Rethinking Failure-Tolerant Traffic Engineering with Demand Prediction

***Abstract*—Network failure uncertainty and traffic demand uncertainty pose a series of challenges to efficiently solve network traffic engineering (TE) problems in the production wide-area networks (WANs). Prior works especially predict-then-optimize paradigm in TE systems encounter a significant mismatch between the upstream prediction task and the downstream optimization task, leading to potential demand loss risk and further degrading the quality of services in the cloud. In this paper, we present TUFTTE, a risk-driven TE optimization framework, where the objective function and constraints can be flexibly specified by the network operators. We employ a differentiable optimization layer to guide the prediction model with the objective of minimizing the network risks. When the network risk is defined as the demand loss, our approach can provably converge to the global optimum. Experimental results on real-world topology demonstrate that TUFTTE reduces the demand loss by 11.59% on average compared to the state-of-the-art algorithm. We believe that our paper deeply explore the potential of prediction and promote the understanding on how to develop and deploy a general learning module into the commercial TE optimization.**

***Index Terms*—failure tolerance, traffic engineering, machine learning, decision-focused learning**

I. Introduction

Wide-area networks (WANs) that connect globally distributed data centers are essential infrastructure to carry a variety of cloud services. The quality of services (QoS) is inextricably linked to the efficiency and resilience of WAN infrastructure. In order to fully utilize and improve reliability, software-defined networking (SDN) is increasingly adopted to enable centralized management and dynamic traffic routing in WANs [1]–[5]. Numerous modern traffic engineering (TE) systems have emerged thanks to the popularity of SDN [6]–[14].

Two types of uncertainty have a detrimental effect on the performance of TE: network failure uncertainty and traffic demand uncertainty. The unpredictable nature of infrastructure or operational failures is inherent in the networking system [5], [10], [15], [16]. The occurrence of unexpected events can lead to congestion and delays, which significantly impact traffic flow and network efficiency. On the other hand, due to the multiplicity of variables at play, a certain degree of unpredictability remains in customer-facing traffic demands (e.g., web and e-mails) [12].

To address the issue of failure uncertainty, a series of failure-tolerant routing mechanisms have been proposed, which can be classified into two categories: congestion-free TE [7], [10], [17] and risk-driven TE [6], [16], [18]–[20]. The congestion-free TE (FFC [10] as a typical example) is designed to operate without congestion in all failure scenarios. Rate limiters are employed to broker and explicitly specify demands [2], [7], [10], [21]. Alternatively, they utilize the *dominating demand* to make decisions [17], [19]. The concept of the *dominating demand* can be simply understood as the maximum possible demand. However, this scheme has the potential to result in low utilization and over-provisioning almost all of the time [6]. Additionally, it is impractical to limit the sending rate of the customer-facing traffic.

As a consequence, network operators turn to assume risks. The risk-driven TE enables high-priority flows to fully transmit their traffic and drops low-priority flows in the event of congestion [3], [5], [16], [22]. Network risks such as availability, demand loss and latency stretch are defined [6], [22], and the problem becomes how to minimize the risks. However, the uncertainty in traffic demands leads to uncertainty in the risk function. The current solution of risk-driven TE is to predict the traffic demands of all node pairs, namely the demand matrix, and then solve an optimization problem with the predicted demand matrix. If we ignore the inherent traffic uncertainty and base the solution to the problem on prediction, it will not work well, because there is a mismatch between the upstream prediction task and the downstream optimization task [11], [12], [23]. For example, demand loss is a rigid risk metric that represents the greatest traffic loss when a component of the network fails [22]. Our experimental results demonstrate that the demand loss of prediction-based methods is on average 5.77% greater than that of the ideal case (§ II-B). We therefore advocate taking the traffic uncertainty into the risk-driven TE.

Recently, DOTE [12] also observed the impact of unpredictable traffic demands, and employed a deep learning module to directly optimize the TE decisions. This end-to-end design motivates us to rethink the failure-tolerant TE. The objectives of DOTE include convex objectives like maximum-link-utilization (MLU) and quasi-concave objectives like maximum-multicommodity-flow. Unfortunately, it does not support failure-tolerant design by simply adjusting the loss function (§ III-B). As a result, this paper develops a pragmatic framework, TUFTTE, for addressing **T**raffic **U**ncertainty in **F**ailure-**T**olerant **T**raffic **E**ngineering, with the objective of reducing network risk across uncertain scenarios.

The initial step in the development of our idea is the formulation of a linear program targeted at minimizing the demand loss. We find that this risk-driven optimization problem is a generalization of FFC [10]. It provides configurations for both congestion-free TE and risk-driven TE, which we refer to as the *proactive* mode and *reactive* mode, respectively. This indicates

that network risk can assist in identifying a superior solution among the congestion-free routing solutions.

To connect the prediction and network risk, we employ a cutting-edge technique in the field of machine learning, namely decision-focused learning [23]–[29]. It integrates prediction and optimization into an end-to-end system, and focuses on how to guide the predictive method with a downstream task. Our method offers a theoretical guarantee of convergence to the global optimum with regard to the problem of minimizing demand loss. Decision-focused learning has the additional benefit of being flexibly adapted to a large range of optimization problems. This allows us to attempt to combine it with another risk-driven TE, TEAVAR [6].

We evaluate TUFTTE with prediction-based methods and DOTE on real-world topologies, in terms of network risks. A variety of scenarios and traffic matrices are used for simulations. The results show that TUFTTE decreases the demand loss by 11.59% on average in comparison to FFC, offering a benefit as high as 52.7% in certain cases. Moreover, our framework enhances the performance of prediction-based TEAVAR. Our code is available online.

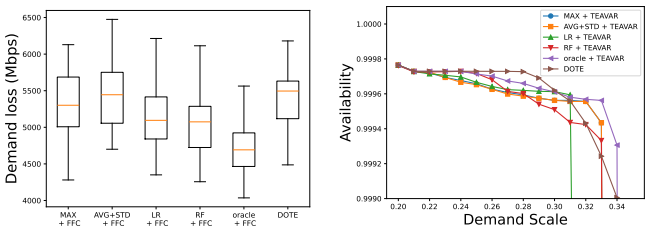
II. Background

A. Network risk

TUFTTE falls under the category of risk-driven TE. The primary goal is to optimize the network risk, which necessitates the definition of risk. While the maximum link utilization (MLU) can be considered as a type of risk, it is not a practical approach because the significance lies in whether the load on the link exceeds its capacity. The operators are more concerned with the amount of traffic lost when an accident occurs. The risk model adopted here is derived from the **R**isk **S**imulation **S**ystem (RSS) developed by Facebook [22]. This model defines three types of network risk as follows:

- Demand loss: maximum amount of unsatisfied demand across all failure scenarios.
- Unavailability: the possibility of unexpected scenarios where traffic loss occurs.
- Latency stretch: path dilation of a flow against the shortest path.

The objective of minimizing demand loss focuses on network risks in the worst case, as demand loss is defined as the maximum traffic loss across all the scenarios. Many previous works including congestion-free TE [7], [10], [30] have sought to improve the worst-case scenario. In contrast, the objective of unavailability focuses on network risks in the average case. It is anticipated that the number of lossless scenarios will be as high as possible when considering availability. The current online applications require high availability for customer satisfaction [3], [22], [31]. Finally, latency stretch is concerned with the quality of network service. Consequently, these risk metrics quantify the quality of different TE policies from a multitude of perspectives. Depending on the type of application, one can choose which risk metric to use (e.g., latency stretch for latency-sensitive applications).



(a) (b)

Fig. 1. Impact of traffic demand uncertainty: (a) Comparison between different predictive models combined with FFC: three horizontal lines of the boxes are 75^{th} percentile, 50^{th} percentile, and 25^{th} percentile values respectively. (b) Comparison between different predictive models combined with TEAVAR: each point is the average availability across a number of demand matrices.

B. Motivation

A resilient routing mechanism ought to be capable of performing well under both demand and failure uncertainty. Therefore, we examine the performance of several failure-tolerant routing algorithms (FFC [10] and TEAVAR [6]) with predicted demand matrices, using the aforementioned risk metrics. We also compare them with a state-of-the-art TE algorithm designed to cope with traffic unpredictability (DOTE [12]).

We evaluate these algorithms on the GEANT topology from the SNDlib [32]. Four simple prediction methods are employed: (i) MAX: the maximum value of historical traffic, as utilized in [18], (ii) AVG+STD: the mean plus two standard deviations, as employed in SWAN [2], (iii) LR: linear regression, a linear prediction, and (iv) RF: random forest regression, a common machine learning method. The ground truth knowledge is designated as *oracle*. The performance of the *oracle* is considered as the ideal performance of an algorithm. More details of experimental settings are provided in § V-A.

The FFC combined with the above prediction methods is tested using multiple demand matrices. We choose the demand loss risk because FFC focuses on worst-case scenarios. The distributions of their demand loss risks are shown in Fig. 1a. The boxes in the figure range from 25^{th} to 75^{th}, and the whiskers go from minimum to maximum value. It can be observed that the demand loss of the four methods is more or less higher than that of the ideal solution, with the random forest method exhibiting the most robust performance. The performance of DOTE is inferior to that of oracle FFC because it is designed to minimize the MLU.

We find similar results corresponding to TEAVAR with availability as the risk metric. For each scenario we check whether the demands for each node pair are fully satisfied under the control of the TE decision. The sum of the probabilities for lossless scenarios reflects the availability shown in Fig. 1b. We compute the average availability of 30 demand matrices, and then scale up the demand matrix to track the relationship between demand scale and the associated availability. Fig. 1b shows that random forest performs worse, and AVG+STD is

the best prediction. Recall that random forest performs the best in the FFC experiment (Fig. 1a). This suggests that it is difficult to determine which prediction method is better suited to a downstream task. Since DOTE is designed to minimize MLU, it provides high availability when the amount of demand is small, but deteriorates notably as the total demand increases. In summary, DOTE is not robust enough, and prediction-based TE cannot perform as well as expected.

With the experimental results, we conclude the weakness of prediction-based TE as follows:

- The traffic demands for each source-destination pair are each predicted, but there is an implicit relationship between them. For instance, service changes will result in a variation in traffic demand between two data centers, yet the total traffic to and from a data center is more stable [33].
- The goal of prediction differs from that of TE. These prediction methods are intended to optimize the prediction error and do not consider how the prediction will be used in the downstream optimization problem [11], [12]. For example, mean square error (MSE) is widely used to measure the bias between the prediction and the ground truth. However, it is irrelevant for the components of networking system such as link capacity and topology. We have to acknowledge the fact that reducing the network risk is the ultimate goal, rather than the accurate prediction. Consequently, prediction-based TE systems make decisions without prior knowledge of the distribution of traffic demand.
- There is no single prediction method that is suitable for all downstream tasks. It is difficult to develop a guideline for selecting a prediction method. The selection process requires extensive experimentation.

All of above taken into consideration, we explore to design a direct optimization mechanism with exceptional robustness.

III. TUFTTE Design

In this section, we initially formulate the problem that TUFTTE seeks to address. Subsequently, we elucidate the methodology for approximating the unknown traffic matrix with the aid of network risks. Finally, we proffer a few recommendations to expedite the training process.

We model the network topology as a graph $G = (V, E)$, where V and E are sets of switches and links between switches. Each link $e_i \in E$ has a capacity c_i. Let K be the set of commodity flows, and each flow is associated with a source u, a destination v, a demand $d_{u,v}$ and a set of preconfigured tunnels $T_{u,v}$. A tunnel is a path from the source to the destination. The key notations are summarized in Table I.

TABLE I
Key notations in the TUFTTE.

Sets	
$G = (V, E)$	The network topology; switches and links.
K	Commodities.
$T_{u,v}$	Set of tunnels from u to v.
S	Scenarios.
Parameters	
$n = \|V\|$	The number of nodes.
$m = \|E\|$	The number of edges.
c_i	The capacity of edge $e_i \in E$.
$d_{u,v}$	The amount of traffic demand from u to v.
$D = (d_{u,v})_{n\times n}$	Demand matrix.
$\mathbf{s} \in S$	Scenario; a binary vector to record which link is failed.
$\lambda_{t,s}$	1 if tunnel t is available in scenario s, 0 otherwise.
Variables	
x_t	The amount of traffic assigned on tunnel t.
$\mathbf{x}$	The vector consisting of all x_t i.e., $\mathbf{x} = (x_t \forall t \in \cup_{(u,v\in K)} T_{u,v})$.
$l_{s,u,v}$	The loss of traffic demand from u to v in scenario s.
L	The maximum summation of traffic loss across all scenarios.

A. Minimizing network risk

The objective of TUFTTE is to minimize the network risk in all scenarios considered. To this end, a scenario set S is constructed, based on a criterion (e.g., we choose scenarios where the number of failed links is less than k, or scenarios whose possibility exceeds a threshold). Each scenario in this set is represented by a binary vector $\mathbf{s} = (s_1, s_2, \cdots, s_m)$, where s_i equals 0 if link e_i fails in scenario $\mathbf{s}$, and equals 1 otherwise. With vector $\mathbf{s}$, we are certain with failed tunnels in this scenario, and record it by $\lambda_{t,s}$ (equals 0 if tunnel t fails in scenario s, and equals 1 otherwise). Let variable x_t represents the amount of traffic assigned on tunnel t. Once we decide the routing plan (i.e., confirm values of x_t), demand loss $l_{s,u,v}$ in scenario s for node pair (u, v) can be computed. Finally, demand loss risk L is the maximum summation of traffic loss across all scenarios. The **P**roblem of **D**emand **L**oss (PDL) can be formulated as the following linear program.

$$
\begin{aligned}
\text{(PDL)} \quad & \min_{L, x_t \geq 0, l_{s,u,v} \geq 0} \quad L \\
\text{s.t.} \quad & \sum_{t \in T_{u,v}} \lambda_{t,s} x_t \geq d_{u,v} - l_{s,u,v}, && \forall (u,v) \in K, \forall s \in S, \quad (1) \\
& \sum_{t | e_i \in t} x_t \leq c_i, && \forall e_i \in E, \quad (2) \\
& \sum_{(u,v)\in K} l_{s,u,v} \leq L, && \forall s \in S. \quad (3)
\end{aligned}
$$

The objective function of PDL is minimizing the demand loss risk L, and demand constraints (1) are used to compute the traffic loss $l_{s,u,v}$. Constraints (2) represent that the amount of traffic on an edge cannot exceed its capacity, and constraints (3) formulate the definition of the demand loss. While the problem PDL is designed with demand loss, we coincidentally find that the optimal solutions to PDL are all also best solutions to FFC [10], as stated in the following proposition:

Proposition 1: The set of optimal TE decisions for optimization problem PDL

$$\{\mathbf{x}^* | (\mathbf{x}^*, \mathbf{l}^*, L^*) \text{ is an optimal solution for PDL}\}$$

is a subset of the optimal solution set for optimization problem FFC, if we concern the scenarios where the number of failed links is less than k.

Remark 1. *Proposition 1 shows that PDL formulation is superior to FFC formulation. Moreover, the goal of congestion-free TE is equivalent to minimizing the demand loss risk. The proof of Proposition 1 is omitted due to the length limit.*

By Proposition 1, the optimal solution $\mathbf{x}^*$ for PDL can be applied to both the congestion-free mechanism and the risk-driven mechanism, which we conclude as two properties:

Property 1: If every tunnel t carries traffic of x_t^*, the network is congestion-free across all scenarios.

Property 2: If every tunnel t with source u and destination v carries traffic of $\frac{\lambda_{t,s}x_t^*}{\sum_{t'\in T_{u,v}}\lambda_{t',s}x_{t'}^*}d_{u,v}$, the routing scheme achieves the lowest demand loss across all scenarios.

Remark 2. *The two properties induce two modes of TUFTTE for network operators:* ***proactive*** *mode and* ***reactive*** *mode. On one hand, Property 1 provides a protection routing scheme, which is suitable for congestion-sensitive applications. On the other hand, if the operators are willing to take risks and pursue high utilization, Property 2 helps optimize network utilization with low risk. The proofs of two properties are trivial: constraints (2) indicate that Property 1 holds, and Property 2 holds because the objective function is minimizing the demand loss.*

FFC holds Property 1, yet it does not guarantee Property 2. This is due to the fact that it restricts the sending rate of flow at end hosts. The subset of FFC's optimal solution set exhibits superior properties. Therefore, PDL is a generalization of FFC in that (i) PDL is flexible in scenario selection, and (ii) PDL provides reactive mode.

B. Loss function derivation

In the previous subsection, the solution for PDL is based on a certain traffic demand matrix $D=(d_{u,v})_{n\times n}$, so we denote the solution as $\mathbf{x}^*(D)$. However, we are unaware of the actual demand matrix D^{actual}, and thus use the predicted demand matrix D^{pred} in its place. This results in a decision loss due to inaccurate prediction. The decision loss can be calculated as $R(\mathbf{x}^*(D^{\text{pred}}),D^{\text{actual}})-R(\mathbf{x}^*(D^{\text{actual}}),D^{\text{actual}})$, where $R(\mathbf{x},D)$ is the demand loss risk. By constraints (1) and (3),

$$\begin{aligned}R(\mathbf{x},D)&=\max_{s\in S}\sum_{(u,v)\in K}l_{s,u,v}\\&=\max_{s\in S}\sum_{(u,v)\in K}\max\{d_{u,v}-\sum_{t\in T_{u,v}}\lambda_{t,s}x_t,0\}.\end{aligned}$$

One may consider applying direct optimization, which involves leveraging a deep neural network (DNN) and training it with the demand loss risk directly, as inspired by DOTE [12] and FIGRET [11]. However, this approach is not applicable to the context of failure-tolerant TE, as the output of DNN fails to guarantee the edge capacity constraints (2). Specifically, the use of the loss function $R(\mathbf{x},D)$ for training results in an increase in the value of the output x_t, thereby exceeding the capacity of the edge. This is contrary to the desired outcome. Consequently, direct optimization cannot be applied through mere adjustments.

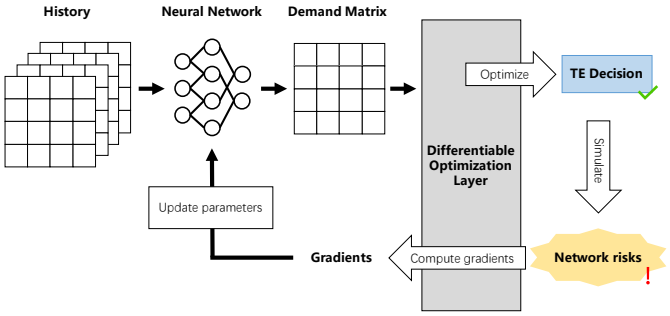


Fig. 2. TUFTTE workflow.

C. Differentiable Optimization Layer

We now address the aforementioned mismatch between the prediction task and the downstream optimization task by introducing a cutting-edge technique, the differentiable optimization layer. This technique is derived from differentiation through optimization problems [23], [27]. Recently, machine learning researches such as OptNet [25] and Cvxpylayer [24] have found that the argmin differentiation can be combined with DNN. The differentiable optimization layer is proposed as a middle layer in neural networks, with the implementation of a series of practical tools. These tools facilitate the forecasting of traffic demand with the goal of minimizing network risks. Next we explain how it can be applied in the failure-tolerant TE.

Fig. 2 depicts the training pipeline. The process begins with a set of historical demand matrices, which are input to the predictive neural network.The output of the neural network is a predicted demand matrix. Subsequently, the differentiable optimization layer solves the problem PDL based on the prediction, and records the gradients of TE decision with respect to the demand matrix. During the training phase, the TE decision is evaluated to obtain the network risk by simulating with the actual traffic matrix. Finally, the gradients are back propagated and the parameters of the predictive neural network are updated.

Recall that the network risk with respect to TE decision is $R(\mathbf{x},D)$. By the chain rule, the partial derivative of R with respect to $d_{u,v}$ is $\frac{\partial R}{\partial d_{u,v}}=\frac{\partial R}{\partial \mathbf{x}}\cdot\frac{\partial \mathbf{x}}{\partial d_{u,v}}$. Here $\frac{\partial R}{\partial \mathbf{x}}$ is easy to compute, so the key to the differentiable optimization layer is how to calculate $\frac{\partial \mathbf{x}}{\partial d_{u,v}}$. Next we explain the procedure of the argmin differentiation, taking PDL as an example.

To begin with, the Lagrangian of PDL is given by

$$\begin{aligned}&\mathscr{L}(\mathbf{x},\mathbf{l},L,\alpha,\beta,\gamma)\\=&L-\sum_{(u,v)\in K,s\in S}\alpha_{s,u,v}(\sum_{t\in T_{u,v}}\lambda_{t,s}x_t-d_{u,v}+l_{s,u,v})\\&-\sum_{i|e_i\in E}\beta_i(c_i-\sum_{t|e_i\in t}x_t)-\sum_{s\in S}\gamma_s(L-\sum_{(u,v)\in K}l_{s,u,v}),\end{aligned}\tag{4}$$

where α, β, and γ are the non-negative dual variables on the constraints (1) (2) (3). The KKT conditions for stationarity and complementary slackness are

$$\begin{aligned}
&\sum_{i|e_i \in t} \beta_i^* - \sum_{s \in S} \alpha_{s,u_t,v_t}^* \lambda_{t,s} = 0, && \forall t \in \cup_{(u,v)\in K} T_{u,v}, \\
&\gamma_s^* - \alpha_{s,u,v}^* = 0, && \forall (u,v) \in K, \forall s \in S, \\
&1 - \sum_{s \in S} \gamma_s^* = 0, \\
&\alpha_{s,u,v}^* \Big(\sum_{t \in T_{u,v}} \lambda_{t,s} x_t^* - d_{u,v} + l_{s,u,v}^* \Big) = 0, && \forall (u,v) \in K, \forall s \in S, \\
&\beta_i^* \Big(c_i - \sum_{t|e_i \in t} x_t^* \Big) = 0, && \forall e_i \in E, \\
&\gamma_s^* \Big(L^* - \sum_{(u,v)\in K} l_{s,u,v}^* \Big) = 0, && \forall s \in S,
\end{aligned} \tag{5}$$

where $\mathbf{x}^*, \mathbf{l}^*, L^*, \alpha^*, \beta^*$, and γ^* are the optimal primal and dual variables. Denote this system of equations (5) as $\mathbf{F}(\mathbf{y}^*, D) = 0$, where $\mathbf{y}^* = (\mathbf{x}^*, \mathbf{l}^*, L^*, \alpha^*, \beta^*, \gamma^*)$. By the implicit function theorem, if the determinant of the Jacobian matrix $J\mathbf{F}_\mathbf{y}(\mathbf{y}^*, D) = \frac{\partial \mathbf{F}}{\partial \mathbf{y}}(\mathbf{y}^*, D)$ is invertible, then there exists a function f, $\mathbf{y} = f(D)$, and its derivatives can be computed by differentiating $\mathbf{F}(f(D), D) = 0$. Therefore, by solving the linear system $J\mathbf{F}_D(\mathbf{y}^*, D) = 0$, we obtain all the $\frac{\partial x_t}{\partial d_{u,v}}$. Note that when $J\mathbf{F}_\mathbf{y}(\mathbf{y}^*, D)$ is not invertible, least squares method can be applied (details in [24]). Finally, the partial derivative of R with respect to the parameters θ of the predictive neural network is $\frac{\partial R}{\partial \theta} = \frac{\partial R}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial D} \cdot \frac{\partial D}{\partial \theta}$.

With the differentiable optimization layer, TUFTTE is trained as summarized in Algorithm 1. After the initialization and configuration of the learning rate (line 1), TUFTTE samples a set of continuous demand matrices $(D_{T-H}, \cdots, D_{T-1})$ and their corresponding actual demand matrix D_T (line 3). The optimal TE decision $\mathbf{x}^*(D_T)$ is pre-calculated. Subsequently, TUFTTE predicts the demand matrix (line 4) and makes a TE decision (line 5). This TE decision is then evaluated by the risk function (line 6). Finally, the parameters of the predictive model are updated with the gradients computed by the differentiable optimization layer (line 7). In the inference stage, TUFTTE only outputs the TE decision for use. In the context of the problem of minimizing demand loss (PDL), we prove that Algorithm 1 converges to the optimal TE function, as stated in Proposition 2.

Proposition 2: Let $B = |K| \cdot D_{\max}, \rho = \sum_{(u,v)\in K} |T_{u,v}|$. Assume that Algorithm 1 is run for T iterations with $\eta = \frac{B}{\rho\sqrt{T}}$, and θ is the parameter in the predictive model, then the demand loss $\mathscr{R}$ satisfies

$$\mathbb{E}[\mathscr{R}(\frac{1}{T}\sum_{t=1}^{T}\theta_t)] - \mathscr{R}(\theta^*) \leq \frac{B\rho}{\sqrt{T}}.$$

D. Acceleration techniques

The time consumed during the training process is acceptable, although the process of computing gradients in the differentiable optimization layer seems complex. Nevertheless, we present several experiences and recent studies that demonstrate how computational resources can be saved.

Algorithm 1 Stochastic gradient descent in decision-focused learning with regret as network risks

Input: Risk function R, optimization problem $\mathscr{F}$, and training data $\{((D_{T-H}, \cdots, D_{T-1}), D_T, \mathbf{x}^*(D_T))\}_{T=H}^{N}$;
Hyperparameters: Learning rate η.
1: Initialize θ in the predictive_model.
2: **for** each epoch **do**
3: **for** random $((D_{T-H}, \cdots, D_{T-1}), D_T, \mathbf{x}^*(D_T))$ **do**
4: $\hat{D} \leftarrow$predictive_model$_\theta(D_{T-H}, \cdots, D_{T-1})$.
5: $\mathbf{x}^*(\hat{D}) \leftarrow \arg\min_\mathbf{x} \mathscr{F}(\mathbf{x}, \hat{D})$.
6: $\mathscr{R} \leftarrow R(\mathbf{x}^*(\hat{D}), D_T) - R(\mathbf{x}^*(D_T), D_T)$.
7: $\theta \leftarrow \theta - \eta \frac{\mathrm{d}\mathscr{R}}{\mathrm{d}\mathbf{x}^*(\hat{D})} \frac{\mathrm{d}\mathbf{x}^*(\hat{D})}{\mathrm{d}\hat{D}} \frac{\mathrm{d}\hat{D}}{\mathrm{d}\theta}$
8: **end for**
9: **end for**

1) Warm start: The pretrain-finetune paradigm has been widely used in deep learning, and TUFTTE can also be beneficial in this context. The MSE loss is irrelevant for our downstream tasks, but it is useful for the DNN to quickly converge to a proper prediction value. Therefore, we pretrain the predictive neural network with the MSE loss before stitching it together with the differentiable optimization layer. Note that this pre-training model can be combined with other downstream tasks, which we will discuss in the next section (§ IV-A).

2) Mini-batch and parallelization.: Mini-batch SGD is a remarkable acceleration technique. One difficulty in applying it is that current state-of-the-art solvers such as Gurobi are not designed to solve multiple optimization problems in parallel. OptNet [25] implements a batched quadratic program solver *qpth* based on the primal-dual interior point method. It efficiently solves the linear system by LU decomposition[1], and takes advantage of the parallel characteristics of GPUs.

3) Surrogate loss: The training process is time-consuming primarily due to the necessity of solving an optimization problem. Researchers are therefore attempting to reduce the number of times an optimization problem is solved, or alternatively, to not solve it at all. Elmachtoub and Grigas [23] derive a convex SPO+ surrogate loss function, and they prove that the SPO+ loss is consistent with respect to the SPO loss. Their framework is applicable to a linear objective function. LODL (Locally Optimized Decision Loss) [28] proposes the utilization of an additional DNN to learn the behavior of the differentiable optimization layer. This enables rapid decision-making and gradient propagation. Most recently, TaskMet [26] integrates task-specific objectives into the process of metric learning. This enables the learning of a more effective metric for prediction error with downstream tasks. We do not employ these methods in our experiments since the training process finishes in a few hours. The utilization of these methods is reserved for instances in which the problem becomes more intricate.

[1]The LU decomposition is performed when solving the quadratic program, so the computation of gradients is relatively fast.

IV. Extensions and limitations

A. Extending with other TE algorithms

TUFTTE is sufficiently flexible to be applied to other problems by following the procedure in § III. In particular, the TUFTTE framework includes three steps to establish: (i) model the concerning problem as a mathematical program $\mathscr{F}$ (§ III-A); (ii) specify the loss function R with the ultimate objective (§ III-B); and (iii) apply the differentiable optimization layer to train the predictive neural network by forward and backward propagation (§ III-C).

With the aid of TUFTTE, many existing TE algorithms, including TEAVAR [6], PCF [7], ARROW [16] and Shoofly [14] can enhance their resilience to uncertain demands. The differentiable optimization layer is capable of solving quadratic programs [25] and disciplined convex programs [24]. This encompasses a wide range of optimization problems defined by TE algorithms. Despite the disparate objectives of these TE algorithms, they typically address tractable problems in real-time to accommodate demand variation and failures, so it is possible for them to be combined with TUFTTE. It is our hope that the integration of TUFTTE with a specific TE algorithm will serve as a general framework for mitigating the demand uncertainty. Because TEAVAR is particularly adept at enhancing availability, we will assess the efficacy of the combined approach in the next section (§ V-C).

The hose model [33], [34] represents a promising avenue for resolving demand uncertainty. Our approach is orthogonal to them: previous works such as network planning and network design focus on reducing the cost of network construction and maintenance in the context of demand and failure uncertainty. Traffic engineering, on the other hand, aims to optimize the choice of routing given a particular network topology. It is possible that TUFTTE could be integrated with a hose model in the future.

B. Dealing with multiple QoS groups

Classifying traffic in backbone network is a common practice currently [2]–[5], [22]. Different routing decisions are made for different QoS classes. In light of this complex situation, it is possible to construct multiple differentiable optimization layers to compute the gradients for each QoS class, thereby assisting them in predicting demand matrices.

Fig. 3 depicts a hierarchical structure for making TE decisions for multiple QoS classes. The QoS class with the highest priority, class 1, is the first to make decisions. The outcome of this decision is then evaluated for risk. Network risk 1 can be used to train the predictive model 1 by back propagation. Once the predictive model 1 has been trained, the TE decision for the QoS class 1 is fed into the differentiable optimization layer of the next QoS class. This procedure is repeated for each subsequent QoS class. It is also possible to train the predictive model 1 with network risk 2, if the network risk 1 is close to 0. This is a common phenomenon because the amount of high-priority flow is usually small, leading to a low risk. Overall, Fig. 3 represents a decision-making process that involves predictive modeling, optimization, decision-making, and risk assessment with each step being iterative and dependent on the previous step's output.

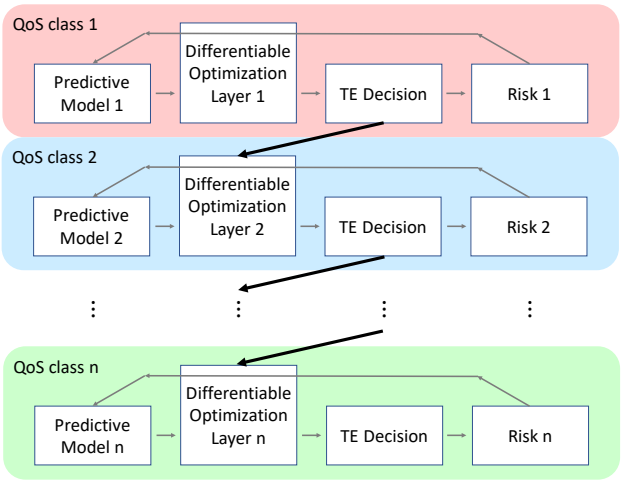


Fig. 3. Multiple layers for multiple QoS groups.

C. Limitations

The end-to-end framework of TUFTTE that incorporates prediction and optimization has a certain scope of applicability. Additionally, it also has some limitations that are left for our future work.

1) Types of traffic suitable for using TUFTTE: Two types of traffic are not recommended for TUFTTE, and they are better handled by traditional predict-then-optimize methods and other effective mechanisms.

- Congestion-sensitive traffic such as control signaling and routing protocol messages. These flows are critical to the management of the network. Increased latency and packet loss on these flows will have serious consequences. Consequently, the congestion-free TE is more appropriate for them.
- Long-running transfers such as data migrations. These flows are typically bandwidth-intensive and machine-to-machine. Their periodicity makes it easier to predict the amount of their traffic demand [8], [12], so they can be delayed appropriately, as long as they are guaranteed to finish before the deadline. Solutions such as bandwidth brokers and rate limiters are practical and effective [2], [8].

2) Evolvable neural network architecture: Traffic patterns and topological information are constantly changing. Our simple neural network will fail to provide appropriate prediction for the optimization layer with the growth of traffic demand. To avoid training the predictive neural network from scratch periodically, more expressive structures such as recurrent neural network, long short-term memory and graph neural network could be helpful.

V. Evaluation

In this section, we assess the efficacy of TUFTTE in a real-world setting. We begin by outlining our experimental settings (§ V-A). Thereafter, we design experiments to address the following questions:

- Can TUFTTE enhance the performance of prediction-based methods? How TUFTTE performs when traffic demand uncertainty increases? (§ V-B)
- Can TUFTTE be integrated with other TE schemes? (§ V-C)
- What distinguishes the predictions of TUFTTE from those of other methods? Does network risk facilitate the prediction of the demand matrix? (§ V-D)

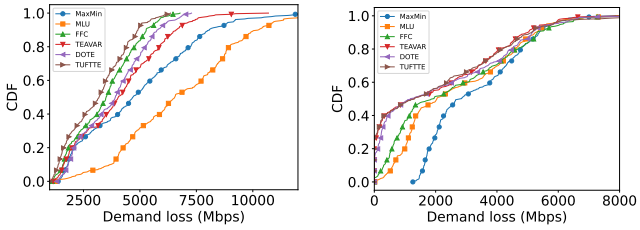
A. Experimental methodology

Datasets. Two production topologies (Abilene and GEANT) from the SNDlib [32] are used in the experiments. We collect about three days of traffic traces for training, and about one day of traffic traces for testing. The traffic demand for each pair of nodes is measured every five and fifteen minutes in Abilene and GEANT, respectively. The future demand matrix is predicted based on the previous H demand matrices ($H = 12$ in our experiments). We simply use k-shortest-paths with $k = 8$ as our tunnels.

Baselines. We compare TUFTTE to two types of TE schemes: prediction-based methods and direct optimization methods.

- *Prediction-based.* These methods predict the demand matrix first, and then solve the downstream optimization problems: (i) MaxMin optimizes max-min fairness for all pairs of nodes [4], [35]. (ii) MLU minimizes the maximum link utilization [9], [36]. (iii) FFC [10] maximizes the throughput while guaranteeing congestion-free. (iv) TEAVAR [6] minimizes the conditional value at risk (CVaR). TEAVAR's availability target (β) is set at 99.9% in our simulation. The random forest regression is used as their predictive method, as it performs well in Fig. 1a.
- *Direct optimization.* DOTE [12] employs a DNN to directly approximate a TE configuration function. We also consider TUFTTE as an end-to-end direct optimization approach. The predictive neural network of TUFTTE framework comprises three fully connected NN layers with 128 neurons each and *ReLU* activation, while DOTE uses five NN layers.

Fault models. Information on failures is not recorded in the public datasets, so we follow the methodology in TEAVAR [6]. We generate the failure probability of each link with a Weibull distribution (shape=0.8, scale=10^{-4} as recommended in TEAVAR). Two types of failure scenario sets are implemented: *Set 1* includes scenarios with k links failed, and *Set 2* includes scenarios whose probability exceeds a cutoff value. In fact, *Set 1* is a special case of *Set 2* when we assume the failure probability of every link is equal. Empirical data from Microsoft WAN [6] and Facebook [22] indicates that the link with the highest probability of failure is more than ten times more likely to fail than the link with the lowest probability of failure. Therefore, *Set 2* is a more general and practical approach.



(a) GEANT (b) Abilene

Fig. 4. The CDF of demand loss for distinct methods.

Metrics. As a resilient routing scheme, TUFTTE focuses on the aforementioned network risks outlined in § II-A. Given a TE configuration, we can calculate the load of traffic on each link, and derive the following two risks. (i) Demand loss is defined as the maximum amount of the summation of the unsatisfied demand for all node pairs across all failure scenarios. In this context, we utilize *Set 1* to quantify the demand loss, as FFC is optimized for scenarios within *Set 1*. (ii) Availability is defined as the summation of the probabilities for congestion-free scenarios (i.e., demands are fully satisfied). In this instance we utilize *Set 2* to compute the availability, as the scenario pruning achieves high scenario coverage and low standard error [6].

Failure recovery. In the event of failure in the network, whereby a link is unable to forward packets, the tunnel containing this link is rendered inaccessible and the source switch of the tunnel is alerted to this incident. Until the centralized controller intervenes, the source switch attempts to distribute the traffic to other tunnels in order to maintain the functionality of the network. The conventional approach to rerouting the traffic on failed tunnels is to rescale and redistribute the traffic proportionally [2], [4], [10], [18]. For instance, if the traffic split ratios are $(0.2, 0.3, 0.5)$, they will change to $(0.4, 0.6, 0)$ if the last tunnel goes down. In the event that the TE system is unable to make a decision regarding the source switch, the source switch will distribute the traffic across all available tunnels in an even manner.

B. Effectiveness of TUFTTE

Fig. 4 shows the cumulative distribution function (CDF) of demand loss in Mbps for distinct methods. A total of up to 300 demand matrices are utilized to evaluate these methods. For the Abilene topology, the traffic demands are scaled up by a factor of 15 to simulate network congestion. The experimental results show that TUFTTE exhibits the lowest demand loss in the majority of cases. This is because TUFTTE handles link failure by modeling PDL and learning an appropriate prediction of traffic demands. Note that this prediction is not precise, but useful (further investigation will be conducted in § V-D).

Notably, TUFTTE reduces the demand loss by an average of 11.59% on the GEANT topology compared to FFC (Fig. 4a). In the most different case, the demand loss is reduced by 52.71%. FFC also achieves a low demand loss, which is

TABLE II

THE RELATION BETWEEN NOISE MAGNITUDE AND AVERAGE DEMAND LOSS OF TUFTTE'S DECISION ON THE GEANT TOPOLOGY. THE INCREMENT IN DEMAND LOSS IS CALCULATED BASED ON RUNNING TUFTTE WITH TRAFFIC DEMANDS WITHOUT NOISE.

noise ε in traffic	FFC's increment on average	TUFTTE's increment on average
10%	10.40%	1.03%
15%	11.62%	1.30%
20%	13.08%	2.69%
25%	14.36%	4.49%
30%	15.54%	6.42%

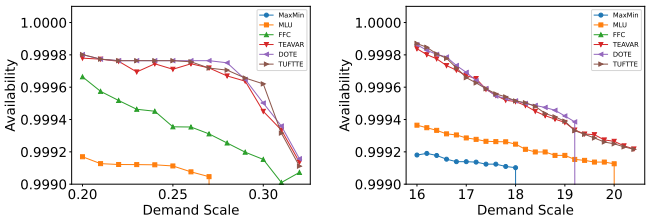
consistent with its objective: maximizing the throughput while guaranteeing no congestion loss across all scenarios. In contrast, DOTE and TEAVAR result in a comparatively greater demand loss on GEANT. This could be because they are targeted at other objectives. Note that the discrepancy between DOTE and MLU can be attributed to the resilience of DOTE.

With regard to the Abilene topology (Fig. 4b), DOTE and TEAVAR demonstrate superior performance, approaching that of TUFTTE. This may be attributed to the symmetry of the Abilene topology and the predictability of its traffic demands. It is relatively straightforward to identify a robust configuration on a simple network topology. The use of FFC results in an increased risk to the network due to the scaling up of the traffic demands. FFC is a conservative TE scheme, which may contribute to this increased risk.

Subsequently, the resilience of TUFTTE to unforeseen traffic fluctuations is evaluated. Random noise is introduced into the datasets. To illustrate, let the actual demand be denoted by $d_{u,v}^{\text{actual}}$, then the noisy demand is $d_{u,v}^{\text{actual}}(1+\varepsilon)$, where ε is sampled from a uniform distribution. As summarized in Table II, we calculate the increment in demand loss for TUFTTE and FFC relative to the demand loss of TUFTTE without noise. It can be observed that the noise has a relatively small effect on the solution quality. For instance, when the noisy demand is within 20% of the actual demand, the demand loss of TUFTTE is within 3% of the solution obtained with the actual demand.

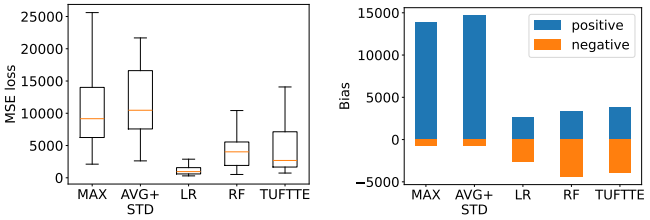
C. Flexibility of TUFTTE

We next present the performance of TUFTTE in conjunction with TEAVAR in order to demonstrate its flexibility. For a comprehensive evaluation of availability, each term in the demand matrix is multiplied by a demand scale factor, and the simulation is then conducted with a range of demand scale factors. The resulting trends in availability with the variation of demand scale for distinct methods are presented in Fig. 5. Only those methods whose availability exceeds 99.9% are shown in the figure, because the networking system requires high availability [3], [22], [31]. TEAVAR and DOTE achieve high availability on the both topologies in comparison to other TE algorithms, which is consistent with the results of prior works. It can be observed that TUFTTE enhances the performance of TEAVAR. Note that improving availability by 1% is significant for large networking systems [3], [22]. The availability of DOTE drops notably when the demand scale factor is large;



(a) GEANT (b) Abilene

Fig. 5. Availability vs. demand scales for TUFTTE and various TE schemes. The availability is not monotonically decreasing because of traffic uncertainty.



(a) (b)

Fig. 6. (a) Prediction error of distinct methods. (b) Positive and negative biases of distinct methods.

however, it is faster than any other methods, so the use of DOTE depends on the specific case.

D. A closer look at prediction

In § II-B, four prediction methods (MAX, AVG+STD, LR, RF) are combined with FFC to evaluate the resulting demand loss. In this section, we revisit the aforementioned prediction methods, and compare their predictions with those of TUFTTE. Firstly, their mean square error (MSE) losses are presented in Fig. 6a, where the prediction of TUFTTE is the output of its predictive neural network. It is notable that both the MAX and AVG+STD prediction methods tend to overestimate the traffic demand, which is reflected in the relatively high MSE losses observed. With the exception of these two prediction methods, the prediction of TUFTTE has the highest MSE loss. However, this is not a cause for concern, as the prediction is useful for minimizing the demand loss. The LR method achieves the lowest MSE loss, which is consistent with its target. It is crucial to acknowledge that, as illustrated in the Fig. 1a, LR results in a greater demand loss than RF in the majority of cases. This indicates that low prediction error does not necessarily translate to low network risks. The definition of prediction error plays a pivotal role in this regard. In conclusion, measuring the prediction error with the ultimate objective is beneficial for TE systems to make informed decisions.

Next we decompose the prediction error into two parts: *positive bias* and *negative bias*. Let $d_{u,v}^{\text{pred}}$ and $d_{u,v}^{\text{actual}}$ represent the predicted and actual demand, respectively. Then the *positive bias* is defined as $\sum_{(u,v)\in K}\max(d_{u,v}^{\text{pred}}-d_{u,v}^{\text{actual}},0)$ and the *negative bias* is defined as $\sum_{(u,v)\in K}\min(d_{u,v}^{\text{pred}}-d_{u,v}^{\text{actual}},0)$. In other words, if the predicted demand $d_{u,v}^{\text{pred}}$ is greater than the actual demand $d_{u,v}^{\text{actual}}$, then the difference between them is counted in the *positive bias*, otherwise in the *negative bias*.

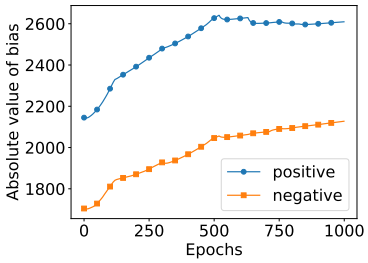


Fig. 7. The tendency of the absolute value of bias with training epochs.

Fig. 6b shows the average positive and negative biases of distinct prediction methods. Bars above the x-axis represent positive bias, while bars below the x-axis represent negative bias. Note that the negative bias of MAX is not equal to 0 because it is possible that future demand may exceed all historical demand. Fig. 6b yields similar conclusions as Fig. 6a, and serves to illustrate the concept of positive and negative biases.

In order to observe the prediction behavior of TUFTTE, a sample instance is created which includes H historical demand matrices and one actual demand matrix. Subsequently, the predictive neural network is trained on the sample instance, incorporating differentiable optimization layer, for a total of 1,000 epochs. The tendency of resulting positive and negative biases is shown in Fig. 7. To facilitate clear illustration, the absolute value of bias is plotted on the figure. It is evident that the absolute values of both positive and negative biases exhibit an upward trend over the course of training. This behavior differs from that observed when training with the MSE loss. When the predictive neural network is trained with the MSE loss, the bias will no doubt decrease with the descent of MSE loss. However, the biases diverge from 0 as the number of training epochs increases (Fig. 7). This implies that the traffic demands between specific pairs of nodes have been overestimated, while those between other pairs have been underestimated. The overestimation and underestimation are beneficial in enhancing the resilience to uncertain demands. By examining which pairs of nodes are overestimated, it is possible to develop a theory to explain this phenomenon. This highlights the necessity for greater scrutiny of specific pairs of nodes, while affording less importance to others. We leave to future work further investigation of the relationship between specific pair of nodes and the demand loss risk.

VI. Related Work

Risk-driven TE. In modern communication networks, ensuring uninterrupted service in the face of failures is of paramount importance. Risk-driven TE, or risk-aware TE, is designed to optimize the network risks from equipment failures [6], [16], [18]–[20]. As a typical example, TEAVAR [6] incorporates the likelihood of different failure events and balances utilization and availability by applying value at risk. ARROW [16] restores capacities in optical layer and solves a cross-layer schedule problem. It improves the network throughput under the failure scenarios by reconfiguring the mapping between wavelengths and IP links. The MLU is considered as another risk metric, thus prior works that aim to minimize the MLU [18]–[20] count as risk-driven TE. For instance, R3 [19], targeted at MLU, models the link failures as *virtual demands*, and reserves capacities for them. R3 is proven to be congestion-free when the resulting MLU is less than 1. Additionally, TUFTTE represents a type of risk-driven TE, and serves to bridge the gap between prediction and optimization tasks.

Demand uncertainty reduction. The history of research on demand uncertainty is longer. Mitra and Wang [37] analyze the impact of demand uncertainty and utilize stochastic optimization for their TE framework. They focus on revenue management, rather than the tolerance to failures. Demand oblivious routing [38], [39] achieves low competitive ratio without the knowledge of historical traffic demands. COPE [36] optimizes the MLU over a range of demand matrices in order to achieve greater robustness. SMORE [9] utilizes diverse paths by constructing randomized routing tree for robustness. Recently, DOTE [12] directly optimizes the prediction task by deep learning and stochastic gradient descent. These studies provide inspiration for us to rethink the failure-tolerant TE with demand prediction.

Decision-focused learning. Decision-focused learning prioritizes the direct optimization of decision-making outcomes over merely improving predictive accuracy. Elmachtoub and Grigas [23] formally define the "Smart Predict, Then Optimize" (SPO) framework, emphasizing the importance of training predictive models with respect to the downstream decision objective. This represents a pivotal shift from conventional machine learning paradigms. Amos and Kolter [25] analyze the convergence properties of learning frameworks that integrate optimization objectives, offering a rigorous mathematical foundation for understanding the convergence behavior of decision-focused learning algorithms. Agrawal et al. [24] implement differentiable layers for disciplined convex programs. A number of recent works [26], [28], [29] propose surrogate loss functions as a means of accelerating the training process. These works provide both the theoretical and practical basis for the TUFTTE.

VII. Conclusion

This paper presents three contributions. First, we investigate the impact of the traffic demand uncertainty in failure-tolerant TE, and analyze the limitations of prediction-based methods. Second, we formulate a linear program targeted at the demand loss risk, which is proven to be a generalization of FFC. Third, we propose TUFTTE, which employs a differentiable optimization layer to guide the predictive method with the downstream task. Our method can provably converge to the global optimum. Empirical experiments on real-world topologies demonstrate that TUFTTE's schemes result in an average of 11.59% less demand loss risk than the prediction-based FFC. TUFTTE's framework is adaptable to incorporate other TE schemes. Our findings on the prediction of TUFTTE also have implications for future research.

References

[1] M. Denis, Y. Yao, A. Hatch, Q. Zhang, C. L. Lim, S. Zhang, K. Sugrue, H. Kwok, M. J. Fernandez, P. Lapukhov, S. Hebbani, G. Nagarajan, O. Baldonado, L. Gao, and Y. Zhang, "Ebb: Reliable and evolvable express backbone network in meta," in *Proceedings of the ACM SIGCOMM 2023 Conference*, p. 346–359.

[2] 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 the ACM SIGCOMM 2013 Conference on SIGCOMM*, p. 15–26.

[3] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, K. N. B., C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, S. Padgett, F. Rabe, S. Ray, M. Tewari, M. Tierney, M. Zahn, J. Zolla, J. Ong, and A. Vahdat, "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *Proceedings of the ACM SIGCOMM 2018 Conference*, p. 74–87.

[4] 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 the ACM SIGCOMM 2013 Conference on SIGCOMM*, p. 3–14.

[5] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, "Decentralized cloud wide-area network traffic engineering with BLASTSHIELD," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 325–338.

[6] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "Teavar: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM SIGCOMM 2019 Conference*, p. 29–43.

[7] C. Jiang, S. Rao, and M. Tawarmalani, "Pcf: Provably resilient flexible routing," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, p. 139–153.

[8] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, p. 515–526.

[9] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-Oblivious traffic engineering: The road not taken," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pp. 157–170.

[10] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, p. 527–538.

[11] X. Liu, S. Zhao, Y. Cui, and X. Wang, "Figret: Fine-grained robustness-enhanced traffic engineering," in *Proceedings of the ACM SIGCOMM 2024 Conference*, p. 117–135.

[12] Y. Perry, F. V. Frujeri, C. Hoch, S. Kandula, I. Menache, M. Schapira, and A. Tamar, "DOTE: Rethinking (predictive) WAN traffic engineering," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 1557–1581.

[13] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pp. 201–216.

[14] R. Singh, N. Bjørner, S. Shoham, Y. Yin, J. Arnold, and J. Gaudette, "Cost-effective capacity provisioning in wide area networks with shoofly," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, p. 534–546.

[15] 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 the 2016 ACM SIGCOMM Conference*, p. 58–72.

[16] Z. Zhong, M. Ghobadi, A. Khaddaj, J. Leach, Y. Xia, and Y. Zhang, "Arrow: restoration-aware traffic engineering," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, p. 560–579.

[17] Y. Chang, C. Jiang, A. Chandra, S. Rao, and M. Tawarmalani, "Lancet: Better network resilience by designing for pruned," *SIGMETRICS Perform. Eval. Rev.*, vol. 48, no. 1, p. 53–54, jul 2020.

[18] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," *SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 1, p. 97–108, jun 2011.

[19] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang, "R3: resilient routing reconfiguration," in *Proceedings of the ACM SIGCOMM 2010 Conference*, p. 291–302.

[20] J. Zheng, H. Xu, X. Zhu, G. Chen, and Y. Geng, "We've got you covered: Failure recovery with backup tunnels in traffic engineering," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pp. 1–10.

[21] Z. Xu, F. Y. Yan, R. Singh, J. T. Chiu, A. M. Rush, and M. Yu, "Teal: Learning-accelerated optimization of wan traffic engineering," in *Proceedings of the ACM SIGCOMM 2023 Conference*, p. 378–393.

[22] Y. Xia, Y. Zhang, Z. Zhong, G. Yan, C. L. Lim, S. S. Ahuja, S. Bali, A. Nikolaidis, K. Ghobadi, and M. Ghobadi, "A social network under social distancing: Risk-Driven backbone management during COVID-19 and beyond," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pp. 217–231.

[23] A. N. Elmachtoub and P. Grigas, "Smart "predict, then optimize"," *Management Science*, vol. 68, no. 1, pp. 9–26, 2022.

[24] A. Agrawal, B. Amos, S. T. Barratt, S. P. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Neural Information Processing Systems*.

[25] B. Amos and J. Z. Kolter, "Optnet: differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, p. 136–145.

[26] D. Bansal, R. T. Q. Chen, M. Mukadam, and B. Amos, "Taskmet: Task-driven metric learning for model learning," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 505–46 519.

[27] P. Donti, B. Amos, and J. Z. Kolter, "Task-based end-to-end model learning in stochastic optimization," in *Advances in Neural Information Processing Systems*, vol. 30.

[28] S. Shah, K. Wang, B. Wilder, A. Perrault, and M. Tambe, "Decision-focused learning without decision-making: Learning locally optimized decision losses," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 1320–1332.

[29] A. Zharmagambetov, B. Amos, A. Ferber, T. Huang, B. Dilkina, and Y. Tian, "Landscape surrogate: Learning decision losses for mathematical optimization under partial information," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 27 332–27 350.

[30] Y. Chang, S. Rao, and M. Tawarmalani, "Robust validation of network designs under uncertain demands and failures," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 347–362.

[31] T. Hauer, P. Hoffmann, J. Lunney, D. Ardelean, and A. Diwan, "Meaningful availability," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pp. 545–557.

[32] ZIB Solver Collection Team, "SNDlib: Survivable network design library," http://sndlib.zib.de, Zuse Institute Berlin, 2024, accessed: April 9, 2024.

[33] S. S. Ahuja, V. Gupta, V. Dangui, S. Bali, A. Gopalan, H. Zhong, P. Lapukhov, Y. Xia, and Y. Zhang, "Capacity-efficient and uncertainty-resilient backbone network planning with hose," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, p. 547–559.

[34] J. P. Eason, X. He, R. Cziva, M. Noormohammadpour, S. Balasubramanian, S. S. Ahuja, and B. Lu, "Hose-based cross-layer backbone network design with benders decomposition," in *Proceedings of the ACM SIGCOMM 2023 Conference*, p. 333–345.

[35] 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*, pp. 846–854.

[36] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "Cope: traffic engineering in dynamic networks," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, p. 99–110.

[37] D. Mitra and Q. Wang, "Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 221–233, 2005.

[38] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, p. 313–324.

[39] H. Racke, "Minimizing congestion in general networks," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 43–52.