DATE: Disturbance-Aware Traffic Engineering with Reinforcement Learning in Software-Defined Networks

Minghao Ye*, Junjie Zhang†, Zehua Guo‡, H. Jonathan Chao*
*Department of Electrical and Computer Engineering, New York University, New York City, NY 11201, USA
†Fortinet, Inc., Sunnyvale, CA 94086, USA
‡Beijing Institute of Technology, Beijing 100081, China
{minghao.ye, junjie.zhang, chao}@nyu.edu, guo@bit.edu.cn

***Abstract*—Traffic Engineering (TE) has been applied to optimize network performance by routing/rerouting flows based on traffic loads and network topologies. To cope with network dynamics from emerging applications, it is essential to reroute flows more frequently than today's TE to maintain network performance. However, existing TE solutions may introduce considerable Quality of Service (QoS) degradation and service disruption since they do not take the potential negative impact of flow rerouting into account. In this paper, we apply a new QoS metric named *network disturbance* to gauge the impact of flow rerouting while optimizing network load balancing in backbone networks. To employ this metric in TE design, we propose a disturbance-aware TE called DATE, which uses Reinforcement Learning (RL) to intelligently select some *critical flows* between nodes for each traffic matrix and reroutes them using Linear Programming (LP) to jointly optimize network performance and disturbance. DATE is equipped with a customized actor-critic architecture and Graph Neural Networks (GNNs) to handle dynamic traffic and single link failures. Extensive evaluations show that DATE can outperform state-of-the-art TE methods with close-to-optimal load balancing performance while effectively mitigating the 99th percentile network disturbance by up to 31.6%.**

***Index Terms*—Traffic Engineering, Software-Defined Networking, Reinforcement Learning, Routing, Network Disturbance, Link Failure**

I. Introduction

In Wide Area Networks (WANs), Internet Service Providers (ISPs) control traffic distribution by routing/rerouting traffic across their networks using a network optimizing operation called Traffic Engineering (TE). The objective of TE is to minimize the probability of network congestion by rerouting flows[1] to balance the load on links when network traffic fluctuates dynamically [1]–[4]. Such flow rerouting operation based on network traffic demands and link states is heavily dependent on network observability techniques, for instance, SNMP [5] and NetFlow [6]. Due to the complexity of collecting traffic demands and other statistics, these techniques are typically operated in the control plane, resulting in a long time interval (e.g., 5 minutes). Hence, network control for performance optimization can only be conducted at a coarse-grained minute level.

The corresponding author is Zehua Guo. This paper was supported by the National Natural Science Foundation of China under Grant 62002019 and the Beijing Institute of Technology Research Fund Program for Young Scholars.

[1]In this paper, a flow is defined as a source-destination pair.

Recently, fast and accurate network observability in the data plane has become possible by programmable switching technology [7], offering new opportunities to improve network performance in a fine-grained sub-minute fashion. For example, In-band Network Telemetry (INT) [8] provided by P4 switches [7] can precisely and rapidly observe network states (e.g., packet delay, queue length, and link utilization). P4 switches have been tested and deployed in the industry (e.g., AT&T [9], Alibaba [10]), and INT has been used in production networks (e.g., Alibaba Cloud [10]). With Software-Defined Networking (SDN) [11], the control plane can react to traffic change quickly by generating and deploying deliberate routing policies for different flows at the SDN switches based on its global view of the network. Meanwhile, new applications such as high-quality real-time video streaming [12]–[14] and Augmented/Virtual Reality (AR/VR) [15] pose different bandwidth and latency requirements on backbone networks. In the foreseeable future, with emerging network observability techniques and applications, flows should be adaptively and frequently rerouted to improve network performance under various network traffic dynamics.

However, flow rerouting could undesirably interrupt existing connections. Traditional TE solutions realize optimal or near-optimal load balancing performance at the cost of rerouting a large number of flows. In WANs, a flow usually aggregates many micro-flows (i.e., five-tuple TCP flows) of different applications. For the Sprint IP backbone network with around 40 nodes, the average number of active micro-flows measured on a 2.5Gbps link is more than 250,000 per minute [16]. In this case, changing the path of a flow could momentarily degrade the Quality of Service (QoS) of many TCP flows. For example, when rerouting a flow, out-of-order packets of some TCP flows between their old and new paths could be generated. Duplicated ACKs triggered by out-of-order packets could falsely signal network congestion to the senders, which in turn reduces the sending rates of the TCP flows to accommodate the network variation and eventually increases the flows' completion time or reduces throughput [17]. A recent work [18] shows that the aggregated throughput can be degraded by half after rerouting multiple TCP flows. As a result, frequently rerouting a large number of flows can potentially lead to severe service disruption of hundreds of thousands of TCP flows.

To address the abovementioned issue, we propose a new QoS metric called *network disturbance* to measure the impact

of flow rerouting. Network disturbance is defined as the percentage of total traffic in the network that is rerouted to different paths compared to the previous routing (please refer to Eq. (2)). With the consideration of network disturbance, ISPs can optimize their network performance while mitigating the rerouting impact on a large number of TCP flows. However, they face a new challenge: how to efficiently control flows to achieve joint optimization of network performance and disturbance for different traffic scenarios. Given various possible route changes, it is very difficult, if not impossible, to quantify network disturbance using a precise mathematical formulation according to the above definition. Therefore, network disturbance cannot be incorporated into traditional TE optimization problems as a part of the objective along with network performance.

In this paper, we propose Disturbance-Aware TE (DATE) to achieve low network disturbance and close-to-optimal network performance. DATE combines SDN and Equal-Cost Multi-Path (ECMP) to reduce network disturbance. Specifically, a majority of the flows are forwarded with existing ECMP routing by equally splitting traffic on equal-cost shortest paths while SDN deliberately chooses and reroutes some *critical flows* to balance link utilization of the network. Here, critical flow is defined as a flow with a dominant impact on network performance. To effectively handle varying network traffic scenarios, we use Reinforcement Learning (RL) to learn a critical flow selection policy and reroute these critical flows using Linear Programming (LP) to optimize network performance. To achieve joint optimization of network performance and disturbance, we design an effective reward function for RL to collect a performance-related reward from LP and impose penalty on high disturbance to mitigate network disturbance during training. As long as the reward signals correlate with the objective, RL is able to train for the objectives that cannot be directly optimized due to lack of precise models. Moreover, we adopt Graph Neural Networks (GNNs) [19], [20] to handle single link failures since GNNs offer unique advantages to model network topologies which are basically represented as graphs. By leveraging the graph representation learning techniques, GNN can naturally outperform other traditional neural network architectures (e.g., Convolutional Neural Networks).

The main contributions of this paper are summarized as follows:

1) We introduce a new QoS metric named network disturbance to evaluate the impact of flow rerouting on WANs.
2) We propose a disturbance-aware TE solution combining RL and LP with an effective reward function designed to achieve joint optimization of network performance and disturbance in different traffic scenarios.
3) We use GNN to generalize over different single link failure scenarios such that DATE can maintain good performance with low disturbance when link failure happens.
4) Our extensive simulations show that the proposed DATE outperforms state-of-the-art TE approaches with close-to-optimal performance and the lowest network disturbance, and thus improves QoS in normal operations as well as single link failure scenarios.

The remainder of this paper is organized as follows. Section II provides the definition of network disturbance. Section III overviews DATE's system design. Section IV explains how to learn a critical flow selection policy using RL and further describes the LP formulations for rerouting critical flows. Section V discusses the implementation details of DATE. Section VI evaluates the performance of DATE by presenting and analyzing the simulation results. Section VII describes the related works, and Section VIII concludes the paper.

II. Definition of Network Disturbance

In this section, we formally define network disturbance as a QoS metric and briefly explain how to calculate network disturbance caused by flow rerouting.

A. Notations

$G(V, E)$	network with nodes V and directed edges E ($\|V\| = N, \|E\| = M$)
f_K	selected critical flows
$c_{i,j}$	the capacity of link $\langle i, j\rangle$ ($\langle i, j\rangle \in E$)
$l_{i,j}$	the traffic load on link $\langle i, j\rangle$ ($\langle i, j\rangle \in E$)
$D^{s,d}$	the traffic demand from source s to destination d ($s, d \in V,\ s \neq d$)
$\sigma_{i,j}^{s,d}$	the percentage of traffic demand from source s to destination d routed on link $\langle i, j\rangle$ ($s, d \in V, s \neq d, \langle i, j\rangle \in E,\ \langle s, d\rangle \in f_K$)
$\Delta\sigma^{s,d}$	the percentage of traffic demand from source s to destination d being rerouted ($s, d \in V,\ s \neq d$)

B. Network Disturbance

Once TE updates routing to improve network performance, flows might be rerouted to different paths or distributed with different split ratios along original paths compared to the previous routing. Thus, we have the following definitions.

Definition 1. ***Rerouted Traffic:*** *Given the traffic demand of flow $\langle s, d\rangle$ as $D^{s,d}$ and the percentage of flow $\langle s, d\rangle$ that are rerouted from original paths as $\Delta\sigma^{s,d}$, the total amount of traffic that would be rerouted from original paths is defined as rerouted traffic*[2]*:*

$$T_R = \sum_{s,d \in V, s \neq d} D^{s,d} \cdot \Delta\sigma^{s,d}, \tag{1}$$

Definition 2. ***Network Disturbance:*** *Given the rerouted traffic T_R, the network disturbance is defined as:*

$$DB = \frac{T_R}{\sum_{s,d \in V, s \neq d} D^{s,d}}, \tag{2}$$

i.e., the percentage of total rerouted traffic in the network for a given traffic matrix, where $\sum_{s,d \in V, s \neq d} D^{s,d}$ is the total traffic of all flows.

[2]A portion of traffic of each rerouted flow might stay on the same routing paths when using consistent hashing to 5-tuple micro-flows, which is not considered as rerouted traffic.

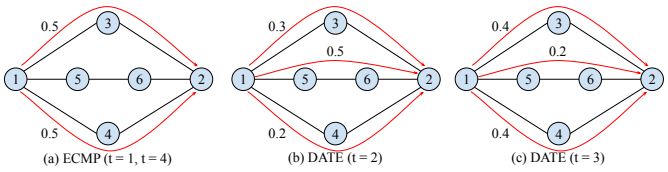


Fig. 1. Paths and split ratios for flow $\langle 1, 2\rangle$ at different time steps t. Each link is bidirectional with link weight equal to 1.

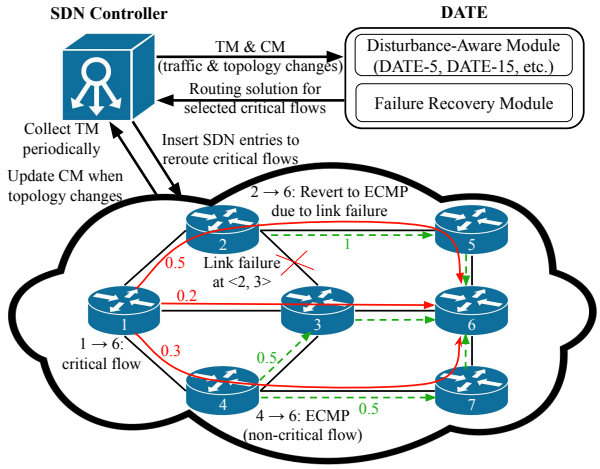


Fig. 2. Overview of DATE's system design. DATE selects and reroutes critical flows periodically or when topology changes. The red solid lines represent the paths for critical flows with different split ratios, and the green dashed lines are the ECMP paths for non-critical flows. Each link is bidirectional with link weight equal to 1.

DATE will select a small set of critical flows f_K to be rerouted for each given Traffic Matrix (TM) to accommodate traffic dynamics. We assume that ECMP is a default routing solution for all non-critical flows. For DATE, it is possible that the critical flows f_K^{t-1} and f_K^t selected for two consecutive TMs (TM_{t-1} and TM_t) are different, which means some of the critical flows identified in time step $t-1$ could be forwarded by ECMP in time step t. Therefore, these flows should also be considered as rerouted flows (i.e., $\langle s, d\rangle \in f_K^{t-1} \cup f_K^t$) when calculating network disturbance at time step t.

Figure 1 provides an example of measuring network disturbance caused by DATE. We assume that flow $\langle 1, 2\rangle$ is the only flow affected by routing updates from $t = 1$ to $t = 4$. As shown in Fig. 1(a), flow $\langle 1, 2\rangle$ is forwarded by ECMP at time $t = 1$. Therefore, the split ratio is (0.5, 0.5) for the two shortest paths 1-3-2 and 1-4-2. When $t = 2$, flow $\langle 1, 2\rangle$ is selected as one of the critical flows for rerouting as illustrated in Fig. 1(b). Then, the split ratio becomes (0.3, 0.5, 0.2) for paths 1-3-2, 1-5-6-2, and 1-4-2, which means 50% of the traffic of flow $\langle 1, 2\rangle$ is rerouted to the non-shortest path 1-5-6-2 and thus should be counted as rerouted traffic for calculating network disturbance. At time $t = 3$, flow $\langle 1, 2\rangle$ remains as a critical flow, but the routing changes to adapt to traffic dynamics such that the split ratio becomes (0.4, 0.2, 0.4) with the same paths as $t = 2$. As a result, 30% of the traffic of flow $\langle 1, 2\rangle$ is rerouted from path 1-5-6-2 to the other two paths. In the last time step $t = 4$, flow $\langle 1, 2\rangle$ is not selected as a critical flow, which means that it is forwarded by ECMP again. Therefore, the percentage of flow $\langle 1, 2\rangle$'s traffic that is rerouted to different paths is 20%.

III. System Design

Figure 2 illustrates DATE's system design. DATE is located together with an SDN controller and consists of two modules, one for normal operations in the network (without any link failures) and another with single link failures. The SDN controller is responsible for collecting TM periodically from the network and updating the network topology if any changes (e.g., link failure/recovery). The disturbance-aware module is responsible for flow routing/rerouting for normal operations to accommodate traffic variations captured by TM, while the failure recovery module is responsible for flow routing/rerouting for single link failures, which are reflected by a network Connectivity Matrix (CM). Based on the TM and CM, DATE identifies a set of critical flows in the network using RL and provides the corresponding routing solution with LP. The evaluation results in Section VI-A shows that selecting 20% of total flows is a good tradeoff between network performance and disturbance. According to the updated routing solution, the SDN controller installs the flow entries in corresponding SDN switches to reroute the critical flows. As depicted in Fig. 2, flow $\langle 1, 6\rangle$ is selected as a critical flow and rerouted over three paths with different split ratios, while flow $\langle 4, 6\rangle$ is forwarded by ECMP since it is not selected as a critical flow.

To cope with drastic traffic change scenarios, we can pre-train several DATE models with different disturbance targets as described in Section VI-B. When the network performance is relatively stable from small traffic variations, we can adopt a DATE model with a low target disturbance (e.g., DATE-5 with a target disturbance of 5%) to maintain good performance while having a small impact on QoS. If network performance degrades severely due to a sudden large spike of traffic demands, DATE could switch to another model with a high target disturbance (e.g., DATE-15 with a target disturbance of 15%) to allow more route changes to improve network performance. As shown in Section VI-B, DATE can handle different traffic scenarios and achieve near-optimal performance with lower network disturbance compared to other TE methods.

When a link failure occurs, a simple local recovery mechanism can be implemented to timeout existing SDN entries for critical flows and completely revert to ECMP forwarding. As illustrated in Fig. 2, flow $\langle 2, 6\rangle$ is rerouted to the remaining shortest path 2-5-6 due to a single link failure at link $\langle 2, 3\rangle$, no matter whether it is a critical flow or not. After that, the CM would be updated and TE-level resilience can be performed immediately by the failure recovery module to select and reroute critical flows. As described in Section V-D, we can pre-train a DATE model for the failure recovery module with an augmented dataset by including all possible single link failure scenarios. The evaluation results in Section VI-C shows that DATE is able to improve network performance with mitigated network disturbance in the presence of different single link failures.

IV. Selecting and Rerouting Critical Flows

In this section, we explain how to learn a critical flow selection policy using a customized RL approach and further

provide the LP formulation for critical flow rerouting.

A. Reinforcement Learning Formulation

Input / State Space: To include topology information and reflect traffic changes, an RL agent takes a state $s_t = (CM, TM_t, TM_{t-1})$ as an input, where CM is the topology CM and (TM_t, TM_{t-1}) are TMs at time step t and $t-1$, respectively.

Action Space: For each state s_t, DATE will select K critical flows. Given that there are $N*(N-1)$ flows in a network with N nodes, this RL problem would require a large action space of size $C^K_{N*(N-1)}$. Inspired by [21], [22], we define the action space as $\{0, 1, ..., N*(N-1)-1\}$ and allow the agent to sample K different actions in each time step t (i.e., $a_t^1, a_t^2, ..., a_t^K$). It is worth noting that the number of critical flows K is fixed to reduce solution space and accelerate convergence. We will investigate the possibility to dynamically adjust the K value for different states in our future work.

Reward: After sampling K different critical flows (i.e., f_K) for a given state s_t, DATE reroutes these critical flows and obtains the maximum link utilization U by solving the rerouting optimization problem (9a) described in Section IV-D. To minimize the maximum link utilization U and mitigate network disturbance DB, we design a reward function as follows:

$$r = \begin{cases} \dfrac{1}{U} & \text{if } DB \le DB_{TH} \\ \dfrac{1}{U \cdot (1 + P(DB - DB_{TH}))} & \text{if } DB > DB_{TH} \end{cases} \tag{3}$$

where DB_{TH} is a preset target network disturbance, and $P(DB - DB_{TH})$ is a penalty function depicted in Figure 3 when the target network disturbance cannot be satisfied:

$$P(DB - DB_{TH}) = (100(DB - DB_{TH}))^2 \tag{4}$$

The reward function is set to encourage the selection of critical flows that can achieve better network performance with lower network disturbance. When $DB \le DB_{TH}$, the target network disturbance can be achieved. Therefore, the $1/U$ term reflects the network performance after rerouting critical flows to balance link utilization. The smaller U, the better the performance and thus a higher reward can be received. When $DB > DB_{TH}$, an increasing penalty would be imposed since the target network disturbance cannot be satisfied. Given the penalty function design in Eq. (4), a small amount of violation (e.g., $DB - DB_{TH} = 1\%$) can be tolerated while a huge penalty would be imposed for a higher network disturbance.

B. Training Algorithm

The critical flow selection policy is represented by a neural network. This policy network takes a state $s_t = (CM, TM_t, TM_{t-1})$ as an input as described above and outputs a probability distribution $\pi(a_t|s_t)$ over all available actions. Since K different actions are sampled for each state s_t and their order doesn't matter, we define a solution

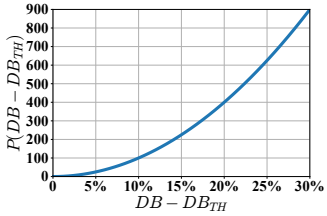


Fig. 3. A penalty function when network disturbance exceeds the preset target disturbance.

$a_{t_K} = (a_t^1, a_t^2, ..., a_t^K)$ as a combination of K sampled actions. For selecting a solution a_{t_K} with a given state s_t, a stochastic policy $\pi(a_{t_K}|s_t)$ parameterized by θ can be approximated as follows[3]:

$$\pi_\theta(a_{t_K}|s_t) \approx \prod_{i=1}^{K} \pi_\theta(a_t^i|s_t). \tag{5}$$

The goal of the training is to maximize the network performance over various TMs, i.e., to maximize the expected reward $E[r_t]$. Thus, we optimize $E[r_t]$ with gradient ascend, using REINFORCE algorithm with a baseline $b(s_t)$. The policy parameter θ is updated according to the following equation:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta log \pi_\theta(a_{t_K}|s_t)(r_t - b(s_t)). \tag{6}$$

where α is the learning rate for the policy network. A good baseline $b(s_t)$ reduces gradient variance and thus increases speed of learning. In this paper, we use a learned estimate of the value function $V^{\pi_\theta}(s_t)$ as the baseline $b(s_t)$. The critic network in Fig. 4 is trained to learn an estimate of $V^{\pi_\theta}(s_t)$. The critic network parameter θ_v is updated according to the following equation:

$$\theta_v \leftarrow \theta_v - \alpha_v \sum_t \nabla_{\theta_v}(r_t - V^{\pi_\theta}_{\theta_v}(s_t))^2, \tag{7}$$

where $V^{\pi_\theta}_{\theta_v}(\cdot)$ is outputted by the critic network as the estimate of $V^{\pi_\theta}(\cdot)$, and α_v is the learning rate for the critic network. Note that the critic network is only trained to estimate the expected reward r_t, and solely helps train the policy network. Once training is done, only the policy network is required to execute the action selection. $(r_t - V^{\pi_\theta}_{\theta_v}(s_t))$ indicates how much better the reward of a specific solution is compared to the expected reward for a given state s_t according to the policy π_θ. Intuitively, Eq. (6) can be explained as follows. If $(r_t - V^{\pi_\theta}_{\theta_v}(s_t))$ is positive, $\pi_\theta(a_{t_K}|s_t)$ (i.e., the probability of the solution a_{t_K}) is increased by updating the policy parameters θ in the direction $\nabla_\theta log \pi_\theta(a_{t_K}|s_t)$ with a step size of $\alpha(r_t - V^{\pi_\theta}_{\theta_v}(s_t))$. Otherwise, the probability of the solution is decreased. The net effect of Eq. (6) is to reinforce actions that empirically lead to better rewards.

To ensure that the RL agent explores the action space adequately during training to discover good policies, the entropy of the policy π is added to Eq. (6). This entropy

[3]To select K distinct actions, we conduct the action sampling without replacement. The right hand side of Eq. (5) is the probability of the solution when sampling with replacement, where Eq. (5) is used to approximate the probability of the solution a_{t_K} given a state s_t for simplicity.

term can improve the exploration by discouraging premature convergence to suboptimal deterministic policies [23]. Then, Eq. (6) is modified to the following equation:

$$\theta \leftarrow \theta + \alpha \sum_{t} \nabla_\theta log \pi_\theta(a_{t_K}|s_t)(r_t - V_{\theta_v}^{\pi_\theta}(s_t)) + \beta \nabla_\theta H(\pi_\theta(\cdot|s_t)), \tag{8}$$

where H is the entropy of the policy (the probability distribution over actions). The hyperparameter β controls the strength of the entropy regularization term. Algorithm 1 shows the pseudo-code for the training algorithm.

Algorithm 1 Training Algorithm

```
Initialize θ and θ_v
for each iteration do
    Δθ ← 0, Δθ_v ← 0
    {s_t} ← Sample a batch of states with size B
    for t = 1, ..., B do
        Sample a solution a_{t_K} according to policy π_θ(a_{t_K}|s_t)
        Receive reward r_t
    end for
    for t = 1, ..., B do
        Δθ ← Δθ + α(∇_θ log π_θ(a_{t_K}|s_t)(r_t − V^{π_θ}_{θ_v}(s_t)) +
              β∇_θ H(π_θ(·|s_t)))
        Δθ_v ← Δθ_v − α_v ∇_{θ_v}(r_t − V^{π_θ}_{θ_v}(s_t))^2
    end for
    θ ← θ + Δθ
    θ_v ← θ_v + Δθ_v
end for
```

C. Actor-Critic Network Architecture

Figure 4 shows the architecture of GNN-based actor-critic network. The inputs to the network are node features and topology CM, where the features for a given node are a series of demands originated from that node. The encoder computes the initial node embedding using a shared feed-forward layer, and then each node's embedding is updated by exchanging messages with its neighbors. Similar to the transformer model presented in [22], [24], the embedding update module consists of a stack of H (e.g., 6) identical attention layers. Each attention layer is composed of two sub-layers. The first is a multi-head attention layer that performs message exchanging between neighbor nodes, and the second is a node-wise fully connected feed-forward layer that performs a nonlinear transformation. In addition, a skip connection [25] and layer normalization [26] are applied to each sub-layer. Employing H attention layers can be interpreted as executing H iterations of embedding update process, and one iteration of embedding update process can be essentially considered as a feature propagation. After H iterations, each node's embedding would include H hops away neighbors' information. According to the "small-world" property [27], any two nodes in most real network topologies are reachable in less than 6 hops. Thus, a small value of H would be enough for each node to capture the complete information of the whole network (i.e., TMs and topology information).

All node embeddings outputted by the encoder are then concatenated to form a graph embedding, which is passed to a policy network to generate a probability distribution over actions and passed to a critic network to predict the baseline of the input state.

D. Rerouting Critical Flows

By default, traffic is distributed according to ECMP routing. We reroute a small set of critical flows (i.e., f_K) by conducting explicit routing optimization for these critical flows. Given a network $G(V, E)$ with a set of traffic demands $D^{s,d}$ for the selected critical flows ($\forall \langle s, d\rangle \in f_K$) and the background link load $\{\bar{l}_{i,j}\}$ contributed by the remaining non-critical flows using the default ECMP routing, our objective is to obtain the optimal explicit routing ratios $\{\sigma_{i,j}^{s,d}\}$ for each critical flow such that U is minimized. To search all possible under-utilized paths for the selected critical flows, we formulate the rerouting problem as an optimization problem as follows.

$$minimize \quad U + \epsilon \cdot \sum_{\langle i,j\rangle \in E} \sum_{\langle s,d\rangle \in f_K} \sigma_{i,j}^{s,d} \tag{9a}$$

subject to

$$l_{i,j} = \sum_{\langle s,d\rangle \in f_K} \sigma_{i,j}^{s,d} \cdot D^{s,d} + \bar{l}_{i,j} \qquad i,j : \langle i,j\rangle \in E \tag{9b}$$

$$l_{i,j} \le c_{i,j} \cdot U \qquad i,j : \langle i,j\rangle \in E \tag{9c}$$

$$\sum_{k:\langle k,i\rangle \in E} \sigma_{k,i}^{s,d} - \sum_{k:\langle i,k\rangle \in E} \sigma_{i,k}^{s,d} = \begin{cases} -1 & \text{if } i = s \\ 1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases} \qquad i \in V,\ s,d : \langle s,d\rangle \in f_K \tag{9d}$$

$$0 \le \sigma_{i,j}^{s,d} \le 1 \qquad s,d : \langle s,d\rangle \in f_K,\ i,j : \langle i,j\rangle \in E \tag{9e}$$

In Eq. (9a), $\epsilon \cdot \sum_{\langle i,j\rangle \in E} \sum_{\langle s,d\rangle \in f_K} \sigma_{i,j}^{s,d}$ is added to avoid unnecessarily long paths in the optimal solution, where ϵ ($\epsilon > 0$) is a sufficiently small constant to ensure that the minimization of U takes higher priority. Eq. (9b) indicates that the traffic load on link $\langle i, j\rangle$ is contributed by the traffic demands routed by explicit routing and also the traffic demands routed by the default ECMP routing. Eq. (9c) is the link capacity utilization constraint. Eq. (9d) is a flow conservation constraint for the selected critical flows.

After solving the above problem using LP solvers (e.g., Gurobi [28]), we can obtain an optimal explicit routing solution $\{\sigma_{i,j}^{s,d}\}$ ($\forall \langle s, d\rangle \in f_K$) for the selected critical flows. Then, the SDN controller installs and updates flow entries at the SDN switches accordingly.

V. Implementation

In this section, we describe the experiment setup as well as implementation details of DATE.

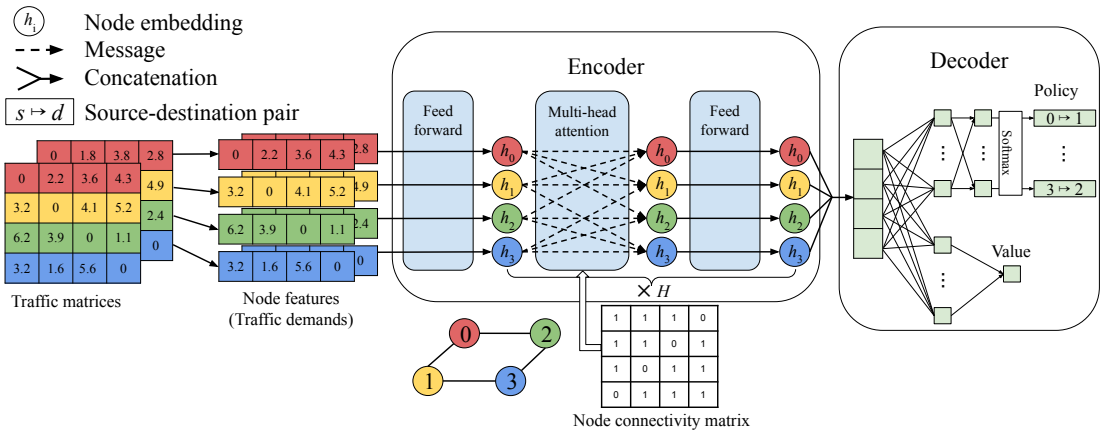


Fig. 4. GNN-based actor-critic architecture.

TABLE I
NETWORK TOPOLOGIES FOR EVALUATION

Topology	Nodes	Directed Links	S-D Pairs
Abilene	12	30	132
GÉANT	23	74	506
EBONE (Europe)	23	76	506
Sprintlink (US)	44	166	1892
Tiscali (Europe)	49	172	2352

A. Dataset

In our evaluation, we use five different real-world network topologies, including Abilene network, GÉANT network, and three ISP networks collected by Rocketfuel [29]. The numbers of nodes, directed links, and source-destination pairs of the networks are listed in Table I. For the Abilene network, the network topology information (such as link connectivity, weights, and capacities) and measured TMs can be found in [30]. Since the Abilene TMs are measured every 5 minutes, there are a total of 2016 continuous TMs each week. To evaluate the performance of DATE and correctly measure network disturbance with consecutive TMs, we choose a total of 2016 TMs in the first week (starting from Mar. 1st, 2004) as our training set and test our scheme on the TMs in the following week (starting from Mar. 8th, 2004). For the GÉANT network, the topology information including link capacities and costs are provided in [31]. The GÉANT TMs are available at [32] and they are collected every 15 minutes for a continuous period of 4 months. Similarly, we select a total of 672 continuous TMs in the first week (starting from Jan. 1st, 2005) as our training set and test our scheme on the TMs in the following week (starting from Jan. 8th, 2005). For the remaining three ISP networks from Rocketfuel, only the link costs are given while the link capacities and TMs are not provided. Therefore, we infer the link capacities as the inverse of link costs based on the default link cost setting in Cisco routers, which is a commonly adopted approach in literature [33]–[35]. As for the dataset, we synthesize 200 TMs for each of the three networks using the gravity model [36]. We first compute a gravity TM where each entry $D_g^{s,d}$ represents the demand volume from source node s to destination node d. Then, we generate the TM sequence based on the gravity TM. The traffic demand $D^{s,d}$ of each TM is sampled from a uniform distribution of $[0.5 \cdot D_g^{s,d}, 4.5 \cdot D_g^{s,d}]$ to simulate the dynamics of network traffic. In our evaluation, the first 100 TMs are used for training and the remaining 100 TMs are used for testing.

B. Hyperparameters

The GNN-based actor-critic architecture is implemented using TensorFlow [37]. For the encoder, we set the embedding dimension d_h to 64 and the number of attention heads M to 8. For the feed-forward sub-layer in each attention layer, the intermediate layer dimension d_f is set to 256. Besides, the number of attention layers H is set to 6. For the decoder, the feed-forward layer of the policy network has an output dimension of $N * (N-1)$, which represents the number of source-destination pairs in the network since N is the number of network nodes. The softmax function is applied upon the output of the policy network to generate the probabilities for all available actions. The critic network is similar to the policy network except that the last layer is a fully connected linear layer with only one neuron corresponding to the baseline $b(s_t)$. The learning rates α and α_v are configured to decay from 0.001 to 0.0001 over 0.5×10^5 iterations. Additionally, the entropy factor β is set to 0.01. We do not use sophisticated hyperparameter tuning methods [38], and fix all these hyperparameters throughout our experiments. The experiment results in Section VI demonstrate that DATE works well in different environments with a single set of hyperparameters.

C. Parallel Training

To speed up training, we spawn multiple actor agents in parallel as suggested by [23]. DATE uses 20 actor agents by default. Each actor agent is configured to experience a different subset of the training set. Then, these agents continually forward their tuples (state, action, reward) to a central learner agent, which aggregates them to train the policy network and critic network. The central learner agent performs a gradient update using Eq. (7) and Eq. (8) according to the received tuples and then sends back the updated network parameters to

the actor agents. In our evaluation, we use 21 CPU cores (one 3.0GHz core for each agent) and 16GB memory to train DATE. The training process is greatly facilitated due to our scalable neural network architecture design. It takes less than 2 hours to train a DATE model from scratch for small networks (i.e., Abilene, GÉANT and EBONE network) with 3,000 iterations. For the two large networks, it requires 8,000 iterations with approximately 6 hours' training to converge since the action space is much larger. It is also worth noting that all the training costs are incurred offline and the number of CPU cores can be adjusted based on actual hardware specifications. For online deployment, the inference time is less than one second and the resource consumption is relatively low (e.g., one CPU core).

D. Resilience

In addition to normal operations, TE should maintain good performance in the presence of link failures. Therefore, we trained DATE with an augmented dataset including the TMs in all possible single link failure scenarios[4]. In each scenario, we remove the failed link from the network environment and recalculate the shortest paths for ECMP. All the preconfigured paths containing the failed link will also be excluded from the LP formulation. To simulate single link failure in GNN, we change the CM such that the message exchange between the two nodes of the failed link is temporarily disabled. After training is done, we evaluate DATE under different single link failure scenarios and the results are presented in Section VI-C. In this paper, we only consider single link failure scenarios. For multiple link failures, including all possible link failure scenarios will make the augmented training dataset extremely large since there are too many link failure combinations. We will investigate this problem in our future works.

E. Baselines

Traditional TE solutions introduce considerable QoS degradation due to frequent and substantial flow rerouting in the network without consideration of network disturbance. To demonstrate the advantages of DATE in mitigating network disturbance and improving network performance, we compare DATE to the following schemes, including the state-of-the-art TE solutions and a rule-based heuristic:

1) **CFR-RL** [39]: customizes an RL approach to learn a critical flow selection policy with an objective of optimizing network performance, and further utilizes LP to reroute these critical flows.
2) **SMORE** [40]: computes a set of paths using an oblivious routing algorithm and deploys a centralized controller to dynamically adapt sending rates of all flows according to these preconfigured paths.
3) **Top-K**: selects K flows with the largest demand volume from a given TM and only reroutes these flows using LP. It is based on the assumption that elephant flows would have a dominate impact on network performance.

[4]For the nodes with degree 1 in the two networks, the failures on their inbound/outbound links are excluded from the training scenarios since any failure on these links will result in traffic loss.

F. Evaluation Metrics

To demonstrate the performance of DATE, we define a performance ratio as follows:

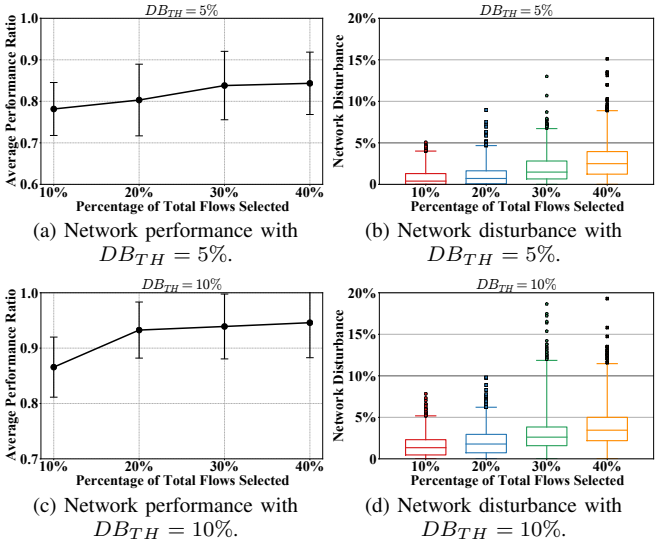
$$PR = \frac{U_{\text{optimal}}}{U_{\text{DATE}}}, \tag{10}$$

where U_{optimal} is the maximum link utilization achieved by an optimal explicit routing for all flows. $PR = 1$ means that the proposed DATE has the same performance as the optimal routing. A lower ratio indicates that the performance of DATE is farther away from that of the optimal routing. Note that we can compute the performance ratio of other TE solutions by replacing the denominator as the maximum link utilization achieved by other schemes. Besides, the corresponding network disturbance DB is calculated using Eq. (2) in our evaluation to measure the impact of flow rerouting on the network.

VI. Evaluation

Extensive experiments are conducted based on five real-world network topologies and their TMs to evaluate the performance and network disturbance of DATE in normal operations and single link failure scenarios. We compare DATE with state-of-the-art TE solutions and a rule-based heuristic to show the effectiveness of DATE in mitigating network disturbance while improving network performance.

A. Number of Critical Flows



(a) Network performance with $DB_{TH} = 5\%$.
(b) Network disturbance with $DB_{TH} = 5\%$.
(c) Network performance with $DB_{TH} = 10\%$.
(d) Network disturbance with $DB_{TH} = 10\%$.

Fig. 5. Comparison of average performance ratio and network disturbance with an increasing number of critical flows K selected in the Abilene network using different preset target disturbance. In (a)(c), error bars span $\pm$ one standard deviation from the average on the entire test set. In (b)(d), the horizontal grey line is the target disturbance. The line in the box represents the median value, and the upper whisker is customized as the 99th percentile disturbance. The data points beyond the upper whisker are outliers, which represents the worst 1% cases with the highest network disturbance.

It is important for DATE to select and reroute a proper number of critical flows to improve network performance with low network disturbance. If the number of critical flows

is too small, it might be difficult to achieve good network performance. On the contrary, rerouting too many critical flows may introduce considerable network disturbance. To investigate the influence of different numbers of critical flows, we conduct a series of experiments with an increasing number of critical flows selected as a fraction of the total flows under two preset disturbance targets. Note that the comparison is mainly performed on the Abilene network since the results for other four networks are similar.

Figure 5 shows the average performance ratio and network disturbance achieved by DATE with an increasing number of critical flows, where the X-axis is the percentage of total flows selected as critical flows. Recall that the total number of flows in the network is $N * (N-1)$, where N is the number of network nodes. In this case, 10% of total flows selected means that the number of critical flows is $K = 10\% * N * (N-1)$. In Fig. 5(a), we can see that the average performance ratio becomes higher as the number of critical flows increases. However, the network disturbance is also increasing as illustrated in the boxplot of Fig. 5(b), which demonstrates the tradeoff between network performance and disturbance. For extreme cases, $K = 20\% * N * (N-1)$ results in a 99th percentile disturbance of 4.7% with several outliers up to 9%. If we select 30% or 40% of total flows as critical flows, the 99th percentile disturbance exceeds the target disturbance and the maximum disturbance is higher than 13%. From Fig. 5(c)(d), we have similar observations when the target disturbance is set to 10%. Comparing to $K = 10\% * N * (N-1)$, selecting 20% of total flows can achieve 6.7% performance improvement on average and the target disturbance can be satisfied even in the most extreme case. Rerouting more than 20% of total flows lead to slight performance improvement while the worst-case disturbance is close to 20%. As a result, selecting and rerouting 20% of total flows could be a good choice since the target disturbance can be satisfied in at least 99% of traffic scenarios. In the following experiments, the number of critical flows is set to $K = 20\% * N * (N-1)$ such that DATE can achieve better network performance with mitigated network disturbance.

B. Comparison between Different Schemes

For normal operations, we train three DATE models for each network with different disturbance targets (i.e., $DB_{TH} = 5\%$, $DB_{TH} = 10\%$, $DB_{TH} = 15\%$) and denote them as DATE-5, DATE-10, and DATE-15, respectively. As discussed in Section III, we can switch between different DATE models on demand. For comparison, we also calculate the performance ratio of CFR-RL, SMORE, Top-K, and ECMP according to Eq. (10) and measure the corresponding network disturbance using Eq. (2) in each network. Figure 6 illustrates the comparison of different schemes in terms of performance ratio and network disturbance in the five networks, where the X-axis represents the average network disturbance and the Y-axis is the average performance ratio achieved by each scheme. The end of each line is the 99th percentile disturbance corresponding to the worst cases. For the baseline methods, CFR-RL performs significantly well in both networks. This is because the objective

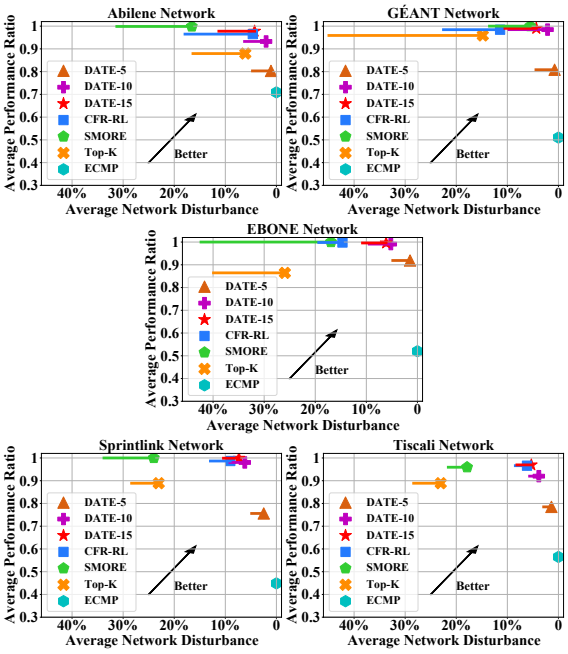


Fig. 6. Comparison of average performance ratio, average network disturbance (marker) and 99th percentile network disturbance (end of each line) between different schemes on the entire test set of the five networks. The upper-right corner represents the best solution with the highest performance ratio and lowest network disturbance.

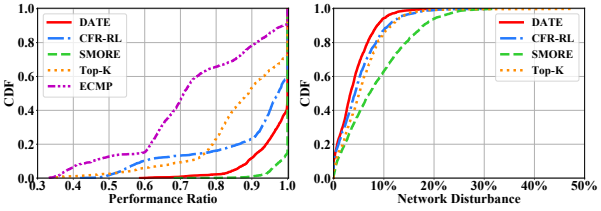
of CFR-RL is to solely optimize network performance such that it can identify the most critical flows in the network for rerouting. As for other baselines, SMORE can adjust the sending rate of all flows in the network with preconfigured paths to achieve close-to-optimal performance, while Top-K can only improve the network performance to some extent since the largest K flows are not always the best choices.

Even though some of the baseline methods can achieve promising performance, they can also incur high network disturbance since none of these approaches takes network disturbance into consideration. For example, SMORE generates an average of 16.6% and 24.2% network disturbance in the Abilene network and Sprintlink network, respectively, which is much higher than other schemes. In the worst cases, more than 42.4% of the total traffic would be rerouted to different paths by SMORE in the EBONE network, which means almost half of the network traffics suffers from service disruption. This is because SMORE adaptively changes the sending rate of all flows for each measured TM without consideration of disturbance. On the contrary, CFR-RL has lower disturbance compared to SMORE in all of the networks except for the GÉANT network. This is reasonable since CFR-RL only reroutes a smaller set of critical flows and part of the traffic of the critical flows may stay on their previous paths. However, at least 22.5% of the total traffic would be rerouted by CFR-RL in the worst cases of the GÉANT network. Unlike DATE, CFR-RL is not disturbance-aware and it would incur higher disturbance in some cases. For Top-K, the network disturbance is much larger than CFR-RL and DATE. In the GÉANT network, Top-K generates 44.9% disturbance in the

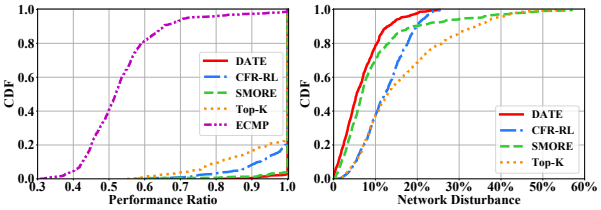
worst cases by rerouting the largest K flows in the network, which is one of the limitations of the rule-based heuristic.

Compared to these baselines, DATE can effectively trade off network performance and disturbance. As shown in Fig. 6, the best operating region of TE is the upper right region with the highest performance ratio and lowest network disturbance. Most of the baselines focus on network performance while ignoring network disturbance, which means they are operating in the upper left region. On the contrary, DATE can consistently operate in the upper right region in all of the five networks. In the Abilene network, DATE-15 can achieve an average of 26.9% performance improvement over ECMP while the 99th percentile disturbance is only 11.3%. In the GÉANT network, the average network performance can be improved by 29.8%, 47.5%, and 48.0% compared to ECMP with DATE-5, DATE-10, and DATE-15, respectively. At the meantime, the target network disturbance can be satisfied in 99% of the test TMs for the three DATE models. In the rest of the three ISP networks, DATE-15 can achieve the same performance as the best performing scheme with lower disturbance. For example, DATE-15 can achieve 99.6% performance on average with EBONE TMs while mitigating the 99th percentile disturbance by 8.6% and 31.6% compared to CFR-RL and SMORE, respectively. It is worth noting that DATE-10 can also achieve at least 92% average performance with low disturbance in the five networks, which demonstrates the capability of DATE to improve network performance within the target disturbance. As a result, DATE can achieve good performance with mitigated network disturbance in different traffic scenarios.

C. Resilience



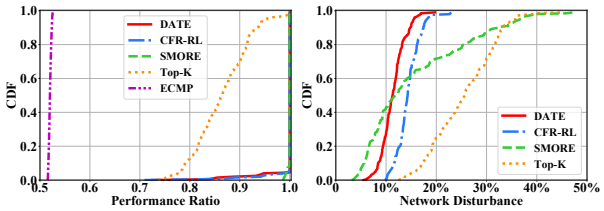
(a) Abilene network.



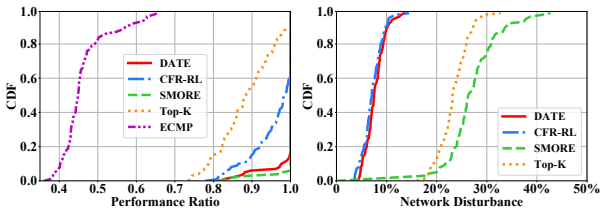
(b) GÉANT network.

Fig. 7. Comparison of performance ratio and network disturbance in CDF among DATE and the baseline methods in the 2nd week of the Abilene network and GÉANT network with random single link failures.

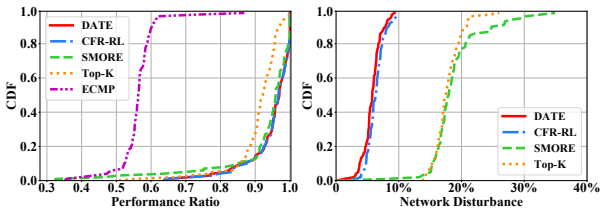
As discussed in Section V-D, we test the resilient routing provided by DATE with different single link failures. A target disturbance $DB_{TH} = 20\%$ is specified to maintain good performance and reduce network disturbance in extreme



(a) EBONE network.



(b) Sprintlink network.



(c) Tiscali network.

Fig. 8. Comparison of performance ratio and network disturbance in CDF among DATE and the baseline methods on the entire test set of the three ISP networks with random single link failures.

conditions. For the Abilene network and GÉANT network, we randomly fail one link every hour such that twelve Abilene TMs and four GÉANT TMs are evaluated before the next link failure. If some of the flows are rerouted due to link failure (e.g., the previous paths are not available), they are not counted as rerouted traffic when we measure network disturbance. Figure 7 presents the Cumulative Distribution Function (CDF) of performance ratio and network disturbance achieved by DATE and baseline methods in the 2nd week of the Abilene network and GÉANT network under different single link failure scenarios. Note that the network disturbance of ECMP is zero since the routing is static and the impact of link failure is omitted from the calculation of network disturbance. In the Abilene network, DATE outperforms CFR-RL and Top-K in different traffic scenarios with random single link failures. There is a small performance gap between DATE and SMORE, but the disturbance of SMORE is much larger compared to DATE. For the GÉANT network, DATE can achieve similar performance as SMORE with the lowest network disturbance among all schemes. In terms of worst-case performance, DATE can even outperform SMORE with the lowest performance ratio of 80% while SMORE's performance can be degraded to 56.9%, which reveals the capability of DATE to handle single link failure by leveraging the graph representation learning techniques. To further explore the potential capability of DATE, we conduct experiments using synthesized TMs with random single link failures in the three ISP networks. Similarly, we randomly fail a unique link every 2 TMs to evaluate the resilient routing from DATE. Figure 8 shows the CDF of

performance ratio and network disturbance achieved by DATE and the baseline methods on the test set of the three ISP networks under different single link failure scenarios. In the EBONE network and Sprintlink network, DATE can achieve the same performance as CFR-RL with lower disturbance or outperforms CFR-RL with similar disturbance. For the Tiscali network, DATE can even slightly outperform SMORE and the disturbance is much lower compared to SMORE. Therefore, DATE is able to maintain promising network performance and mitigate network disturbance in the presence of single link failures.

VII. Related Works

SDN-based TE is widely applied to optimize network performance. Dynamic hybrid routing [33] relies on a flexible routing policy from SDN and dynamically re-balances traffic to accommodate traffic fluctuations based on a preconfigured routing policy for better load balancing. Agarwal et al. [41] consider a network with partially deployed SDN switches and try to improve network utilization by strategically placing SDN switches. Guo et al. [42] design a TE solution named SOTE to achieve load balancing in an SDN/OSPF hybrid network.

Machine learning has been used in TE design recently. To minimize signaling delay in large SDNs, Lin et al. [43] propose QoS-aware Adaptive Routing, which employs RL for designing a distributed three-level control plane architecture. Xu et al. [44] use RL to optimize performance metrics (i.e., throughput and delay) in backbone networks. However, none of the above works takes mitigating the impact of network disturbance and service disruption caused by flow rerouting into account when designing TE solutions.

VIII. Conclusion

In this paper, we apply a new QoS metric called network disturbance to measure the impact of flow rerouting on WANs. To incorporate network disturbance in TE design, we propose DATE, a disturbance-aware TE solution leveraging RL and GNN to learn a critical flow selection policy that can accommodate dynamic network traffics and different link failure scenarios. For each given TM and topology, DATE smartly selects a small set of critical flows and reroutes them with LP to balance link utilization of the network within a target network disturbance. Extensive evaluations show that DATE is able to achieve a high load balancing performance and reduce network disturbance in normal operations as well as single link failure scenarios.

References

[1] Y. Wang and Z. Wang, "Explicit routing algorithms for internet traffic engineering," in *IEEE ICCCN'99*.

[2] E. D. Osborne and A. Simha, *Traffic engineering with MPLS*. Cisco Press, 2002.

[3] J. Chu and C.-T. Lea, "Optimal link weights for ip-based networks supporting hose-model vpns," *IEEE/ACM ToN*, 2009.

[4] J. Zhang *et al.*, "Optimizing network performance using weighted multipath routing," in *IEEE ICCCN'12*.

[5] C. Hare, "Simple network management protocol (snmp)." 2011.

[6] R. Sommer and A. Feldmann, "Netflow: Information loss or win?" in *ACM SIGCOMM IMW'02*.

[7] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM CCR*, 2014.

[8] C. Kim *et al.*, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM'15*.

[9] "AT&T picks barefoot networks for programmable switches," 2017. [Online]. Available: https://www.sdxcentral.com/articles/news/att-picks-barefoot-networks-programmable-switches/2017/04/

[10] Y. Li *et al.*, "Hpcc: High precision congestion control," in *ACM SIGCOMM'19*.

[11] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, 2008.

[12] "Tiktok," 2020. [Online]. Available: https://www.tiktok.com/

[13] "Kuaishou," 2020. [Online]. Available: https://www.kuaishou.com/en

[14] "Vimeo," 2020. [Online]. Available: https://vimeo.com/

[15] "Google lens," 2020. [Online]. Available: https://lens.google.com/

[16] C. Fraleigh *et al.*, "Packet-level traffic measurements from the sprint ip backbone," *IEEE network*, 2003.

[17] W. Reda *et al.*, "Path persistence in the cloud: A study of the effects of inter-region traffic engineering in a large cloud provider's network," *ACM SIGCOMM CCR*, 2020.

[18] R. Carpa *et al.*, "Evaluating the impact of sdn-induced frequent route changes on tcp flows," in *IEEE CNSM'17*.

[19] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2018.

[20] P. Veličković *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2018.

[21] H. Mao *et al.*, "Resource management with deep reinforcement learning," in *ACM HotNets'16*.

[22] W. Kool *et al.*, "Attention, learn to solve routing problems!" in *ICLR'19*.

[23] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *ICML'16*.

[24] A. Vaswani *et al.*, "Attention is all you need," in *NIPS'17*.

[25] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR'16*.

[26] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[27] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, 1998.

[28] Gurobi Optimization LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

[29] N. Spring *et al.*, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM CCR*, 2002.

[30] Yin Zhang's Abilene TM. [Online]. Available: https://www.cs.utexas.edu/~yzhang/research/AbileneTM

[31] S. Uhlig *et al.*, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM CCR*, 2006.

[32] GÉANT. The TOTEM project. [Online]. Available: https://totem.info.ucl.ac.be/dataset.html

[33] J. Zhang *et al.*, "Dynamic hybrid routing: Achieve load balancing for changing traffic demands," in *IEEE/ACM IWQoS'14*.

[34] J. Zhang *et al.*, "Load balancing for multiple traffic matrices using sdn hybrid routing," in *IEEE HPSR'14*.

[35] M. Kodialam *et al.*, "Oblivious routing of highly variable traffic in service overlays and ip backbones," *IEEE/ACM ToN*, 2008.

[36] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM CCR*, 2005.

[37] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *USENIX OSDI'16*.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS'10*.

[39] J. Zhang *et al.*, "CFR-RL: Traffic engineering with reinforcement learning in sdn," *IEEE JSAC*, 2020.

[40] P. Kumar *et al.*, "Semi-oblivious traffic engineering: The road not taken," in *USENIX NSDI'18*.

[41] S. Agarwal *et al.*, "Traffic engineering in software defined networks," in *IEEE INFOCOM'13*.

[42] Y. Guo *et al.*, "Traffic engineering in sdn/ospf hybrid network," in *IEEE ICNP'14*.

[43] S.-C. Lin *et al.*, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *IEEE SCC'16*.

[44] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM'18*.