Online Joint Optimization on Traffic Engineering and Network Update in Software-defined WANs

Jiaqi Zheng†, Yimeng Xu†, Li Wang†, Haipeng Dai†, Guihai Chen†¶
†*State Key Laboratory for Novel Software Technology, Nanjing University, China*
¶*Shanghai Jiao Tong University, Shanghai, China*

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488837

***Abstract*—State-of-the-art inter-datacenter WANs rely on centralized traffic engineering (TE) to improve the network performance, where TE computation is a periodical procedure and timely performs routing configurations (*i.e.*, enforces TE polices via add, remove and modify forwarding rules) in response to the changing network conditions. The TE computation determines the routing configurations corresponding to the current network conditions and the network update operations change the routing configurations from last TE to current TE solution. Existing works take centralized TE computation and network update as two individual optimization procedures, which inevitably leads to suboptimal solution in the long run. In this paper we initiate the study of online joint optimization on TE computation and network update with the objective of minimizing the sum of TE cost and network update cost. We formulate this problem as an optimization program and propose a set of provable online algorithms with rigorous competitive and regret analysis. Trace-driven simulations on two empirical topologies demonstrate that our algorithms can significantly decrease the total cost.**

I. Introduction

Software defined networking (SDN), as a state-of-the-art implementation of centralized traffic engineering (TE), is increasingly adopted in inter-datacenter WANs to orchestrate the data transmission. Since an inter-datacenter WAN is a highly expensive network infrastructure, the centralized TE needs to periodically run an optimization program to improve network performance. For example, Google [1], [2], [3] and Microsoft [4] optimize the data plane performance frequently via timely dispatching TE plan to the data plane. Once the TE computation in the current time interval is done, the network update operations can be subsequently performed to switch the route from last to current TE configurations.

TE plays an important role when scheduling inter-datacenter WANs. Specifically, SWAN [4] and B4 [1], [2], [3] use the max-min fairness principle to improve the network throughput. NUMFabric [5] takes different utility functions [6] to capture the correlation between the allocated bandwidth resource and the quality-of-service. Calendaring [7] and Amoeba [8] strengthen the performance especially for deadline-aware applications. In general, the TE optimization can be abstracted as a fundamental minimum-cost flow problem [9], where the cost here can characterize the network delay, link load, flow completion time, deadline missing ratio, *etc.* Network update operations are triggered periodically along with the TE computation and change the routes between two adjacent TE configurations [10].

Existing works take centralized TE computation [11], [12], [13], [14] and network update [4], [15], [16], [17], [18] as two individual optimization procedures, which inevitably leads to suboptimal solution in nature. Essentially, directly optimizing TE objective [19] such as minimizing transmission delay, minimizing flow completion time *etc.* can produce a TE solution with minimum cost at each single time slot. However, it may not perform well in the long run with continuous time slots, especially in terms of competitive and regret analysis [20]. Furthermore, if network update cost cannot be respected and fails to integrate into TE optimization, frequent network update procedure between two adjacent TE solutions can bring route oscillation [21] and incur a large amount of unnecessary operations resulting from adding, removing or modifying the route in the data plane. This essentially drives the network operators to jointly optimize TE computation and network update procedure. Furthermore, network update plan between two adjacent TE solutions is usually not unique [16], which can navigate network operators a broader optimization space when designing our online framework. To the best of our knowledge, our work proposes the first online optimization framework that jointly considering TE cost and network update cost, which has not been done before.

In this paper we propose an online joint optimization framework on TE computation and network update that aims to minimize the total cost, *i.e.*, the sum of TE cost and network update (rerouting) cost, in the long run. The resulting TE solutions determine how the flows are routed in current time interval and the network update plan indicates which routes need to be changed from last to current TE solution. These two procedures need to be jointly optimized in the time horizon to improve the whole performance.

Our first contribution is that we propose an online optimization framework for jointly minimizing TE cost and network update cost in the time horizon. Generally speaking, given the inter-datacenter WAN topology and the available tunnels for each ingress-egress switch pairs, the optimization program aims to determine how each flow is splitted at the ingress switch onto the available tunnels (TE solution) and which flow needs to be rerouted (network update plan) such that the network capacity constraints are respected (Sec. II). Here the flow is an aggregate of all TCP flows between the same ingress-egress switch pair.

Our second contribution is that we develop a set of provable online algorithms to solve our problem and conduct rigorous

competitive and regret analysis. As the optimization on the network update cost is coupled by two adjacent TE time slots, we exploit regularization techniques to stabilize the solution and decouple the original problem into a series of subproblems, where each subproblem can be solved in a single time slot. We first propose a competitive regularization-based algorithm (RA) where the demand of each flow can be accurately predicted in one future time slot. Furthermore, we relax the prediction accuracy and propose receding horizon control-based (RHC) and averaging fixed horizon control-based (AFHC) algorithms that allow the network operators to predict the flow demand in multiple future time slots with bounded prediction errors (Sec. III).

Our third contribution is that we evaluate our algorithms using two empirical topologies with real dataset. The candidate tunnel set is produced by k-shortest paths, equal-cost multipath routing, Räcke's oblivious routing and edge-disjoint k-shortest paths, respectively. We first use the time series to predict the flow demand and then obtain the solutions by iteratively solving each subproblem. Evaluation results show that RA, RHC and AFHC can reduce 12.1%, 12.3% and 15.4% cost on average compared with state-of-the-art and can perfectly match the theoretical analysis (Sec. IV).

II. An Online Optimization Framework

We introduce an online optimization framework in this section, which aims to minimize the TE cost and network update cost in the long run.

A. A Motivating Example

The traditional centralized TE periodically reroutes the traffic to optimize the performance once the network conditions change such as traffic matrix variations or the switches fail in the data plane, where the TE is usually informed by this event in a proactive manner [2], [3]. Consider the motivating example in Fig. 1, there are totally seventeen switches and twenty links in the network. The capacity of each link is one unit. There are three aggregate flows colored red, blue and green, respectively and their demands vary with time. We assume each flow's demand can be accurately predicted within one future time slot in our motivating example. We will relax this assumption in our algorithm design section since it can be replaced by an estimation of the real demand with bounded prediction errors.

Specifically, we can see that in Fig. 1, the flow F_A originates from switch R_3 to switch R_7, where it has three available tunnels $\langle R_3, R_1, R_2, R_7\rangle$, $\langle R_3, R_4, R_5, R_6, R_7\rangle$ and $\langle R_3, R_8, R_9, R_7\rangle$. The demand of flow F_A increases from one unit at t_0 to two units at t_1, and drop to one unit at t_2. The flow F_B originates from switch R_{10} to switch R_{11}, where it has two available tunnels $\langle R_{10}, R_8, R_9, R_{11}\rangle$ and $\langle R_{10}, R_{12}, R_{13}, R_{11}\rangle$. The demand of F_B keeps the same at t_0, t_1 and t_2. The flow F_C originates from switch R_{14} to switch R_{15}, where it has two available tunnels $\langle R_{14}, R_{12}, R_{13}, R_{15}\rangle$ and $\langle R_{14}, R_{16}, R_{17}, R_{15}\rangle$. The demand of F_C increases from one unit at t_1 to two units at t_2. Without loss of generality, we assume that the TE objective in Fig. 1(a) aims to minimize the total cost, *i.e.*, the product of flow demand and the number of hops that it passes through. At t_0, the cost incurred by F_A, F_B and F_C is $1\times 3 = 3$, $1\times 3 = 3$, and $1\times 3 = 3$, where each flow is routed through on the tunnel with the minimum number of hops. Then the demand of flow F_A increases to two units at t_1. Accordingly the TE reroutes the flow F_B and F_C to make the link $\langle R_8, R_9\rangle$ accommodate the increased demand of flow F_A without congestion. Thus the flow F_A can be splitted onto two tunnels $\langle R_3, R_1, R_2, R_7\rangle$ and $\langle R_3, R_8, R_9, R_7\rangle$, one unit flow demand on each tunnel. The flow F_A prefers the tunnel $\langle R_3, R_8, R_9, R_7\rangle$ to the tunnel $\langle R_3, R_4, R_5, R_6, R_7\rangle$ as the former has less number of hops and is with less TE cost. That's why the flow F_A is not routed through the tunnel $\langle R_3, R_4, R_5, R_6, R_7\rangle$ at t_1. Hence, the cost incurred by F_A, F_B and F_C at t_1 becomes $1\times 3+1\times 3 = 6$, $1\times 3 = 3$, and $1\times 3 = 3$. Next at t_2, the demand of F_A drops to one unit and the demand of F_C increases to two units. The TE reroutes the flow F_B from $\langle R_{10}, R_{12}, R_{13}, R_{11}\rangle$ to $\langle R_{10}, R_8, R_9, R_{11}\rangle$ to make the link $\langle R_{12}, R_{13}\rangle$ accommodate the increased demand of flow F_C and simultaneously ensures that each link is congestion-free. The cost incurred by F_A, F_B and F_C at t_2 is $1\times 3 = 3$, $1\times 3 = 3$ and $1\times 3+1\times 3 = 6$.

TABLE I
Cost comparisons in Fig. 1.

Strategies	TE cost			Rerouting cost		Total cost
	t_0	t_1	t_2	$t_0 \to t_1$	$t_1 \to t_2$	
Traditional TE	9	12	12	15	12	60
Our approach	9	13	12	4	7	45

However, traditional TE computation only focuses on TE cost minimization in a single time slot and does not take network update (rerouting) cost into consideration. The network update operations involve adding, removing and modifying forwarding rules in the data plane. Frequently switching forwarding rules can lead to route oscillation and degrade application performance. Hence, it is necessary to exert a joint optimization on TE computation and network update, aiming to minimize the total cost, *i.e.*, the sum of TE cost and network update cost. We use our example in Fig. 1(b) to illustrate how our approach works. The main differences between Fig. 1(a) and Fig. 1(b) are the route configurations resulting from TE at t_1. At this moment, our approach splits flow F_A on two paths $\langle R_3, R_1, R_2, R_7\rangle$ and $\langle R_3, R_4, R_5, R_6, R_7\rangle$, one unit flow demand on each path. At the same time, we keep the route of flow F_B unchanged. This inevitably increases the TE cost since the path $\langle R_3, R_4, R_5, R_6, R_7\rangle$ that the flow F_A passes through has more number of hops. But we avoid the frequent rerouting behavior of flow F_B and thus decrease the network update cost. For our approach, the TE cost at t_0, t_1 and t_2 is 9, 13 and 12, respectively. The network update cost from t_0 to t_1 is $1\times 4 = 4$, while that from t_1 to t_2 is $1\times 4+1\times 3 = 7$. Table I shows the cost comparisons for traditional TE and our approach, from which we can observe that our approach can reduce total cost by 25%. Note that here the unit of TE cost and network update cost are both one. For more general cases,

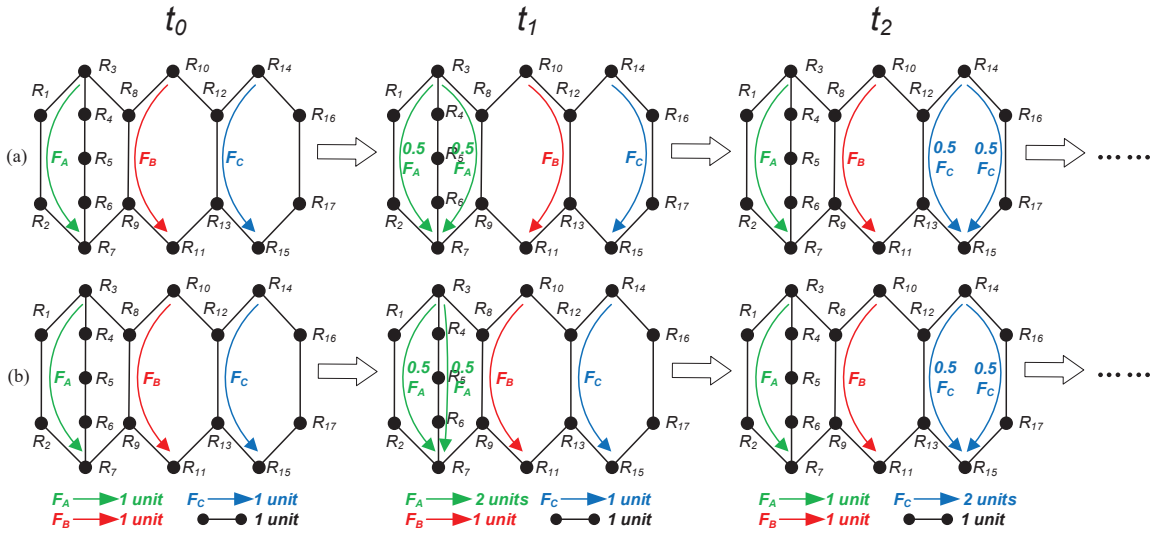


Fig. 1. A motivating example for (a) traditional TE and (b) our approach.

we will introduce two parameters α and β to normalize these two different costs in the following section.

B. Network Model

Before presenting the problem definitions, we first discuss our network model. A network is a directed graph $G = (V, E)$, where V is the set of switches and E the set of links with capacities L_e for each link $e \in E$. F represents the set of aggregate flows in the network and each flow is associated with a demand $d_{p,t}$. Note that we slightly abuse the notation and use the index p to refer to the flow f for convenience. Essentially, each ingress-egress switch pairs correspond to one aggregate flow and each aggregate flow corresponds to a available tunnel set. This aggregate flow will be splitted among the available tunnels according to $x_{p,t}$ (unknown variables). For example, if an ingress-egress switch pair has three tunnels p_1, p_2, p_3 at t and the flow demand for this ingress-egress switch pair is d, then $d_{p_1,t} = d_{p_2,t} = d_{p_3,t} = d$. Accordingly, the demand on each tunnel is the product between $d_{p_i,t}$ and $x_{p,t}$. The flows enter the network in an online manner, where the flow demand can vary with the time and it can be viewed as zero once the transmission of this flow is terminated. The flow demand in one future time slot can be predicted [4], [1]. We will discuss more general cases that the flow demand prediction has bounded errors in the next section. $P(f,t)$ represents the set of tunnels for flow f at t and $\mathbf{P}$ represents the set of tunnels for all flows in the network. The reason why we add the index t in $P(f,t)$ is that the number of tunnels may vary with time due to unexpected link failures. $\alpha_{p,t}$ and $\beta_{p,t}$ are the coefficients which can normalize rerouting cost and TE cost, respectively. For convenience, we summarize important notations in Table II.

TABLE II
KEY NOTATIONS IN THIS PAPER.

Input	F	The set of aggregate flows f from the source-destination switch pairs in the network
	V	The set of switches v
	E	The set of links e
	G	The directed network graph $G = (V, E)$
	L_e	The capacity of link e
	$P(f,t)$	The set of tunnels for flow f at t
	$\mathbf{P}$	The set of tunnels for all flows in the network. *i.e.*, $\mathbf{P} = \cup_{f\in F, t\in T} P(f,t)$
	$d_{p,t}$	The total demand of flow f at t, where $p \in P(f,t)$
	$\alpha_{p,t}$	The coefficients to normalize rerouting cost
	$\beta_{p,t}$	The coefficients to normalize TE cost
	c	A configurable parameter
Output	$x_{p,t}$	The allocated bandwidth for flow f at tunnel p
	$y_{p,t}$	The absolute value of difference between two variable $x_{p,t-1}$ and $x_{p,t}$

C. Problem Formulation

Based on the network model above, we begin to formulate our problem. The program (1) is a joint optimization on the network update and traffic engineering, *i.e.*, minimize the sum of the rerouting cost and the TE cost in the long run such that the network capacity constraints can be respected. The available tunnel set is given and how the flows are routed onto the available tunnels needs to be determined by TE. We use two-phase update protocol to guarantee consistency when rerouting flows.

$$\text{minimize} \quad \sum_{t=1}^{T}\sum_{p\in\mathbf{P}} \alpha_{p,t} y_{p,t} + \sum_{t=1}^{T}\sum_{p\in\mathbf{P}} \beta_{p,t} x_{p,t} \tag{1}$$

$$\text{subject to} \quad \sum_{p\in P(f,t)} x_{p,t} \geq 1, \quad \forall f \in F, \forall t \in T, \tag{1a}$$

$$\sum_{p|e\in p} d_{p,t} x_{p,t} \leq L_{e,t}, \quad \forall e \in E, \forall t \in T, \forall p \in \mathbf{P}, \tag{1b}$$

$$y_{p,t} \geq |x_{p,t} - x_{p,t-1}|, \quad \forall p \in \mathbf{P}, \forall t \in T, \tag{1c}$$

$$x_{p,t} \geq 0, \quad \forall p \in \mathbf{P}, \forall t \in T. \tag{1d}$$

The first term in the objective function captures the rerouting cost, *i.e.*, the cost that we perform network update to reroute flows from time slot $t-1$ to t. The second term is the TE cost that can characterize the transmission delay or the path cost at time slot t. Constraint (1a) is the flow demand conservation constraint, indicating how the flow is splitted at the ingress switch among its available tunnel set, which can be implemented using group table with `select` type [22]. Constraint (1b) indicates that the load at each link e should be less than or equal to link capacity $L_{e,t}$. In the constraint (1c), if the difference between two variables $x_{p,t}$ and $x_{p,t-1}$ is larger than zero, the rerouting cost comes from the operations of adding or modifying the rules in Openflow flow table or group table. Otherwise, it comes from the operations of deleting the rules. Without loss of generality, we can remove the absolute value sign and use $\max$ function instead in $\Lambda_1(\mathbf{x})$ of program (2). This can be fixed by folding the cost into the coefficient $\alpha_{p,t}$. Hence, we rewrite the program (1) to the program (2) by introducing the $\max$ function and thus the variables $y_{p,t}$ can be removed. For convenience, we denote $\Lambda_1(\mathbf{x})$ and $\Lambda_2(\mathbf{x})$ as the following.

$$\Lambda_1(\mathbf{x}) = \sum_{t=1}^{T}\sum_{p\in\mathbf{P}} \alpha_{p,t}\max\{0, x_{p,t}-x_{p,t-1}\}$$
$$\Lambda_2(\mathbf{x}) = \sum_{t=1}^{T}\sum_{p\in\mathbf{P}} \beta_{p,t}x_{p,t}$$

$$\begin{aligned} \text{minimize} \quad & \Lambda_1(\mathbf{x}) + \Lambda_2(\mathbf{x}) && (2)\\ \text{subject to} \quad & (1a), (1b), (1d). \end{aligned}$$

Now we can see that the constraint (1b) makes the program (2) harder to solve, *i.e.*, it is difficult to decouple the original problem into a series of subproblems. Hence, we lift this constraint into the objective function and transform program (2) to program (3). We will prove that the violation of constraint (1b) is bounded.

$$\Lambda_2'(\mathbf{x}) = \sum_{t=1}^{T}\sum_{p\in\mathbf{P}} \left(\beta_{p,t} + c\cdot d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\right)x_{p,t}$$

$$\begin{aligned} \text{minimize} \quad & \Lambda_1(\mathbf{x}) + \Lambda_2'(\mathbf{x}) && (3)\\ \text{subject to} \quad & (1a), (1d). && (3a) \end{aligned}$$

III. Online Algorithms

In this section we develop a set of provable online algorithms to solve our problem and conduct rigorous competitive and regret analysis. Inspired by [23], [24], we first develop a regularization-based algorithm where each flow's demand can be accurately predicted in a future time slot. Furthermore, we develop two algorithms — a RHC-based algorithm and an AFHC-based algorithm, which can predict the flow demand in multiple future time slots with bounded prediction errors.

A. Accurate Prediction in One Single Step

$$\begin{aligned} \text{minimize} \quad & \frac{1}{\eta}\sum_{p\in\mathbf{P}}\alpha_{p,t}\left(\left(x_{p,t}+\frac{\epsilon}{n}\right)\ln\left(\frac{x_{p,t}+\frac{\epsilon}{n}}{x_{p,t-1}+\frac{\epsilon}{n}}\right)-x_{p,t}\right)\\ & +\sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\left(\beta_{p,t}+c\cdot d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\right)x_{p,t} && (4)\\ \text{subject to} \quad & (1a), (1d). \end{aligned}$$

We first discuss the case that the flow demand $d_{p,t}$ can be accurately predicted in a future time slot. We introduce relative entropy function [25] to regularize the objective in the program, *i.e.*, replace the $\max$ function with a logarithmic function. We transform our program to the above program (4) and solve it in an iterative way using Algorithm 1.

Algorithm 1: A Regularization-based Algorithm

Input : The parameters $\epsilon > 0$, $\eta = \ln(1+\frac{n}{\epsilon})$ and $c > 0$.
Output: The solutions $\{\tilde{x}_{p,t}\}$

Initialize $\tilde{x}_{p,0} = 0$, $\forall p \in \mathbf{P}$.
for $t = 1 \to T$ **do**
  Obtain the solution $\{\tilde{x}_{p,t}\}$ via substituting $\{\tilde{x}_{p,t-1}\}$ into the program (4), *i.e.*, solving the program (4) iteratively.

Before analyzing the performance of Algorithm 1, we first introduce related definitions.

Definition 1. *Let SL^* and SL be the objective function value of program* (2) *and program* (3) *when obtaining $\{\tilde{x}_{p,t}\}$ from Algorithm 1,* i.e.*, $SL^* = \Lambda_1(\tilde{x}_{p,t}) + \Lambda_2(\tilde{x}_{p,t})$ and $SL = \Lambda_1(\tilde{x}_{p,t}) + \Lambda_2'(\tilde{x}_{p,t})$*

Definition 2. *Let OPT^* and OPT be the optimal solution of program* (2) *and program* (3)*.*

Definition 3. *Let $\{x^*_{p,t}\}$ be the optimal solutions in program* (2)*,* i.e.*, $OPT^* = \Lambda_1(x^*_{p,t}) + \Lambda_2(x^*_{p,t})$*

Lemma 1. *[25] $SL \le \tau OPT$, where $\tau = (1+\epsilon')\log(1+\frac{k}{\epsilon'})$, $\epsilon' = \frac{\epsilon n}{k}$, $k = \max_{f,t}|P_{f,t}|$.*

Remark: Lemma 1 indicates that Algorithm 1 can produce a solution with constant competitive ratio to program (3) when using the regularization-based decomposition.

Theorem 1. *Algorithm 1 is $\tau\cdot(c\cdot M+1)$-competitive and the average load violation on each link is bounded by $\tau\left(\frac{1}{c\cdot m}+1\right)$ in time horizon,* i.e.*, Algorithm 1 outputs the solutions $\{\tilde{x}_{p,t}\}$ that satisfy the following two conditions:*
(R1): $SL^* \le \tau\cdot(c\cdot M+1)\cdot OPT^*$
(R2): $\sum_{t=1}^{T}\sum_{e\in E}\frac{1}{L_{e,t}}\sum_{p|e\in p} d_{p,t}\tilde{x}_{p,t} \le \tau\left(\frac{1}{c\cdot m}+1\right)|T||E|$
where $\tau = (1+\epsilon')\log(1+\frac{k}{\epsilon'})$, $M = \max_{p,t}\left\{\frac{d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}}{\beta_{p,t}}\right\}$ and $m = \min_{p,t}\left\{\frac{d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}}{\alpha_{p,t}+\beta_{p,t}}\right\}$.

Proof. We first prove the condition (R1). From Lemma 1, we obtain

$$SL^* \le SL \le \tau\cdot OPT \tag{5}$$

According to Definition 2 and Definition 3, we can know that $x^*_{p,t}$ is the optimal solution in program (2). Hence, it is a feasible solution in program (3).

$$\begin{aligned} OPT &\le \Big(\sum_{t=1}^{T}\sum_{p\in\mathbf{P}} \alpha_{p,t}\max\{0, x^*_{p,t} - x^*_{p,t-1}\} \\ &\quad + \sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\left(\beta_{p,t} + c\cdot d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\right)x^*_{p,t}\Big) \\ &\le (cM+1)\left(\sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\alpha_{p,t}\max\{0, x^*_{p,t} - x^*_{p,t-1}\} + \sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\beta_{p,t}x^*_{p,t}\right) \\ &= (cM+1)OPT^* \end{aligned} \tag{6}$$

Combining the equation (5) and (6), the condition (R1) holds. Next we begin to prove that the condition (R2) holds. The average link load $\bar{B}$ in the time horizon can be defined as:

$$\frac{\sum_{t=1}^{T}\sum_{e\in E}\frac{1}{L_{e,t}}\sum_{p|e\in p} d_{p,t}\tilde{x}_{p,t}}{|T||E|} = \frac{\sum_{t=1}^{T}\sum_{p\in\mathbf{P}} d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\tilde{x}_{p,t}}{|T||E|} \tag{7}$$

Similarly, from Lemma 1 and Definition 3, we obtain

$$\begin{aligned} SL &\le \tau \times OPT \\ &\le \tau(\sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\alpha_{p,t}\max\{0, x^*_{p,t} - x^*_{p,t-1}\} \\ &\quad + \sum_{t=1}^{T}\sum_{p\in\mathbf{P}}(\beta_{p,t} + c\cdot d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}})x^*_{p,t}) \\ &\le \tau\sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\left(\alpha_{p,t} + \beta_{p,t} + c\cdot d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\right)x^*_{p,t} \\ &\le \tau\left(\frac{1}{m}+c\right)\sum_{t=1}^{T}\sum_{e\in E}\frac{1}{L_{e,t}}\sum_{p|e\in p} d_{p,t}x^*_{p,t} \end{aligned} \tag{8}$$

According to Definition 1, we derive that

$$c\sum_{t=1}^{T}\sum_{p\in\mathbf{P}} d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\tilde{x}_{p,t} \le SL \tag{9}$$

Combining the inequation (7), (8) and (9), we have

$$\begin{aligned} \bar{B} &= \frac{c\sum_{t=1}^{T}\sum_{p\in\mathbf{P}} d_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}\tilde{x}_{p,t}}{c|T||E|} \le \frac{SL}{c|T||E|} \\ &\le \frac{\tau}{c|T||E|}\left(\frac{1}{m}+c\right)\sum_{t=1}^{T}\sum_{e\in E}\frac{1}{L_{e,t}}\sum_{p|e\in p} d_{p,t}x^*_{p,t} \\ &\le \tau\left(\frac{1}{cm}+1\right) \end{aligned} \tag{10}$$

Hence, the condition (R2) holds. □

Remark: Theorem 1 indicates that Algorithm 1 has also a constant competitive ratio to program (1) and (2) and the violation of the constraint (1b) is bounded. Note that we can make the constraint (1b) feasible by scaling down all allocated bandwidth, since a (γ_1,γ_2)-competitive algorithm and a ($\gamma_1\cdot\gamma_2$,1)-competitive algorithm is equivalent in nature [26].

B. Bounded Error Prediction with Multiple Steps

We develop (RHC-based) Algorithm 2 and (AFHC-based) Algorithm 3 that can predict the flow demand with multiple steps, *i.e.*, if the prediction window size is w, we can predict the flow demand at $\tau+1, \cdots, \tau+w$ when the current time slot is τ. We first give a novel definition of our prediction error model, which is more practical and significantly different from that in the prior work [23], [24].

Definition 4. *The actual flow demand $d_{p,t}$ and the corresponding prediction $d_{p,t|\tau}$ satisfies $|d_{p,t} - d_{p,t|\tau}| \le \delta(t-\tau)$, where $\delta(t-\tau)$ is a bounded prediction error, $d_{p,t|\tau}$ indicates the predicted flow demand $d_{p,t}$ in the future time t when we are being at time τ ($\tau < t$).*

Remark: $\delta(t-\tau)$ captures the upper bound of prediction error and it becomes larger with the increment of the time gap $t-\tau$.

Now we explain the high-level working of Algorithm 2. It solves a cost minimization problem at t over the prediction window $[t, t+w]$ and obtains a set of solutions $\{x^{FHC}_{p,t:t+w}\}$ (line 3). We choose the latest solution $x^{FHC}_{p,t}$ as its value in each iteration (line 4). For example, when $t=1$, we obtain the solutions $\{x^{FHC}_{p,1:1+w}\}$ over the window $[1, 1+w]$ and choose $x^{FHC}_{p,1}$. When $t=2$, we obtain the solutions $\{x^{FHC}_{p,2:2+w}\}$ over the window $[2, 2+w]$ and choose $x^{FHC}_{p,2}$, and so on.

Algorithm 2: A RHC-based Algorithm

Input : The prediction window size w.
Output: The solutions $\{x^{RHC}_{p,t}\}$

1 Initialize $x^{RHC}_{p,0} = 0$, $\forall p \in \mathbf{P}$.
2 **for** $t = 1 \to T$ **do**
3 $\quad\{x^{FHC}_{p,t:t+w}\} = \arg\min_x \sum_{\tau=t}^{t+w}\sum_{p\in\mathbf{P}}[\alpha_{p,\tau}y_{p,\tau} + (\beta_{p,\tau} + c\cdot d_{p,\tau|t}\sum_{e\in p}\frac{1}{L_{e,\tau}})x_{p,\tau}]$
4 $\quad x^{RHC}_{p,t} = x^{FHC}_{p,t}$

Lemma 2. *$cost(STA_{new}) - cost(STA_{old}) \le c\cdot|E|\cdot|T|$, where $cost(STA_{new})$ and $cost(STA_{old})$ is the objective value resulting from the solution x^*_p and $\hat{x}_p$, where the definition of x^*_p and $\hat{x}_p$ is shown in Fig. 6.*

Proof. From the definition of x^*_p in Fig. 6, we can derive that,

$$\begin{aligned} &cost(STA_{new}) - cost(STA_{old}) \\ &= \sum_{p\in\mathbf{P}}\alpha_{p,t}x^*_p + \sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\left[\beta_{p,t} + cd_{p,t|\tau}\sum_{e\in p}\frac{1}{L_{e,t}}\right]x^*_p \\ &\quad - \sum_{p\in\mathbf{P}}\alpha_{p,t}\hat{x}_p - \sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\beta_{p,t}\hat{x}_p \\ &\le \sum_{t=1}^{T}\sum_{p\in\mathbf{P}} cd_{p,t|\tau}\sum_{e\in p}\frac{1}{L_{e,t}}\hat{x}_p = \sum_{t=1}^{T} c\sum_{e\in E}\sum_{p|e\in p}\frac{d_{p,t}\hat{x}_p}{L_{e,t}} \\ &\le c\cdot|E|\cdot|T| \end{aligned}$$

□

Remark: The first inequation follows the minimum property of x^*_p (Fig. 6) and the second follows the constraint (1b).

Theorem 2. *The regret of Algorithm 2 is* $regret(RHC) = \mathcal{O}\left(T \cdot \max_{p,t}\left\{\sum_{p\in\mathbf{P}}\left[c\cdot\delta(0)\sum_{e\in p}\frac{1}{L_{e,t}}+\alpha_{p,t+1}\right]\right\}\right) + c|E||T|$, *where* $\delta(0) = \sup_t\left\{|d_t - d_{t|t}|\right\}$.

Proof. The regret of Algorithm 2 is

$$\begin{aligned} &regret(RHC) = cost(RHC) - cost(STA_{old}) = cost(RHC) \\ &- cost(STA_{new}) + cost(STA_{new}) - cost(STA_{old}) \end{aligned} \tag{11}$$

We define $\boldsymbol{\xi_i} = (x_{p,1}^{RHC}, x_{p,2}^{RHC}, \cdots, x_{p,i}^{RHC}, x_{p,i+1}^{*} \cdots, x_{p,T}^{*})$ to affiliate our proof $(i = 0, 1, ..., T)$, specifically,

$$\xi_{i,t} = \begin{cases} x_{p,t}^{RHC} & t \le i \\ x_p^* & i < t \le T \end{cases}$$

where x_p^* is defined in Fig. 6. Accordingly, we have

$$cost(\boldsymbol{\xi_0}) = cost(x_{p,1}^{*}, x_{p,2}^{*}, \cdots, x_{p,T}^{*}) = cost(STA_{new}(1,T))$$

$$cost(\boldsymbol{\xi_T}) = cost(x_{p,1}^{RHC}, x_{p,2}^{RHC}, \cdots, x_{p,T}^{RHC}) = cost(RHC)$$

We introduce $\{\xi\}$ sequences and the regret can be transformed to the accumulation of the difference between two adjacent terms — $cost(\xi_t)$ and $cost(\xi_{t-1})$, which is used to solve the optimization at the interval $[t, t+1]$. From the proof in Fig. 6,

$$\begin{aligned} &cost(RHC) - cost(STA_{new}) = \sum_{t=1}^{T} cost(\boldsymbol{\xi_t}) - cost(\boldsymbol{\xi_{t-1}}) \\ &\le \sum_{t=1}^{T}\sum_{p\in\mathbf{P}}\left[c\delta(0)\sum_{e\in p}\frac{1}{L_{e,t}} + \alpha_{p,t+1}\right] \\ &= \mathcal{O}\left(T\cdot\max_{p,t}\left\{\sum_{p\in\mathbf{P}}\left[c\delta(0)\sum_{e\in p}\frac{1}{L_{e,t}}+\alpha_{p,t+1}\right]\right\}\right) \end{aligned} \tag{12}$$

Lemma 2, inequation (11) and (12) end our proof. □

Remark: Theorem 2 indicates that the regret of Algorithm 2 is proportional to $\delta(0)$ and cannot be affected by the prediction window size w.

Since RHC cannot benefit the window size increase especially when the number of tunnels is more than one. This motivates us to develop Algorithm 3 to make use of the look-ahead value. Instead of choosing the latest solution in Algorithm 2, Algorithm 3 computes an average of solutions for $w+1$ prediction windows as its value (line 4).

Algorithm 3: An AFHC-based Algorithm

Input : The prediction window size w.
Output: The solutions $\{x_{p,t}^{AFHC}\}$

1 Initialize $x_{p,0}^{AFHC} = 0$, $\forall p \in \mathbf{P}$.
2 **for** $t = 1 \to T$ **do**
3 $\quad \{x_{p,t:t+w}^{FHC(t \mod (w+1))}\} = \arg\min_x \sum_{\tau=t}^{t+w}\sum_{p\in P}[\alpha_{p,\tau}y_{p,\tau} + (\beta_{p,\tau} + c\cdot d_{p,\tau|t}\sum_{e\in p}\frac{1}{L_{e,\tau}})x_{p,\tau}]$
4 $\quad x_{p,t}^{AFHC} = \frac{1}{w+1}\sum_{n=1}^{w+1} x_{p,t}^{FHC(n)}$

Theorem 3. *The regret of Algorithm 3 is* $regret(AFHC) = \mathcal{O}\left(T\max_{p,t,\tau}\left\{\sum_{p\in P}\left[\frac{\alpha_{p,\tau}}{w+1} + c\delta(t-\tau)\sum_{e\in p}\frac{1}{L_{e,t}}\right]\right\}\right) + c|E||T|$.

Proof. Without loss of generality, we define $x_p^*(t)$ as the following equation.

$$x_p^*(t) = \begin{cases} x_p^* & t \ge 1 \\ 0 & t \le 0 \end{cases}$$

From Fig. 7, we can derive that

$$\begin{aligned} &cost(AFHC) - cost(STA_{new}) \\ &\le \frac{1}{w+1}\sum_{\tau=1-w}^{T}[cost(FHC(\tau, \tau+w)) - cost(STA_{new}(\tau, \tau+w))] \\ &= O\left(T\max_{p,t,\tau}\left\{\sum_{p\in P}\left[\frac{\alpha_{p,\tau}}{w+1} + c\delta(t-\tau)\sum_{e\in p}\frac{1}{L_{e,t}}\right]\right\}\right) \end{aligned} \tag{13}$$

Lemma 2 and inequation (13) end our proof. □

Remark: Theorem 3 indicates that the regret of Algorithm 3 is proportional to $\max\{\delta\}$ and the performance may be improved when enlarging the prediction window size w. Since $\max\{\delta\} \ge \delta(0)$, Algorithm 3 is usually more sensitive to prediction errors than Algorithm 2.

Theorem 4. *The competitive ratio of Algorithm 3 is* $(cM+1)\rho$, *where* $\rho = \frac{cost(AFHC(1,T))}{OPT}$.

Proof. According to line 4 in Algorithm 3, we can obtain $\left(1 - \max_{t,\tau}\left\{\frac{\delta(t-\tau)}{d_{p,t}}\right\}\right)\rho \le 1 + \frac{1}{w+1}\max_{p,t}\frac{\alpha_{p,t}}{\beta_{p,t}+cd_{p,t}\sum_{e\in p}\frac{1}{L_{e,t}}}$.

Combining inequation (6), the competitive ratio of Algorithm 3 is

$$\frac{cost(AFHC(1,T))}{OPT^*} \le (cM+1)\frac{cost(AFHC(1,T))}{OPT} = (cM+1)\rho$$

□

IV. Experimental Evaluation

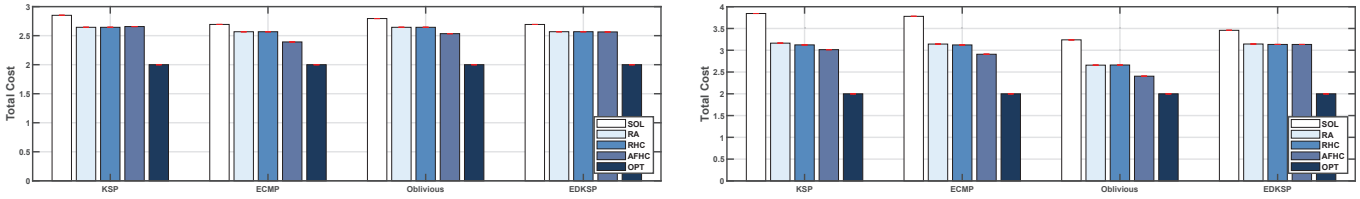
We conduct extensive experiments to evaluate our algorithms in this section.

Setup: We consider two realistic topologies and the traffic matrices of them are obtained from [27].

- **Geant topology**: It has 22 nodes and 72 links in total. The corresponding dataset for this topology is produced by 15-minute time slots over four months.
- **Nobel-Germany topology**: It has 17 nodes and 52 links in total. The corresponding dataset for this topology is produced by 5-minute time slots over one day.

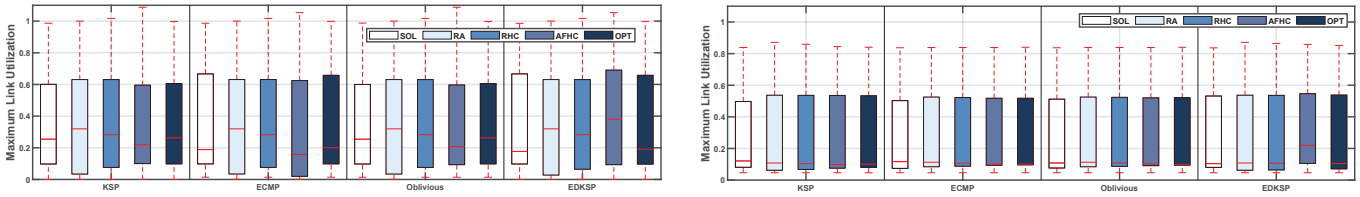
Benchmark Schemes: We evaluate the following schemes.

- **SOL**: The SDN application optimization APIs [19]. We implement SOL with the objective of minimizing the cost [9] in one single time slot and does not take network update cost into consideration.
- **RA**: Our proposed regularization-based algorithm shown in Algorithm 1.
- **RHC**: Our proposed RHC-based algorithm shown in Algorithm 2.
- **AFHC**: Our proposed AFHC-based algorithm shown in Algorithm 3.
- **OPT**: Offline optimum that assumes all future inputs are known as priori.



(a) Geant topology (b) Nobel-Germany topology

Fig. 2. Normalized total cost variations of different schemes under KSP, ECMP, Oblivious and EDKSP.



(a) Geant topology (b) Nobel-Germany topology

Fig. 3. Maximum link utilization variations of different schemes under KSP, ECMP, Oblivious and EDKSP.

We take the time series [28] to predict the flow demand, which can be an input of our prediction-based algorithms. Furthermore, in order to explore the influence resulting from different tunnels, we use **KSP** (k-shortest paths), **ECMP** (equal-cost multipath routing), **Oblivious** (Räcke's oblivious routing [11], [29], [30]) and **EDKSP** (edge-disjoint k-shortest paths) to generate the candidate tunnel set for each ingress-egress switch pairs, where k ranges from 2 to 6. The ingress switches can configure the split ratio accordingly.

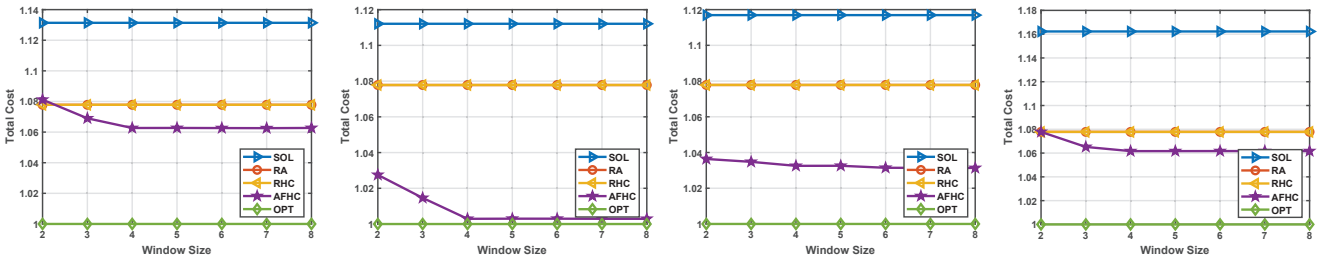
Experiment Results: As shown in Fig. 2, we first investigate the total cost of each scheme under different routing algorithms. We can observe that AFHC performs best both in Geant topology and Nobel-Germany topology, whose cost is only 14.4% larger than that of OPT on average. The essential reason is that AFHC is an average of several future time steps using look-ahead values, which takes fully advantage of prediction and explores the properties of different routes, leading to a relatively low cost. SOL may occasionally obtain the same TE cost as other schemes, but it always ends up with a higher network update cost. RA lifts the capacity constraint into the objective function and takes network update cost into account, leading to reduced total cost. Regarding for RHC and AFHC, we discovered that even though both of them integrate the look-ahead information from the prediction window, the performance is different. AFHC can obtain a better solution, but it may introduce $\max\{\delta\}$ in terms of the regret and competitive ratio analysis (Theorem 3), which makes the solution sensitive to prediction errors. On the contrary, RHC just introduces $\delta(0)$ ($\delta(0)$ approaches zero as it can be claimed that the latest prediction is more accurate than a relatively far prediction) in terms of its regret and competitive ratio analysis (Theorem 2), where δ is the prediction error.

Fig. 3 shows the maximum link utilization under different routes. Intuitively, when the link utilization is larger than one, the congestion may happen. A larger value indicates more severe congestion in the network. Note that the link utilization of SOL is always less than one due to the link capacity constraints. SOL tends to pick the tunnel with the least cost and thus makes some links heavily loaded. RA, RHC and AFHC lift the capacity constraints into the objective function to obtain a stable solution in the long run. Hence, the link utilization may be slightly larger than one and Theorem 1 claims that the link capacity violation can be bounded. In Fig. 3(a), we can observe that the link capacity violation is within 6% for RA, RHC and AFHC. We argue that this can be acceptable in practice because today's commercial switches are deeply buffered [31] and many applications can tolerate temporary rate reduction [32]. Furthermore, state-of-the-art transport protocols [33] can detect congestion early with the additional function in the programmable switch [34] to adjust sending rate in a fine granularity.

Fig. 4 and Fig. 5 show that the total cost variations with different window sizes. Without loss of generality, we normalize the value of OPT to one. From Fig. 4 and Fig. 5, we can see that the window size has no impact on the cost produced from SOL, RA, and OPT, since their optimizations don't involve the prediction window. Obviously, a larger window size can benefit AFHC significantly, which can perfectly match the statement in Theorem 3, *i.e.*, the total cost of AFHC decreases when the prediction window size becomes large. The total cost of RHC is relatively stable and we can also observe a weak correlation between the performance and window size as Theorem 2 claims — the regret of Algorithm 2 is proportional to $\delta(0)$ and cannot be affected by the window size w.

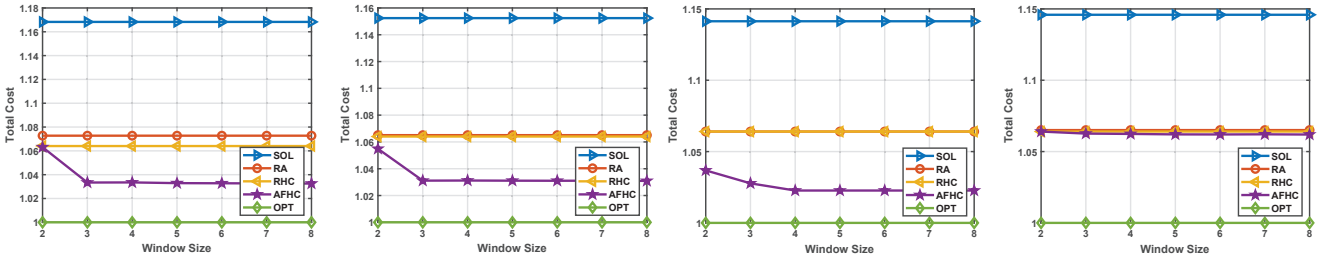
V. Related Work

Optimization in Traffic Engineering: SOL [19] developed an optimization API that enables the applications to model the optimization objective and constraints. In terms of performance and robustness optimization in TE, Kumar et al. [11] presented an approach using a set of paths by running Räcke's



(a) KSP (b) ECMP (c) Oblivious (d) EDKSP

Fig. 4. Total cost comparisons with different window sizes in Geant topology.



(a) KSP (b) ECMP (c) Oblivious (d) EDKSP

Fig. 5. Total cost comparisons with different window sizes in Germany-Nobel topology.

oblivious routing [30], and a centralized controller could accordingly adjust the sending rates. Furthermore, CG4SR [13] applied the column generation method to the TE optimization. To tradeoff between the bandwidth consumption and rerouting overheads, Kuo et al. [14] designed an incremental tree updating algorithm and a multi-tree update scheduling algorithm. For the node-constrained TE, Trimponias et al. [35] conducted a rigorous theoretical analysis where the traffic needs to pass through a set of middleboxes. Tseng et al. [36] investigated a rate adaptation planning problem under the reconfiguration delay and introduced the perseverance constraints to handle the temporary disruption. TeaVaR [12] allowed the operators to optimize the bandwidth allocation and provide provable availability, which leveraged the empirical dataset to generate a probabilistic model in terms of network failures and maximize the bandwidth allocation to satisfy an operator-specified availability target. Sentinel [37] aimed to reduce failure-induced packet losses and integrated fast failover functions into TE.

Network Rerouting in SDNs: SWAN [4] and B4 [3] tried to determine a congestion-free update plan in inter-datacenter WANs at the expense of parts of the link capacity. Instead of seeking for a congestion-free update plan, Zheng et al. [38] advocated to find an update plan that minimizes the transient congestion, which can significantly improve the bandwidth utilization. However, it cannot be applied to latency-sensitive data transfers. Furthermore, Dionysus [16] employed dependency graphs to determine a fast update plan according to switch runtime conditions. To reduce the dependency complexity, Cupid [17] divided global update dependencies among switches into a set of local restrictions to minimize the cost. Another work by Ludwig et al. [39] aimed to minimizing the number of interactions when transitioning from the initial to the final update stage. The authors proved that finding a shortest update sequence that avoids forwarding loops is NP-hard. Later, they considered secure network updates [40] in the presence of middleboxes such as firewalls and NAT.

Though the idea of TE optimization and network update have surfaced in the literature, the novelty of our work lies in a comprehensive exploration of the online joint optimization and a rigorous competitive and regret analysis, which to our knowledge has not been done before.

VI. Conclusion

In this paper, we proposed an online optimization framework for jointly minimizing TE cost and network update cost in the time horizon. Furthermore, we developed a set of provable online algorithms and conduct rigorous competitive and regret analysis. They can handle the cases that the demand of each flow can be accurately predicted or they have bounded prediction errors. Finally, extensive evaluations using two empirical topologies with real dataset demonstrated that our online algorithms can decrease the total cost and close to the offline optimum.

Appendix

The detailed proof can be found in Fig. 6, Fig. 7 and Fig. 8.

Acknowledgment

We thank the anonymous reviewers for their helpful comments on drafts of this paper. This work was supported in part by the National Key Research and Development Program of China (2018YFB1004700), China NSF grants (61802172, 61832005, 61872178, 61972254), China NSF of Jiangsu Province (BK20201248, BK20181251) and Open Fund of PDL (WDZC20205500109).

We denote $x_p^* = \arg\min_x \sum_{p\in P} \alpha_{p,t} x_p + \sum_{t=1}^{T} \sum_{p\in P} \left[\beta_{p,t} + cd_{p,t|\tau} \sum_{e\in P} \frac{1}{L_{e,t}}\right] x_p$ and $\hat{x}_p = \arg\min_x \sum_{p\in P} \alpha_{p,t} x_p + \sum_{t=1}^{T} \sum_{p\in P} \beta_{p,t} x_p$

$x_{p,\tau}^{RHC} = \arg\min_x \sum_{p\in P} \left[\alpha_{p,\tau}(x - x_{p,\tau-1}^{RHC})^+\right] + \sum_{p\in P} \alpha_{p,\tau+1}(x_{p,\tau+1}^{FHC} - x)^+ + (\beta_{p,\tau} + c \cdot d_{p,\tau|\tau} \sum_{e\in P} \frac{1}{L_{e,\tau}})x$

$\because cost(\boldsymbol{\xi_t}) - cost(\boldsymbol{\xi_{t-1}}) = \sum_{p\in P} \left[\alpha_{p,t}(x_{p,t}^{RHC} - x_{p,t-1}^{RHC})^+\right] + \sum_{p\in P} \alpha_{p,t+1}(x_p^* - x_{p,t}^{RHC})^+ + (\beta_{p,t} + cd_{p,t} \sum_{e\in P} \frac{1}{L_{e,t}})x_{p,t}^{RHC}$

$- \sum_{p\in P} \left[\alpha_{p,t}(x_p^* - x_{p,t-1}^{RHC})^+ + (\beta_{p,t} + cd_{p,t} \sum_{e\in P} \frac{1}{L_{e,t}})x_p^*\right]$

$\therefore cost(\boldsymbol{\xi_t}) - cost(\boldsymbol{\xi_{t-1}})$

$\leq cost(\boldsymbol{\xi_t}) - cost(\boldsymbol{\xi_{t-1}}) + \sum_{p\in P} \left[\alpha_{p,t}(x_p^* - x_{p,t-1}^{RHC})^+ + \alpha_{p,t+1}(x_{p,t+1}^{FHC} - x_p^*)^+\right] + \sum_{p\in P} (\beta_{p,t} + cd_{p,t|t} \sum_{e\in P} \frac{1}{L_{e,t}})x_p^*$

$- \sum_{p\in P} \left[\alpha_{p,t}(x_{p,t}^{RHC} - x_{p,t-1}^{RHC})^+ + \alpha_{p,t+1}(x_{p,t+1}^{FHC} - x_{p,t}^{RHC})^+\right] - \sum_{p\in P} (\beta_{p,t} + cd_{p,t|t} \sum_{e\in P} \frac{1}{L_{e,t}})x_{p,t}^{RHC}$

$\leq \sum_{p\in P} \left\{ \left[c(d_{p,t} - d_{p,t|t}) \sum_{e\in P} \frac{1}{L_{e,t}}\right] (x_{p,t}^{RHC} - x_p^*) + \alpha_{p,t+1}[(x_{p,t+1}^{RHC} - x_p^*)^+ + (x_p^* - x_{p,t+1}^{RHC})^+] \right\} \leq \sum_{p\in P} \left[c\delta(0) \sum_{e\in P} \frac{1}{L_{e,t}} + \alpha_{p,t+1}\right]$

Fig. 6. The detailed proof that $cost(\xi_t) - cost(\xi_{t-1})$ can be bounded.

$cost(FHC(\tau, \tau+w)) - cost(STA_{new}(\tau, \tau+w))$

$= \sum_{p\in P} \left[\alpha_{p,\tau}(x_{p,\tau}^{FHC} - x_{p,\tau-1}^{AFHC})^+ + \sum_{t=\tau+1}^{\tau+w} \alpha_{p,t}(x_{p,t}^{FHC} - x_{p,t-1}^{FHC})^+\right] + \sum_{p\in P} \sum_{t=\tau}^{\tau+w} \left(\beta_{p,t} + cd_{p,t} \sum_{e\in P} \frac{1}{Le,t} x_{p,t}^{FHC}\right)$

$- \sum_{t=\tau+1}^{\tau+w} \sum_{p\in P} \left\{\alpha_{p,t}[x_p^*(t) - x_p^*(t-1)]^+ + \left(\beta_{p,t} + cd_{p,t} \sum_{e\in P} \frac{1}{Le,t} x_p^*(t)\right)\right\}$

$\leq cost(FHC(\tau, \tau+w)) - cost(STA_{new}(\tau, \tau+w)) + \sum_{p\in P} \left[\alpha_{p,\tau}(x_p^*(\tau) - x_{p,\tau-1}^{AFHC})^+ + \sum_{t=\tau+1}^{\tau+w} \alpha_{p,t}(x_p^*(t) - x_p^*(t-1))^+\right]$

$+ \sum_{p\in P} \sum_{t=\tau}^{\tau+w} \left(\beta_{p,t} + cd_{p,t|\tau} \sum_{e\in P} \frac{1}{Le,t} x_p^*(t)\right) - \sum_{p\in P} \left[\alpha_{p,\tau}(x_{p,\tau}^{FHC} - x_{p,\tau-1}^{AFHC})^+ + \sum_{t=\tau+1}^{\tau+w} \alpha_{p,t}(x_{p,t}^{FHC} - x_{p,t-1}^{FHC})^+\right]$

$- \sum_{p\in P} \sum_{t=\tau}^{\tau+w} \left(\beta_{p,t} + cd_{p,t|\tau} \sum_{e\in P} \frac{1}{Le,t} x_{p,t}^{FHC}\right)$

$= \sum_{p\in P} \alpha_{p,\tau} \left\{[x_p^*(\tau) - x_{p,\tau-1}^{AFHC}]^+ - [x_p^*(\tau) - x_p^*(\tau-1)]^+\right\} + \sum_{t=\tau}^{\tau+w} \sum_{p\in P} c(d_{p,t} - d_{p,t|\tau}) \sum_{e\in p} \frac{1}{L_{e,t}} [x_{p,t}^{FHC} - x_p^*(t)]$

$\leq \sum_{p\in P} \left[\alpha_{p,\tau} + \sum_{t=\tau}^{\tau+w} c\delta(t-\tau) \sum_{e\in p} \frac{1}{L_{e,t}}\right]$

Fig. 7. The detailed proof that $cost(FHC(\tau, \tau+w)) - cost(STA_{new}(\tau, \tau+w))$ can be bounded.

$cost(FHC(\tau, \tau+w)) \leq cost(FHC(\tau, \tau+w)) + \sum_{t=\tau+1}^{\tau+w} \sum_{p\in P} \alpha_{p,t}(x_{p,t}^{OPT} - x_{p,t-1}^{OPT})^+ + \sum_{p\in P} \alpha_{p,\tau}(x_{p,\tau}^{OPT} - x_{p,\tau-1}^{AFHC})^+ +$

$\sum_{t=\tau}^{\tau+w} \sum_{p\in P} \left(\beta_{p,t} + cd_{p,t|\tau} \sum_{e\in p} \frac{1}{L_{e,t}} x_{p,t}^{OPT}\right) - \sum_{t=\tau+1}^{\tau+w} \sum_{p\in P} \alpha_{p,t}(x_{p,t}^{FHC} - x_{p,t-1}^{FHC})^+$

$- \sum_{p\in P} \alpha_{p,\tau}(x_{p,\tau}^{FHC} - x_{p,\tau-1}^{AFHC})^+ - \sum_{t=\tau}^{\tau+w} \sum_{p\in P} \left(\beta_{p,t} + cd_{p,t|\tau} \sum_{e\in p} \frac{1}{L_{e,t}} x_{p,t}^{FHC}\right)$

$\leq \sum_{t=\tau+1}^{\tau+w} \sum_{p\in P} c\delta(t-\tau) \sum_{e\in p} \frac{1}{L_{e,t}} x_{p,t}^{FHC} + OPT(\tau, \tau+w) + \sum_{p\in P} \alpha_{p,\tau}(x_{p,\tau}^{OPT} - x_{p,\tau-1}^{AFHC})^+$

$\therefore \frac{cost(AFHC(1,T))}{OPT(1,T)} = \frac{1}{w+1} \sum_{t=1-w}^{T} \frac{cost(FHC(\tau,\tau+w))}{OPT(1,T)}$

$\leq 1 + \frac{1}{w+1} \sum_{t=1-w}^{T} \frac{\sum_{p\in P} \alpha_{p,t}(x_{p,t}^{OPT} - x_{p,t-1}^{AFHC})^+}{OPT(1,T)} + \frac{1}{w+1} \sum_{t=1-w}^{T} \frac{\sum_{t=\tau}^{\tau+w} \sum_{p\in P} c\delta(t-\tau) \sum_{e\in p} \frac{1}{L_{e,t}} x_{p,t}^{FHC}}{OPT(1,T)}$

$\leq 1 + \frac{1}{w+1} \sum_{t=1-w}^{T} \frac{\sum_{p\in P} \alpha_{p,t} x_{p,t}^{OPT}}{\sum_{t=1}^{T} \sum_{p\in P} (\beta_{p,t} + cd_{p,t} \sum_{e\in p} \frac{1}{L_{e,t}}) x_{p,t}^{OPT}} + \max_{t,\tau}\{\frac{\delta(t-\tau)}{d_{p,t}}\} \sum_{t=1-w}^{T} \frac{\sum_{p\in P} cd_{p,t} \sum_{e\in p} \frac{1}{L_{e,t}} x_{p,t}^{AFHC}}{OPT(1,T)}$

$\leq 1 + \frac{1}{w+1} \max_{p,t} \frac{\alpha_{p,t}}{\beta_{p,t} + cd_{p,t} \sum_{e\in p} \frac{1}{L_{e,t}}} + \max_{t,\tau}\{\frac{\delta(t-\tau)}{d_{p,t}}\} \sum_{t=1-w}^{T} \frac{\sum_{p\in P} cd_{p,t} \sum_{e\in p} \frac{1}{L_{e,t}} x_{p,t}^{AFHC}}{OPT(1,T)}$

Fig. 8. The competitive ratio analysis of Algorithm 3.

REFERENCES

[1] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined wan," in *SIGCOMM*, 2013, pp. 3–14.

[2] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandeistavila *et al.*, "Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing," in *SIGCOMM*, 2015, pp. 1–14.

[3] C. Hong, S. Mandal, M. Alfares, M. Zhu, R. Alimi, B. K. Naidu, C. Bhagat, S. Jain, J. Kaimal, S. Liang *et al.*, "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *SIGCOMM*, 2018, pp. 74–87.

[4] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *SIGCOMM*, 2013, pp. 15–26.

[5] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, "Numfabric: Fast and flexible bandwidth allocation in datacenters," in *SIGCOMM*, 2016, pp. 188–201.

[6] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[7] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," in *SIGCOMM*, 2014, pp. 515–526.

[8] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 579–595, 2017.

[9] S. Paris, A. Destounis, L. Maggi, G. S. Paschos, and J. Leguay, "Controlling flow reconfigurations in SDN," in *INFOCOM*, 2016, pp. 1–9.

[10] K. He, J. Khalid, A. Gember-Jacobson, S. Das, C. Prakash, A. Akella, L. E. Li, and M. Thottan, "Measuring control plane latency in sdn-enabled switches," in *SOSR*, 2015, pp. 25:1–25:6.

[11] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-oblivious traffic engineering: The road not taken," in *NSDI*, 2018, pp. 157–170.

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

[13] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "Cg4sr: Near optimal traffic engineering for segment routing with column generation," in *INFOCOM*, 2019, pp. 1333–1341.

[14] J.-J. Kuo, S.-H. Chiang, S.-H. Shen, D.-N. Yang, and W.-T. Chen, "Dynamic multicast traffic engineering with efficient rerouting for software-defined networks," in *INFOCOM*, 2019, pp. 793–801.

[15] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. A. Maltz, "zupdate: updating data center networks with zero loss," in *SIGCOMM*, 2013, pp. 411–422.

[16] X. Jin, H. H. Liu, X. Wu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *SIGCOMM*, 2014, pp. 539–550.

[17] W. Wang, W. He, J. Su, and Y. Chen, "Cupid: Congestion-free consistent data plane update in software defined networks," in *INFOCOM*, 2016, pp. 1–9.

[18] G. Li, Y. R. Yang, F. Le, Y.-s. Lim, and J. Wang, "Update algebra: Toward continuous, non-blocking composition of network updates in sdn," in *INFOCOM*, 2019, pp. 1081–1089.

[19] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using sol," in *NSDI*, 2016, pp. 223–237.

[20] L. L. H. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman, "A tale of two metrics: simultaneous bounds on competitiveness and regret," in *SIGMETRICS*, 2013, pp. 329–330.

[21] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, K. Shiomoto, and T. Hashimoto, "Hierarchical model predictive traffic engineering," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1754–1767, 2018.

[22] "Openflow switch specification," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[23] J. Comden, S. Yao, N. Chen, H. Xing, and Z. Liu, "Online optimization in cloud resource provisioning: Predictions, regrets, and algorithms," *POMACS*, vol. 3, no. 1, pp. 16:1–16:30, 2019.

[24] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *IGCC*, 2012, pp. 1–10.

[25] N. Buchbinder, S. Chen, and J. S. Naor, "Competitive analysis via regularization," in *SODA*, 2014, pp. 436–444.

[26] N. Buchbinder and J. Naor, "The design of competitive online algorithms via a primal-dual approach," *Foundations and Trends in Theoretical Computer Science*, vol. 3, no. 2-3, pp. 93–263, 2009.

[27] S. library, "http://sndlib.zib.de/home.action," 2014.

[28] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*, 2016.

[29] H. Räcke, "Minimizing congestion in general networks," in *FOCS*, 2002, pp. 43–52.

[30] ——, "Optimal hierarchical decompositions for congestion minimization in networks," in *STOC*, 2008, pp. 255–264.

[31] J. Gettys and K. M. Nichols, "Bufferbloat: dark buffers in the internet," *Communication of the ACM*, vol. 55, no. 1, pp. 57–65, 2012.

[32] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *SIGCOMM*, 2011, pp. 74–85.

[33] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "HPCC: high precision congestion control," in *SIGCOMM*, 2019, pp. 44–58.

[34] A. Sivaraman, S. Subramanian, M. Alizadeh, S. Chole, S.-T. Chuang, A. Agrawal, H. Balakrishnan, T. Edsall, S. Katti, and N. McKeown, "Programmable packet scheduling at line rate," in *SIGCOMM*, 2016, pp. 44–57.

[35] G. Trimponias, Y. Xiao, X. Wu, H. Xu, and Y. Geng, "Node-constrained traffic engineering: Theory and applications," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1344–1358, 2019.

[36] S.-H. Tseng, "Perseverance-aware traffic engineering in rate-adaptive networks with reconfiguration delay," in *ICNP*, 2019, pp. 1–10.

[37] J. Zheng, H. Xu, X. Zhu, G. Chen, and Y. Geng, "Sentinel: Failure recovery in centralized traffic engineering," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1859–1872, 2019.

[38] J. Zheng, H. Xu, G. Chen, and H. Dai, "Minimizing transient congestion during network update in data centers," in *ICNP*, 2015, pp. 1–10.

[39] A. Ludwig, J. Marcinkowski, and S. Schmid, "Scheduling loop-free network updates: It's good to relax!" in *PODC*, 2015, pp. 13–22.

[40] A. Ludwig, S. Dudycz, M. Rost, and S. Schmid, "Transiently secure network updates," in *SIGMETRICS*, 2016, pp. 273–284.