$), and with restrictions on how the network can adapt. R3 replaces failures with virtual demands (the traffic to be rerouted on failures) and computes an oblivious protection routing (MCF) for the virtual demand associated with each link. The formulation is only valid when $MLU \leq 1$, since the virtual demand on each link is assumed to not exceed the link capacity. In contrast, our formulation (G), and the associated first-level RLT relaxation is valid for any MLU, which can aid in tasks such as determining which failure scenarios are bad when the network is not sufficiently provisioned, and how best to augment link capacities to handle failures (§6.3). When $MLU \leq 1$, the bounds from R3 are conservative for our validation problem owing to the restriction on adaptations and since the impact of the failures is over-estimated. We have been able to show:

Proposition 4. *The first-level RLT relaxation of (G) provides at least as tight a bound as R3, whenever R3 provides a valid bound.*

In fact, we can impose similar restrictions as R3 on how traffic is rerouted in response to failures by appropriately choosing a subset of RLT constraints. Yet, the MLU will reduce because we optimally chose slack a_{ij} instead of assuming it is c_{ij}. The proof of Proposition 4 considers a special affine policy for $y = (r, U, a)$ in (4), where U does not adapt with x and $a_{ij} = \alpha_{ij} x_{ij}$. We show that all such policies that yield $U \leq 1$ can be made feasible to R3, and, therefore, the bound for R3 is no better than the one obtained with this policy restriction. Since RLT encompasses search over these policies, the result follows. We will show in §6 that RLT yields tighter bounds than R3 whenever the network utilization is less than 1.

5 Aiding synthesis and generalizations

§4 has shown how our framework applies to two validation case studies. We next discuss applications to robust design (§5.1), and to other validation problems (§5.2).

5.1 Augmenting capacities to bound utilization

To see how our validation framework can help in robust design, consider the problem of incrementally adding capacity to existing links to ensure all failure scenarios of interest can be handled (with $U \leq 1$), while minimizing the costs of augmented capacity. We can extend (1) to model the capacity augmentation problem as follows:

$$\min_{\delta \geq 0} \max_{x^f \in X} \min \left\{ \sum_{\langle i,j \rangle \in E} w_{ij} \delta_{ij} \;\middle|\; \begin{array}{l} (c_{ij} + \delta_{ij})(1 - x^f_{ij}) \geq \sum_t r_{ijt} \\ r \text{ is a routing for } d \end{array} \right\}$$

where X is the set of failure scenarios, and δ_{ij} and w_{ij} are respectively the incremental capacity added to link $\langle i,j \rangle$, and the cost per unit capacity. Further, r is a routing for d if r satisfies the flow balance constraints of an MCF formulation. Then, dualizing the inner minimization problem results in a two-stage formulation whose

inner problem is an IP since X is a discrete set. However, using the RLT relaxation technique presented in our framework, we replace the inner problem by an upper-bounding LP which can be dualized to upper-bound the cost of augmentation. This yields an LP based approach to conservatively augmenting capacity.

The above discussion also motivates an iterative approach to design. At each iteration, we solve a capacity augmentation problem considering failure scenarios identified in earlier rounds. Then, with the new capacities, we solve the failure validation problem to identify additional failure scenarios and iterate. At any stage, this provides a lower bound on the optimal capacity augmentation. Although the iterative procedure works well empirically for capacity augmentation, in other robust design problems, finding the worst uncertainty may be hard and the procedure may require too many iterations. In contrast, the LP based approach presented above always yields a conservative robust design quickly.

5.2 More general validation problems

In this section, we discuss how our framework can tackle other validation problems beyond our case studies.

Routing with middlebox constraints. Our framework may be used to obtain bounds on MLU when routing is constrained to satisfy middlebox policies [39, 47, 7]. The requirement that traffic from s to t be routed across a series of middleboxes can be modeled by associating each flow with a state variable which indicates a given middlebox has been traversed. The state is modified by each middlebox on the path. (2) is reformulated by introducing variables $r_{ijst\phi}$ which denote the flow on link $\langle i,j\rangle$ from s to t and for packets with state ϕ, and appropriately modifying the flow balance equations, and capacity constraints. The validation problem may now be formulated and solved across failures, or demands using the same approach as our two case studies.

Simultaneously varying failures and demands. We may desire to ensure utilizations are acceptable across any combination of failures and demands. This can be achieved by directly taking (F), and replacing demand variables d_{st} with variables x^d_{st}, and adding constraints for both x^d and x^f using previously studied models. A similar RLT relaxation applies in this case as well.

Handling shared risk link groups (SRLGs). We have considered a model where at most f links fail simultaneously. In practice, multiple links may fail together (e.g., a fiber cut may impact all links in the affected fiber bundle) [48]. The set of link groups G is considered, and each group g is associated with a set of links that may fail together. We introduce variables x^f_g which indicates whether a particular link group has failed. The validation problem is modeled by considering formulation (F), and replacing the constraints $x^f \in X$ with the constraints $x^f_{ij} = 1 - \prod_{\langle i,j\rangle \in g, g\in G}(1-x^f_g)$, where all x^f_{ij} and x^f_g variables are binary, and $\sum_{g\in G} x^f_g \leq f$. This captures that link $\langle i,j\rangle$ has failed iff any group that it belongs to has failed, and at most f link groups may fail simultaneously. To eliminate the product terms, the first constraint can be linearized with the constraints $x^f_{ij} \geq x^f_g, \langle i,j\rangle \in g, g \in G$, and the constraint $x^f_{ij} \leq \sum_{\langle i,j\rangle \in g, g\in G} x^f_g$. An RLT relaxation may now be applied as normal. Alternately, other linearized constraints can be derived from exploiting this relationship within the RLT scheme that we do not detail.

Network	Nodes	Edges	Date	Link Capacity
Abilene	11	28	2004	homogeneous
ANS	18	50	2011	homogeneous
GEANT	41	118	2014	heterogeneous

Table 1: Topologies

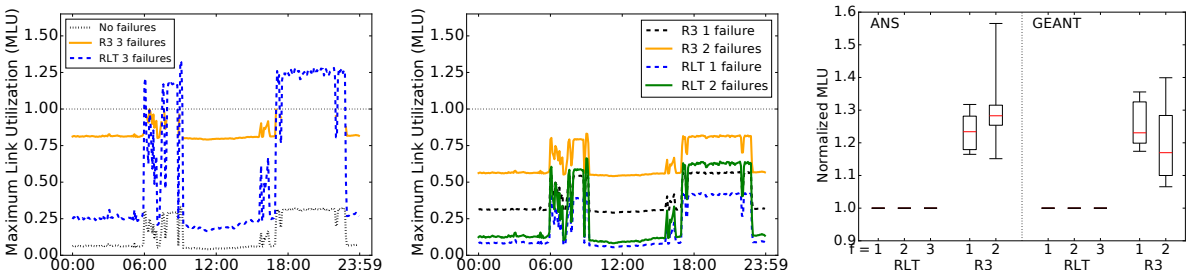
6 Evaluation

We evaluate the effectiveness of our framework in validating topology design under failures (§6.1), and tunnel selection under variable demands (§6.4). We compare our performance bounds with those obtained using existing approaches. Further, we show we can (i) identify bad failure scenarios (§6.2), (ii) optimally augment network capacity to handle failures (§6.3), and (iii) evaluate common design heuristics for tunnel selection (§6.4).

We evaluate our work using real topologies obtained from the Internet TopologyZoo [6]. We focus on three topologies: Abilene, ANS and GEANT [2] (Table 1), where Abilene and ANS have homogeneous link capacities, and GEANT has heterogeneous link capacities. All our LPs and IPs were run using CPLEX [3] (version 12.5.1.0). Our primary performance metric is MLU (§2.2) though we also consider how MLU impacts latency through emulation on an SDN testbed (§6.2).

6.1 Validation across failure scenarios

We evaluate the efficacy of our approach for determining MLU across failure scenarios, comparing MLU bounds produced by our RLT-based LP (§4.2) with (i) the IP (G) which can determine the optimal MLU value (§4.2); and (ii) R3 [50] (§4.4), the best known current approach. (G) is an intractable problem used only for comparison, and the running time of both our RLT relaxation and (G) is shown at the end of this section. We report the MLU returned by the R3 formulation instead of just the binary decision of whether $MLU \leq 1$ used in the original work. Recall R3 only provides valid bounds on MLU when $MLU \leq 1$ (§4.4). We study failure scenarios involving f arbitrary link failures, f ranging from 1 to 3, which practitioners indicated were important to consider. To ensure connectivity after multiple failures, we eliminated one-degree nodes from ANS and GEANT topologies, and modeled each edge as consisting of 2 sub-links



(a) Abilene: baseline and 3 failures (b) Abilene: 1 and 2 failures (c) ANS and GEANT: normalized MLU

Figure 4: Validation across failure scenarios, comparing RLT and R3, for various topologies.

# of Failures	RLT (sec)	IP (sec)	% IP Completed
1	640.58	60.50	100
2	622.60	394.97	100
3	607.68	3890.16	60
4	598.31	–	0
5	586.79	–	0

Table 2: Average running time of the RLT scheme and the optimal IP for GEANT. For the optimal IP, the average running time is computed with instances that completed in 2 hours.

of equal capacity for all topologies. The resulting ANS (GEANT) network has 17 (32) nodes and 96 (200) edges.

We begin by presenting results with the Abilene topology using real traffic data [1]. Figure 4a shows the MLU for $f = 3$, for the RLT and R3 schemes for all traffic matrices measured on April 15th, 2004, a day which experienced a wide variety of traffic patterns. The MLU under normal conditions (no failures) is shown as a baseline. The RLT scheme matches the optimal IP scheme for all traffic matrices, and hence we do not present the IP scheme. The graph shows that several traffic matrices stress the network to achieve $MLU > 1$, indicating it is not provisioned to handle all three simultaneous link failures. Further, the RLT scheme achieves a tighter bound than R3 for all cases where $MLU \leq 1$, and unlike R3, it can provide valid bounds even when $MLU \geq 1$.

Figure 4b presents results for Abilene, but for $f = 1$ and 2. Again, the optimal IP is not shown, since RLT matches optimal. The graph shows the MLU is under 1 for all matrices, indicating the network can handle all possible 2 link failures. Moreover, RLT achieves a tighter bound on MLU than R3 for all matrices. We repeat the experiments with ANS and GEANT topologies. Since actual traffic matrices were not available to us, we generated multiple traffic matrices for each topology using the gravity model [52]. The traffic matrices were chosen so as to keep the link utilizations between 0.3 and 0.45 under normal conditions. Figures 4c presents the normalized MLU for R3 and RLT, relative to the optimal IP for each f. Boxplots depict variation across the matrices. The graph shows that for all f and all traffic matrices, RLT always achieves a normalized MLU of 1, indicating it always matches optimal. The normalized MLU with R3 is higher, e.g., ranging from 1.15 to 1.57 for ANS $f = 2$. Note that results for R3 are not shown for $f = 3$ because all traffic matrices with GEANT, and all but 2 matrices with ANS achieved an optimal MLU above 1, indicating the network was not sufficiently provisioned for them. In contrast, the optimal MLU was under 1 for $f = 1$ and 2, for both topologies, and all traffic matrices.

A surprising aspect of our results is that across all topologies and traffic matrices, the RLT scheme matches the optimal IP. We have also investigated this further for other synthetic topologies and other settings, and have found RLT to match optimal across all the examples. We leave to future work further investigation of whether the first-level RLT in fact can be proven to match the optimal for this case study, or if counter-examples exist.

Running time. We report the running time (Table 2) from experiments with GEANT, the largest topology in our set, on a machine with 8-core 3.00 GHz Intel Xeon CPU and 94 GB memory. To create an even larger topology, we modeled each edge as consisting of 10 sub-links of equal capacity. The resulting network has 32 nodes and 1000 edges. Table 2 shows the average running time of RLT and the optimal IP using 10 traffic matrices generated by the gravity model. Since many IP instances didn't finish even after several hours, we set a 2-hour limit to the solver. Results show that the running time stays stable for RLT, but explodes for the optimal IP, as the number of failures increases. At $f = 3$, 40% of the IP instances did not converge. At $f = 4$ and 5, none of the IP instances converged, and the gaps[1] are larger than 0.5 in all the cases, indicating that the IP solutions found by the solver within 2 hours are still far from the optimal.

6.2 Impact of failures on application performance

Our validation framework can be used to identify failure scenarios that result in high MLU, which could then be emulated on a network testbed to study application performance metrics such as latency under such scenarios.

[1] gap = (UB - LB) / UB, where UB and LB denote the upper bound and the lower bound of the optimal objective value.

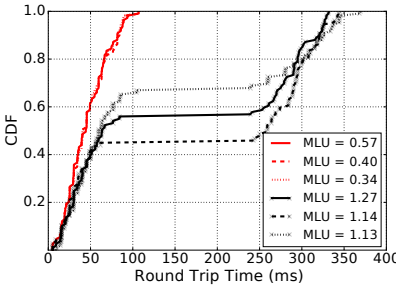


Figure 5: Latency CDF of different failure scenarios.

Finding bad failure scenarios. In general, it is hard to find failure scenarios for the original validation problem (G) that result in a high MLU since it is an IP. A random search is inefficient – e.g., for a certain Abilene traffic matrix, a brute-force search revealed only 0.05% of 3-failure scenarios achieved $MLU > 1$, while 0.08% cases achieved $MLU > 0.8$.

We use a branch and bound algorithm leveraging our RLT LP relaxation. At each exploration step, the failure status of a subset of links is fixed at each node (in the initial step, none of the links are fixed), and the relaxation LP is run to determine a (possibly fractional) solution that results in the highest MLU for the LP. The link with the highest fractional failure (say $\langle i, j \rangle$) is considered, and the LP is rerun fixing x_{ij}^f as each of 0 and 1. Branches where the $MLU < 1$ are pruned. Of the remaining candidate unexplored nodes, the node with the highest MLU is visited. Ties are broken by picking the node at the lowest level in the search tree. The process is run until an integral solution is found, and the search procedure could be continued to determine multiple integral solutions. If the LP relaxation is tight, the search procedure solves at most as many LPs as the number of edges in the topology to find a failure scenario that results in the highest MLU, and our empirical experiments show it takes much fewer steps in practice.

Emulation on an SDN testbed. We emulated the Abilene topology on Mininet [4]. Traffic was generated using the Ostinato traffic generator [5], and an actual Abilene traffic matrix snapshot. We used the procedure above together with our validation framework to identify multiple failure scenarios where MLU exceeded 1. Figure 5 presents measured Round Trip Time (RTT). Each curve corresponds to a failure scenario, and shows a CDF of the median RTT across all source-destination pairs for that scenario. The three curves to the right (black) represent failure scenarios identified by our framework with $MLU > 1$. To contrast, we show three other randomly generated 3-link failure scenarios with lower MLU (red, and to the left – note the curves overlap). The results illustrate that RTTs are significantly higher for the high MLU scenarios identified by our framework.

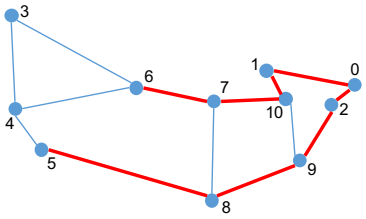


Figure 6: Abilene with links augmented shaded in red.

6.3 Deriving valid capacity augmentations

Our robust validation framework also guides operators in how best to augment link capacities to guarantee $MLU < 1$ across failure scenarios. As discussed in §5, our framework can be applied in an iterative approach that achieves optimal, or may be formulated as a single LP that does not guarantee optimality but solves efficiently.

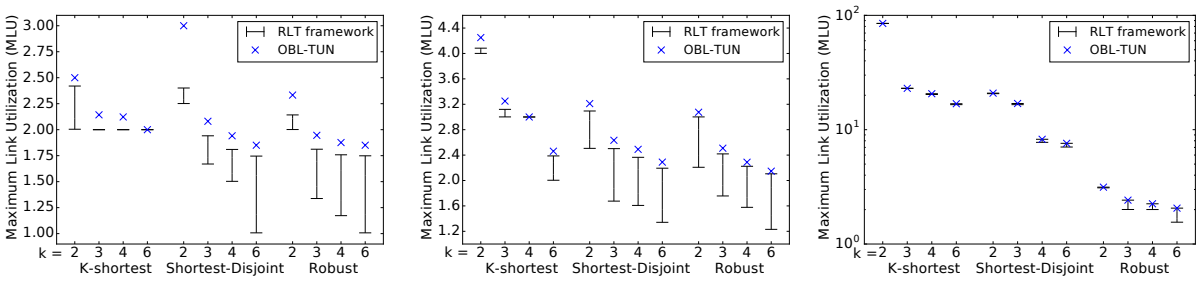
Table 3 illustrates the iterative procedure for an Abilene traffic matrix under three simultaneous link failures. Recall that each iteration consists of (i) a validation step, which either certifies $MLU \leq 1$ for the topology (augmented by capacity increase suggested in prior iteration), or identifies a violating failure scenario; and (ii) an augmentation step, which identifies minimum capacity augmentation needed to handle all failure scenarios identified in prior iterations. The procedure terminates when the validation step certifies $MLU \leq 1$. The augmentation step is a small variant of (2) (see Appendix for details). It is an LP and can easily incorporate practical constraints that limit which links can have their capacity augmented.

We have also formulated the problem as an LP with the stricter requirement that the RLT relaxation of the validation problem achieves $MLU \leq 1$ (§5). The LP achieves the same optimal augmentation as the iterative approach above, which is not surprising given that in all instances we have tried the integrality gap has been 1. More generally, the design LP yields an augmentation cost no worse than $\alpha\text{OPT} + (\alpha - 1)\text{BASE}$, where OPT is the optimal augmentation cost, BASE is the cost of the base network and α is the integrality gap of the RLT relaxation.

6.4 Validation across traffic demands

We next consider how our approach can validate that utilizations are acceptable across demands, focusing on the tunneling case study. For each topology, we consider tunnels pre-selected using the following strategies:

Non-robust strategies. These strategies pick tunnels without explicitly considering tolerance to a range of demands. Specifically, we consider: (i) *K-shortest:* Here, the K shortest paths between each source and destination pair are chosen. Prior works [30, 29] have used this approach to generate an initial candidate set of tunnels, and [30] ultimately picks a subset in a demand-sensitive manner; (ii) *Shortest-Disjoint:* Here, the shortest path is selected. Among other paths, one that overlaps the least with prior choices is selected in an iterative fash-



(a) Abilene (b) ANS (c) GEANT

Figure 7: MLU of RLT framework and Oblivious Tunneling (OBL-TUN) for different tunnel designs and topologies.

ion. Combining path lengths and disjointness is a natural approach to tunnel selection [46].

Robust strategies. We also consider a heuristic called *Robust*, which derives tunnels by decomposing the optimal oblivious routing [25] (details in the Appendix). Since oblivious routing derives an MCF that performs well across all demands, tunnels derived from such a flow have the potential to perform well across demands.

For each tunnel selection approach, our goal is to determine MLU with an adaptive strategy, where traffic is split optimally across tunnels for each demand by solving (3). Since the associated validation problem is nonlinear, we obtain bounds on MLU using (i) our RLT-based framework and (ii) an *Oblivious Tunneling* formulation (abbreviated as OBL-TUN) which minimizes MLU across all demands under the constraint that the fraction of traffic on each tunnel cannot vary with demand (§4.4). While both the RLT framework and OBL-TUN provide upper-bounds on the actual MLU, our framework can also be used to derive a lower bound. Specifically, we solve (V) after fixing the demand to the worst-performing demand for the RLT (or oblivious) relaxation. While this already provides a lower bound, we improve the initial lower bound using a local search procedure on (V), which involves alternating minimization on (v, λ) and x^d. These are tractable problems since (V) is linear if either (v, λ) or x^d are fixed.

Results. We evaluate a total of six schemes, combining our three tunnel selection heuristics with the two ways to obtaining bounds on MLU. For the set of demands, we consider all demands that can be routed with given capacities (§4.3). An MLU higher than 1 indicates the amount of over-provisioning required if tunneling were used to support all demands that the topology could handle with MCF routing.

Figures 7a, 7b and 7c present the MLU across all traffic demands for each of three strategies and three topologies, and different number of selected tunnels (K). Each cross shows the upper bound determined by OBL-TUN, while the vertical bar shows the upper and lower bounds obtained with our RLT-based framework. For GEANT, our current RLT implementation had a high memory requirement that can be addressed using standard decomposition techniques [36] in the future – hence we only report upper bounds achieved by OBL-TUN.

Step	Counter Examples	MLU	Total New Capacity
1	$(1,10,H)$, $(2,9,F)$	1.274	2.744 Gbps
2	$(2,9,H)$, $(1,10,F)$	1.274	5.488 Gbps
3	$(9,8,H)$, $(10,7,F)$	1.217	7.653 Gbps
4	$(10,7,H)$, $(9,8,F)$	1.217	9.818 Gbps
5	$(0,2,H)$, $(1,10,F)$	1.192	11.743 Gbps
6	$(1,0,H)$, $(1,10,F)$	1.071	12.452 Gbps
7	$(7,6,H)$, $(8,5,F)$	1.006	12.509 Gbps
8	$(8,5,H)$, $(7,6,F)$	1.006	12.566 Gbps
9	–	1.000	–

Table 3: Iterative optimal capacity augmentation for Abilene (Figure 6). Each row shows MLU and counter example generated by the validation step, and the total capacity that must be added across all links as per the augmentation step to address all prior counter-examples. H (F) indicates one (both) sub-link(s), (each initially 5 Gbps) associated with the edge fails.

Several points can be made. First, our RLT framework often obtains tighter upper bounds than OBL-TUN strengthening Proposition 2. For example, for Abilene with $K = 2$, and Shortest-Disjoint tunnel selection, the upper bounds with OBL-TUN and the RLT framework are 3 and 2.4 respectively. Second, by providing lower bounds as well, our framework can exactly solve the non-linear problem (V) in quite a few cases. For instance for Abilene and the K-shortest heuristic, a single horizontal line is shown for $K = 3$ and higher, indicating that our framework can determine the optimal MLU.

Third, through a combination of lower and upper bounds, our framework can provide valuable insights on tunnel selection heuristics used by system practitioners. For example, for K-shortest $K = 6$, not only does our framework determine exact MLU, but also the MLU is the same as OBL-TUN. This indicates that when tunnels are selected using K-shortest, adapting how traffic is split across tunnels with demand performs no better than a non-adaptive approach. The trend is particularly pronounced for GEANT where our framework indicates the lower-bounds on MLU are higher than 16 even for $K = 6$, and very close to the oblivious solution. While

recent work has suggested picking the K shortest tunnels and then picking a subset in a demand-sensitive manner [30], this result shows the possibility for this heuristic to perform poorly under certain demand patterns. The Shortest-Disjoint heuristic performs much better for Abilene and ANS, but performs poorly for GEANT – for $K = 6$, the lower bound is 7.05, close to the MLU of 7.59 with an oblivious approach.

While the non-robust design strategies perform poorly, *Robust* performs much better. The benefits are particularly stark for GEANT, e.g., for $K = 6$, the MLU ranges between 1.54 and 2.05. We have also experimented with robust tunnels and predicted demands obtained from real traffic matrices that are scaled so as to stress the Abilene network. The RLT framework achieves the optimal MLU (Proposition 3). OBL-TUN however results in MLU that is 6.84 times worse than optimal. Overall, these results show the value of our RLT-based framework.

7 Related work

Like work on network verification (e.g., [34, 33]), robust validation ensures that network designs meet operator intent. While verification efforts have focused on correctness of the network data-plane, and switch configurations, robust validation is an early attempt at verifying quantifiable network properties. Our framework complements topology synthesis tools [43] by allowing specification of robust design requirements, and providing the underlying optimization substrate.

Prior work on traffic engineering has focused on adaptive settings [20, 32] or has derived a robust routing that optimizes for multiple demands assuming that the routing does not change across demands [9, 8, 49, 51]. Robust routing schemes include oblivious schemes which do not use prior traffic data (e.g., [9, 8]), that route based on multiple historical traffic matrices (e.g., [51]), and those that combine these techniques [49]. Oblivious schemes arose from pioneering work in the theoretical computer science community [40, 41]. In contrast, we obtain worst-case utilization bounds for network designs, where topology and tunnels are invariant, but routing may adapt in practical yet richer ways. It has been shown that adaptive tunnels may, in the worst-case, not benefit much relative to oblivious routing [25]. Instead, we show that provable gains are achieved for specific topologies which have also been observed in practice [35].

Several works have looked at traffic engineering in the presence of failures [8, 48, 50, 37], and we have extensively compared our work with [50]. [8, 48] studied partial adaptation to failures as a way to balance flexible adaptation with the cost for adaptation. [37] optimizes bandwidth assignments to flows, guaranteeing that no congestion occurs with failures. While we do not elaborate, this model can be expressed using our framework. Prior work [22] developed ways to choose OSPF weights which are robust to single link failures. In contrast, we allow flexible adaptation, minimize MLU, and aid robust design of networks that cope well with failures.

Many recent works have looked at how traffic must be routed in the presence of middleboxes (e.g., [39, 47, 7]). There is a growing trend for virtualization of middleboxes, which may allow placements to change on the fly [45, 7]. Our framework can accommodate problems that adapt routing to handle uncertain demands/failures while satisfying middlebox constraints both for fixed placements, and when allowing placements to adapt along with routing.

Beyond networking, the complexity status of robust optimization formulations has been investigated and tractable formulations derived for various special cases [12, 14]. Recent literature has considered limited adaptability in robust binary programming applications including supply chain design and emergency route planning [26, 10]. Instead, our work considers more general forms of adaptivity, focuses on the networking domain, and brings relaxation hierarchies from non-convex optimization to bear on robust optimization problems.

8 Conclusions

In this paper, we have made three contributions. First, we have presented a general framework that network architects can use to validate that their designs perform acceptably across a (possibly exponential and non-enumerable) set of failure and traffic scenarios. Second, by explicitly modeling richer ways in which networks may adapt to failures, and traffic patterns, we have obtained tighter bounds on MLU than current theoretical tools, which consider more limited forms of adaptation for tractability reasons. Third, we have demonstrated the practical applicability of our framework. While the first-level RLT can provably solve the validation problem for predicted demand, surprisingly, it also determines optimal MLU for for all our experiments with the failure case study. Empirical results confirm that our techniques consistently out-perform oblivious methods that can be unduly conservative. Finally, our framework can enable operators to understand performance under failures, guide incremental design refinements, and shed new light on commonly accepted design heuristics. Our initial results encourage us to explore larger networks, study the quality of bounds on other validation problems, and consider network design more extensively in the future.

Acknowledgements

We thank our shepherd Nate Foster, and the reviewers for their insightful feedback. This work was supported in part by the National Science Foundation (Award Number 1162333), and by a Google Research Award.

A Appendix

Proof of Proposition 1: Clearly, the optimal value of (G) is no more than that of (F') because (G) has the following additional constraints (i) for all $\langle i,j\rangle \in E$, $\lambda_{ij} = v_{ij}$, and (ii) and for all nodes t, $v_{tt} = 0$. Therefore, we only need to show that the optimal value of (F') is no more than that of (G). Let (λ^*, v^*, x^{f*}) be optimal in (F'). Denote by $\mathrm{SP}_{it}(\lambda)$ the shortest path between i and t with edge-lengths λ. For any path P_{it} connecting nodes i and t, it follows from the first constraint in (F) that $v^*_{it} - v^*_{tt} \leq \sum_{\langle i,t\rangle P_{it}} \lambda^*_{ij}$ and, so, minimizing rhs over paths yields $v^*_{it} - v^*_{tt} \leq \mathrm{SP}_{it}(\lambda^*)$. For any link $\langle i,j\rangle$ this implies that $\lambda^*_{ij} \leq v^*_{ij} - v^*_{jj} \leq \mathrm{SP}_{ij}(\lambda^*) \leq \lambda^*_{ij}$, where the first inequality is from the slack-induced constraint, the second inequality follows from discussion above, and the third inequality because $\langle i,j\rangle$ is a valid path from i to j. Therefore, equality holds throughout. Now, consider the solution (v', x^{f*}) such that $v'_{it} = \mathrm{SP}_{it}(\lambda^*)$. We show that this solution is feasible to (G). Clearly, $v'_{it} - v'_{jt} \leq v'_{ij}$ because the shortest path from j to t can be augmented with $\langle i,j\rangle$ to yield a path from i to t. Next, because $v'_{ij} = \mathrm{SP}_{ij}(\lambda^*) = \lambda^*_{ij}$, where the last equality was shown above, it follows that $\sum_{\langle i,j\rangle} v'_{ij} c_{ij}(1 - x^{f*}_{ij}) = 1$. Moreover, $v'_{it} = \mathrm{SP}_{it}(\lambda^*) \geq 0$ because $\lambda^*_{ij} \geq 0$ and, trivially, $v'_{tt} = \mathrm{SP}_{tt}(\lambda^*) = 0$. Therefore, (v', x^{f*}) is feasible to (G). Finally, $\sum_{i,t} d_{it} v'_{it} = \sum_{i,t} d_{it} \mathrm{SP}_{it}(\lambda^*) \geq \sum_{i,t} d_{it}(v^*_{it} - v^*_{tt})$, where the equality follows from the definition of v' and the inequality by summing products of $\mathrm{SP}_{it}(\lambda^*) \geq (v^*_{it} - v^*_{tt})$ with $d_{it} \geq 0$. Therefore, the optimal value of (G) is at least as large as that of (F').

Proof that (G) can be formulated as an Integer Program after a polynomial time verification of graph connectivity: The objective of (G) is not finite if the minimum edge-cut set contains f or fewer links, a fact that can be verified in polynomial time [18]. Now consider that the topology is not disconnected after any simultaneous set of f link failures. We show that $v_{it} \leq \frac{1}{c_{min}}$, where $c_{min} = \min_{\langle i,j\rangle \in E} c_{ij}$. To prove the bounds, let NF denote the set of links that do not fail when the optimal value of (G) is achieved. For any pair of nodes i and t, there exists a path (whose edges we denote as P) on the failure of this set of links. By adding the first constraint of (G) for all edges along P, $v_{it} = \sum_{\langle i,j\rangle \in P} v_{ij} \leq \sum_{\langle i,j\rangle \in NF} v_{ij}$. From the second constraint of (G), $\sum_{\langle i,j\rangle \in NF} v_{ij} c_{ij} = 1$, and hence $\sum_{\langle i,j\rangle \in NF} v_{ij} \leq 1/c_{min}$. The bounds follow.

Multiplying the bound constraints $0 \leq v_{ij} \leq \frac{1}{c_{\min}}$ with x^f_{ij} and $1 - x^f_{ij}$ allows us to linearize the above mixed-integer non-linear program into an integer program. This is achieved by replacing $v_{ij} x^f_{ij}$ with a new variable $v x^f_{ij}$ and observing that it is automatically constrained to be $v_{ij} x^f_{ij}$ when $x^f_{ij} \in \{0,1\}$.

Capacity augmentation procedure. For a given scenario, the capacity augmentation problem is easy to model and solve as a linear program. Specifically, (2) is modified by setting the utilization bound $U = 1$, and replacing capacity c_{ij} with $c_{ij} + \delta_{ij}$, where δ_{ij} is the incremental capacity that must be added to link $\langle i,j\rangle$. The objective is $\sum_{ij} w_{ij}\delta_{ij}$, where w_{ij} is the cost associated with each unit of capacity added to link $\langle i,j\rangle$. The formulation is easily extended to multiple scenarios, by replicating the set of constraints (2) modified as above, for each scenario. Practical cabling constraints that constrain which links can have their capacity augmented and by how much are easily incorporated by adding bounds to δ_{ij}.

Robust tunnel design heuristic. To generate a set of tunnels by decomposing the optimal oblivious routing, a derived graph is considered which has the same nodes and edges as the original topology, but with each edge having a weight equal to the flow from the oblivious routing. The widest path (the path with the highest bottleneck link capacity) is chosen as a tunnel. The bottleneck capacity of this path is now decremented from all other edges on this path in the derived graph. This procedure is repeated until k tunnels are obtained.

References

[1] Abilene traffic matrices. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM/`.

[2] GEANT network. `http://geant3.archive.geant.net/Network/NetworkTopology/pages/home.aspx`.

[3] IBM ILOG CPLEX optimization studio. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer`.

[4] Mininet. `http://mininet.org/`.

[5] Ostinato network traffic generator and analyzer. `http://ostinato.org/`.

[6] Topology zoo. `http://www.topology-zoo.org/`.

[7] B. Anwer, T. Benson, N. Feamster, and D. Levin. Programming slick network functions. In *Proceedings of ACM SIGCOMM Symposium on Software Defined Networking Research*, pages 14:1–14:13, 2015.

[8] D. Applegate, L. Breslau, and E. Cohen. Coping with network failures: Routing strategies for optimal demand oblivious restoration. In *Proceedings of ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS)*, pages 270–281, 2004.

[9] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 313–324, 2003.

[10] P. Awasthi, V. Goyal, and B. Y. Lu. On the adaptivity gap in two-stage robust linear optimization under uncertain constraints. `http://www.columbia.edu/~vg2277/column-wise.pdf`, 2015.

[11] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke. Optimal oblivious routing in polynomial time. *J. Comput. Syst. Sci.*, 69(3):383–394, 2004.

[12] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, Princeton, NJ, 2009.

[13] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[14] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[15] D. Bertsimas and V. Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical programming*, 134(2):491–531, 2012.

[16] M. Bienkowski, M. Korzeniowski, and H. Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 24–33, 2003.

[17] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian. Fabric: A retrospective on evolving sdn. In *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software-Defined Networks*, pages 85–90, 2012.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press Cambridge, 2001.

[19] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[20] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings of IEEE INFOCOM*, pages 1300–1309, 2001.

[21] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *Proceedings of Symposium on Foundations of Computer Science*, pages 184–193, 1975.

[22] B. Fortz, M. Thorup, et al. Robust optimization of OSPF/IS-IS weights. In *Proceedings of International Network Optimization Conference*, pages 225–230, 2003.

[23] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 58–72, 2016.

[24] A. Gupta, M. T. Hajiaghayi, and H. Räcke. Oblivious network design. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 970–979, 2006.

[25] M. T. Hajiaghayi, R. D. Kleinberg, and T. Leighton. Semi-oblivious routing: Lower bounds. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[26] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[27] D. Handelman. Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific Journal of Mathematics*, 132(1):35–62, 1988.

[28] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 34–43, 2003.

[29] V. Heorhiadi, M. K. Reiter, and V. Sekar. Simplifying software-defined network optimization using SOL. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, pages 223–237, 2016.

[30] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 15–26, 2013.

[31] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4:

Experience with a globally-deployed software defined wan. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, 2013.

[32] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 253–264, 2005.

[33] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: Static checking for networks. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, pages 113–126, 2012.

[34] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, pages 15–27, 2013.

[35] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, and R. Soulé. Kulfi: Robust traffic engineering using semi-oblivious routing. *arXiv:1603.01203 [cs.NI]*, 2016.

[36] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical programming*, 69(1-3):111–147, 1995.

[37] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter. Traffic engineering with forward fault correction. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 527–538, 2014.

[38] R. Potharaju and N. Jain. When the network crumbles: An empirical study of cloud network failures and their impact on services. In *Proceedings of ACM Annual Symposium on Cloud Computing*, pages 15:1–15:17, 2013.

[39] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simple-fying middlebox policy enforcement using sdn. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 27–38, 2013.

[40] H. Räcke. Minimizing congestion in general networks. In *IEEE Symposium on Foundations of Computer Science*, pages 43–52, 2002.

[41] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of Annual ACM Symposium on Theory of Computing*, pages 255–264, 2008.

[42] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *RFC 3031*, 2001. `https://tools.ietf.org/html/rfc3031`.

[43] B. Schlinker, R. N. Mysore, S. Smith, J. C. Mogul, A. Vahdat, M. Yu, E. Katz-Bassett, and M. Rubin. Condor: Better topologies through declarative design. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 449–463, 2015.

[44] H. D. Sherali and W. P. Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*. Springer Science & Business Media, Dordrecht, 1999.

[45] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 13–24, 2012.

[46] D. Sidhu, R. Nair, and S. Abdallah. Finding disjoint paths in networks. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 43–51, 1991.

[47] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, and N. Foster. Merlin: A language for provisioning network resources. In *Proceedings of ACM CoNEXT Conference*, pages 213–226, 2014.

[48] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford. Network architecture for joint failure recovery and traffic engineering. In *Proceedings of ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS)*, pages 97–108, 2011.

[49] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. COPE: Traffic engineering in dynamic networks. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 99–110, 2006.

[50] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang. R3: Resilient routing reconfiguration. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 291–302, 2010.

[51] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Towsley. Optimal routing with multiple traffic matrices tradeoff between average and worst case performance. In *Proceedings of International Conference on Network Protocols*, 2005.

[52] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proceedings of ACM Internet Measurement Conference*, pages 30–30, 2005.