DOTE: Rethinking (Predictive) WAN Traffic Engineering

Yarin Perry[1], Felipe Vieira Frujeri[2], Chaim Hoch[1], Srikanth Kandula[2], Ishai Menache[2], Michael Schapira[1], and Aviv Tamar[3]

[1]Hebrew University of Jerusalem, [2]Microsoft Research, [3]Technion

Abstract– We explore a new design point for traffic engineering on wide-area networks (WANs): *directly* optimizing traffic flow on the WAN using only *historical* data about traffic demands. Doing so obviates the need to explicitly estimate, or predict, future demands. Our method, which utilizes stochastic optimization, provably converges to the global optimum in well-studied theoretical models. We employ deep learning to scale to large WANs and real-world traffic. Our extensive empirical evaluation on real-world traffic and network topologies establishes that our approach's TE quality almost matches that of an (infeasible) omniscient oracle, outperforming previously proposed approaches, and also substantially lowers runtimes.

1 Introduction

To meet the constant rise in traffic, service providers invest huge effort into traffic engineering (TE)—optimizing traffic flow across their backbone WANs [11, 22, 24, 28, 37, 39, 57], which interconnect their datacenters with each other and with external networks. The production state-of-the-art involves periodically solving a (logically centralized) optimization problem to determine how to best split traffic across network paths. Changes to TE configurations are realized using software-defined control of network hardware [11, 22, 24, 35, 38, 39].

A key challenge for WAN TE is uncertainty regarding future traffic demands. The standard approach for contending with this is twofold. For time-sensitive traffic, providers measure application-specific usage data from switches (e.g., using sampled netflow or ipfix counters) and attempt to *predict* future usage. For bandwidth-hungry, scavenger-class traffic [22], providers deploy so called agents/shims in the OS of hosts from which traffic originates. These agents explicitly signal applications' traffic demands to "brokers" that, in turn, aggregate demands, relay them to the centralized optimizer, and enforce the resulting rate allocations [22, 24].

Both of the above approaches for handling traffic uncertainty have drawbacks. Demand predictions can naturally be erroneous and, more importantly, there is an objective mismatch between the loss functions to predict future traffic demands (e.g., mean-squared-error, L1 norm error) and the *end-to-end objective* of producing high-performance TE configurations. For example, mean-squared-error would weight error in *any demand* equally, yet errors on demands that are more problematic to carry on a given topology will exert a disproportionately large effect on TE quality. The other approach – brokering and explicitly specifying demands – entails nontrivial operational overheads, including changes to end-hosts and applications. This can increase the lag experienced by application requests (which is why this approach is used in practice only for bandwidth-hungry, scavenger-class traffic [22]).

The demand uncertainty challenge is further amplified for *customer-facing traffic* (web, images, e-mails, videos, etc.), which constitutes a large and growing share of the total traffic traversing some providers' backbones. For such traffic, which originates in unmodified apps or clients, brokering in the host OS is not applicable. Moreover (see §2.1), such traffic exhibits high variability and is difficult to predict accurately.

We explore a new design point for WAN TE: training a TE decision model on *historical* data about traffic demands to *directly* output high-quality TE configurations. We present the *DOTE* (Direct Optimization for Traffic Engineering) TE framework. *DOTE* applies *stochastic optimization* to *learn* how to map recently observed traffic demands (e.g., empirically-derived traffic demands from the last hour) to the next choice of TE configuration. Using *DOTE*, providers need only *passively* monitor traffic to/from datacenters and do not have to onboard applications onto brokers. Directly predicting TE outcomes that optimize TE performance also resolves the objective mismatch between demand prediction and TE performance, yielding TE outcomes that are more robust to traffic unpredictability. We show how *DOTE* can scale to handle large WANs and real-world traffic by harnessing the expressiveness of deep learning.

We evaluate *DOTE* both analytically and empirically. Our theoretical results establish that if the TE optimization objective satisfies desirable convexity/concavity properties, *DOTE provably* converges to the optimum. We prove that this is indeed the case for standard TE optimization objectives such as minimizing the maximum-link-utilization (MLU) [8, 14, 27], maximizing network throughput [4, 22, 24, 37], and maximizing concurrent-flow [11, 29].

Our empirical evaluation compares *DOTE*, in terms of both quality and runtimes, to TE with explicit demand estimates from end-hosts, demand-prediction-based TE, demand-oblivious TE, deep-reinforcement-learning-based TE, and more. Evaluating data-driven TE schemes like *DOTE* requires substantial empirical data regarding traffic conditions for both training and performance analysis. We conduct a large-scale empirical study using both publicly available datasets and historical data from Microsoft's private WAN. These datasets span months of traffic demands at few-minutes granularity, amounting to tens of thousands of demand snapshots. Our evaluation covers small (10s of nodes) and large (100s of nodes) WANs, different types of traffic (including inter-datacenter and customer-facing), and different TE optimization objectives. To facilitate reproducibility, our code is available at [2].

Our evaluation results show that:

- ***DOTE* achieves TE quality almost matching that of an *infeasible* oracle with *perfect knowledge* of future demands**. Across all evaluated network topologies, traffic traces, and considered TE objectives, *DOTE* compares favorably to all other evaluated TE schemes. We also demonstrate *DOTE*'s robustness to changes in traffic conditions and to network failures.
- **By invoking a DNN for the online computation of TE configurations, *DOTE* achieves runtimes 1-2 orders of magnitude faster** than solving a linear program (LP), even for large WANs, matching the gains from recent proposals for fast (approximate) LP optimizations [4,40]. Our approach thus also holds promise for expediting decision making for TE.

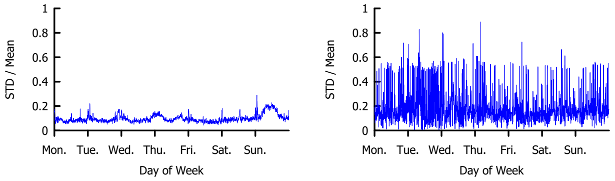
We view our investigation of direct optimization for WAN TE as a first step and discuss current limitations of our approach that we hope future research can address.
This work does not raise any ethical concerns.[1]

2 Motivation and Key Insights

2.1 Inter-DC *vs.* Customer-Facing Traffic

Enterprise WANs carry traffic between the provider's own datacenters (e.g., geo-replication of datasets, newly computed search indices) as well as traffic traversing the backbone towards/from customers (e.g., web traffic, videos).

To motivate our direct optimization approach, we present analyses of traffic on Microsoft's production WAN. Figure 1(a) plots the standard deviation in *inter-datacenter traffic* demands, normalized by the mean, across 11 consecutive weeks, for the pair of datacenters with the highest average demand. Demands are collected at 5-minute granularity. Similarly, Figure 1(b) plots the normalized standard deviation in *customer-facing traffic* demands over 4 consecutive weeks for the pair of nodes with the highest average demand. Observe the substantial difference; in the inter-datacenter traffic trace, demands are *significantly* less variable.



(a) Inter-data-center traffic (b) Customer-facing traffic

Figure 1: Variability in traffic demands for inter-datacenter traffic and customer-facing traffic across different weeks.

High variability in customer-facing traffic demands can accrue from different sources, e.g., (1) flash-crowds that may cause a surge in search requests, e-mail volume, etc., (2) congestion on the WAN's peering links with ISP networks, and (3) route changes and outages that cause traffic to ingress or egress the WAN at different sites. We have observed that customer-facing demands can exceed 100× the average value for extended stretches of time. Thus, customer-facing traffic is harder to accurately predict than inter-datacenter traffic. See Figure 10(a)–Figure 10(b) in the appendix for differences in demand-prediction accuracy between the above discussed two traffic traces.

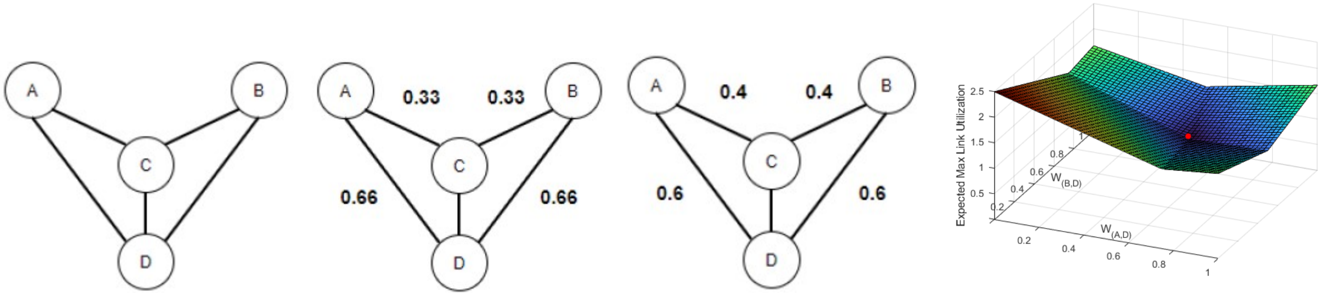
To summarize: for customer-facing traffic, which is a large and growing share of overall WAN traffic, not only is direct inference of traffic demands by the host OS infeasible, but accurate demand prediction also appears elusive. We seek a method that can achieve nearly optimal TE outcomes even for the unpredictable traffic demands.

2.2 Demand Prediction *vs.* Direct Optimization

We illustrate key insights underlying *DOTE* using the example in Figure 2(a). Each of nodes A and B wishes to send traffic to node D, and can do so either via its direct link to D or its 2-hop path to D through node C. All link capacities are 1. Every fixed time interval (say, 5 minutes), the TE system must determine, for each of the two source nodes, A and B, traffic splitting ratios specifying which fraction of its demand is forwarded along each of its assigned two paths to D. A and B's traffic demands for each time interval are drawn (i.i.d) at the beginning of each time interval from a *fixed* probability distribution: with probability $\frac{1}{2}$ node A's demand is $\frac{5}{3}$ and node B's demand is $\frac{5}{6}$ and with probability $\frac{1}{2}$ node B's demand is $\frac{5}{3}$ and node A's demand is $\frac{5}{6}$. The TE system has no a priori knowledge of the realization of the traffic demands; splitting ratios must be determined before actual traffic demands are revealed.

Demand-prediction-based TE and its shortcomings. A natural solution is training a predictor on empirical data containing past demands for A and B to predict the combination of demands closest (in expectation) to the realized combination of demands (*e.g.*, in terms of mean-squared-error), and then

[1] In particular, the measured traffic demands, used in our evaluation, are aggregate counters between pairs of datacenters at the granularity of minutes (or coarser). They do not contain user IP addresses or packet contents.



(a) Network topology

(b) Induced splitting ratios for a demand-predictor

(c) Optimal splitting ratios

(d) Expected MLU as a function of the splitting ratios

Figure 2: Simple WAN TE example

performing global optimization with respect to the predicted demands. In our simple example, this leads to the predicted demand-combination being $(\frac{5}{4}, \frac{5}{4})$ and the induced splitting ratios presented in Figure 2(b). Under these splitting ratios, *regardless* of the realization of the demands, either link (A,D) or link (B,D) will carry more traffic than its capacity can accommodate. In the optimal solution shown in Figure 2(c), however, *regardless of the realized demands, no link carries more traffic than its capacity can support*.

Of course, instead of predicting a single demand-combination, one could have predicted a *probability distribution* over the traffic demands and optimized splitting ratios with respect to that. This entails two nontrivial challenges, which are significantly amplified for large WANs and real-world traffic: (1) We must impose a specific structure on the probability distribution to be predicted (e.g., Gaussian, bimodal), which might not be a good fit for actual WAN traffic. This is particularly true when there hidden correlations between demands (as in our example); (2) Optimizing an LP with respect to a distribution over *multiple* demand-combinations can be prohibitively time consuming for large WANs.

On *direct* optimization of traffic splitting ratios and why it might do better. An alternative approach, which avoids presuppositions regarding the traffic, and also LP optimizations, is training a decision model on past realizations of A and B's traffic demands to *directly* output traffic splitting ratios that are close to the global optimum. This approach can outperform the demand-prediction-based approach in scenarios where traffic is volatile and hard to predict but a certain configuration of splitting ratios performs well on most traffic realizations. Directly inferring the splitting ratios also obviates the need for solving an LP to optimize splitting ratios with respect to predicted traffic. As our evaluation results in §4 show, this significantly accelerates TE runtimes for large WANs. In our example, after sufficient training, the model is expected to learn the splitting ratios in Figure 2(c) (the unique global optimum). Indeed, *DOTE*, which is a manifestation of this approach, quickly converges to this global outcome.

Exploiting convexity/concavity for direct optimization of splitting ratios via gradient descent. A key insight is that for classical TE optimization objectives, the function mapping splitting ratios to *expected* performance scores satisfies desirable properties, namely, *convexity/concavity*. This facilitates utilizing elegant direct optimization methods, like (stochastic) gradient descent, circumventing explicit demand prediction.

To illustrate this, we consider the classical TE objective of minimizing maximum-link-utilization (MLU). We visualize in Figure 2(d) the impact of different choices of splitting ratios on MLU, i.e., the maximum ratio, across all network links, between the traffic traversing a link and the link capacity. x-axis values specify the fraction of A's traffic sent on the direct path (A,D). Since A only has two available paths, this value also uniquely determines the fraction of A's traffic sent on the indirect path (A,C,D). Similarly, y-axis values specify the fraction of B's traffic sent on (B,D) and so also on (B,C,D). z-axis values represent the *expected* MLU for different choices of splitting ratios for A (x-axis) and B (y-axis) for the underlying demand distribution described above. For instance, the scenario where A and B send all of their traffic on (A,D) and (B,D), respectively, is captured by $w_{(A,D)} = 1$ (x-axis) and $w_{(B,D)} = 1$ (y-axis), and the derived expected MLU is $\frac{5}{3}$ (z-axis). Indeed, in this scenario, regardless of which of the two demand combinations is realized, the traffic injected into either link (A,D) or link (B,D) will be $\frac{5}{3}$x its capacity. The *unique* global minimum for MLU, in which no link capacity is exceeded, is achieved for $w_{(A,D)} = 0.6$ and $w_{(B,D)} = 0.6$ (the red dot in Figure 2(d), which corresponds to the splitting ratios in Figure 2(c)).

As seen in Figure 2(d), the expected MLU exhibits a desirable structure—*convexity* in the traffic splitting ratios. This suggests the following procedure for converging to the optimum: start with *arbitrary* splitting ratios, and adapt the splitting ratios in the direction of the steepest slope of the (expected) MLU (i.e., the opposite direction of the *gradient* with respect to the splitting ratios) until converging to the global minimum. We show (in §3.3) that the convexity of the expected MLU extends to *any* network topology, *any* choice of network paths (tunnels), and *any* underlying demand distribution, and so, this elegant optimization procedure is guaranteed to converge to the global optimum in general.

2.3 TE as Stochastic Optimization

How to estimate the gradient of the *expected* MLU? Executing gradient descent on the *expected* MLU requires repeatedly evaluating the gradient for different traffic splitting configurations. However, *exact knowledge of the gradient is impossible without exact knowledge of the underlying demand distribution*. Once again, the specific structure of the TE setting gives rise to opportunities for effective optimization. We show how the gradient can be closely approximated from data samples of past realizations of the demands. Our approach builds on the following two observations that, while illustrated using our toy example, generalize to arbitrary network topologies, tunneling schemes, and distributions over traffic demands (see §3).

- **For any realized demand-combination, the MLU gradient *with respect to these specific demands* can be expressed *in closed form*.** Suppose that the realized demands in our simple example are $\frac{5}{3}$ for A and $\frac{5}{6}$ for B. The MLU as a function of A's splitting ratios, $w_{(A,D)}$ and $(1-w_{(A,D)})$, and B's splitting ratios, $w_{(B,D)}$ and $(1-w_{(B,D)})$, can be expressed as:

  $$\max\{\frac{5}{3}w_{(A,D)}, \frac{5}{3}(1-w_{(A,D)})+\frac{5}{6}(1-w_{(B,D)}), \frac{5}{6}w_{(B,D)}\}$$

  (i.e., the maximum load across the links (A,D), (C,D), and (B,D), respectively[2]). This representation of the MLU for the realized demands as a convex function of the splitting ratios enables deriving a closed form expression of the (sub)*gradient* of the MLU[3], as shall be discussed in §3.

- **Averaging over the MLU gradients *for past realized demands* closely approximates the gradient of the *expected* MLU.** Exact knowledge of the underlying probability distribution over demands is elusive in most real-world scenarios. Hence, the gradient of the *expected* MLU for a given configuration of splitting ratios cannot be precisely derived. However, this gradient can be well-approximated by averaging over the gradients *for realized demands* at those splitting ratios. In our example, deriving the expected MLU gradient for specific traffic splitting ratios for A and B can be achieved by sampling sufficiently many past realizations of A and B's demand-combinations, deriving the MLU gradient with respect to each such realized demand combination (at these splitting ratios), and averaging over these.

Why is reinforcement learning (RL) *not* a good fit? (Deep) RL methods have been applied to many networking domains, including routing [54]. Similarly to *DOTE*, RL approaches to TE also replace explicit demand prediction with end-to-end optimization, mapping recent traffic demands to TE configurations [54]. However, while RL can be applied to essentially any sequential decision making context, RL suffers from higher data-sample complexity, notorious sensitivity to noisy training, and a brittle optimization process that necessitates painstakingly sweeping hyperparameters [21]. A key observation underlying *DOTE* is that WAN TE exhibits a desirable structure that gives rise to opportunities for much simpler and more robust optimization, rendering RL an "overkill".

2.4 Harnessing Deep Learning

In our simple example, traffic demands were repeatedly drawn from the *same* probability distribution. Real-world traffic exhibits intricate temporal (hourly, diurnal, weekly), and other, patterns. To pick up on such regularities, the TE system could take into account the recent history of observed traffic demands (e.g., traffic demands from the last hour). However, there are infinitely many possible recent histories of traffic demands the TE system might observe. To address this, *DOTE* trains a deep neural network (DNN) to *approximate the optimal mapping from traffic histories to TE configurations*, exploiting the capability of DNNs to automatically identify complex patterns in large, high-dimensional data (§3.4). *DOTE* builds on recent developments in large-scale optimization, namely, the ADAM stochastic gradient descent optimizer [30], to accommodate efficient training on extensive empirical data (10s of thousands of traffic demand snapshots in our experiments).

3 Direct Optimization for TE (*DOTE*)

Below, we present our model for WAN TE with uncertain traffic demands, which extends the classical WAN TE model. We then delve into the the *DOTE* stochastic optimization framework, provide theoretical guarantees, and discuss how *DOTE* can be implemented in practice.

3.1 Modeling WAN TE

Network. The network is modeled as a capacitated graph $G = (V,E,c)$. V and E are the vertex and edge (link) sets,

[2] Observe that the load on (A,C) and (B,C) is always dominated by the load on (C,D), and so we disregard these links.

[3] Note that even though this function is not differentiable for all inputs due to the maximum operator, the subgradient always exists and can be explicitly derived.

respectively, and $c : E \to \mathbb{R}^+$ assigns a capacity to each edge.

Tunnels. Each source vertex s communicates with each destination vertex t via a set of network paths, or "*tunnels*", P_{st}.

Traffic demands. A *demand matrix* (DM) D is an $|V| \times |V|$ matrix whose (i, j)'th entry $D_{i,j}$ specifies the traffic demand between source i and destination j.

Optimization objective. To simplify exposition, we first describe *DOTE* for the case of one classical TE objective: minimizing maximum-link utilization (MLU) [9, 13, 17]. We discuss other optimization objectives (maximum network throughput and maximum-concurrent-flow) in §3.5.

TE configurations. We focus on how traffic should be split across a *given* set of tunnels so as to achieve the optimization objective. *DOTE* is compatible with any tunnel-selection method. We discuss an extension that incorporates data-driven tunnel selection in §5.

Given a network graph and demand matrix, a *TE configuration* $\mathcal{R}$ specifies for each source vertex s and destination vertex t how the $D_{s,t}$ traffic from s to t is split across the tunnels in P_{st}. Thus, a TE configuration specifies for each tunnel $p \in P_{st}$ a value x_p, where x_p is the fraction of the traffic demand from s to t forwarded along tunnel p (and so $\sum_{p\in P_{st}} x_p = 1$).

Given a demand matrix D and TE configuration $\mathcal{R}$, the total amount of flow traversing edge e is $f_e = \sum_{s,t\in V, p\in P_{st}, e\ni p} D_{s,t} \times x_p$. The objective is minimizing the maximum link utilization induced by $\mathcal{R}$ and D, $\max_{e\in E} \frac{f_e}{c(e)}$, which we will refer to as MLU and represent as $\mathcal{L}(\mathcal{R}, D)$. WAN operators seek to reduce the MLU to keep more headroom open for unplanned failures and traffic spikes. Typically, operators spend to increase link capacities when MLU exceeds a threshold value, and so reducing MLU can reduce CAPEX [14, 27].

In this work, we aim to select TE configurations without a priori knowledge of the traffic demands. To do so, we augment the above model as follows:

WAN TE under traffic uncertainty. Time is divided into consecutive intervals, called "epochs", of length δ_t. δ_t is determined by the network operator (e.g., at some large service providers [22, 24], δ_t is a few minutes). At the beginning of each epoch t, the TE configuration $\mathcal{R}^{(t)}$ for that epoch is decided based only on the history of *past* demand matrices and TE configurations. We also assume that the demand matrix is fixed within an epoch and can be approximately estimated after the fact.[4] Such periodic changes to TE configuration reflect the current practice in private WANs [22–24].

After selecting the TE configuration $\mathcal{R}^{(t)}$ for epoch t, the demand matrix D_t is revealed. To minimize MLU, the goal for direct optimization is to devise a *TE function* $\pi(D_{t-1}, \ldots D_1)$ that, for every $t > 0$, maps the history of DMs from the previous $t-1$ time epochs to a TE configuration $\mathcal{R}^{(t)}$ for the upcoming time epoch t so as to minimize $\frac{1}{T}\Sigma_{x=1}^{t}\mathcal{L}(\mathcal{R}^{(x)}, D_x)$, where T represents the length of time in which TE configurations are computed according to π.

To reason about WAN TE in the presence of traffic uncertainty, we assume that the demand matrix D_t at each epoch t is generated from some probability distribution. We also make the following two assumptions, which are fundamental to any data-driven approach to WAN TE. First, we assume that there is some sufficiently large $H > 0$ such that the finite window of H recent historical observations of DMs is sufficient for informing the decision of the next TE configuration. (Our empirical results in §4 suggest that $H = 12$ suffices for attaining high performance on our datasets.) Formally, we model the demand matrix D_t as generated according to an unknown H-Markov process with transition probabilities such that $P(D_t|D_{t-1}, \ldots, D_{t-H}) = P(D_t|D_{t-1}, \ldots, D_1)$. Second, we assume that the probability of observing a particular sequence of H DMs in the training data and during real-time system execution is the same. This formally translates to the Markov process being in a steady state. Let $P(D_{t-1}, \ldots, D_{t-H})$ denote the Markov process' stationary distribution, which determines the probability for any specific H-long recent history of DMs. The *expected* MLU for a TE configuration $\mathcal{R}$ at epoch t is therefore $\mathbb{E}_{D_t}[\mathcal{L}(\mathcal{R}, D_t)]$, where the expectation is with respect to the (unknown) probability distributions $P(D_{t-1}, \ldots, D_{t-H})$ and $P(D_t|D_{t-1}, \ldots, D_{t-H})$ defined above.

3.2 The *DOTE* TE Framework

DOTE leverages stochastic optimization to compute a TE function $\pi_\theta(D_{t-1}, \ldots, D_{t-H})$, parametrized by θ, which maps the H-long recent history of DMs to the TE configuration for the next time epoch, $\mathcal{R}^{(t)}$. If the TE function is sufficiently expressive, there should exist parameters that closely approximate the optimal TE function. As we shall discuss in §3.4, in *DOTE*, π_θ is realized by a deep neural network (DNN), and the parameters θ correspond to the DNN's link weights. We thus consider the optimization problem of seeking parameters θ for which the following expression is minimized: $\mathbb{E}[\mathcal{L}(\pi_\theta(D_{t-1}, \ldots, D_{t-H}), D_t)]$, where the expectation is with respect to choosing t uniformly at random from $\{1, \ldots, T\}$, and the probability distributions $P(D_{t-1}, \ldots, D_{t-H})$ and $P(D_t|D_{t-1}, \ldots, D_{t-H})$ defined above. Observe that by the linearity of expectation and the above equation, $\mathbb{E}[\mathcal{L}(\pi_\theta(D_{t-1}, \ldots, D_{t-H}), D_t)] = \frac{1}{T}\Sigma_{t=1}^{T}\mathbb{E}_{D_t}\left[\mathcal{L}(\mathcal{R}^{(t)}, D_t)\right]$, which is precisely our optimization objective in *DOTE*.

The training data for *DOTE* is a trace of historical DMs, consisting of N sequences of DMs of the form $\left\{D_t^i, D_{t-1}^i, \ldots, D_{t-H}^i\right\}$, where each sequence consists of $H+1$ DMs and captures a specific realization of a H-long history of DMs and the subsequent realized DM. We assume that

[4] For e.g., by sampling ipfix (or equivalent) data at each node in the WAN, as is done in production in SWAN [22] and B4 [24]. This data contains source and destination nodes and volume of bytes exchanged. Alternatively, traditional ISP backbones use network tomography on measured link usage data (see, e.g., [46, 58]).

these N observations of DM sequences are sampled i.i.d. from $t \in [1,\dots,T]$, $P(D_{t-1},\dots,D_{t-H})$ and $P(D_t|D_{t-1},\dots,D_{t-H})$.[5]

DOTE executes *stochastic gradient descent* (SGD) [51] to optimize the parameters θ by sequentially sampling m-sized mini-batches of data, where each data point in the mini-batch is drawn from the data uniformly at random. For each mini-batch of sampled data points, the parameters θ are updated as follows:

$$\theta := \theta - \alpha \frac{1}{m} \sum_{i \text{ in batch}} \nabla_\theta \mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H})),$$

where α is a step size parameter and $\nabla_\theta \mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H}))$ is the gradient of the loss function with respect to θ. Our realization of stochastic optimization in *DOTE* follows the ADAM [30] method, which incorporates momentum and an adaptive step size.

A closer look at *DOTE*'s parameter update step. Recall that our objective is to reach a performant TE configuration with respect to the *expected* loss (MLU). The success of *DOTE*'s SGD is thus crucially dependent on *DOTE*'s ability to well-approximate the gradient with respect to the expected loss. Unfortunately, in most real-world TE environments, exact knowledge of the underlying distribution over traffic demands is unattainable. To address this, *DOTE*'s parameter update step (see above) incorporates the expression $\frac{1}{m}\sum_{i \text{ in batch}} \nabla_\theta \mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H}))$. As discussed above, each sequence of $H+1$ demand matrices $\left\{D^i_t, D^i_{t-1},\dots,D^i_{t-H}\right\}$ in the batch is assumed to be independently drawn from the underlying stationary distribution of the Markov process. Hence, $\frac{1}{m}\sum_{i \text{ in batch}} \nabla_\theta \mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H}))$ is an *unbiased* estimate of the gradient of the expected loss, and closely approximates the gradient of the expected loss for a large enough m. Approximating the gradient of the expected loss in this manner is termed Sample Average Approximation (SAA) in stochastic optimization literature [51]. Relying on unbiased stochastic gradients for SGD guarantees convergence to a *global* optimum with respect to the *expected* loss [49] when the loss function is concave (as in our context, see §3.3).

We are left with the challenge of deriving $\frac{1}{m}\sum_{i \text{ in batch}} \nabla_\theta \mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H}))$. An important technical observation is that each data point i in the batch, $\mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H}))$ is a composition of *differentiable* computations. *DOTE* capitalizes on this for calculating the gradient $\nabla_\theta \mathcal{L}(D^i_t, \pi_\theta(D^i_{t-1},\dots,D^i_{t-H}))$ in closed form via backpropagation. We revisit this point in §3.4.

3.3 Analytical Optimality Results

We prove that, for a *perfectly expressive* TE function, i.e., when the TE function can be *any* mapping from demand histories to TE configurations, and in the limit of infinite empirical data sampled from the underlying Markov process' stationary distribution, *DOTE* attains optimal performance. In practice, we relax both assumptions: in *DOTE*, we sample from a large, but *finite*, dataset of historical demands, and use a *parametric* model (specifically, a neural network) to map from the set of possible histories to valid TE configurations. Our theoretical result below, however, establishes that our approach is *fundamentally sound*, and so high performance in practice can be achieved by acquiring sufficient empirical data and employing a sufficiently expressive decision model (e.g., a deep enough neural network). Our empirical results in §4 corroborate this.

For the sake of analysis, we assume that the set of possible history realizations, which we denote by $\mathbf{H}$, is finite. Let $\pi : \mathbf{H} \to \mathbf{R}$ denote a mapping from history to TE configuration[6]. We consider an idealized stochastic gradient descent (SGD) algorithm that, at each iteration k samples *a single* data point $D_t, D_{t-1},\dots,D_{t-H}$ from the probability distributions $P(D_{t-1},\dots,D_{t-H})$ and $P(D_t|D_{t-1},\dots,D_{t-H})$, and updates $\pi_{k+1} = Proj\left\{\pi_k - \eta v_k\right\}$, where $v_k \in \partial\mathcal{L}(\pi_k(D_{t-1},\dots,D_{t-H}), D_t)$ denotes the subgradient of the objective function, and $Proj$ denotes a projection onto the simplex for each (s,d) pair. The final output after K iterations is $\bar{\pi} = \frac{1}{K}\sum_{k=1}^{K}\pi_k$. Let $\bar{\mathcal{L}}(\pi) = \mathbb{E}\left[\mathcal{L}(\pi(D_{t-1},\dots,D_{t-H}), D_t)\right]$ denote the expected MLU of a TE function, and let $\pi^* \in \arg\min_\pi \bar{\mathcal{L}}(\pi)$ denote the optimal TE function. We prove the following theorem:

Theorem 1. *For any $\varepsilon > 0$, there exists $\eta > 0$ and finite K such that $\left|\mathbb{E}\left[\bar{\mathcal{L}}(\bar{\pi})\right] - \bar{\mathcal{L}}(\pi^*)\right| \le \varepsilon$, where the expectation is w.r.t. the sampling by the algorithm.*

The proof of Thoerem 1, which crucially relies on the convexity of the MLU objective, appears in Appendix B.

3.4 Scalability and Real-World Traffic

Direct TE optimization aims at computing a mapping from the history of recent traffic demands to a TE configuration that optimizes expected performance for the next demands. A key insight is that with real-world traffic, one may expect certain *patterns* in this mapping; for example, if two histories of traffic conditions are very similar, their corresponding optimal TE configurations should also be similar. However, measuring and explicitly quantifying such similarities is nontrivial. Our approach is to exploit deep neural networks, which have demonstrated remarkable success in identifying complex patterns in high dimensional data, for this task.

[5] When the data is a long trace of historical DMs, the samples are not necessarily independent. However, we assume that the mixing time of the Markov process is fast enough such that correlations between the data samples are negligible. This is a common assumption in time series prediction.

[6] Note that we dropped the subscript θ in π, as in our analysis we consider the space of all possible TE configurations, and not a specific parametrization.

DOTE employs a DNN to realize the TE function $\pi_\theta(D_{t-1}, \dots, D_{t-H})$. Specifically, *DOTE*'s DNN maps an input of H (12 in our experiments) most recent DMs into an output vector specifying the splitting ratios across tunnels for all source-destination pairs. In our implementation of *DOTE*, we use the popular Fully Connected DNN architecture. See Appendix E for a formal exposition of how the DNN's output and the realized DM are fed into the loss function to derive the induced MLU. Importantly, the sequence of steps for mapping the DNN output to the MLU value $\mathcal{L}(D_t^i, \pi_\theta(D_{t-1}^i, \dots, D_{t-H}^i))$ involves only *differentiable* computations; the loss as a function of the TE configuration is a composition of a max and a linear function, and the neural network is differentiable by design. Hence, the gradient $\nabla_\theta \mathcal{L}(D_t^i, \pi_\theta(D_{t-1}^i, \dots, D_{t-H}^i))$ can be calculated in closed form via backpropagation. In our implementation, the Pytorch [41] auto-differentiation package is used to calculate the gradients.

3.5 On Maximum and Concurrent Flow

We next explain how *DOTE* extends to two other central TE objectives: maximizing network throughput [18, 22, 24, 25] (maximum multicommodity flow) and maximum concurrent-flow [11, 29, 48].

TE configurations for flow maximization. TE objectives that capture different notions of flow maximization require that the outputs of the TE mechanism satisfy strict capacity constraints. To address this, we revise our definition of TE configuration $\mathcal{R}$ from §3.1: for each source-destination pair $s,t \in V$, $\mathcal{R}$ now specifies (1) traffic splitting ratios x_p over the paths (tunnels) $p \in P_{st}$ (as in §3.1), (2) for each path (tunnel) $p \in P_{st}$, a "*cap*" $\omega_p \geq 0$. ω_p represents the maximum permissible flow between s to t along the path p (enforced via rate limiting). $\mathcal{R}$ must satisfy that no link capacity is exceeded (*regardless of the realized demands*), i.e., that for each link $e \in E$, $\Sigma_{s,t \in V, p \in P_{st}, e \ni p} \omega_p \leq c(e)$. A TE configuration $\mathcal{R}$ and demand matrix D induce, for each tunnel p a flow $f_p(\mathcal{R}, D) = \min\{x_p \times D_{s,t}, \omega_p\}$. The total flow between s and t is thus $f_{st}(\mathcal{R}, D) = \Sigma_{p \in P_{st}} f_p(\mathcal{R}, D)$.

The maximum-multicommodity-flow and maximum-concurrent-flow objectives. In maximum-multicommodity-flow [18, 22, 24, 25], the performance objective $\mathcal{L}(\mathcal{R}, D)$ is to compute, for a given demand matrix D, a TE configuration $\mathcal{R}$ that maximizes the expression $\mathcal{L}(\mathcal{R}, D) = \Sigma_{s,t \in V} f_{st}(\mathcal{R}, D)$ (the total network throughput). For a TE configuration $\mathcal{R}$ and demand matrix D, let $\alpha(\mathcal{R}, D)$ denote the maximum value $\alpha \in [0,1]$ for which at least an α-fraction of each $D_{s,t}$ is routed *concurrently*, i.e., such that for all $s,t \in V$, $f_{st}(\mathcal{R}, D) \geq \alpha D_{s,t}$. The goal in maximimum-concurrent-flow is to compute, for an input DM D, the TE configuration $\mathcal{R}$ for which $\mathcal{L}(\mathcal{R}, D) = \alpha(\mathcal{R}, D)$ is maximized. Relative to maximum-multicommodity-flow above, the maximum-concurrent-flow objective enhances fairness. Practical TE systems [22, 24] use a sequence of optimizations wherein they employ different objectives for different priority classes. For example, they may use maximum-multicommodity-flow or minimizing MLU for high priority traffic and maximum-concurrent-flow for scavenger-class traffic.

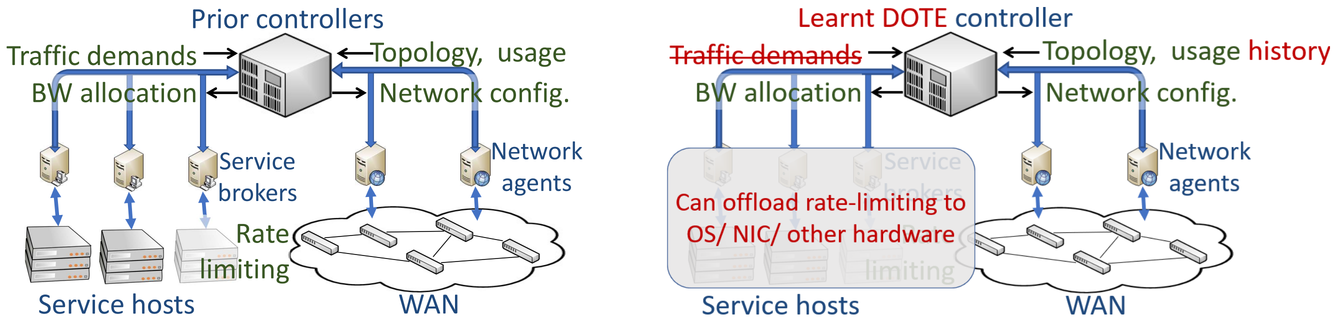
***DOTE* for maximum-multicommodity-flow and maximum-concurrent-flow.** Adapting *DOTE* to the above two flow-maximization objectives is accomplished along the lines described in §3.4. In particular, a DNN is again utilized to map the recent observations of DMs to the next TE configuration. Recall from the above discussion that the (revised) TE configuration consists of both traffic splitting ratios across tunnels and a "flow cap" for each tunnel. In our design, the DNN outputs $w_p \geq 0$ for each tunnel p. The w_p's are used to derive traffic splitting ratios and flow caps as follows. We set $\omega_p = \frac{w_p}{\gamma}$, where $\gamma = \max\left(\max_{e \in E} \frac{\Sigma_{p: e \in p} w_p}{c(e)}, 1\right)$. Observe that this guarantees that no link capacity can be exceeded even if each tunnel p carries its maximum permissible flow ω_p (i.e., that $\Sigma_{s,t \in V, p \in P_{st}^e} \omega_p \leq c(e)$). We then set the traffic split share on tunnel p to simply be its proportional weight: $x_p = \frac{\omega_p}{\Sigma_{q \in P_{st}} \omega_q}$. Since the objective is now *maximizing* a performance metric, *DOTE* now involves stochastic gradient *ascent*.

Optimality via stochastic quasi-concave optimization. In Appendix B, we prove the analogues of Theorem 1 for maximum-multicommodity-flow and for maximum-concurrent-flow, establishing *DOTE*'s optimality for these two objectives. Similarly to Theorem 1 (for MLU), this implies that with sufficient training data and a sufficiently expressive decision model, *DOTE* attains near-optimal TE configurations. Our evaluation results for maximum-multicommodity-flow and for maximum-concurrent-flow exemplify this (§4.3).

Our proofs for maximum-multicommodity-flow and for maximum-concurrent-flow are considerably more subtle than that of Theorem 1, as both objectives are not concave (the analogue of convexity for maximization problems). Instead, we show that the average maximum-multicommodity-flow / maximum-concurrent-flow score of a TE configuration over any set of DMs is *quasi*-concave. This result, which may be of independent interest, allows us to leverage the analytical arguments in [20] to show convergence of a suitable stochastic gradient ascent algorithm to the global optimum, and bound the number of required iterations.

3.6 Realizing *DOTE*

Figure 3 illustrates key differences between *DOTE* and prior software-defined WAN TE schemes. One key difference is the use of historical traffic demands and a learnt controller instead of running an optimization solver, leading to substantial decrease in deployment overheads and runtimes. In particular, bandwidth brokers are no longer needed to estimate application demands. Furthermore, rate allocations can, if necessary, be enforced by piggy-backing on novel traffic



(a) Architecture of prior SD-WAN TE schemes [22, 24].

(b) *DOTE* with differences shown in red.

Figure 3: Illustration of the key differences from previous SD-WAN TE schemes.

shaping techniques that are deployed in modern cloud servers at the OS-level as well as in NIC/FPGA offloads [1, 12, 43, 47].

Training *DOTE*. Since *DOTE*'s decision model is trained *offline* on historical data, its operational model can be periodically replaced by a model trained in the background on more recent and up to date data, to gracefully adapt to *planned* changes in WAN topology (adding capacity, planned addition/removal of nodes or links) and to temporal drifts in traffic demand distributions. Our evaluation results (§4) indicate that *DOTE* produces high performance TE configurations even weeks after being deployed, and even if the network topology changes during this time (e.g., due to failures). This provides ample time for training substitute TE functions (a process that requires less than a day on large networks for our empirical datasets without code and hardware optimizations).

Handling network failures. Tunnelling protocols (e.g., MPLS) identify tunnels with failed nodes/links. A traditional approach in TE to rerouting traffic around failed tunnels is to let traffic sources redistribute traffic proportionally among their remaining tunnels [22, 24, 39, 52].[7] We incorporate this simple approach into *DOTE* and evaluate its effectiveness in §4), showing that it achieves high resiliency to failures. We discuss other possible approaches in §5.

4 Evaluation

Using actual traffic demands from three different production WANs (Abilene, GEANT, and Microsoft's WAN), we ask the following questions: (1) How does *DOTE* compare against an omniscient oracle with perfect knowledge of future demands? (2) How does *DOTE* compare with state-of-the-art prediction-based TE [4, 22, 24, 36, 40], demand-oblivious TE [8, 32], and RL-based TE [54]? (3) Can *DOTE* support different TE objectives (e.g., MLU [8], maximum-multicommodity-flow [4, 22, 24])? (4) How long does *DOTE* take to train and to apply online at each solver activation? (5) How does *DOTE* perform under network faults and drift in traffic patterns?

	#Nodes	**#Edges**	**Length**	**Granularity**
Abilene	11	14	4.5months	5 min.
GEANT	23	37	4 months	15 min.
PWAN	O(100)	O(100)	O(1) months	minutes
$PWAN_{DC}$	O(10)	O(10)	O(1) months	minutes
GtsCe	149	193	Synthetic	
Cogentco	197	243		
KDL	754	895		

Table 1: Datasets used to evaluate *DOTE*

4.1 Methodology

Datasets: Data-driven TE is best evaluated on *real-world* datasets; we use the production topology and the traffic demands from GEANT [53], Abilene [3], and PWAN, a private WAN at Microsoft. Traffic traces were collected at a few-minute granularity over several months. We also use three topologies (GtsCe, Cogentco and KDL) from Topology Zoo [31] with synthetic traffic (generated using the gravity model [8, 45]). Table 4 lists the topology sizes and traffic demands. Nodes in these WAN topologies are datacenters, edge sites, or peering points. Traffic on PWAN includes both traffic between datacenters and traffic to/from end users. To better understand how *DOTE* performs for each traffic class, we consider a subset–PWAN_{DC}–which only contains large datacenters and the traffic between them. For each WAN, we use the earlier 75% of demand matrices (DMs) for training and the later 25% DMs as the test set.

Tunnel choices are k-shortest-paths, edge-disjoint paths, and SMORE trees. More specifically, we use (1) Yen's algorithm for k-shortest-paths, with $k = 8$ per commodity (pair of nodes), (2) edge-disjoint shortest-paths where, for each commodity, we iteratively compute a shortest-path in the network and remove all links on that path from consideration until no more paths exist for that commodity, and (3) tunnels from SMORE [37] generated using Yates [36].

Comparables to *DOTE* include: (1) **Omniscient oracle**, which is an optimization with perfect knowledge of future demands and bounds the quality of *any* WAN TE scheme. (2) **Demand-prediction-based TE** methods [4, 22, 24, 36, 40], which are in production today [22, 24]. We consider a rich

[7] Traffic split $(0.6, 0.3, 0.1)$ becomes $(0, 0.75, 0.25)$ if the first path goes down.

WAN	Online Lat. (s)		Precomp. Lat. (s)		
	DOTE	LP	*DOTE*	COPE	Oblivious
Abilene	0.0005	0.02	1800	180	1
PWAN_{DC}	0.003	0.05	1200	7200	15
Geant	0.002	0.04	2400	10800	180
PWAN	0.2	1	36000	> 345600	~ 86400

Table 2: Comparing the online latency (to compute a TE configuration for a demand matrix) as well as the precomputation latency (to train models, to compute demand-oblivious configurations, etc.) for various TE schemes. 8 shortest paths are used per demand across all WANs and TE schemes.

collection of possible predictors of future demands: linear regression, ridge regression, random forest, DNN models, and autoregressive models (§C). (3) **RL-based** WAN TE [54], which leverages a neural network of the same size as *DOTE*'s (see below). (4) **Demand-oblivious TE** [8], which optimizes the *worst-case performance* over *all* traffic demands. (5) **SMORE** [37], which picks source-rooted trees for worst-case demands but adapts splitting ratios over the chosen trees based on *predicted* future demands. (6) **COPE** [55], which enhances demand-oblivious schemes by also optimizing over a set of predicted traffic demands.

Metrics: Our TE quality metric is the ratio between the value obtained by the evaluated TE scheme and the performance obtained by the omniscient oracle, which has perfect information about future traffic demands. We consider three TE objectives: minimize maximum link utilization (MLU) [8, 14, 27], maximize multicommodity flow [4, 22, 24, 37] and maximize concurrent-flow [11, 29]. Note that this ratio is ≥ 1 for MLU (because lower max-link utilization is better) and ≤ 1 for the other metrics (because carrying more flow is better). We refer to the relative gap from 1 as the *optimality gap*. We also measure the runtimes (latency) of the evaluated TE schemes on the same physical machine.[8]

DNN architecture: Unless otherwise specified, results for *DOTE* use five fully connected NN layers with 128 neurons each and $ReLU(x)$ activation except for the output layer which uses $Sigmoid(x)$. For different TE objectives, *DOTE* uses a similar architecture with small changes. We chose this architecture because it empirically outperformed other investigated architectures.

Infrastructure and code: We ran our experiments in cloud VMs and made use of cloud ML training systems. To enable further research, we have released our code at [2].

Fault model: To examine TE behaviour under network faults, we randomly bring down a certain number of links (e.g., 1 to 20 while ensuring network is not partitioned), and compare the performance of *DOTE* (see *DOTE*'s failure-recovery scheme at the end of §3.6) and alternatives with an omniscient oracle with perfect knowledge of both future failures and future traffic demands.

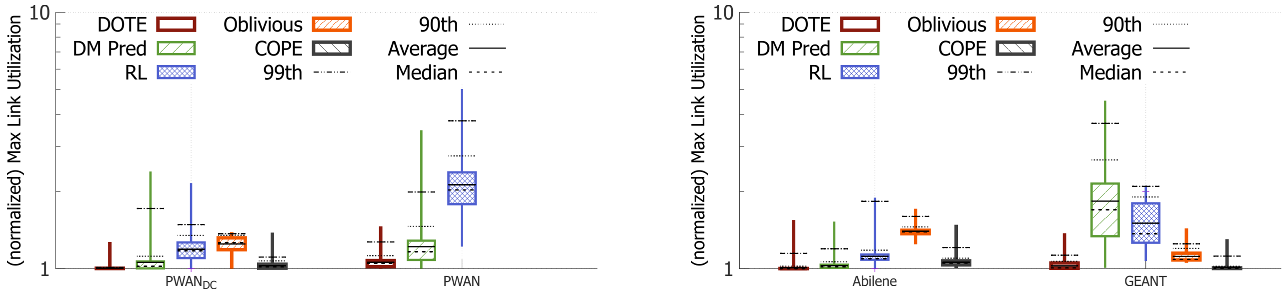
4.2 Comparing *DOTE* with Other TE Schemes

TE quality. Figure 4 compares *DOTE* with the other TE schemes described in §4.1, with the exception of SMORE (to be discussed in §4.3). The values plotted here are the maximum link utilization (MLU) normalized by that of the ominiscient oracle with perfect knowledge of future demands. The figure shows results on four different topologies. Each candlestick shows the distribution of MLUs achieved on the various demand matrices with the boxes ranging from 25th to 75th percentile and the whiskers going from minimum to maximum value. The figure also plots values achieved at various other percentile values. We note a few findings.

- First, optimizing for *predicted* demands can lead to poor TE quality (see results for GEANT and PWAN). Note that the y axis is in log scale. A value of $y = 2$ indicates that the link most utilized by the TE scheme is twice as utilized as the most utilized link in the optimal solution (produced by the oracle). Optimizing with respect to predicted demands performs well only on Abilene and PWAN_{DC}, where the traffic demands are predictable. These results are for a linear-regression-based predictor that outperforms all other considered predictors on our real-world traffic datasets (see Appendix C).
- Next, we observe that the RL-based TE scheme [54] has extremely poor TE quality even on Abilene. This could be due to the infamous training complexity of RL.
- Third, demand-oblivious TE [8] results in somewhat decent TE quality on GEANT but not on any of the other WANs. This could be because optimizing *worst-case* performance across *all* possible demands fails to take advantage of the specific characteristics of real-world traffic demands.
- Fourth, COPE [55], which explicitly accounts for historically observed demands, significantly outperforms demand-oblivious TE. The key issue with COPE is its extremely high runtimes. Our analysis (see Table 2 and discussion below) suggests that COPE's applicability does not extend beyond topologies with tens of nodes.
- Finally, note that *DOTE* achieves TE quality that is almost always significantly better than the alternatives' and nearly as good as the omniscient oracle's. The difference in TE quality is especially stark at the higher percentiles. Relative to the compared TE schemes, *DOTE* offers MLU up to 25% better at the median and 170% better at the 99th percentile.

Runtimes. Table 2 presents a comparison of runtimes across TE schemes. The table presents the latency of applying each TE scheme to a new demand matrix and, wherever appropriate, the required precomputation time. Demand-oblivious schemes [8] and COPE [55] do not change the TE configuration online but involve very long precomputation latency and require very large memory. *DOTE* performs both precomputation

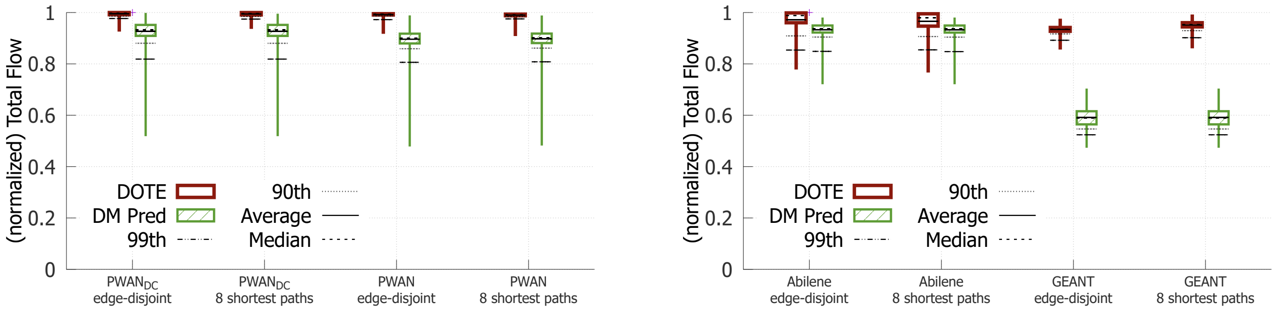
[8] VM with 8 vCPUs and 256GB RAM.



(a) *PWAN$_{DC}$* & PWAN

(b) Abilene & GEANT

Figure 4: TE quality when aiming to minimize the maximum link utilization with 8 shortest paths per demand. Candlesticks depict results across hundreds of demands; the boxes are from the 25th to the 75th percentile, the whiskers range from min to max value, dashed lines capture other percentiles of interest. *DOTE* achieves much lower MLU compared to the alternatives.



(a) PWAN$_{DC}$ & PWAN

(b) Abilene & GEANT

Figure 5: TE quality when aiming to maximize total flow with two different tunnel choices.

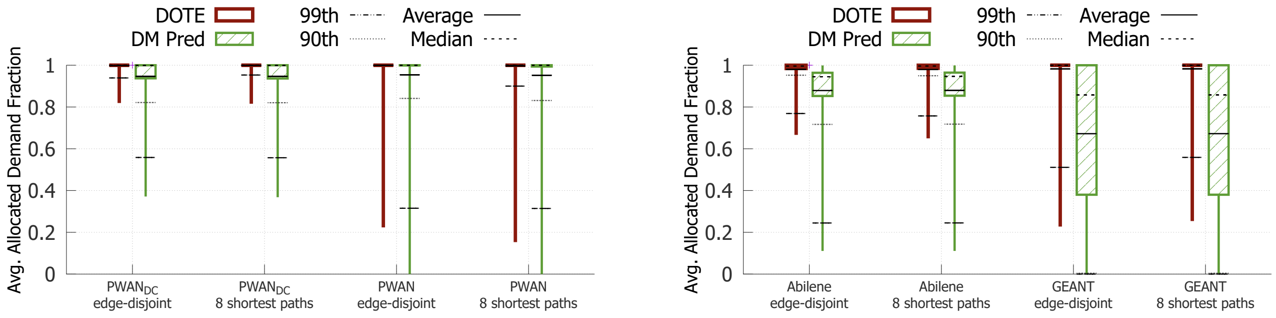
tation on historical demands (training the DNN) and online computation (invoking the DNN). SMORE's online computation involves solving an LP to optimize over predicted demand matrices and so its latency is roughly as high as the LP's latency in the table. To compute Racke's routing trees, SMORE requires several hours on the larger topologies.

The table shows that *DOTE*'s inference time is faster than the latency of using LPs to optimize over one (predicted) DM. The LP's latency is on par with results in recent studies [4, 40]. *DOTE*'s online computation is short because it is effectively a few matrix multiplications.[9] LP computation latency increases super-linearly with the network size and prior work notes that solver times can exceed several minutes on networks with thousands of nodes and edges [4, 40]; *DOTE*'s inference latency on large WANs, such as KDL (see Table 4), is still within a few seconds. *DOTE*'s training time is less than 12 hours for PWAN and can be accelerated using standard methods (e.g., by parallelization, SIMD and other model training enhancements).

COPE's precomputation latency is a few orders of magnitude higher than that of the demand-oblivious TE, which is, itself, a couple orders of magnitude higher than that of prediction-based TE. COPE also has much higher memory requirements (over 256GB on PWAN); in fact, on PWAN, COPE did not finish pre-computation even after four days on a 8-core VM with 256GB running Gurobi [19] vers. 9.1, and hence Figure 4 includes no results for COPE on PWAN. To understand COPE's runtime complexity better, we ran it on WAN topologies from Topology Zoo [31] that are larger than GEANT and PWAN$_{DC}$ but smaller than PWAN. On Janet-Backbone which has 29 nodes and 45 edges, COPE ran for 1.5 hours and on SurfNet (50 nodes, 68 edges), COPE did not finish even in 10 hours. These results suggest that COPE is inapplicable to large WANs.[10]

[9] Input is 12 demand matrices and output is splitting ratios or one double per tunnel per demand. On the large PWAN network, both the input and output are a few tens of MBs.

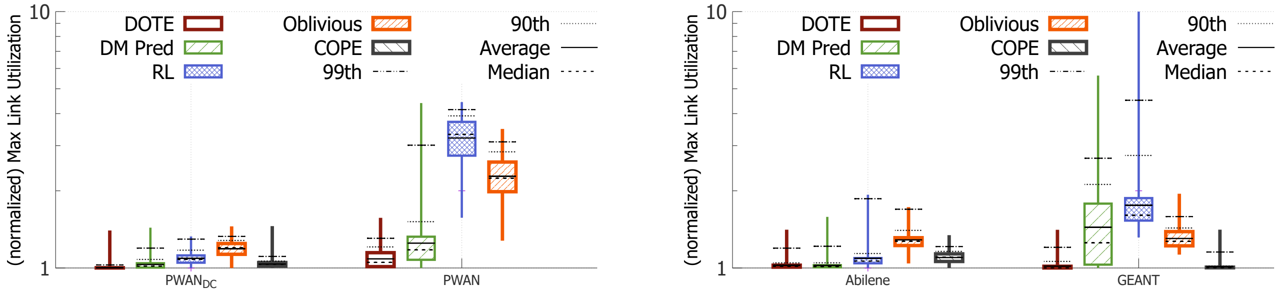
[10] Per Table 1 in [55], the previously published results on COPE are on much smaller topologies than considered here.



(a) PWAN$_{DC}$ & PWAN

(b) Abilene & GEANT

Figure 6: TE quality when aiming to maximize the concurrent flow for two different tunnel choices. For each demand matrix, we compute the fraction of demand satisfied for each source and destination, and sort these values into a vector. Across many hundreds of demand matrices, the candlesticks plot the average over all such allocation vectors. Note: allocating more flow is better. The box in each candlestick is the 25th and 75th percentile (fractional allocation) and the whiskers go from min to max value.



(a) PWAN$_{DC}$ & PWAN

(b) Abilene & GEANT

Figure 7: TE quality when aiming to minimize MLU with all possible edge-disjoint paths.

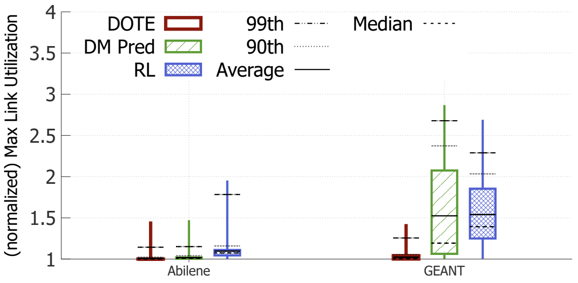


Figure 8: TE quality when aiming to minimize MLU with routing trees chosen by SMORE.

4.3 Generalizing to Other TE Objectives and Tunnel Choices

Here, we present results for two additional TE objectives – maximizing multi-commodity-flow and maximizing concurrent flow– as well as two other choices for tunnels.

Note that some of the compared alternatives to *DOTE*, namely, demand-oblivious TE [8] and COPE [55], do not readily apply to these TE objectives (as both build on results from oblivious routing theory that provide provable guarantees for MLU minimization), and it is not clear how to extend them to other objectives. Our evaluation of *DOTE* for these metrics is therefore restricted to benchmarking against the omniscient oracle and prediction-based TE.

Maximizing Total Flow: Figure 5 compares *DOTE* with prediction-based TE on all four WANs for two different tunnel choices when the TE objective is to carry as much total flow as possible while respecting capacity constraints. Observe that *DOTE* carries substantially more flow and closely approximates the TE quality of the omniscient oracle. As before, the gap between *DOTE* and prediction-based TE is larger on WANs where demands are less predictable (i.e., all WANs but Abilene) and at the higher percentiles. Generally, *DOTE* may be able to carry 10% to 20% more flow.

Maximizing Concurrent Flow: Figure 6 compares *DOTE* with the omniscient oracle and prediction-based TE when the TE objective is to maximize the minimum fraction of demand satisfied across all demands. Observe that *DOTE* fully allocates almost all of the demands (the upper candlesticks are at $y = 1$), whereas prediction-based TE allocates a smaller fraction of the demanded volume for many more demands.

Tunnel choice does not qualitatively change our results for TE performance; contrast Figure 7 and Figure 8 with Figure 4. Note that when using Racke's routing trees (as in SMORE) prediction-based TE coincides with SMORE.

4.4 Coping with Network Failures

Figure 9 shows how *DOTE* performs, in terms of MLU, when different numbers of (randomly chosen) links fail in the PWAN topology. As noted in §3.6, *DOTE* assumes that source nodes (or tunnel heads [44]) identify tunnels that fail and rebalance traffic proportionally among the surviving tunnels. The figure compares *DOTE* with two variants of prediction-based TE: DM Pred. which, similar to *DOTE*, has no a priori knowledge of future traffic demands or the link faults, and FA DM Pred. which is identical to DM Pred. except that it is fault-aware, i.e., knows the links that will fail. Our quality metric is still normalized MLU except that we now normalize based on an omniscient oracle that has perfect knowledge of *both* future traffic demands *and* the failures.

Our results show that *DOTE* outperforms both demand-prediction-based TE (DM Pred.) and demand-prediction-based TE with oracle access to future failures (FA DM Pred.) for many concurrent link failures with different tunnel choices. We interpret this result as indicating that the error in demand predictions weights more heavily on attaining a good TE objective than the confusion induced by these link failures. Our results on other topologies (Abilene, GEANT, and $\mathrm{PWAN}_{\mathrm{DC}}$) and for the maximum-multicommodity-flow objective show a similar trend (Figure 13 and Figure 14).

4.5 Robustness to Traffic Noise and Drift

Robustness to unexpected traffic changes. To assess *DOTE*'s robustness to noisy traffic, we evaluate *DOTE* on the GEANT, Cogentco, and GtsCe WANs [31], where each demand in the realized DM is independently multiplied by a factor chosen uniformly at random from $[1-\alpha, 1+\alpha]$ for $\alpha \in \{0.1, 0.25, 0.35\}$. Our results (see §D) show that under such traffic perturbations, the distance, in terms of MLU, from the omniscient oracle remains low across all evaluated WANs (e.g., 2%, 2.9%, and 3.8% for $\alpha = 0.1, 0.25, 0.35$ for GEANT with edge-disjoint tunnels).

Robustness to natural traffic drift. We investigate to what extent the quality of *DOTE*'s TE configurations deteriorates when *DOTE* is not frequently retrained. We quantify the distance from the omniscient oracle, in terms of both MLU and maximum-multicommodity-flow, of the average *weekly* value achieved by *DOTE* on the Abilene and GEANT WANs over 4 consecutive weeks (without retraining *DOTE*). See Table 3 and Table 4 in the Appendix. Our results show that while the distance from the optimum increases over time, in general, *DOTE* remains close to the optimum (within a few % on average) even weeks after the model is trained. This suggests that *DOTE* can provide high quality TE even if it was re-trained once every month. *DOTE*'s training time (see Table 2) allows for much more frequent retraining.

5 Limitations and Future Research

We believe that our investigation of direct optimization for WAN TE has but scratched the surface and outline below current limitations of our approach, as well as intriguing directions for future research.

Extending *DOTE* to support latency-sensitive traffic. To accommodate latency-sensitive traffic, the following strategy (similarly to [34]) could be employed: reserve shortest paths (tunnels) for such traffic and always schedule short/latency-sensitive traffic flows to these paths.

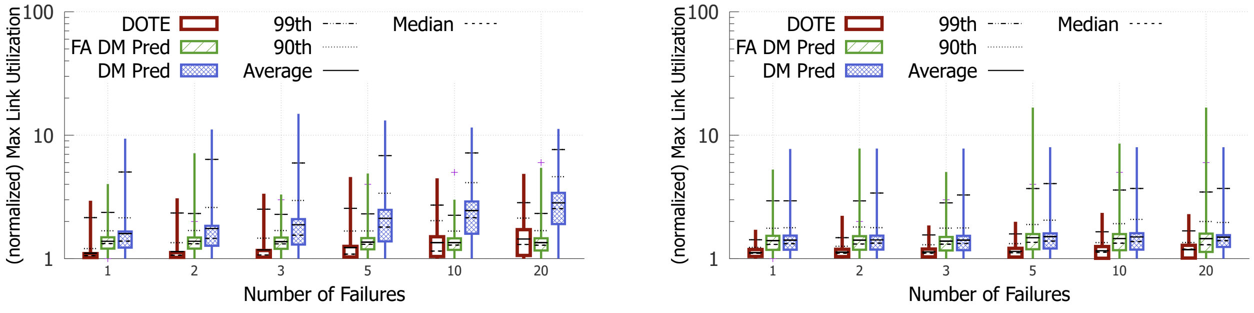
More expressive neural network architectures. Our realization of *DOTE* uses a relatively simple neural network that does not leverage knowledge of the WAN topology. Consequently, the neural network has to (implicitly) learn the network topology during training. Directly incorporating the WAN structure into *DOTE* using Graph Convolutional Networks [56] could potentially lead to faster training and/or better quality solutions.

Extending *DOTE* to incorporate data-driven tunnel selection. Our discussion of *DOTE* assumed an underlying tunnel-selection scheme. *DOTE* can be extended to support *data-driven* tunnel-selection by adding DNN output variables specifying a probability distribution over a finite set of candidate tunnels (e.g., shortest-path, edge disjoint, SMORE). At the beginning of each time epoch, the tunnels to be used in that time epoch would be chosen according to this probability distribution. *DOTE*'s optimality results extend to this setting. We defer a more thorough study of data-driven tunnel selection (e.g., not limited to a finite set of predetermined candidate tunnels) to future research.

Learning to contend with link failures. We described (§3.6) an approach for dealing with link failures in the data plane. An alternative is incorporating fault tolerance into the DNN training process by introducing random link failures.

6 Related Work

(WAN) TE. TE has been extensively studied [5, 7, 10, 11, 14, 16, 22, 24, 26–28, 37, 39, 57, 59] in a broad variety of settings, including legacy networks [13, 17], datacenter networks [6],



(a) 8SP

(b) edge-disjoint

Figure 9: Coping with different numbers of random link failures on PWAN; the candlesticks show the distribution over 1700 different randomly chosen failure cases.

and backbone networks [27]. SDN-controlled WAN TE has also received extensive attention [11, 22, 24, 35, 37–39, 59].

TE via oblivious routing, COPE, and SMORE. Oblivious routing optimizes *worst-case MLU* across *all* possible DMs [8, 9, 42]. Since oblivious routing does not exploit *any* information about past traffic demands, it naturally yields suboptimal solutions [8, 37]. COPE [55] optimizes *MLU* across a set of DMs spanned by previously observed DMs, while retaining a worst-case performance guarantee. Since COPE both extends oblivious routing *and* optimizes over *ranges* of demand matrices, its optimization phase is extremely time-consuming (§4.2). The key conceptual difference between *DOTE* and such "robust TE" schemes is in the goal of the pre-computation. Instead of emitting a single TE configuration that minimizes some cost function (specifically, MLU) over some predetermined set of DMs, DOTE's objective is to identify a mapping from a vector of DMs from the recent past to the next TE configuration. DOTE thus achieves higher flexibility by being able to emit different TE configurations on a case-by-case basis, and is also able to pick up on temporal patterns in traffic demands. SMORE [37] employs Racke's oblivious routing trees [42] to produce static tunnels that are robust to traffic uncertainty, with traffic splitting ratios still optimized with respect to the (inferred/predicted) future traffic demands. Thus, SMORE can be thought of as a instantiation of prediction-based TE.

Online TE [14, 15, 27], wherein traffic configurations (such as splitting ratios) adapt automatically and in short timescales to the observed demands is an enticing design point for TE, but is challenging to achieve. TexCP [27] requires WAN routers to offer novel explicit feedback, while MATE [14] relies on changes in end-to-end latency and hence takes much longer to react and converge and is also less stable [27]. Recently deployed TE schemes [22, 24] (see §2 and Figure 3) are simpler and easier to deploy because they replace such distributed, closed-loop, short-timescale control with centralized, open-loop and periodic adaptation. We view online TE as complementary to *DOTE*; *DOTE* could be used to *periodically* compute a TE configuration while online TE could be *continuously* used in between *DOTE* updates to tweak this TE configuration in response to changes in network conditions.

Reinforcement-learning-based TE. Demand-prediction-based and RL approaches to TE are contrasted in [54] in terms of MLU only on a small network (12 nodes and 32 edges) for *synthetic* traffic patterns and a model of hop-by-hop routing that does not capture routing along tunnels. Our theoretical and empirical results reveal that *DOTE*'s stochastic optimization scheme outperforms both demand-prediction-based and RL-based TE.

Some recent work on TE [4, 40] speeds up the multicommodity flow computations that underpin TE optimization by effectively breaking the large LPs into smaller problems that can be solved in parallel. However, these approaches still rely on predicted demand matrices (unlike *DOTE*). *DOTE* offers an alternate way to speed up TE: replacing the LP solver with invocations of a fairly small DNN. This has the potential to be innately more efficient.

7 Conclusion

We presented a new framework for WAN TE: data-driven end-to-end stochastic optimization using only historical information about traffic demands. Our theoretical and empirical results establish that this approach closely approximates the optimal TE configuration, significantly outperforming previously proposed TE schemes in terms of both solution quality and runtimes.

Acknowledgements: We thank our shepherd, Mojgan Ghasemi, and the NSDI reviewers, for their valuable feedback. We thank Umesh Krishnaswamy, Himanshu Raj and the SWAN team at Microsoft for their help and feedback. Yarin Perry and Michael Schapira were partially supported by BSF grant 2019798 and a grant from Microsoft. Aviv Tamar is funded by ERC grant 101041250.

References

[1] Google cloud armor: Rate limiting overview. https://bit.ly/3TnI1mO.

[2] *Github repo containing our code*. 2022. https://github.com/PredWanTE/DOTE.

[3] Abilene/Internet2. http://www.internet2.edu/.

[4] Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, and Peter Bailis. Contracting wide-area network topologies to solve flow problems quickly. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 175–200, 2021.

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

[6] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, 2010.

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

[8] David Applegate and Edith Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands. In *SIGCOMM*, 2003.

[9] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 383–388, 2003.

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

[11] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. TEAVAR: striking the right utilization-availability balance in WAN traffic engineering. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019*, pages 29–43, 2019.

[12] Adrian M. Caulfield, Eric S. Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, Daniel Lo, Todd Massengill, Kalin Ovtcharov, Michael Papamichael, Lisa Woods, Sitaram Lanka, Derek Chiou, and Doug Burger. A cloud-scale acceleration architecture. In *MICRO*, 2016.

[13] Marco Chiesa, Gábor Rétvári, and Michael Schapira. Lying your way to better traffic engineering. CoNEXT, 2016.

[14] A. Elwalid, C. Jin, S. Low, and I. Widjaja. Mate: Mpls adaptive traffic engineering. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1300–1309 vol.3, 2001.

[15] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. Replex: Dynamic traffic engineering based on wardrop routing policies. In *Proceedings of the 2006 ACM CoNEXT Conference*, 2006.

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

[17] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 2004.

[18] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

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

[20] Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

[21] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[22] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. SIGCOMM, 2013.

[23] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B.,

Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. *SIGCOMM '18*, pages 74–87, 2018.

[24] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a globally-deployed software defined wan. SIGCOMM, 2013.

[25] William S. Jewell. *Multi-commodity Network Solutions*. 1966.

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

[27] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *SIGCOMM*. ACM, 2005.

[28] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. Calendaring for wide area networks. In *SIGCOMM*, 2014.

[29] George Karakostas. Faster Approximation Schemes for Fractional Multicommodity Flow Problems. *ACM Trans. Algorithms*, 2008.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization learning. *arXiv preprint arXiv:1412.6980*, 2014.

[31] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 2011.

[32] M Kodialam, T V Lakshman, and S Sengupta. Traffic-oblivious routing in the hose model. *IEEE/ACM Transactions on Networking*, 19(3):774 – 787, 2011.

[33] Igor V Konnov. On convergence properties of a subgradient method. *Optimization Methods and Software*, 18(1):53–62, 2003.

[34] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized cloud wide-area network traffic engineering with BlastShield. Technical Report MSR-TR-2021-31, Microsoft Research, 2021.

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

[36] Praveen Kumar, Chris Yu, Yang Yuan, Nate Foster, Robert Kleinberg, and Robert Soulé. Yates: Rapid prototyping for traffic engineering systems. In *Proceedings of the Symposium on SDN Research*, SOSR '18, pages 11:1–11:7, New York, NY, USA, 2018. ACM.

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

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

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

[40] Deepak Narayanan, Fiodar Kazhamiaka, Firas Abuzaid, Peter Kraft, Akshay Agrawal, Srikanth Kandula, Stephen Boyd, and Matei Zaharia. Solving large-scale granular resource allocation problems efficiently with POP. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 521–537, 2021.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[42] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, 2002.

[43] Sivasankar Radhakrishnan, Yilong Geng, Vimalkumar Jeyakumar, Abdul Kabbani, George Porter, and Amin Vahdat. Senic: Scalable nic for end-host rate limiting. In *NSDI*, 2014.

[44] E. Rosen, A. Viswanathan, and R. Callon. Multi-Protocol Label Switching Architecture. RFC 3031.

[45] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. IMW, 2002.

[46] Matthew Roughan, Mikkel Thorup, and Yin Zhang. Performance of estimated traffic matrices in traffic engineering. In *SIGMETRICS*, 2003.

[47] Ahmed Saeed, Nandita Dukkipati, Vytautas Valancius, Vinh The Lam, Carlo Contavalli, and Amin Vahdat. Carousel: Scalable traffic shaping at end hosts. In *SIGCOMM*, 2017.

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

[49] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[50] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, volume 2, page 5, 2009.

[51] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

[52] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. Network architecture for joint failure recovery and traffic engineering. In *Proceedings of the 2011 ACM SIGMETRICS Conference*, 2011.

[53] Steve Uhlig, Bruno Quoitin, Jean Lepropre, and Simon Balon. Providing public intradomain traffic matrices to the research community. *SIGCOMM Comput. Commun. Rev.*, 36(1):83–86, jan 2006.

[54] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, 2017.

[55] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. Cope: Traffic engineering in dynamic networks. In *SIGCOMM*, 2006.

[56] Zonghanu Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596, 2019.

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

[58] Yin Zhang, M. Roughan, C. Lund, and D.L. Donoho. Estimating point-to-point and point-to-multipoint traffic matrices: an information-theoretic approach. *IEEE/ACM Transactions on Networking*, 13(5):947–960, 2005.

[59] Zhizhen Zhong, Manya Ghobadi, Alaa Khaddaj, Jonathan Leach, Yiting Xia, and Ying Zhang. ARROW: restoration-aware traffic engineering. In *ACM SIGCOMM 2021 Conference, Virtual Event, USA, August 23-27, 2021*, pages 560–579, 2021.

Appendix

A Predictability of WAN TE Traffic

Figure 10(a) plots the inter-data-center traffic demand between the pair of data centers with the highest average demand over the course of a week. Similarly, Figure 10(b) plots the normalized volume of customer-facing traffic for the pair of nodes with the highest average demand over the course of a week. Demands are shown at 5-minute granularity and are normalized by the peak demand. As shown in Figure 10(a), inter-data-center traffic demands exhibit very distinct diurnal and hourly patterns. Indeed, the figure also presents the predictions of a linear regression model trained on data from the 3 preceding weeks, which takes as input the traffic demands observed in the previous hour (at 5-minute granularity), and outputs the predicted traffic demand for the upcoming 5 minutes. In contrast, the predictions of a linear regressor for customer-facing traffic, as shown in Figure 10(b), are quite often far from the actual traffic demands.

B Analytical Results

B.1 Minimizing Max-Link Utilization

We next prove that, for an infinitely expressive TE function, i.e., when each history of DMs can be *independently* mapped to a TE configuration, and in the limit of infinite empirical data sampled from the underlying Markov process' stationary distribution, *DOTE* attains optimal performance. This establishes that our approach is *fundamentally sound*, and so high performance in practice can be achieved by acquiring sufficient empirical data and employing a sufficiently expressive decision model (e.g., a deep enough neural network).

For the sake of analysis, we make the following simplifying assumptions. We first assume that the set of possible history realizations, which we denote by $\mathbf{H}$, is finite. Let D_{max} denote an upper bound on the maximum traffic demand between a source-destination pair, c_{min} denote the minimum link capacity, and p_{max} denote the maximum number of tunnels interconnecting a source-destination pair. Note that any valid TE configuration specifies, for each source-destination pair, a point in the p_{max}-dimensional simplex (specifying its splitting ratios across at most p_{max} tunnels); let $\mathbf{R}$ denote the space of valid TE configurations. Let $\pi : \mathbf{H} \rightarrow \mathbf{R}$ denote a mapping from history to TE configuration. π can be represented as a vector with $|\mathbf{H}| \times n^2 \times (p_{max} - 1)$ components.[11] Since each element in this vector is itself a vector in the p_{max}-dimensional simplex, we have that $\|\pi\| \leq \sqrt{|\mathbf{H}| n^2 (p_{max} - 1)} \doteq B$, where $\|\cdot\|$ is the Euclidean norm. We make the following observation.

Proposition 1. *The loss function $\mathcal{L}(\pi(D_{t-1}, \ldots, D_{t-H}), D_t)$ is convex in π and ρ-Lipschitz, with $\rho = D_{max}/c_{min}$.*

Proof. f_e is, by definition, linear in the traffic splitting ratios and so in π. Since the max is a convex function, we have that $\mathcal{L}$ is convex in π. Similarly, since each component in $\frac{f_e}{c(e)}$ is D_{max}/c_{min}-Lipschitz, the maximum is also D_{max}/c_{min}-Lipschitz. □

We now consider an idealized stochastic gradient descent (SGD) algorithm where at each iteration k we sample $D_t, D_{t-1}, \ldots, D_{t-H}$ from the probability distributions $P(D_{t-1}, \ldots, D_{t-H})$ and $P(D_t | D_{t-1}, \ldots, D_{t-H})$, and update $\pi_{k+1} = Proj\{\pi_k - \eta v_k\}$, where $v_k \in \partial \mathcal{L}(\pi_k(D_{t-1}, \ldots, D_{t-H}), D_t)$ denotes a subgradient of the objective function[12], and $Proj$ denotes a projection onto the simplex for each (s,d) pair. The final output after K iterations is $\bar{\pi} = \frac{1}{K} \sum_{k=1}^{K} \pi_k$.

The next theorem, based on Theorem 14.12 in [49], bounds the loss of this algorithm. Let $\bar{\mathcal{L}}(\pi) = \mathbb{E}[\mathcal{L}(\pi(D_{t-1}, \ldots, D_{t-H}), D_t)]$ denote the expected loss of a TE function, and let $\pi^* \in \arg\min_\pi \bar{\mathcal{L}}(\pi)$ denote the optimal TE function.

Theorem 2. *For every $\varepsilon > 0$, if SGD is run for $K \geq \frac{B^2 \rho^2}{\varepsilon^2}$ iterations with $\eta = \sqrt{\frac{B^2}{\rho^2 K}}$, then the output of SGD satisfies*

$$\mathbb{E}\left[\bar{\mathcal{L}}(\bar{\pi})\right] \leq \bar{\mathcal{L}}(\pi^*) + \varepsilon,$$

where the expectation is w.r.t. the sampling by the algorithm.

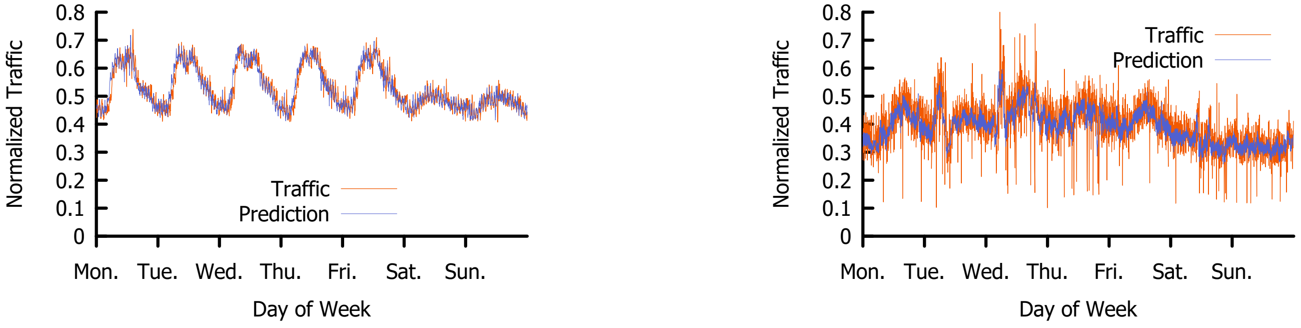
Theorem 2 shows that without function approximation (the TE function space spans *all* possible mappings from history to TE configuration), and with infinite data (the algorithm continuously samples from the true demand distribution), SGD converges to the optimal TE function with arbitrary precision. In practice, we relax both assumptions. In *DOTE* we sample from a large, but *finite*, dataset of historical demands, and use a *parametric* model (specifically, a neural network) to map from an *infinite* set of possible histories to valid TE configurations. Our empirical results show that, with enough data and a deep enough neural network, the approximate TE function *DOTE* learns is still very close to optimal.

B.2 Maximum-Multicommodity-Flow and Maximum- Concurrent-Flow

We begin by stating a general convergence result for quasi-convex functions that satisfy certain assumptions. We then proceed to show that both maximum-multicommodity-flow and maximum-concurrent-flow indeed satisfy these assumptions, implying their convergence.

[11] Note that we dropped the subscript θ in π, as in our analysis we consider the space of all possible TE configurations, and not a specific parametrization.

[12] The objective is not necessarily differentiable everywhere because of the max, but the subgradient exists for every π.



(a) Inter-data-center traffic

(b) Customer-facing traffic

Figure 10: Inter-data-center traffic and customer-facing traffic over the course of a week, along with the predictions of a linear regression model for the time-series.

B.2.1 General results

We begin by providing an analysis of stochastic quasi-convex optimization, under general assumptions. In the next section, we will show that maximum-multicommodity-flow and maximum- concurrent-flow are special cases of this setting.[13] Our analysis builds on two studies – the analysis of stochastic normalized subgradient of [20], which is for smooth and unconstrained problems, and the study of [33], which considered non-smooth quasi-convex optimization.

A quasi-convex function $f(x)$ satisfies that its level sets, $L(f;\alpha) = \{x | f(x) \leq \alpha\}$, are convex sets for all α.

We first define a normalized subgradient in the context of quasi-convex functions, following [33]. The normal cone to a convex set X at point x is defined by $N(X,x) = \{q \in \mathbb{R}^n | \langle q, y-x \rangle \leq 0 \quad \forall y \in X\}$. The set of subgradients at a point x are given by $N(L(f;f(x)),x)$. The set of normalized subgradients, $Q(f;x)$, at a point x, are given by $Q(f;x) = \mathbb{S}(0,1) \cap N(L(f;f(x)),x)$, where $\mathbb{S}(0,1)$ is the n-dimensional sphere of radius 1. These are directions of ascent – normalized vectors such that taking an infinitely small step in their direction is guaranteed to not decrease the function.

In the following, we consider a general stochastic optimization problem:

$$\min_{x \in X} \mathbb{E}_{D \sim P(D)} \left[f(x,D) \right], \tag{1}$$

where f is quasi-convex in x for every D.

We will further assume the following. Let $\mathbb{B}(z,r)$ denote the n-dimensional ball centered on z with radius r.

Assumption 1. *Set X is convex and bounded by $\mathbb{B}(0,\bar{B})$. The function f is bounded by B. It is also G-Lipschitz and quasi-convex in x for every D. Furthermore, $Q(\frac{1}{M}\sum_{i=1}^{M} f(x,D_i);x) \neq \emptyset$ for any $x \notin \arg\min_y \frac{1}{M}\sum_{i=1}^{M} f(y,D_i)$, and for every $D_1,\ldots,D_M$, we have that $\frac{1}{M}\sum_{i=1}^{M} f(x,D_i)$ is quasi-convex in x.*

Note that the last requirement in Assumption 1 is not immediate, as the sum of quasi-convex functions is not necessarily quasi-convex.

The stochastic normalized subgradient method we consider works as follows [20]. At each iteration k we sample a minibatch $\{D_i\}_{i=1}^{b} \sim P(D)$ and define $f_k = \frac{1}{b}\sum_{i=1}^{b} f(x,D_i)$. We then update $x_{k+1} = Proj\{x_k - \eta v_k\}$, where $v_k \in Q(f_k;x_k)$ denotes a subgradient of the minibatch, and $Proj$ denotes a projection onto the set X. The final output after K iterations is $\bar{x}_K = \arg\min_{x_1,\ldots,x_K} f_k(x_k)$.

The analysis in [20] bounds the error of the normalized subgradient method, for smooth and unconstrained functions. We next adapt it to our setting.

The next definition adapts a central definition from [20] to our non-smooth setting.

Definition 1. *(SLQC) Let $x, x^* \in \mathbb{R}^n$, $\kappa, \varepsilon > 0$. We say that f is (ε,κ,x^*)-strictly-locally-quasi-convex (SLQC) in x if at least one of the following applies. (1) $f(x) - f(x^*) \leq \varepsilon$. (2) $Q(f;x) \neq \emptyset$ and for any $\Delta \in Q(f;x)$, and every $y \in \mathbb{B}(x^*, \frac{\varepsilon}{\kappa})$, it holds that $\langle \Delta, y-x \rangle \leq 0$.*

We next show that the Lipschitz and quasi-convex properties in Assumption 1 suffice to establish SLQC.

Lemma 1. *Let f satisfy Assumption 1. Fix D, and let $x^* \in \arg\min_{x \in X} f(x;D)$. Then f is (ε,G,x^*)-SLQC for all $x \in X$.*

Proof. Assume $f(x;D) - f(x^*;D) > \varepsilon$. Let Z denote the $f(x;D)$-level set of $f(x;D)$. Let ∂Z be the boundary of Z. By definition of the level set, for every $z \in \partial Z$, $f(z) - f(x^*) > \varepsilon$. From the Lipschitz property then, for every $z \in \partial Z$ we must have $\|z - x^*\| \geq \frac{\varepsilon}{G}$. Since Z is convex, we therefore have that $\mathbb{B}(x^*, \frac{\varepsilon}{G}) \subset Z$. From Assumption 1, $Q(f;x) \neq \emptyset$, and from the definition of $Q(f;x)$, we have that for every $y \in \mathbb{B}(x^*, \frac{\varepsilon}{G})$, if $\Delta \in Q(f;x)$ then $\langle \Delta, y-x \rangle \leq 0$. □

[13] While we present results for quasi-convexity, the extension of these results to quasi-concave problems is immediate.

We next show that with high probability, the subgradient of each minibatch is a descent direction for the expected objective in (1).

Lemma 2. *Let Assumption 1 hold, and let* $x^* \in \arg\min_{x\in X} \mathbb{E}_{D\sim P(D)}\left[f(x,D)\right]$*. Assume that the minibatch size satisfies* $b = O\left(\frac{8nB^2\log(G\bar{B}/\delta)}{\varepsilon^2}\right)$*. Then, with probability at least* $1-\delta$*, we have that the minibatch average* $f_k = \frac{1}{b}\sum_{i=1}^{b} f(x,D_i)$ *is* $(\varepsilon, 2G, x^*)$*-SLQC in* x_k*.*

Proof. Let

$$\xi = \frac{1}{b}\sum_{i=1}^{b} f(x^*,D_i) - \mathbb{E}_{D\sim P(D)}\left[f(x^*,D)\right].$$

From Hoeffding's inequality, we have that

$$P(|\xi| \geq t) \leq 2\exp\left(-\frac{2bt^2}{B^2}\right).$$

Thus, if $b \geq \frac{B^2\log(2/\delta)}{2t^2}$ we have that with probability $1-\delta$, $|\xi| < t$.

Let $x_k^* \in \arg\min_{x\in X} \frac{1}{b}\sum_{i=1}^{b} f(x,D_i)$. Let

$$\xi' = \frac{1}{b}\sum_{i=1}^{b} f(x_k^*,D_i) - \mathbb{E}_{D\sim P(D)}\left[f(x_k^*,D)\right].$$

Then, using a covering number argument [50], we have that for $b \geq \frac{nB^2\log(G\bar{B}/\delta)}{2t^2}$, with probability $1-\delta$, $|\xi'| < t$. We have that

$$\frac{1}{b}\sum_{i=1}^{b} f(x_k^*,D_i) \leq \frac{1}{b}\sum_{i=1}^{b} f(x^*,D_i) \leq \mathbb{E}_{D\sim P(D)}\left[f(x^*,D)\right] + \xi,$$

and

$$\mathbb{E}_{D\sim P(D)}\left[f(x^*,D)\right] - \xi' \leq \mathbb{E}_{D\sim P(D)}\left[f(x_k^*,D)\right] - \xi' \leq \frac{1}{b}\sum_{i=1}^{b} f(x_k^*,D_i).$$

Therefore,

$$\frac{1}{b}\sum_{i=1}^{b} f(x^*,D_i) - \frac{1}{b}\sum_{i=1}^{b} f(x_k^*,D_i) \leq \xi + \xi'.$$

Now, similarly to the proof of Lemma 1, assume that $\frac{1}{b}\sum_{i=1}^{b} f(x_k,D_i) - \frac{1}{b}\sum_{i=1}^{b} f(x_k^*,D_i) > \varepsilon$. We choose $b = O\left(\frac{8nB^2\log(G\bar{B}/\delta)}{\varepsilon^2}\right)$ such that with probability $1-\delta$, $\xi + \xi' \leq \varepsilon/2$.

We therefore have:

$$\frac{1}{b}\sum_{i=1}^{b} f(x_k,D_i) - \frac{1}{b}\sum_{i=1}^{b} f(x^*,D_i) > \varepsilon - (\xi + \xi') \geq \frac{\varepsilon}{2}.$$

For simplicity, we denote $\bar{f}(x) = \frac{1}{b}\sum_{i=1}^{b} f(x,D_i)$. Note that $\bar{f}$ is quasi-convex, by Assumption 1. Let Z denote the $\bar{f}(x_k)$-level set of $\bar{f}(x)$. Let ∂Z be the boundary of Z. By definition of the level set, for every $z \in \partial Z$, $\bar{f}(z) - \bar{f}(x^*) > \varepsilon/2$. From the Lipschitz property then, for every $z \in \partial Z$ we must have $\|z - x^*\| \geq \frac{\varepsilon}{2G}$. Since Z is convex, we therefore have that $\mathbb{B}(x^*, \frac{\varepsilon}{2G}) \subset Z$. From Assumption 1, $Q(\bar{f};x) \neq \emptyset$, and from the definition of $Q(\bar{f};x)$, we have that for every $y \in \mathbb{B}(x^*, \frac{\varepsilon}{2G})$, if $\Delta \in Q(\bar{f};x)$ then $\langle \Delta, y - x\rangle \leq 0$. □

We are finally ready to present the converge result.

Theorem 3. *Let Assumption 1 hold. Suppose we run the stochastic normalized subgradient method for* $K \geq \frac{4G^2\|x_1 - x^*\|^2}{\varepsilon^2}$ *iterations,* $\eta = \varepsilon/2G$*, and the minibatch size satisfies* $b = O\left(\frac{8nB^2\log(KG\bar{B}/\delta)}{\varepsilon^2}\right)$*. Then with probability* $1-2\delta$*, we have that* $f(\bar{x}_K) - f(x^*) \leq 3\varepsilon$*.*

Proof. This is a direct application of Theorem 5.1 of [20], where we used Lemma 2 to guarantee that at each iteration the minibatch is SLQC, as required in [20]. We note that by our Definition 1, the proof in [20] holds without change to the non-smooth setting. The projection onto the set X requires a straightforward modification to the proof of [20], where the first equality in their proof of Theorem 4.1 should be a $\leq$. The rest of the proofs remain unchanged. □

B.2.2 Results for Maximum-Multicommodity-Flow

We formally define the problem as follows.

for each tunnel T, let x_T denote the flow on that tunnel, and let $x_e = \sum_{T:e\in T} x_T$, for each edge e, denote the total flow on edge e. We define

$$\gamma = \max\left(\max_e \frac{x_e}{C_e}, 1\right),$$

and normalize the flows by γ, yielding normalized flows on a tunnel,

$$y_T = \frac{x_T}{\gamma},$$

and correspondingly, total normalized flows from source s to target t, $y_{s,t} = \sum_{T\in P_{st}} y_T$. Let $x = \{x_T\}$ denote our decision variables. Given a demand matrix D, the Max-MCF objective is

$$f_{MMCF}(x,D) = \sum_{s,t} \min\left(D_{s,t}, y_{s,t}\right).$$

We next show that f_{MMCF} is Lipschitz.

Lemma 3. *For any tunnel* T *and* $x \geq 0$*,* $\frac{x_T}{\gamma(x)} \leq C_{max}$*.*

Proof. Let $e \in T$, then by the definitions of $\gamma(x)$ and x_e, $\gamma(x) \geq \frac{x_e}{c_e} \geq \frac{x_T}{C_{max}}$. □

Lemma 4. $f_T(x) = \frac{x_T}{\gamma(x)}$ *is Lipschitz on* $\mathbb{R}^n_+$*, and its Lipschitz constant is at most* $K = 2\cdot\frac{C_{max}}{C_{min}}$*.*

Proof. Assume, without loss of generality, that $f(x) \geq f(y)$.
Case 1: $\gamma(y) = 1$

$$\begin{aligned}
&|f(x)-f(y)| \\
&= f(x)-f(y) \\
&= \frac{x_T}{\gamma(x)} - \frac{y_T}{\gamma(y)} \\
&= \frac{x_T}{\gamma(x)} - y_T \\
&\leq x_T - y_T \\
&\leq |x_T - y_T| \\
&\leq \|x-y\|_1 \\
&\leq 2 \cdot \frac{C_{max}}{C_{min}} \cdot \|x-y\|_1,
\end{aligned}$$

where the first inequality is since $\gamma(x) \geq 1$, and the third inequality is by the definition of $\|x\|_1$.
Case 2: $\gamma(y) = \frac{y_{e_0}}{C_{e_0}} > 1$, for some edge e_0.

$$\begin{aligned}
&|f(x)-f(y)| \\
&= f(x)-f(y) \\
&= \frac{x_T}{\gamma(x)} - \frac{y_T}{\gamma(y)} \\
&= \frac{x_T}{\gamma(x)} - \frac{y_T}{\gamma(x)} + \frac{y_T}{\gamma(x)} - \frac{y_T}{\gamma(y)} \\
&= \frac{1}{\gamma(x)} \cdot (x_T - y_T) + \frac{y_T}{\gamma(y)} \cdot \frac{1}{\gamma(x)}(\gamma(y) - \gamma(x)) \\
&\leq \frac{1}{\gamma(x)} \cdot (x_T - y_T) + \frac{y_T}{\gamma(y)} \cdot \frac{1}{\gamma(x)} \left(\frac{y_{e_0}}{C_{e_0}} - \frac{x_{e_0}}{C_{e_0}} \right) \\
&\leq \left| \frac{1}{\gamma(x)} \cdot (x_T - y_T) + \frac{y_T}{\gamma(y)} \cdot \frac{1}{\gamma(x)} \left(\frac{y_{e_0}}{C_{e_0}} - \frac{x_{e_0}}{C_{e_0}} \right) \right| \\
&\leq \frac{1}{\gamma(x)} \cdot |x_T - y_T| + \frac{y_T}{\gamma(y)} \cdot \frac{1}{\gamma(x)} \left| \frac{y_{e_0}}{C_{e_0}} - \frac{x_{e_0}}{C_{e_0}} \right| \\
&\leq |x_T - y_T| + \frac{C_{max}}{C_{min}} \cdot |y_{e_0} - x_{e_0}| \\
&\leq 2 \cdot \frac{C_{max}}{C_{min}} \cdot \|x-y\|_1,
\end{aligned}$$

where the first inequality is since $\gamma(y) = \frac{y_{e_0}}{C_{e_0}}$, $\gamma(x) \geq \frac{x_{e_0}}{C_{e_0}}$, $y_T \geq 0$, and $\gamma > 0$, the third inequality is since $|a+b| \leq |a| + |b|$, the fourth inequality is by Lemma 3 and since $\gamma(x) \geq 1$, and the last inequality is by the definitions of $\|x\|_1$, x_e and since $|a+b| \leq |a| + |b|$. □

Proposition 2. *The function f_{MMCF} is Lipschitz, and its Lipschitz constant is at most $\sum_{s,t} \sum_{p \in P_{st}} 2 \cdot \frac{C_{max}}{C_{min}}$.*

Proof. By Lemma 4 and as a sum and minimum of Lipschitz functions. □

We next state two lemmas that we will use in our analysis.

Lemma 5. *For any $a, b \geq 0$, $c, d > 0$ and $\lambda \in [0,1]$, we have that $\min\left(\frac{a}{c}, \frac{b}{d}\right) \leq \frac{\lambda a + (1-\lambda) b}{\lambda c + (1-\lambda) d}$.*

Proof. Let $f(\lambda) = \frac{\lambda a + (1-\lambda) b}{\lambda c + (1-\lambda) d}$. Then,

$$\begin{aligned}
f'(\lambda) &= \frac{(a-b)(\lambda c + (1-\lambda)d) - (c-d)(\lambda a + (1-\lambda)b)}{(\lambda c + (1-\lambda)d)^2} \\
&= \frac{ad - bc}{(\lambda c + (1-\lambda)d)^2}.
\end{aligned}$$

Also, $f(0) = \frac{b}{d}, f(1) = \frac{a}{c}$, and $f'(\lambda)$ has a fixed sign for any $\lambda \in [0,1]$. Therefore, $f(\lambda) \geq \min\left(\frac{a}{c}, \frac{b}{d}\right)$. □

Lemma 6. *Let $x = \{x_T\}$, $x' = \{x'_T\}$, and $\lambda \in [0,1]$. Let $x'' = \{\lambda x_T + (1-\lambda) x'_T\}$, and let γ, γ' and γ'' be the respective normalization constants. Then $\gamma'' \leq \lambda\gamma + (1-\lambda)\gamma'$.*

Proof. We have that

$$\begin{aligned}
\gamma'' &= \max\left(\max_e \frac{x''_e}{C_e}, 1\right) \\
&= \max\left(\max_e \frac{\lambda x_e + (1-\lambda) x'_e}{C_e}, 1\right) \\
&\leq \max\left(\max_e \frac{\lambda x_e}{C_e}, \lambda\right) + \max\left(\max_e \frac{(1-\lambda) x'_e}{C_e}, 1-\lambda\right) \\
&= \lambda \max\left(\max_e \frac{x_e}{C_e}, 1\right) + (1-\lambda) \max\left(\max_e \frac{x'_e}{C_e}, 1\right) \\
&= \lambda\gamma + (1-\lambda)\gamma'.
\end{aligned}$$

□

We next show that Max-MCF satisfies Assumption 1.

Proposition 3. *The function f_{MMCF} is Lipschitz and bounded. Its maximum is obtained inside a convex set X. Furthermore, for every $D^1, \ldots, D^M$, we have that $\frac{1}{M}\sum_{i=1}^{M} f(x, D^i)$ is quasi-concave in x*

Proof. By definition, $x_T \geq 0$ for all T. Let $C_{max} = \max_e C_e$, and consider T-dimensional hypercube $X = [0, C_{max}]^T$. By definition, for every $x \geq 0$ that is outside X, there is an $x' \in X$ with an equivalent objective value. To see this, let γ the normalizing constant for x, and set $x' = x/\gamma$. Then,

$$x'_T = \frac{x_T}{\max\left(\max_e \frac{x_e}{C_e}, 1\right)} \leq \frac{x_T}{\max\left(\max_e \frac{x_e}{C_{max}}, 1\right)} \leq \frac{x_T}{\frac{x_T}{C_{max}}} = C_{max}.$$

But the normalizing factor for x' is 1, so x and x' have the same objective value.

Clearly, f_{MMCF} is bounded by $\sum_{s,t} \sum_{T: e \in T} C_{max}$.

The function is Lipschitz by proposition 2.

Let $\bar{f}_{MMCF}(x) = \frac{1}{M}\sum_{i=1}^{M} f_{MMCF}(x, D^i)$. We shall now show that for any $x, x' \in X$, and $\lambda \in [0,1]$, $\bar{f}_{MMCF}(\lambda x + (1-\lambda) x') \geq$

$\min\{\bar{f}_{MMCF}(x), \bar{f}_{MMCF}(x')\}$, proving that $\bar{f}_{MMCF}$ is quasiconcave. We denote by γ' and y' the respective normalization constant and normalized flows corresponding to x'. We also denote $x'' = \lambda x + (1-\lambda)x'$, and let γ'' and y'' denote its corresponding normalization constant and normalized flows, respectively.

$$
\begin{aligned}
&\min\big(\sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, y_{s,t}), \sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, y'_{s,t})\big)\\
=&\min\big(\sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, \sum_{T\in P_{st}}\frac{x_T}{\gamma}), \sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, \sum_{T\in P_{st}}\frac{x'_T}{\gamma'})\big)\\
=&\min\big(\frac{1}{\gamma}\sum_{i=1}^{M}\sum_{s,t}\min(\gamma D^i_{s,t}, \sum_{T\in P_{st}} x_T), \frac{1}{\gamma'}\sum_{i=1}^{M}\sum_{s,t}\min(\gamma' D^i_{s,t}, \sum_{T\in P_{st}} x'_T)\big)\\
\leq&\frac{\lambda\sum_{i=1}^{M}\sum_{s,t}\min(\gamma D^i_{s,t}, \sum_{T\in P_{st}} x_T) + (1-\lambda)\sum_{i=1}^{M}\sum_{s,t}\min(\gamma' D^i_{s,t}, \sum_{T\in P_{st}} x'_T)}{\lambda\gamma + (1-\lambda)\gamma'}\\
=&\frac{\sum_{i=1}^{M}\sum_{s,t}\min(\lambda\gamma D^i_{s,t}, \lambda\sum_{T\in P_{st}} x_T) + \min((1-\lambda)\gamma' D^i_{s,t}, (1-\lambda)\sum_{T\in P_{st}} x'_T)}{\lambda\gamma + (1-\lambda)\gamma'}\\
\leq&\frac{1}{\lambda\gamma + (1-\lambda)\gamma'}\sum_{i=1}^{M}\sum_{s,t}\min\Big(\lambda\gamma D^i_{s,t} + (1-\lambda)\gamma' D^i_{s,t},\\
&\lambda\sum_{T\in P_{st}} x_T + (1-\lambda)\sum_{T\in P_{st}} x'_T\Big)\\
=&\sum_{i=1}^{M}\sum_{s,t}\min\Big(D^i_{s,t}, \frac{1}{\lambda\gamma + (1-\lambda)\gamma'}\sum_{T\in P_{st}}(\lambda x_T + (1-\lambda)x'_T)\Big)\\
\leq&\sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, \sum_{T\in P_{st}}\frac{x''_T}{\gamma''})\\
=&\sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, y''_{s,t}),
\end{aligned}
$$

where the first inequality is by Lemma 5, the second inequality is since $\min(a,b)+\min(c,d) \leq \min(a+c, b+d)$, and the third inequality is by Lemma 6. □

Lemma 7. *Let $x \notin \arg\max_y f(y)$, $x^* \in \arg\max_y f(y)$, and let γ, γ^* be the respective normalization constants. If $f(x+\lambda(x^*-x)) \geq \frac{\lambda\gamma^* f(x^*)+(1-\lambda)\gamma f(x)}{\lambda\gamma^*+(1-\lambda)\gamma}$ for any $\lambda \in [0,1]$, then $Q(f;x) \neq \emptyset$.*

Proof. The directional derivative of f along x^*-x at x:

$$
\begin{aligned}
\nabla_{x^*-x} f(x) &= \lim_{h\to 0^+}\frac{f(x+h(x^*-x)) - f(x)}{h\|x^*-x\|}\\
&\geq \lim_{h\to 0^+}\frac{\frac{h\gamma^* f(x^*)+(1-h)\gamma f(x)}{h\gamma^*+(1-h)\gamma} - f(x)}{h\|x^*-x\|}\\
&= \lim_{h\to 0^+}\frac{\frac{h\gamma^*}{h\gamma^*+(1-h)\gamma}(f(x^*)-f(x))}{h\|x^*-x\|}\\
&\geq \lim_{h\to 0^+}\frac{\frac{\gamma^*}{\max(\gamma^*,\gamma)}(f(x^*)-f(x))}{\|x^*-x\|} > 0.
\end{aligned}
$$

Therefore, since $L(f; f(x))$ is convex, $-\frac{x^*-x}{\|x^*-x\|} \in Q(f;x)$. □

Lemma 8. *Let $x = \{x_T\}$, $x' = \{x'_T\}$, and $\lambda \in [0,1]$. Let $x'' = \{\lambda x_T + (1-\lambda)x'_T\}$, and let γ, γ' and γ'' be the respective normalization constants. Then, $\bar{f}_{MMCF}(x'') \geq \frac{\lambda\gamma\bar{f}_{MMCF}(x)+(1-\lambda)\gamma'\bar{f}_{MMCF}(x')}{\lambda\gamma+(1-\lambda)\gamma'}$.*

Proof.

$$
\begin{aligned}
&\bar{f}_{MMCF}(x'') = \sum_{i=1}^{M}\sum_{s,t}\min(D^i_{s,t}, \sum_{T\in P_{st}}\frac{x''_T}{\gamma''})\\
\geq&\sum_{i=1}^{M}\sum_{s,t}\min\Big(D^i_{s,t}, \frac{1}{\lambda\gamma + (1-\lambda)\gamma'}\sum_{T\in P_{st}}(\lambda x_T + (1-\lambda)x'_T)\Big)\\
=&\frac{\sum_{i=1}^{M}\sum_{s,t}\min\Big(\lambda\gamma D^i_{s,t} + (1-\lambda)\gamma' D^i_{s,t}, \lambda\gamma\sum_{T\in P_{st}}\frac{x_T}{\gamma} + (1-\lambda)\gamma'\sum_{T\in P_{st}}\frac{x'_T}{\gamma'}\Big)}{\lambda\gamma + (1-\lambda)\gamma'}\\
\geq&\frac{\sum_{i=1}^{M}\sum_{s,t}\Big(\min(\lambda\gamma D^i_{s,t}, \lambda\gamma\sum_{T\in P_{st}}\frac{x_T}{\gamma}) + \min((1-\lambda)\gamma' D^i_{s,t}, (1-\lambda)\gamma'\sum_{T\in P_{st}}\frac{x'_T}{\gamma'})\Big)}{\lambda\gamma + (1-\lambda)\gamma'}\\
=&\frac{\lambda\gamma\bar{f}_{MMCF}(x) + (1-\lambda)\gamma'\bar{f}_{MMCF}(x')}{\lambda\gamma + (1-\lambda)\gamma'},
\end{aligned}
$$

where the first inequality is by Lemma 6 and the second inequality is since $\min(a,b)+\min(c,d) \leq \min(a+c, b+d)$. □

Proposition 4. *$Q(\bar{f}_{MMCF}, x) \neq \emptyset$ for any $x \notin$ argmax$_y \bar{f}_{MMCF}(y)$*

Proof. By Lemma 8 where $x = x^*, x' = x$, and by Lemma 7. □

Since Assumption 1 holds, Theorem 3 guarantees that the stochastic normalized subgradient method will converge to an optimal solution of the Max-MCF objective.

B.2.3 Results for Maximum-Concurrent-Flow

Given a demand matrix D, the Max-Concurrent-Flow objective is

$$f_{MCONC}(x, D) = \min(\{\frac{y_{s,t}}{D_{s,t}}\}_{s,t\in V, D_{s,t}>0} \cup \{1\}).$$

We assume that when $D_{s,t} \neq 0$, there is a minimal value ε for $D_{s,t}$, corresponding, e.g., to a single packet. We next show that Max-Concurrent-Flow satisfies Assumption 1.

Proposition 5. *The function f_{MCONC} is Lipschitz, and its Lipschitz constant is at most $\max_{s,t}\Big(\sum_{p\in P_{st}}\frac{2\cdot C_{max}}{\varepsilon\cdot C_{min}}\Big)$.*

Proof. By Lemma 4 and as a sum, minimum and multiplication by a constant of Lipschitz functions. □

Proposition 6. *The function f_{MCONC} is Lipschitz and bounded. Its maximum is obtained inside a convex set X. Furthermore, for every $D^1,\dots,D^M$, we have that $\frac{1}{M}\sum_{i=1}^{M} f(x,D^i)$ is quasi-concave in x*

Proof. The claims in the beginning of proposition 3 hold for f_{MCONC}, and therefore its maximum is obtained inside a convex set.

Clearly, f_{MCONC} is bounded by 1.

The function is Lipschitz by proposition 5.

Let $\bar{f}_{MCONC}(x) = \frac{1}{M}\sum_{i=1}^{M} f_{MCONC}(x,D^i)$. We shall now show that for any $x,x' \in X$, and $\lambda \in [0,1]$, $\bar{f}_{MCONC}(\lambda x + (1-\lambda)x') \geq \min\{\bar{f}_{MCONC}(x), \bar{f}_{MCONC}(x')\}$, proving that $\bar{f}_{MCONC}$ is quasi-concave. We denote by γ' and y' the respective normalization constant and normalized flows corresponding to x'. We also denote $x'' = \lambda x + (1-\lambda)x'$, and let γ'' and y'' denote its corresponding normalization constant and normalized flows, respectively.

$$
\begin{aligned}
&\min\Big(\sum_{i=1}^{M}\min\Big(\left\{\frac{y_{s,t}}{D^i_{s,t}}\right\}\cup\{1\}\Big), \sum_{i=1}^{M}\min\Big(\left\{\frac{y'_{s,t}}{D^i_{s,t}}\right\}\cup\{1\}\Big)\Big)\\
&=\min\Big(\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}}\frac{x_T}{\gamma}}{D^i_{s,t}}\right\}\cup\{1\}\Big), \sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}}\frac{x'_T}{\gamma'}}{D^i_{s,t}}\right\}\cup\{1\}\Big)\Big)\\
&=\min\Big(\frac{1}{\gamma}\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}} x_T}{D^i_{s,t}}\right\}\cup\{\gamma\}\Big), \frac{1}{\gamma'}\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}} x'_T}{D^i_{s,t}}\right\}\cup\{\gamma'\}\Big)\Big)\\
&\leq\frac{\lambda\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}} x_T}{D^i_{s,t}}\right\}\cup\{\gamma\}\Big)+(1-\lambda)\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}} x'_T}{D^i_{s,t}}\right\}\cup\{\gamma'\}\Big)}{\lambda\gamma+(1-\lambda)\gamma'}\\
&=\frac{\sum_{i=1}^{M}\Big(\min\Big(\left\{\frac{\sum_{T\in P_{st}} \lambda x_T}{D^i_{s,t}}\right\}\cup\{\lambda\gamma\}\Big)+\min\Big(\left\{\frac{\sum_{T\in P_{st}} (1-\lambda)x'_T}{D^i_{s,t}}\right\}\cup\{(1-\lambda)\gamma'\}\Big)\Big)}{\lambda\gamma+(1-\lambda)\gamma'}\\
&\leq\frac{\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}} \lambda x_T+\sum_{T\in P_{st}}(1-\lambda)x'_T}{D^i_{s,t}}\right\}\cup\{\lambda\gamma+(1-\lambda)\gamma'\}\Big)}{\lambda\gamma+(1-\lambda)\gamma'}\\
&=\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}}\frac{\lambda x_T+(1-\lambda)x'_T}{\lambda\gamma+(1-\lambda)\gamma'}}{D^i_{s,t}}\right\}\cup\{1\}\Big)\\
&\leq\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}}\frac{x''}{\gamma''}}{D^i_{s,t}}\right\}\cup\{1\}\Big)\\
&=\sum_{i=1}^{M}\min\Big(\left\{\frac{y''_{s,t}}{D^i_{s,t}}\right\}\cup\{1\}\Big),
\end{aligned}
$$

where the first inequality is by Lemma 5, the second inequality is since $\min(a,b)+\min(c,d) \leq \min(a+c,b+d)$, and the third inequality is by Lemma 6. □

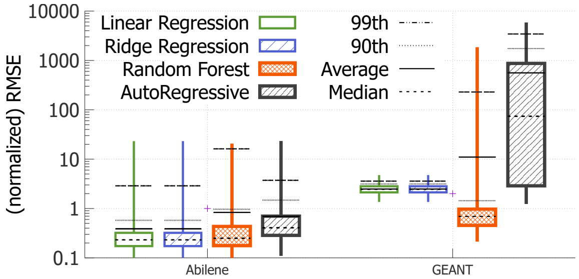


Figure 11: Accuracy of predicting demands; results from different prediction methods.

Lemma 9. *Let $x = \{x_T\}$, $x' = \{x'_T\}$, and $\lambda \in [0,1]$. Let $x'' = \{\lambda x_T + (1-\lambda)x'_T\}$, and let γ, γ' and γ'' be the respective normalization constants. Then,*
$\bar{f}_{MCONC}(x'') \geq \frac{\lambda\gamma\bar{f}_{MCONC}(x)+(1-\lambda)\gamma'\bar{f}_{MCONC}(x')}{\lambda\gamma+(1-\lambda)\gamma'}$.

Proof.

$$
\begin{aligned}
\bar{f}_{MCONC}(x'') &= \sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}}\frac{x''}{\gamma''}}{D^i_{s,t}}\right\}\cup\{1\}\Big)\\
&\geq\sum_{i=1}^{M}\min\Big(\left\{\frac{\sum_{T\in P_{st}}\frac{\lambda x_T+(1-\lambda)x'_T}{\lambda\gamma+(1-\lambda)\gamma'}}{D^i_{s,t}}\right\}\cup\{1\}\Big)\\
&=\frac{\sum_{i=1}^{M}\min\Big(\left\{\frac{\lambda\gamma\sum_{T\in P_{st}}\frac{x_T}{\gamma}+(1-\lambda)\gamma'\sum_{T\in P_{st}}\frac{x'_T}{\gamma'}}{D^i_{s,t}}\right\}\cup\{\lambda\gamma+(1-\lambda)\gamma'\}\Big)}{\lambda\gamma+(1-\lambda)\gamma'}\\
&\geq\frac{\sum_{i=1}^{M}\left(\min\Big(\left\{\frac{\lambda\gamma\sum_{T\in P_{st}}\frac{x_T}{\gamma}}{D^i_{s,t}}\right\}\cup\{\lambda\gamma\}\Big)+\min\Big(\left\{\frac{(1-\lambda)\gamma'\sum_{T\in P_{st}}\frac{x'_T}{\gamma'}}{D^i_{s,t}}\right\}\cup\{(1-\lambda)\gamma'\}\Big)\right)}{\lambda\gamma+(1-\lambda)\gamma'}\\
&=\frac{\lambda\gamma\bar{f}_{MCONC}(x)+(1-\lambda)\gamma'\bar{f}_{MCONC}(x')}{\lambda\gamma+(1-\lambda)\gamma'},
\end{aligned}
$$

where the first inequality is by Lemma 6 and the second inequality is since $\min(a,b)+\min(c,d) \leq \min(a+c,b+d)$. □

Proposition 7. *$Q(\bar{f}_{MCONC},x) \neq \emptyset$ for any $x \notin \mathrm{argmax}_y \bar{f}_{MCONC}(y)$*

Proof. By Lemma 9 where $x = x^*, x' = x$, and by Lemma 7. □

Since Assumption 1 holds, Theorem 3 guarantees that the stochastic normalized subgradient method will converge to an optimal solution of the Max-Concurrent-Flow objective.

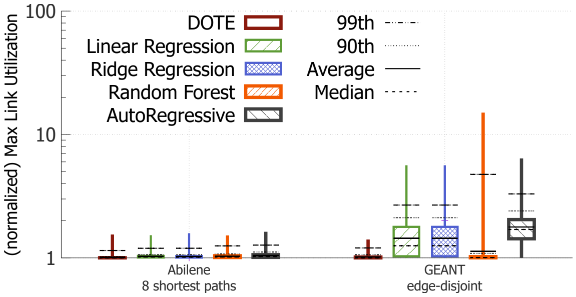


Figure 12: Impact of demand prediction accuracy on max-link-utilization.

C A Closer Look at Demand Prediction

Our results in §4 considered a demand-prediction-based scheme that utilizes linear regression. We next contrast linear regression with other prediction methods on our datasets. Specifically, we consider the following prediction methods: linear regression, ridge regressing, random forrest, and autoregressive model. With the exception of the autoregressive model, each of these schemes predicts the next traffic demand for each source-destination pair using only that specific pair's recently observed 12 most traffic demands, *i.e.*, the prediction for each pair is independent from the prediction for other pairs (as in SWAN [22]). The autoregressive model, in contrast, predicts the entire next DM from the 12 most recently observed DMs, to allow for detecting correlations between different pairs that might be conducive for prediction.

Figure 11 plots the accuracy of the different predictors, as quantified by the root-mean-squared-error, for the two publicly available WAN datasets. The accuracy is normalized by the average traffic demand for the dataset and presented in log-scale. Our results for *PWAN* and *PWAN$_{DC}$* exhibit similar trends. As shown in the figure, linear regression and ridge regression achieve the best results on average on both WANs. We also considered a DNN-based predictor with a single hidden layer with 128 neurons and ReLU activation functions, but its performance was strictly dominated by linear regression on the test data (results omitted). Moreover, treating source-destination pairs individually attains better accuracy than that provided by the autoregressive model. We believe that this is because, on the one hand, the previous traffic demands for a single pair already contain a lot of valuable information and, on the other hand, the much larger input and output of the autoregressive model (entire DMs *vs.* single demands) makes effective learning more difficult.

Figure 12 plots the implications of choosing different predictors for TE performance, as quantified by the max-link-utilization, benchmarked against *DOTE*. Observe that *DOTE* outperforms all considered flavors of demand-based-prediction TE, and also that accuracy in demand prediction does not always translate to better TE performance, exemplifying the potential objective mismatch between the two, discussed in the Introduction.

D Robustness to Unexpected Traffic Changes

We consider the GEANT, Cogentco, and GtsCe network topologies with edge-disjoint tunnels. For Cogentco, and GtsCe we use the gravity model to generate demands for both train and test. To evaluate the implications of unexpected traffic changes, we add noise to the test set by multiplying each demand independently by a factor sampled uniformly at random from the range $[1-\alpha, 1+\alpha]$ for $\alpha \in \{0.1, 0.25, 0.35\}$.

Recall that for GEANT, *DOTE* generates TE configurations that are extremely close to the optimum (less than 2%). Our results show that even under random traffic perturbations, the distance from the omniscient oracle remains low; 2%, 2.9%, and 3.8% for $\alpha = 0.1, 0.25, 0.35$, respectively. For $\alpha = 0.35$, the distance from the omniscient oracle was 0.01% in the median, 13% in the 90th percentile, and no higher than 28% even in the 99th percentile.

For both Cogentco and GtsCe, *DOTE*'s trained model is roughly 0.5% from the omniscient oracle on the test demands are perturbed. This is because traffic is generated using the gravity model naturally does not reflect the intricate *temporal* patterns and complexity of real-world traffic. Even after perturbing the traffic in our experiments *DOTE* achieved near-optimal performance. Specifically, on Cogentco, the average distance from the omniscient oracle was 0.54%, 0.57%, and 0.6% for $\alpha = 0.1, 0.25, 0.35$, respectively. For $\alpha = 0.35$, the distance from the omniscient oracle was 0.56% in the median, 1% in the 90th percentile, and 1.4% in the 99th percentile. On GtsCe, the average distance from the omniscient oracle was 0.51%, 0.56%, and 0.61% for $\alpha = 0.1, 0.25, 0.35$ respectively. For $\alpha = 0.35$, the distance from the omniscient oracle was 0.57% in the median, 1% in the 90th percentile, and 1.4% in the 99th percentile.

	Tunnels	Week 1	Week 2	Week 3	Week 4
Abilene	8 SP	0.7	0.3	1.0	1.5
Abilene	edge-disjoint	2.1	2.4	2.4	2.0
GEANT	8 SP	1.4	2.7	2.9	3.1
GEANT	edge-disjoint	0.7	1.6	2.0	2.5

Table 3: Average weekly distance from the omniscient oracle achieved by *DOTE* for MLU across 4 consecutive weeks

	Tunnels	Week 1	Week 2	Week 3	Week 4
Abilene	8 SP	1.6	2.1	3.9	6.2
Abilene	edge-disjoint	1.1	1.4	3.1	5.5
GEANT	8 SP	4.9	4.7	5.0	4.8
GEANT	edge-disjoint	6.3	6.8	6.9	6.4

Table 4: Average weekly distance from the omniscient oracle achieved by *DOTE* for maximum-multicommodity-flow across 4 consecutive weeks

E Stochastic Optimization Loss Function Pseudocode

Function 1 Stochastic Optimization Loss Function Pseudocode

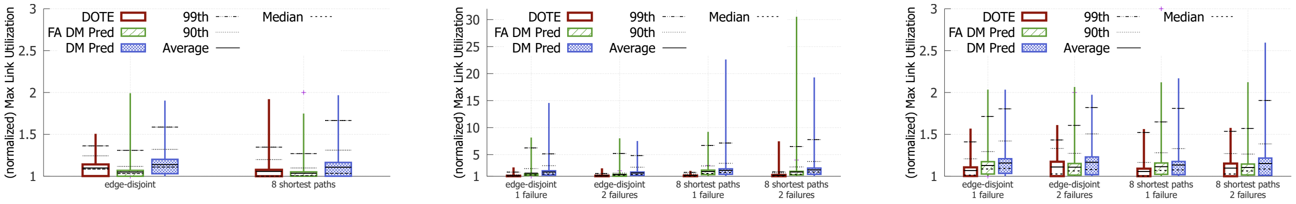
```
$G = (V, E, c)$ // capacitated directed graph that models the WAN topology
$U = \{(i, j) | i \in V, j \in V, i \neq j\}$ // all pairs of nodes
$T = \cup_{(s,t) \in U} P_{s,t}$ // the set of all tunnels
$A^{|U| \times |T|}$ // specifies, for each pair of nodes $i \in U$ and tunnel $j \in T$ whether tunnel $j$ interconnects the nodes in $i$
$A_{i,j} = \begin{cases} 1 & j \in P_i \\ 0 & \text{otherwise} \end{cases}$
$B^{|T| \times |E|}$ // specifies, for each tunnel $i$ and edge $j$, whether tunnel $i$ contains edge $e$
$B_{i,j} = \begin{cases} 1 & j \in i \\ 0 & \text{otherwise} \end{cases}$
$C^{|E| \times 1}$ // vector representing WAN link capacities
$C_{i,1} = c(i)$

function LOSS($DNN_{output}, DM_{next}$)
    $DNN_{output}^{|T| \times 1}$ // the output of the DNN
    $DM_{next}^{|U| \times 1}$ // the (actual) next demand matrix

    // $\times$ and $/$ are element-wise operations
    // 1. Compute the splitting ratios
    $PathsSplit^{|T| \times 1} = DNN_{output} \times (A^T(1.0/A \times DNN_{output}))$
    // 2. Calculate the flow on each edge
    $FlowOnEdges^{|E| \times 1} = B^T((A^T \times DM_{next}) \times PathsSplit)$
    // 3. Compute the maximum-link-utilization
    $MaxLoad = max(FlowOnEdges/C)$
    return $MaxLoad$
end function
```

F Additional Failure Results

Analogous to Figure 9, Figure 13 shows the behavior under faults for the Abilene, GEANT and $\mathsf{PWAN}_{\mathsf{DC}}$ topologies respectively. Figure 14 shows the results for maximum-multicommodity-flow.

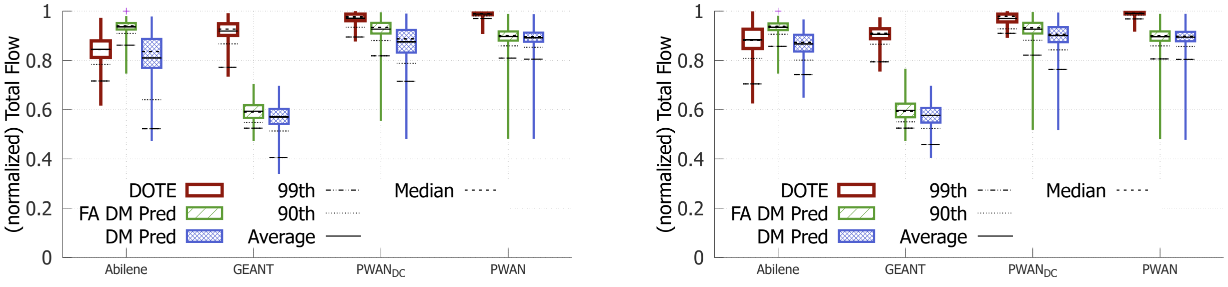


(a) Abilene

(b) GEANT

(c) $\mathsf{PWAN_{DC}}$

Figure 13: Understanding the behavior of *DOTE* under failures on different WAN datasets. The results are qualitatively similar to Figure 9.



(a) 8SP

(b) edge-disjoint

Figure 14: Coping with a random link failure when aiming to maximize the total flow for two different tunnel choices.