FlexDATE: Flexible and Disturbance-Aware Traffic Engineering With Reinforcement Learning in Software-Defined Networks

Minghao Ye, Junjie Zhang, *Member, IEEE*, Zehua Guo, *Senior Member, IEEE, Member, ACM*, and H. Jonathan Chao, *Life Fellow, IEEE*

***Abstract*—Traffic Engineering (TE) is an important network operation that routes/reroutes flows based on network topology and traffic demands to optimize network performance. Recently, new emerging applications pose challenges to TE with dynamic network conditions, where frequent routing updates are required to maintain good network performance with Software-Defined Networking (SDN). However, flow rerouting operations could lead to considerable Quality of Service (QoS) degradation and service disruption, which is often neglected by existing TE solutions. In this paper, we apply a new QoS metric named *network disturbance* to measure the negative impact of flow rerouting operations performed by TE. To achieve near-optimal load balancing performance and mitigate network disturbance together in dynamic network scenarios, we propose a flexible and disturbance-aware TE solution called FlexDATE that combines Reinforcement Learning (RL) and Linear Programming (LP). Specifically, FlexDATE leverages RL to intelligently identify flexible numbers of *critical flows* for each traffic matrix and reroutes these critical flows based on LP optimization to improve network performance with low disturbance. Empowered by a customized actor-critic architecture coupled with Graph Neural Networks (GNNs), FlexDATE can generalize well to unseen traffic scenarios and remain resilient to single link failures. Extensive simulations are conducted on five real-world network topologies to evaluate FlexDATE with real and synthetic traffic traces. The results show that FlexDATE can achieve the performance target (i.e., 90% of optimal performance) in 99% of network scenarios and effectively mitigate the average and maximum network disturbance by up to 9.1% and 38.6%, respectively, compared to state-of-the-art TE solutions.**

***Index Terms*—Traffic engineering, reinforcement learning, software-defined networking, network disturbance, graph neural networks.**

I. Introduction

TRAFFIC Engineering (TE) is an important network operation for Internet Service Providers (ISPs) to control traffic distribution by routing/rerouting traffic across their Wide Area Networks (WANs). With the objective of minimizing network congestion probability, TE usually reroutes network flows[1] periodically to achieve better load balancing in dynamic network traffic conditions [1]–[4]. Given a network topology and traffic demands between all source-destination pairs, TE can be formulated as an optimization problem and solved for optimal routing strategies to redistribute the traffic across the network. Generally speaking, such flow rerouting operations are heavily dependent on the network statistics measured by network observability techniques (e.g., 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 with a long time interval. Therefore, TE is usually conducted at a coarse-grained minute-level to improve network performance.

In recent years, applications, such as high-quality real-time video streaming [7]–[9] and Augmented/Virtual Reality (AR/VR) [10], pose diverse bandwidth and latency requirements to backbone networks with dynamic traffic variations. In this scenario, coarse-grained minute-level TE operations may lead to performance degradation since they cannot adapt to drastic traffic changes with long routing update intervals. Thanks to emerging programmable network technology [11], fast and accurate network observability in the data plane has become possible and provides new opportunities for TE to perform routing updates in a fine-grained sub-minute fashion. For instance, P4 switches [11] have been tested and deployed in the industry (e.g., AT&T [12] and Alibaba [13]) with more flexibility to process packets in the data plane. In Alibaba Cloud [13], In-band Network Telemetry (INT) technique [14] has been implemented in P4 switches to precisely and rapidly observe network states (e.g., packet delay, queue length, and link utilization). Moreover, INT can be combined with Software-Defined Networking (SDN) [15] to implement network monitoring systems in SDN controllers and collect network statistics from the data plane [16], [17]. With INT and SDN, TE can be deployed at the control plane and quickly react to traffic changes based on the global view of the network. Once TE generates an updated routing strategy based on timely measured network states from INT, the SDN controller can deploy deliberate routing policies at underlying SDN switches to reroute flows. For example, SDN-based TE is operating at second-level to satisfy the

An earlier version of this paper was presented in part at the 2021 IEEE/ACM 29th International Symposium on Quality of Service [DOI: 10.1109/IWQOS52092.2021.9521343].

This paper was supported by the National Key Research and Development Program of China under Grant 2021YFB1714800, the National Natural Science Foundation of China under Grant 62002019 and the Beijing Institute of Technology Research Fund Program for Young Scholars. *(Corresponding author: Zehua Guo.)*

Minghao Ye and H. Jonathan Chao are with the Department of Electrical and Computer Engineering, New York University, New York City, NY 11201 USA (e-mail: minghao.ye@nyu.edu; chao@nyu.edu).

Junjie Zhang is with Fortinet, Inc., Sunnyvale, CA 94086 USA (e-mail: junjie.zhang@nyu.edu).

Zehua Guo is with Beijing Institute of Technology, Beijing 100081, China (e-mail: guolizihao@hotmail.com).

[1]In this paper, a flow is defined as a source-destination pair. We use these two terms interchangeably.

requirements of different cloud applications in Amazon's inter-region backbone network [18]. In the foreseeable future, with emerging network observability techniques and applications, flows should be adaptively and frequently rerouted to improve network performance and cope with network traffic dynamics.

However, TE operations may undesirably interrupt existing connections with path changes. In traditional TE solutions, a large number of flows would be rerouted to achieve optimal or near-optimal load balancing performance. Given that a single flow usually aggregates many micro-flows (i.e., five-tuple TCP flows) of different applications, frequent routing updates could momentarily degrade the Quality of Service (QoS) of many TCP flows. For example, packet reordering may happen when TE reroutes the flows. When a TCP flow is migrated from a high latency path to a low latency path, some of the newly sent packets on the new path may arrive at the destination earlier than the previous packets on the old path. As a result, these packets are considered out-of-order packets, and duplicate acknowledgments (dupACKs) would be triggered at the receiver side, which falsely signals network congestion to the sender. Upon receiving three dupACKs, the sender would reduce the congestion window size and thus decrease the sending rate, which eventually reduces the throughput of the TCP flow and increases its completion time. A recent work [19] shows that the aggregated throughput can be degraded by half after rerouting multiple TCP flows. Besides, it is also possible that some delay-sensitive TCP flows would be rerouted to new paths with higher latencies and thus fail to satisfy the latency requirements [18]. Therefore, frequently rerouting a large number of flows could severely disrupt services with a negative impact on hundreds of thousands of TCP flows.

To address the above-mentioned issue, we propose a new QoS metric called *network disturbance* to measure the impact of flow rerouting in TE. Network disturbance is defined as the percentage of total traffic in the network that is rerouted to different paths during routing updates (please refer to Eq. (2)). By considering network disturbance, ISPs can optimize their network performance while mitigating the rerouting impact on a large number of TCP flows. However, there is a new challenge: how to efficiently control flows to achieve joint optimization of network performance and disturbance in different network 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. Meanwhile, it is very challenging to realize a trade-off between maximizing network performance and minimizing network disturbance. To achieve good performance in dynamic network conditions, it might be necessary to reroute many flows at the cost of incurring high network disturbance. On the contrary, ISPs may use a static routing strategy without network disturbance, but the resulting network performance would be far from optimal. Therefore, network disturbance cannot be incorporated in traditional optimization problems as a part of the objective along with network performance.

In this paper, we propose Flexible and Disturbance-Aware Traffic Engineering (FlexDATE) to mitigate network disturbance caused by flow rerouting while ensuring near-optimal network performance in dynamic network scenarios. Specifically, a majority of the flows are forwarded by static oblivious routing with a worst-case performance guarantee to avoid service disruption, while the SDN controller deliberately chooses and reroutes some *critical flows* to balance link utilization of the network. Here, a critical flow is defined as a flow with a dominant impact on network performance. To effectively handle varying network traffic conditions and link failure scenarios, we use Reinforcement Learning (RL) to learn a good policy that determines the number of critical flows and identifies the critical flows simultaneously with a customized actor-critic architecture. Then, we can obtain the optimal routing strategy for critical flows by solving a modified Linear Programming (LP) optimization problem. Upon network traffic/connectivity changes, we can reroute the critical flows by updating their traffic split ratios over a set of preconfigured paths [20], [21] at the sender side. To find a good trade-off between network performance and disturbance, we design an effective reward function for RL with a preset performance target and penalty factors on network disturbance to prioritize performance improvement or disturbance mitigation at different training stages. As long as the reward signals correlate with the objective, RL is able to train for objectives that cannot be directly optimized due to a lack of precise models. In addition, we adopt Graph Neural Networks (GNNs) [22], [23] to handle single link failures. GNN is recently applied in different TE solutions [24]–[26] with unique advantages to model network topologies that are generally represented as graphs. By leveraging the graph representation learning techniques and message passing frameworks, GNN considers link failure scenarios by disabling direct message exchanges between the two nodes connected to a failed link. Moreover, each module/layer in FlexDATE's GNN architecture can be shared and reused by each node, which facilitates the training process and outperforms other traditional neural network architectures (e.g., convolutional neural networks) with lower model complexity.

The main contributions of this paper are summarized as follows:

1) We propose a new QoS metric called network disturbance to evaluate the impact of flow rerouting on WANs.
2) We design a flexible and disturbance-aware TE solution combining RL and LP to select and reroute a few critical flows to achieve near-optimal performance with mitigated disturbance in different traffic scenarios.
3) We leverage GNN architecture to learn from network topology information such that FlexDATE can generalize well to different single link failure scenarios.
4) Our extensive simulations show that FlexDATE can achieve above 90% of optimal performance in most network scenarios and outperform state-of-the-art TE solutions with the lowest average network disturbance.

The remainder of this paper is organized as follows. Section II provides the problem statement of TE and the definition of network disturbance. Section III overviews FlexDATE's system design. Section IV explains how to learn a good policy to determine the number of critical flows and identify the crit-

ical flows with RL. Section V describes the LP formulations for rerouting critical flows and forwarding non-critical flows. Section VI discusses the implementation details of FlexDATE. Section VII evaluates the performance of FlexDATE by presenting and analyzing the simulation results. Section VIII lists the related works, and Section IX concludes the paper.

II. Problem Statement and Network Disturbance

In this section, we provide a brief explanation of TE and formally define network disturbance as a QoS metric to measure the impact caused by flow rerouting.

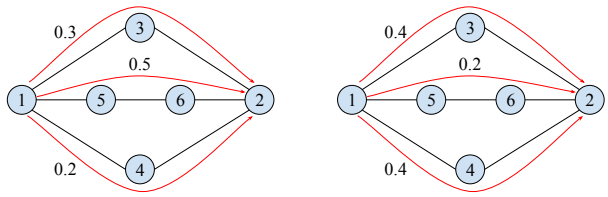
A. Problem Statement

A network can be modeled as a directed graph $G(V, E)$, where V is the set of network nodes and E is the set of directed links. Each directed link $e \in E$ has a capacity that limits the maximum traffic to be carried on that link. Given a series of traffic demands that need to be delivered from each source node $s \in V$ to each destination node $d \in V$, a Traffic Matrix (TM) can be constructed to represent the traffic demands $\{D^{s,d}\}$ of all flows $\langle s, d\rangle$ in a certain period. To properly deliver all traffic demands from different source nodes to different destination nodes, a routing strategy is required to specify how the traffic is routed over the network. A widely-adopted approach is to construct multiple preconfigured paths $\{P^{s,d}\}$ for each source-destination pair $\langle s, d\rangle$ and route the traffic on different paths. In this scenario, the routing strategy can be represented as path split ratios $\{\sigma_p^{s,d}\}$ that specify the percentage of traffic demands of each flow $\langle s, d\rangle$ routed on each path $p \in P^{s,d}$.

Generally speaking, it is important to configure good routing strategies to maintain promising network performance under dynamic traffic scenarios and unexpected link failures. Otherwise, the network may experience severe performance degradation, such as traffic loss and high latency. Therefore, TE operation is required to optimize the routing strategy to reduce network congestion probability and achieve good network performance. Given the network topology and measured TM, we can formulate an optimization problem and solve it for optimal routing. One of the common TE objectives is minimizing the Maximum Link Utilization (MLU) in the network to achieve good load balancing. Here, MLU denotes the utilization of the most congested link in the network and can be computed as $U = \max_{e \in E}(l_e/c_e)$, where l_e and c_e are the traffic load and capacity of link e, respectively. When there is a newly measured TM or failed link, TE can solve the optimization problem and obtain an updated routing to accommodate traffic/connectivity changes.

B. Network Disturbance

Once TE performs a routing update, flows might be rerouted to different paths or distributed with different split ratios along original paths compared to the previous routing. Such flow rerouting operation may lead to service disruption and QoS degradation as described in Section I. Thus, we have the following definitions.



(a) Routing at time step $t = 1$. (b) Routing at time step $t = 2$.

Fig. 1. An example to measure network disturbance caused by flow rerouting. The routing strategies for flow $\langle 1, 2\rangle$ at different time steps t are represented as traffic split ratios over different paths.

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, 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.

Fig. 1 provides an example of measuring network disturbance caused by TE operations. We assume that flow $\langle 1, 2\rangle$ is the only flow affected by routing updates from time step $t = 1$ to $t = 2$. As shown in Fig. 1(a), flow $\langle 1, 2\rangle$ is routed on three different paths 1-3-2, 1-5-6-2, and 1-4-2 at time step $t = 1$, and the corresponding path split ratios are (0.3, 0.5, 0.2). When $t = 2$, flow $\langle 1, 2\rangle$ is rerouted with different path split ratios (0.4, 0.2, 0.4) as illustrated in Fig. 1(b). In this scenario, 30% of flow $\langle 1, 2\rangle$'s traffic at $t = 2$ is rerouted from path 1-5-6-2 to the other two paths, which should be counted as rerouted traffic for calculating network disturbance.

III. System Design

The objective of FlexDATE is to maintain promising network performance with TE operations in different network scenarios and mitigate network disturbance caused by traffic rerouting. Upon traffic changes or link failures, FlexDATE exploits a combination of RL and LP to identify a set of critical flows in the network and solves an optimization problem to reroute these critical flows. When traffic is relatively stable, it is desired to select less or similar critical flows for each TM to reduce network disturbance while achieving good network performance. However, if the traffic variation becomes larger or link failure happens, the network performance could be degraded. In these scenarios, FlexDATE may need to reroute more critical flows to maintain good network performance at the cost of incurring higher network disturbance. To accommodate dynamic traffic variations and various link failure

[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 rerouted traffic.

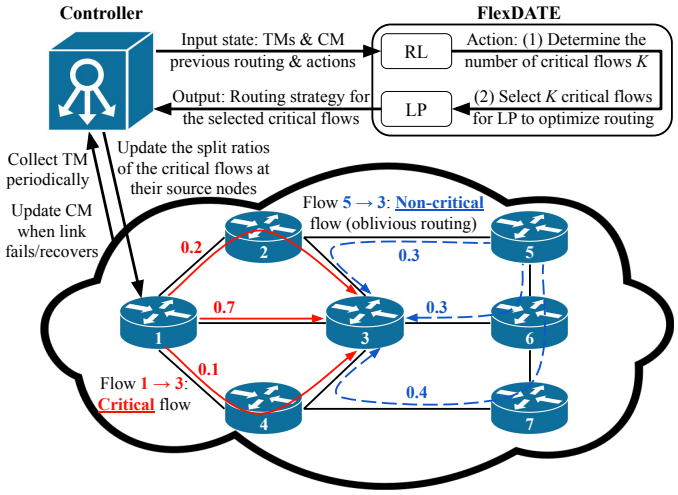


Fig. 2. Overview of FlexDATE's system design. FlexDATE selects and reroutes critical flows periodically or upon drastic traffic changes and link failure/recovery. The red solid arrows represent the path split ratios of a critical flow $\langle 1, 3\rangle$ updated by the controller, and the blue dashed arrows are the path split ratios of a non-critical flow $\langle 5, 3\rangle$ forwarded by the default oblivious routing.

scenarios, we design a novel single-stage RL architecture to determine the number of critical flows and identify the critical flows simultaneously based on different network conditions, as later shown in Section IV.

FlexDATE takes 2 consecutive TMs and a topology Connectivity Matrix (CM) as an input to select a few critical flows for rerouting. In addition, the previous routing strategy and critical flow selection are also provided as input information to assist RL in making better decisions. This is because the measurement of network disturbance is dependent on the difference between the updated routing strategy and the previous routing strategy to obtain the rerouted traffic. Given the critical flow selection from RL, FlexDATE solves an LP optimization problem with the objective to minimize MLU in the network and obtains the optimal routing strategy for critical flows. For the remaining non-critical flows, they should follow a default static routing strategy to avoid network disturbance. To ensure robust performance and keep the routing simple, we first leverage Räcke's oblivious routing algorithm [20] to generate a set of diverse and low-stretch preconfigured paths for all source-destination pairs and set the path budget to 4 as reported in [21]. Then, we optimize routing with respect to all possible traffic demands based on the LP formulation described in [27] with some customizations to support the usage of preconfigured paths (see Section V-A). The resulting optimal oblivious routing strategy is taken as the default static routing strategy for non-critical flows to bound the worst-case performance, which is more robust compared to other traditional routing strategies (e.g., Equal-Cost Multi-Path). As for the critical flows, we solve a modified Multi-Commodity Flow (MCF) problem described in Section V-B to obtain the optimal path split ratios when traffic/connectivity changes (e.g., new TM or link failure). Then, the SDN controller would update the path split ratios of the critical flows accordingly at the sender side to reroute the critical flows. Note that the SDN entries for the previously selected critical flows in the last time interval will timeout when performing routing updates.

Fig. 2 illustrates FlexDATE's system design. FlexDATE is located together with an SDN controller and consists of RL and LP modules. The SDN controller is responsible for collecting TMs periodically from the network and updating the network topology CM if any changes (e.g., link failure/recovery). Given the input from the SDN controller, FlexDATE identifies critical flows and performs flow routing/rerouting to accommodate traffic variations captured by TMs and network connectivity changes reflected by CM. For example, as depicted in Fig. 2, flow $\langle 1, 3\rangle$ is selected as a critical flow under the current network condition while flow $\langle 5, 3\rangle$ is forwarded by the default oblivious routing as a non-critical flow. To achieve better load balancing performance with low disturbance, FlexDATE only needs to update the path split ratios of the critical flow $\langle 1, 3\rangle$ at node 1 such that less amount of traffic would be routed on the two paths that share the same links with flow $\langle 5, 3\rangle$'s paths. When a link failure occurs, some of the preconfigured paths that include the broken link may be temporarily unavailable. Therefore, a simple local recovery mechanism can be implemented to time out existing SDN entries for critical flows and revert to the default oblivious routing. The traffic demands routed on the broken paths should also be rescaled to the available paths to avoid traffic loss. In the meantime, the CM would be updated according to link failures and sent to the FlexDATE system in the SDN controller. Then, TE-level resilience can be performed immediately by FlexDATE to select and reroute a new set of critical flows to maintain good network performance. The evaluation results in Section VII show that FlexDATE is able to achieve near-optimal network performance with mitigated network disturbance in the presence of dynamic traffic and different single link failures.

IV. Learning to Select Critical Flows

In this section, we explain how to learn a critical flow selection policy using a customized RL approach coupled with GNN architecture.

A. Reinforcement Learning Formulation

Input / State Space: It is important for an RL agent to take actions based on observed network conditions, including topology connectivity, traffic changes, and routing strategies. At time step t, an RL agent takes a state $s_t = (CM, TM_t, TM_{t-1}, R_{t-1}, CF_{t-1})$ as an input, where CM is the topology connectivity matrix and (TM_t, TM_{t-1}) are traffic matrices at time step t and $t-1$ representing traffic changes, respectively. R_{t-1} is the routing strategy optimized at the previous time step $t-1$, which can be interpreted as path split ratios. Since all flows are routed/rerouted based on a set of preconfigured paths with a path budget of 4, the routing strategy can be viewed as four $N \times N$ matrices where each routing matrix represents the split ratios of all flows on one of the preconfigured paths. For each flow, the path split ratios distributed over four routing matrices should sum up to 1. In addition, CF_{t-1} is an $N \times N$ matrix indicating the critical flow selection at the previous time step $t-1$. If the flow $\langle s, d\rangle$ is selected as a critical flow in the last time step, the entry (s, d)

of the critical flow matrix CF_{t-1} is set to 1; otherwise, it is set to 0. Thus, the input dimension should be $N \times N \times 8$.

Action Space: For each state s_t, FlexDATE will select K critical flows to accommodate dynamic traffic and different link failure scenarios, which is divided into two steps of action. At first, FlexDATE needs to determine the number of critical flows K to be rerouted. Given that there are $N*(N-1)$ flows in a network with N nodes, the number of critical flows K can be ranged from 1 to $N*(N-1)$. However, there is no need to identify a large number of critical flows since near-optimal network performance can be achieved by rerouting only a few critical flows [28]. Therefore, we run a series of experiments for each topology in Section VII-A to find an appropriate upper bound value K_{ub} to reduce the action space, where K_{ub} should be much smaller than $N*(N-1)$. In other words, the agent can take an action from the reduced action space {1, 2, ..., K_{ub}} to determine the number of critical flows to be selected at each time step t (i.e., a_t^0). Once the value of K is determined, RL should select K critical flows from all possible $N*(N-1)$ candidates in the second step, which requires a large action space of size $C^K_{N*(N-1)}$. Inspired by [29], [30], 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., $a_t^1, a_t^2, ..., a_t^K$).

Reward: After sampling K different critical flows f_K for a given state s_t, FlexDATE reroutes these critical flows and obtains the MLU U by solving the rerouting optimization problem (11) described in Section V-B. By comparing the updated routing with the previous routing, the incurred network disturbance DB can be calculated using Eq. (2). To measure how far the resulting routing strategy is from the optimal routing strategy, we define a performance ratio as follows:

$$PR = \frac{U_{\text{optimal}}}{U}, \tag{3}$$

where U_{optimal} is the MLU achieved by an optimal explicit routing for all flows. Since U cannot be lower than U_{optimal}, the value of PR is always capped at 1. $PR = 1$ means that FlexDATE achieves the same performance as optimal routing. The higher the PR, the better the performance. With consideration of achieving near-optimal network performance PR and mitigating network disturbance DB, we design a reward function as shown below:

$$r = \begin{cases} PR - \lambda * DB & \text{if } PR < PR_{tg} \\ PR - \mu * DB & \text{if } PR \geq PR_{tg} \end{cases} \tag{4}$$

where PR_{tg} is a preset network performance target, λ and μ are penalty factors on network disturbance at different training stages that can be adjusted to trade off performance and disturbance. In our reward function design, we adopt a performance target PR_{tg} as a criterion to separate different training stages. The core idea is to encourage the selection of critical flows that can achieve the performance target with low network disturbance. When $PR < PR_{tg}$, the network performance target cannot be satisfied. Therefore, the penalty factor λ on network disturbance should be set to a smaller value such that RL would benefit more from improving network performance. In other words, RL would focus on improving network performance at the early training stage to learn a good policy. Once the network performance target is achieved (i.e., $PR \geq PR_{tg}$), we should impose a higher penalty μ on network disturbance such that RL would focus on mitigating network disturbance while ensuring good performance. In our evaluation, we set $PR_{tg} = 0.9$ with an aim to achieve near-optimal performance (i.e., 90% of optimal performance), while λ and μ are configured as 0.5 and 1, respectively. We discuss the impact of different penalty factor settings in Section VII-B.

B. Training Algorithm

The K value selection policy is represented by a neural network (denoted as K-net), and the critical flow selection policy is represented by another neural network (denoted as CF-net). Both two policy networks take a shared state $s_t = (CM, TM_t, TM_{t-1}, R_{t-1}, CF_{t-1})$ as an input and output a probability distribution over all available actions. For K-net, we sample an action from the probability distribution $\pi(a_t^0|s_t)$ for each state s_t to determine the K value. As for CF-net, since K different actions are sampled from the probability distribution $\pi(a_t|s_t)$ for each state s_t and their order doesn't matter, we define a solution $a_{t_K} = (a_t^1, a_t^2, ..., a_t^K)$ 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 training is to achieve near-optimal network performance and mitigate network disturbance over various network conditions (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 $b(s_t)$. It is worth mentioning that a good baseline $b(s_t)$ reduces gradient variance and thus increases the speed of learning. In this paper, we use a learned estimate of the value function $V^{\pi_\theta}(s_t)$ from a critic network as the baseline $b(s_t)$. The critic network parameter θ_v is updated according to the following equation:

$$\theta_v \leftarrow \theta_v - \alpha_v \sum_t \nabla_{\theta_v} (r_t - V^{\pi_\theta}_{\theta_v}(s_t))^2, \tag{6}$$

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 networks. Once training is done, only the two policy networks K-net and CF-net are required to execute the action selection. 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 improve the exploration by discouraging premature convergence to suboptimal deterministic policies [31]. Then, the policy parameter θ_k of K-net and the policy parameter θ

[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.

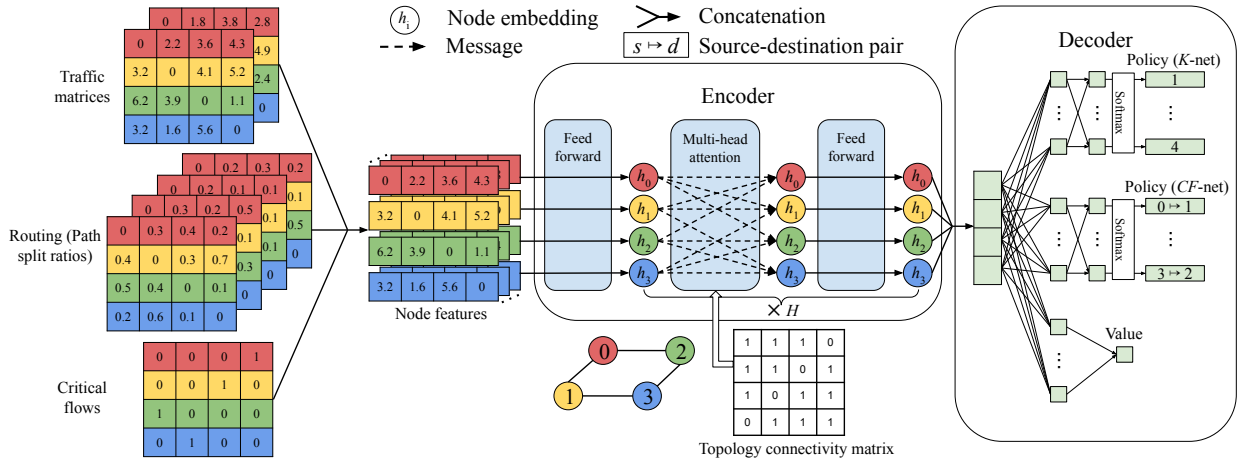


Fig. 3. GNN-based actor-critic architecture. The encoder computes initial node embeddings using a shared Feed-Forward (FF) layer based on the node features concatenated from input states, and then updates per-node embeddings using H attention layers according to the topology connectivity matrix. Each attention layer consists of a Multi-Head Attention (MHA) layer and a node-wise fully connected FF layer. The final node embeddings are concatenated and fed into the decoder for K-net and CF-net policy networks to generate probability distributions over actions, and for the critic network to predict the baseline.

of CF-net are updated according to the following equations:

$$\theta_k \leftarrow \theta_k + \alpha_k \sum_t \nabla_{\theta_k} log\pi_{\theta_k}(a_t^0|s_t)(r_t - V_{\theta_v}^{\pi_\theta}(s_t)) + \beta_k \nabla_{\theta_k} H(\pi_{\theta_k}(\cdot|s_t)), \tag{7}$$

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

where α_k and α are the learning rates for the policy networks K-net and CF-net, respectively, and H is the entropy of the policy (the probability distribution over actions). The hyperparameters β_k and β are set to control the strength of the entropy regularization term. Intuitively, Eq. (8) can be explained as follows. $(r_t - V_{\theta_v}^{\pi_\theta}(s_t))$ 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. If $(r_t - V_{\theta_v}^{\pi_\theta}(s_t))$ 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 $\alpha(r_t - V_{\theta_v}^{\pi_\theta}(s_t))$. Otherwise, the probability of the solution is decreased. For K-net, Eq. (7) can be explained in a similar way. Overall, the net effect of Eqs. (7) and (8) is to reinforce actions that empirically lead to better rewards. Algorithm 1 shows the pseudo-code for the training algorithm.

Algorithm 1 Training Algorithm

```
Initialize θ_k, θ, θ_v
for each iteration do
    {s_t} ← Sample a batch of states with size B
    for t = 1, ..., B do
        Sample an action a_t^0 according to policy π_{θ_k}(a_t^0|s_t)
        K ← a_t^0
        Sample a solution a_{t_K} according to policy π_θ(a_{t_K}|s_t)
        Receive reward r_t
    end for
    Update θ_k and θ according to Eqs. (7) and (8)
    Update θ_v according to Eq. (6)
end for
```

C. Actor-Critic Network Architecture

Fig. 3 shows the architecture of the GNN-based actor-critic network. The inputs to the network are node features and topology connectivity matrix, where the features for a given node are: (1) a series of demands originated from that node, (2) the previous routing (i.e., path split ratios) of all flows generated from that node, and (3) which flows originated from that node are previously selected as critical flows. The encoder computes the initial node embedding using a shared Feed-Forward (FF) layer, and then each node's embedding is updated by exchanging messages with its neighbors. Similar to the transformer model presented in [30], [32], the embedding update module consists of a stack of H identical attention layers. Each attention layer is composed of two sub-layers. The first is a Multi-Head Attention (MHA) layer that performs message exchanges between neighbor nodes, and the second is a node-wise fully connected FF layer that performs a nonlinear transformation. In addition, a skip connection [33] and layer normalization [34] are applied to each sub-layer. For a given node v, the node embedding h_v is updated iteratively with a message function $f(\cdot)$ that is parameterized by GNN to aggregate the messages passed from its neighbors [30], [32]:

$$h_v^{l+1} = \sum_{w \in \chi_v} f(h_v^l, h_w^l, \theta_f^l), \tag{9}$$

where χ_v is the set of nodes that exchange messages with node v, and θ_f denotes learnable function parameters.

Employing H attention layers can be interpreted as executing H iterations of the embedding update process, and one iteration of the 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. Thus, when H is set to the number of max hops in the network, it would be enough for each node to capture the complete information of the whole network. All node embeddings outputted by the encoder are then concatenated to form a graph embedding, which is passed to the two policy networks K-net and CF-net to generate probability distributions over actions and passed to the critic network to predict the baseline of the input state.

TABLE I
NOTATIONS

$G(V, E)$	network with nodes V and directed links E ($\|V\| = N, \|E\| = M$)
f_K	selected critical flows
c_e	the capacity of link e ($e \in E$)
l_e	the traffic load on link e ($e \in E$)
$D^{s,d}$	the traffic demand from source s to destination d ($s, d \in V$, $s \neq d$)
$P^{s,d}$	the preconfigured path set from source s to destination d ($s, d \in V$, $s \neq d$)
$\sigma_p^{s,d}$	the percentage of traffic demand from source s to destination d routed on the preconfigured path p ($p \in P^{s,d}, s, d \in V, s \neq d$)
$\delta_{p,e}^{s,d}$	= 1, if link e belongs to the preconfigured path p from source s to destination d; 0, otherwise ($e \in E, p \in P^{s,d}, s, d \in V, s \neq d$)
$\rho(e, e^*)$	the weight for link pair (e, e^*) ($e, e^* \in E$)
$\varphi^{s,d}(e)$	the path length from source s to destination d according to link weights $\rho(e, e^*)$ for all link e^* ($e \in E, s, d \in V, s \neq d$)

V. Rerouting Critical Flows

In this section, we describe how to reroute the selected critical flows to balance the link utilization of the network. The notations used in this section are listed in Table I.

A. Optimal Oblivious Routing

By default, traffic is distributed according to optimal oblivious routing, which is oblivious to the actual traffic demands. The original oblivious routing LP model presented in [27] computes a routing that specifies the fraction of demands of each flow routed on each link, which requires $O(MN^2)$ routing variables to be solved. To reduce the computation overhead and the complexity of routing input to RL, we modify the LP formulation to route all flows on the preconfigured paths computed by [20] and set the path budget to 4 [21]. The resulting optimal oblivious routing is represented as traffic split ratios of each flow on its corresponding preconfigured path set with at most $4N^2$ routing variables.

Given a topology $G(V, E)$ with a set of preconfigured paths $\{P^{s,d}\}$ for each source-destination pair, the objective of the oblivious routing problem is to obtain the optimal path split ratios $\{\sigma_p^{s,d}\}$, so that the oblivious performance ratio OR with respect to all possible demands $\{D^{s,d} \geq 0\}$ is minimized. Then, the oblivious routing problem can be formulated as an optimization problem as follows.

$$\min OR \tag{10a}$$

subject to

$$\sigma_p^{s,d} \geq 0 \qquad p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{10b}$$

$$\sum_{p \in P^{s,d}} \sigma_p^{s,d} = 1 \qquad s, d \in V,\ s \neq d \tag{10c}$$

$$\sum_{e^* \in E} c_{e^*} \cdot \rho(e, e^*) \leq OR \qquad e \in E \tag{10d}$$

$$\sum_{p \in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} \leq c_e \cdot \varphi^{s,d}(e) \qquad e \in E,\ s, d \in V,\ s \neq d \tag{10e}$$

$$\sum_{e^* \in p} \rho(e, e^*) - \varphi^{s,d}(e) \geq 0 \qquad e \in E,\ p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{10f}$$

$$\rho(e, e^*) \geq 0 \qquad e, e^* \in E \tag{10g}$$

$$\varphi^{s,d}(e) \geq 0 \qquad e \in E,\ s, d \in V,\ s \neq d \tag{10h}$$

Eqs. (10b) and (10c) are the path split ratio constraints. Eqs. (10d)-(10h) are the constraints for oblivious routing with preconfigured paths. Note that $\rho(e, e^*)$ and $\varphi^{s,d}(e)$ are two auxiliary variables used for simplifying the optimal oblivious routing problem as a single polynomial-sized LP. For simplicity, we omit the LP duality analysis that introduces these two auxiliary variables. Please refer to [27] for more details.

B. Explicit Routing for Critical Flows

In FlexDATE, non-critical flows are routed by the static optimal oblivious routing derived from Eq. (10). For the selected critical flows f_K, we can reroute these critical flows by conducting explicit routing optimization.

The critical flow rerouting problem can be described as follows. Given a network $G(V, E)$ with a set of traffic demands $\{D^{s,d}\}$ and preconfigured paths $\{P^{s,d}\}$ for the selected critical flows ($\forall \langle s, d \rangle \in f_K$), our objective is to obtain the optimal explicit routing ratios $\{\sigma_p^{s,d}\}$ of each critical flow such that U is minimized. Note that the background link load $\{\bar{l}_e\}$ is contributed by the remaining non-critical flows with the default oblivious routing. Therefore, we formulate the critical flow rerouting problem as an optimization problem as follows.

$$\min U \tag{11a}$$

subject to

$$\sigma_p^{s,d} \geq 0 \qquad p \in P^{s,d}, s, d : \langle s, d \rangle \in f_K \tag{11b}$$

$$\sum_{p \in P^{s,d}} \sigma_p^{s,d} = 1 \qquad s, d : \langle s, d \rangle \in f_K \tag{11c}$$

$$l_e = \sum_{\langle s,d \rangle \in f_K} \sum_{p \in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} \cdot D^{s,d} + \bar{l}_e \qquad e \in E \tag{11d}$$

$$l_e \leq c_e \cdot U \qquad e \in E \tag{11e}$$

Eqs. (11b) and (11c) are the path split ratio constraints for the selected critical flows. Eq. (11d) indicates that the traffic load on link e is contributed by the traffic demands of critical flows routed by explicit routing and the traffic demands of non-critical flows routed by the default oblivious routing. Eq. (11e) is the link capacity utilization constraint.

After solving the above problem using LP solvers (e.g., Gurobi [35]), we can obtain the optimal explicit routing strategy $\{\sigma_p^{s,d}\}$ ($\forall \langle s, d \rangle \in f_K$) for the selected critical flows. Then, the SDN controller would update the path split ratios of the rerouted critical flows at their source nodes accordingly.

VI. Implementation

In this section, we describe the experimental setup and implementation details of FlexDATE.

A. Dataset

In our evaluation, we use five different real-world network topologies, including the Abilene network, CERNET network, GÉANT network, and two ISP networks collected by Rocketfuel [36]. The numbers of nodes, directed links, and source-destination pairs of the five networks are listed in Table II.

The Abilene, CERNET, and GÉANT networks are the research and education networks from the United States, China, and Europe, respectively. For the Abilene network, the network topology information (such as link connectivity, costs, and capacities) and measured TMs can be found in [37]. For the CERNET network, both the network topology and real TMs are obtained from [38]. Since the Abilene TMs and CERNET TMs are measured every 5 minutes, there are a total of 2016 continuous TMs each week. To evaluate the performance of FlexDATE and correctly measure network disturbance with consecutive TMs in the Abilene network, 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). Similarly, we use the first week's CERNET TMs (starting from Feb. 19th, 2013) for training and the second week's CERNET TMs (starting from Feb. 26th, 2013) for evaluation. For the GÉANT network, the topology information including link capacities and costs are provided in [39]. The GÉANT TMs are available at [40] and they are collected every 15 minutes. 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 Sprintlink and Tiscali networks collected by Rocketfuel, only the link costs are given while the link capacities 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 [41]–[43]. Since there are no measured TMs available for these two networks, we synthesize a series of spatiotemporal TMs using the Modulated Gravity Model (MGM) [44], [45]. MGM can construct spatial properties with gravity-model-like constraints and utilize sinusoids to reflect the cyclical nature of TMs, which is suitable for emulating the characteristics of real traffic. To evaluate FlexDATE in different traffic scenarios, we combine hourly and daily traffic patterns in MGM and use different parameters to control traffic variations. Specifically, half of the TMs are generated with large traffic variations to represent dynamic traffic scenarios, and the rest of the TMs are relatively stable with small traffic variations. For stable TMs, the daily and hourly Peak-to-Mean (PM) ratios are separately configured as 1.1 and 1.05 to limit traffic variations, and the spatial variance of daily traffic is only 1.5. As for dynamic TMs, we use a higher spatial variance of 3 for daily traffic and set the daily and hourly PM ratios to 5 and 1.5, respectively. To simulate extreme conditions, we further introduce an exponential model [44] for dynamic TMs to increase traffic variations. In our evaluation, both the training set and test set include 100 dynamic TMs and 100 stable TMs.

TABLE II
REAL-WORLD NETWORK TOPOLOGIES FOR EVALUATION

Topology	Nodes	Directed Links	S-D Pairs
Abilene	12	30	132
CERNET	14	32	182
GÉANT	23	74	506
Sprintlink (US)	44	166	1892
Tiscali (Europe)	49	172	2352

B. Hyperparameters

The GNN-based actor-critic architecture is implemented using TensorFlow [46]. For the encoder, we set the embedding dimension to 64 and the number of attention heads to 8. For the FF sub-layer in each attention layer, the intermediate layer dimension is set to 256. Besides, the number of attention layers H is set to the number of max hops in each network to ensure complete message exchange. With fewer attention layers, the training speed could be slightly increased, but the performance of FlexDATE would also be degraded due to potential information loss. For the decoder, the intermediate layer dimension is set to 128 with Leaky ReLU as the activation function. The FF layer of the K-net policy network has an output dimension of K_{ub} (i.e., the maximum number of critical flows), where the value of K_{ub} for each topology is later shown in Table III. For the CF-net policy network, the FF layer 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. A softmax function is applied to the output of each policy network to generate the probabilities for all available actions. The critic network is similar to the policy networks except that the last layer is a fully connected linear layer with only one neuron corresponding to the baseline $b(s_t)$. Additionally, the learning rates α_k, α, and α_v are initially configured as 0.0001 with a decay rate of 0.96 every 500 iterations, while the entropy factors β_k and β are set to 0.1. We do not use sophisticated hyperparameter tuning methods [47] and fix all these hyperparameters throughout our experiments. The experiment results in Section VII demonstrate that FlexDATE works well in different network conditions with a single set of hyperparameters.

C. Parallel Training

To speed up training, we spawn multiple actor agents in parallel as suggested by [31]. 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 networks and critic network. The central learner agent performs a gradient update using Eqs. (6)-(8) according to the received tuples and then sends back the updated network parameters to the actor agents. In our evaluation, we use 20 actor agents with 21 CPU cores (one 2.9 GHz core for each agent) and 32 GB memory to train FlexDATE in a high-performance computing cluster. The training process is greatly facilitated due to our scalable neural network architecture design. It takes approximately 8 hours to train a FlexDATE model from scratch for small networks (i.e., Abilene and

CERNET) with 8,000 iterations. In the GÉANT network, 15,000 iterations are needed to reach convergence with 15 hours of training. For the remaining ISP networks with more than 40 nodes, it requires less than 1 day's 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 resource consumption can be adjusted based on different hardware specifications (e.g., 4 CPU cores in a server). Once the training is done, we can deploy FlexDATE in the SDN controller, where the inference time is less than one second and the resource consumption is relatively low (see Section VII-E).

D. 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 FlexDATE in mitigating network disturbance and improving network performance, we compare FlexDATE to the following schemes:

1) **Optimal Oblivious Routing (OR)**: optimizes a routing with respect to all possible TMs using an LP formulation described in Eq. (10) that is modified from [27], and distributes traffic on the preconfigured paths according to optimal oblivious routing without the knowledge of actual traffic demands.
2) **Equal-Cost Multi-Path (ECMP)** [48]: distributes traffic evenly among available next hops along the shortest paths based on the link cost settings.
3) **DATE** [26]: customizes an actor-critic RL approach to select 20% of total flows as critical flows and further utilizes LP to reroute these critical flows, while non-critical flows are routed by ECMP. A fixed network disturbance target is specified during RL training to limit network disturbance.
4) **DATE-OR**: modifies DATE [26] by leveraging OR to forward non-critical flows instead of using ECMP.
5) **Top-K**: selects K flows with the largest demand volume from a given TM and reroutes these flows using the LP formulation in Eq. (11), where the value of K is determined by the K-net policy network of FlexDATE. This heuristic method is based on the assumption that elephant flows would have a dominant impact on network performance.
6) **SMORE** [21]: computes a path set using an oblivious routing algorithm [20] and deploys a centralized controller to dynamically adapt the split ratios of all flows for each TM according to these preconfigured paths.

It is worth mentioning that FlexDATE, DATE-OR, Top-K, and OR are using the same preconfigured paths as SMORE. Both FlexDATE, DATE-OR, and Top-K take OR as the default routing strategy for non-critical flows, while DATE uses ECMP to forward non-critical flows. To compare the performance of FlexDATE and baseline methods to optimal routing, we compute the performance ratio PR of each TE solution according to Eq. (3). Besides, the corresponding network disturbance DB is calculated using Eq. (2) in our evaluation to measure the impact of flow rerouting caused by different TE solutions. Note that OR and ECMP would not incur any network disturbance since they do not perform routing updates when traffic changes.

VII. Evaluation

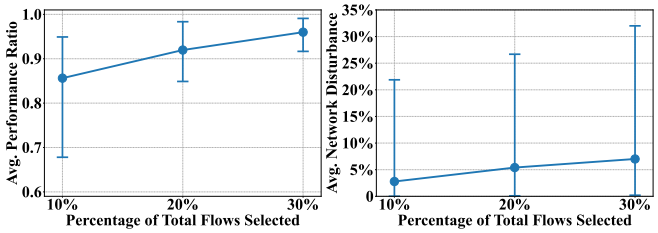
Extensive simulations are conducted based on five real-world network topologies and their TMs to evaluate the performance of FlexDATE in dynamic traffic conditions and single link failure scenarios. We compare FlexDATE to state-of-the-art TE solutions to show the effectiveness of FlexDATE in mitigating network disturbance while achieving near-optimal network performance.

A. The Number of Critical Flows

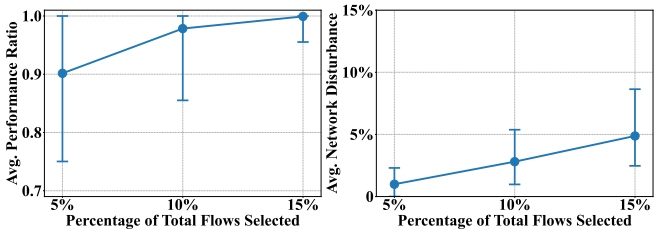
It is important for FlexDATE to select and reroute a proper number of critical flows K to achieve the predefined network performance target with low network disturbance. If the number of critical flows is too small, it might be difficult to achieve good network performance. On the contrary, rerouting too many critical flows may introduce considerable network disturbance. To investigate the influence of rerouting 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. Meanwhile, we need to determine an appropriate upper bound value K_{ub} for each network to reduce the action space of K-net as described in Section IV-A. To find a good candidate for K_{ub}, we temporarily disable K-net and train FlexDATE with a unique K value that is fixed for all TMs in each experiment. Besides, we also temporarily remove the penalty on disturbance in the reward function such that FlexDATE would aim at maximizing network performance without considering network disturbance in these experiments. The reason is that FlexDATE needs to ensure good performance in the worst case when $K = K_{ub}$, even though it could introduce high network disturbance at the moment. Note that the comparison is mainly performed on the training dataset of the Abilene and Sprintlink networks since the test dataset should be unknown when we determine the K_{ub} value for RL training. The results for other networks are similar.

Trade-off between performance and disturbance. Fig. 4 shows the average performance ratio and network disturbance of FlexDATE with an increasing number of critical flows selected in the Abilene and Sprintlink networks, 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. Thus, 10% of total flows selected means that the number of critical flows is $K = 10\% * N * (N - 1)$. From Fig. 4, we can see that the average performance ratio becomes higher as FlexDATE selects and reroutes more critical flows in the two networks. However, the corresponding network disturbance is also increasing as K grows up, which demonstrates the trade-off between network performance and disturbance.

Configuration of upper bound value. As depicted in Fig. 4(a), the worst-case performance ratio is above the performance target $PR_{tg} = 0.9$ when FlexDATE selects 30% of



(a) Network performance and disturbance in the Abilene network.



(b) Network performance and disturbance in the Sprintlink network.

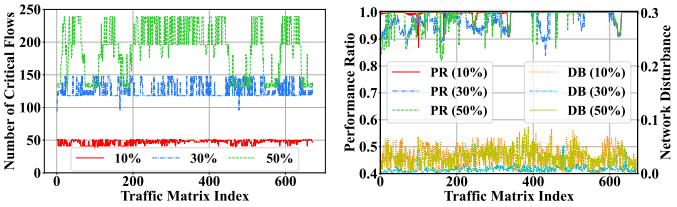
Fig. 4. Evolution of FlexDATE's average performance ratio and network disturbance with an increasing number of critical flows K selected and rerouted in the Abilene and Sprintlink networks. The error bars span from the average value to the highest/lowest value achieved on the training dataset.

TABLE III
UPPER BOUND FOR THE NUMBER OF CRITICAL FLOWS

Topology	Max Critical Flows K_{ub}	% of Total Flows
Abilene	40	30%
CERNET	36	20%
GÉANT	51	10%
Sprintlink	284	15%
Tiscali	118	5%

total flows for rerouting. Thus, we set $K_{ub} = 30\% * N * (N-1)$ for the Abilene network. Similarly, we configure the maximum number of critical flows to $K_{ub} = 15\% * N * (N-1)$ for the Sprintlink network. Such K_{ub} settings are demonstrated to work well in the test dataset, as later shown in Section VII-C. Once the upper bound K_{ub} is determined, FlexDATE can be trained to choose a K value from 1 to K_{ub} and then select K critical flows for rerouting, which provides more flexibility to handle different network scenarios. As shown in Fig. 4(b), when selecting 5% of total flows in the Sprintlink network, FlexDATE can achieve $PR = 1$ in the extreme cases. It means that $K = 5\% * N * (N-1)$ is good enough for some Sprintlink TMs to achieve optimal performance with low disturbance. However, the worst-case performance ratio is only 0.75 with 5% of total flows rerouted, which implies that FlexDATE should reroute more flows (e.g., 15% of total flows) in some traffic scenarios to maintain good performance. In the following experiments, the upper bound value K_{ub} for each network is set to a portion of total flows as shown in Table III, such that FlexDATE can ensure good network performance with reduced network disturbance.

Sensitivity analysis of upper bound value. One potential issue is that it could be too conservative if we determine the upper bound value K_{ub} based on the training dataset. Even though the K_{ub} settings in Table III work well in our evaluation, FlexDATE may need to reroute more critical flows than the current upper bound value K_{ub} to maintain good performance in unexpected traffic scenarios during online deployment. Therefore, we need to consider the trade-off



(a) Number of critical flows K. (b) Performance and disturbance.

Fig. 5. Sensitivity analysis of FlexDATE with different upper bound value K_{ub} settings (10%, 30%, and 50% of total flows) in the GÉANT network.

between performance guarantee in unexpected traffic scenarios and RL model complexity/convergence when determining the K_{ub} value. To understand the effect of different K_{ub} settings, we perform a sensitivity analysis based on the test dataset of the GÉANT network, where FlexDATE is trained with K_{ub} = 10%, 30%, and 50% of total flows in the network, respectively. It is worth noting that K_{ub} = 10% of total flows is generally good enough for FlexDATE to achieve the performance target for all GÉANT TMs, as shown in Table III. In other words, the K_{ub} settings with 30% or 50% of total flows are much larger than required.

Fig. 5 shows the number of critical flows selected for each test TM and the corresponding network performance and disturbance in the GÉANT network under different K_{ub} settings. From Fig. 5(a), we can see that the number of critical flows K becomes larger as K_{ub} increases, and the K value is also changing more drastically and frequently. Although FlexDATE does not converge to a smaller K value when a large K_{ub} is configured, it is still able to achieve the performance target in most cases with low network disturbance incurred, as shown in Fig. 5(b). Compared to the original 10% K_{ub} setting, FlexDATE incurs lower network disturbance with more critical flows selected under the 30% K_{ub} setting. This interesting observation reveals that a large K value does not essentially lead to high disturbance, especially when K is much higher than required. To explain this, we can refer to Definitions 1 and 2, where network disturbance is measured as the percentage of total rerouted traffic by comparing the difference between the updated routing and the previous routing. If the updated routing does not change too much compared to the previous routing (e.g., RL selects similar critical flows or LP's solution is similar), the resulting network disturbance could be lower even though the number of critical flows K is larger. Given the rich information from input states and the effective reward function design for balancing performance and disturbance, FlexDATE can learn a good policy under different K_{ub} settings to achieve good performance with mitigated disturbance.

Another interesting observation from Fig. 5(b) is that there are more traffic scenarios in the test dataset where FlexDATE cannot achieve the performance target ($PR_{tg} = 0.9$) with a larger K_{ub} value employed. This is because the model complexity of RL becomes higher as K_{ub} increases, and thus is more difficult for RL to reach convergence with a higher training overhead. Even though a larger K_{ub} may be useful to guarantee performance in future unexpected traffic scenarios, it would lead to a much larger solution space to explore during the training procedure. Given a network with N nodes, there are $N * (N-1)$ flows in the network and $\sum_{K=1}^{K_{ub}} C_{N*(N-1)}^{K}$

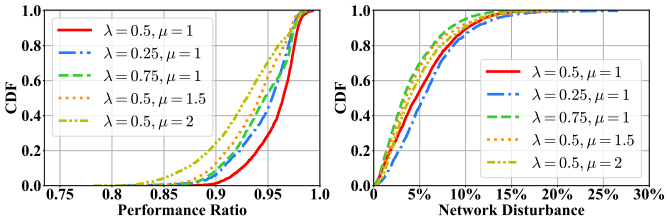


Fig. 6. CDF of performance ratio and network disturbance in the Abilene network with different penalty factor settings in FlexDATE's reward function.

possible combinations of critical flows with consideration of flexible K values, which makes the critical flow selection problem extremely difficult if we configure a large K_{ub} value. In this situation, it is necessary to reduce the action space with a proper upper bound value K_{ub} to ensure convergence. Since we have similar observations in the other four networks, we adopt a relatively conservative method to decide the K_{ub} value as shown in Table III, which is demonstrated to work well in the following experiments. When it comes to actual deployment, network operators can consider using a K_{ub} value that is slightly higher than needed to balance performance guarantee and model complexity.

B. Penalty Factor Settings

The reward function design in Eq. (4) is essential for FlexDATE to learn a good critical flow selection policy towards near-optimal performance and low disturbance. One of the key aspects is to assign appropriate values to the penalty factors λ and μ on network disturbance at different training stages. Thus, we conduct a series of experiments to investigate the impact of penalty factor settings in the Abilene network.

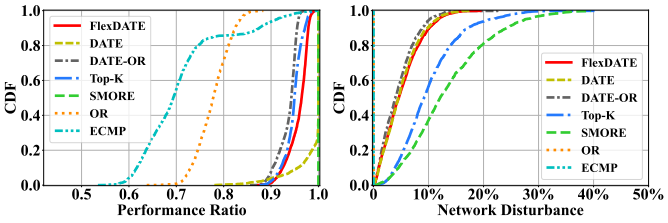
Fig. 6 shows the Cumulative Distribution Function (CDF) of network performance and disturbance with FlexDATE under different penalty factor settings. When λ is fixed to 0.5, FlexDATE would achieve lower network performance and disturbance as μ increases from 1 to 2. Intuitively, larger μ means that RL would be more sensitive to network disturbance when the performance target is satisfied, so the learned policy would avoid high disturbance at the cost of slight performance degradation. When μ is fixed to 1, network disturbance is decreasing as λ increases from 0.25 to 0.75. This is reasonable since RL would prioritize disturbance mitigation over performance improvement with a larger penalty factor. However, there is a pivot at $\lambda = 0.5$ where FlexDATE would achieve the best performance. Given that λ is activated when the performance target is not satisfied, it would affect RL training in the early training stage when RL is exploring the action space without concrete ideas about how to identify critical flows to achieve the performance target. Therefore, it is essential to find a good balance between performance and disturbance to facilitate the training process and achieve convergence. In this scenario, $\lambda = 0.5$ and $\mu = 1$ is suitable for FlexDATE with almost all TMs satisfying the performance target. Since we have similar observations in the other four networks, we adopt $\lambda = 0.5$ and $\mu = 1$ as the default penalty factor settings in our evaluation.

C. Comparison of Performance and Disturbance

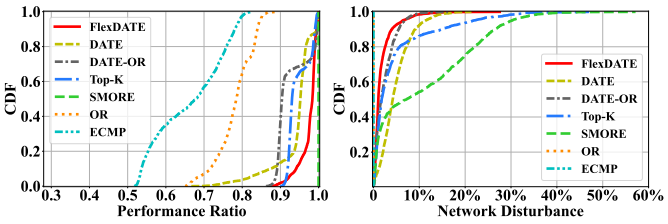
For comparison, we calculate the performance ratio of FlexDATE and the baseline methods according to Eq. (3) and measure the corresponding network disturbance using Eq. (2). As mentioned in Section IV-A, we set the performance target PR_{tg} to 0.9 such that FlexDATE can achieve near-optimal performance (i.e., 90% of optimal performance) and mitigate network disturbance. For DATE and DATE-OR, we use a disturbance target of 15% to balance network performance and disturbance as described in [26].

Performance analysis of FlexDATE. Fig. 7 illustrates the comparison of different schemes in terms of performance ratio and network disturbance in the Abilene, CERNET, and GÉANT networks. For the three networks with real traffic traces, FlexDATE can achieve near-optimal performance and satisfy the performance target of $PR_{tg} = 0.9$ for 99% of the test TMs in the second week, where the average performance ratio in the three networks is 95.8%, 97.5%, and 99.5%, respectively. Meanwhile, the average network disturbance incurred by FlexDATE is only 5.13%, 1.83%, and 2.96% in the Abilene, CERNET, and GÉANT networks, respectively. In other words, FlexDATE has the lowest average network disturbance among different TE solutions in the CERNET and GÉANT networks except for the static OR and ECMP, which demonstrates the effectiveness of FlexDATE to mitigate network disturbance and service disruptions. This is because FlexDATE is able to adaptively adjust the number of critical flows K and properly select and reroute a set of critical flows for different TMs to accommodate dynamic traffic changes. As depicted in Fig. 8, the number of critical flows K selected by FlexDATE is ranging between 21-40 for the Abilene network and 26-36 for the CERNET network in accordance with various traffic scenarios to achieve the performance target with reduced network disturbance.

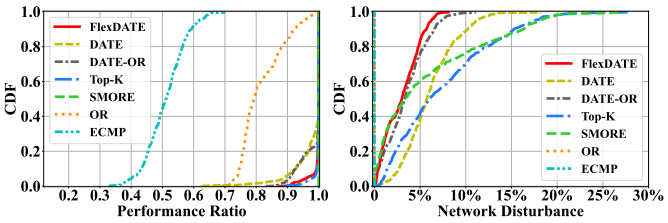
Comparison against baseline methods. For the baseline methods, DATE can effectively limit network disturbance in the extreme cases with a preset disturbance target (i.e., 15%). In the Abilene network, DATE incurs low network disturbance on average that is comparable to FlexDATE, while the maximum network disturbance of DATE is only 17.5%, which is the lowest value among all TE solutions that require routing updates. However, DATE would experience severe performance degradation in the extreme cases even though the average performance looks promising. In the Abilene, CERNET, and GÉANT networks, DATE only achieves a performance ratio of 78%, 66.7%, and 62.7% in the worst case, respectively, which reveals the limitation of DATE when dealing with drastic traffic changes. By substituting the default ECMP routing with the robust OR, DATE-OR can achieve better worst-case performance with lower disturbance compared to the original DATE. However, due to the strict restriction on network disturbance and fixed number of critical flows, both DATE and DATE-OR lack the agility to adapt to large traffic variations and their routing updates are less effective. In contrast, FlexDATE can select flexible numbers of critical flows with two policy networks and use a performance target instead to ensure good performance in



(a) Abilene network.



(b) CERNET network.

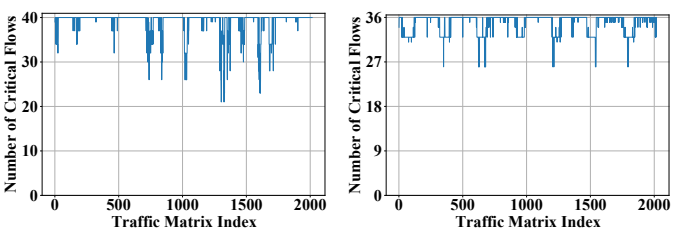


(c) GÉANT network.

Fig. 7. Comparison of performance ratio and network disturbance in CDF among FlexDATE and the baseline methods in the second week of the three networks with real traffic traces. Note that OR and ECMP are static routing strategies without network disturbance.

different traffic scenarios. As shown in Fig. 7, the worst-case performance of FlexDATE in the three networks is 87%, which can outperform DATE and DATE-OR by at most 24.3% and 3.8%, respectively. Although DATE-OR can achieve a comparable worst-case performance ratio as FlexDATE, there is no guarantee for DATE-OR to achieve a performance target since it only constrains network disturbance in the reward function. In Fig. 7(b), there are 37.3% of traffic scenarios in the CERNET network where DATE-OR cannot provide near-optimal performance (i.e., $PR < 0.9$), while FlexDATE can satisfy the performance target for 99% of test TMs. Thus, FlexDATE achieves a better trade-off between network performance and disturbance compared to DATE-OR with an improved RL design.

For Top-K, the value of K is determined by FlexDATE's K-net policy network. From Fig. 7, we can see that Top-K achieves similar performance as FlexDATE but with much higher network disturbance. This is because Top-K would reroute K flows with the largest traffic demands. Such observations demonstrate the importance of the CF-net policy network in FlexDATE to select a proper set of critical flows for rerouting, given that the K value determined by K-net alone cannot effectively mitigate network disturbance when combined with a heuristic approach. SMORE can achieve almost the same performance as optimal routing, which demonstrates the feasibility and good properties of the preconfigured paths used by SMORE. However, SMORE would also incur high network disturbance when updating the path split ratios of all flows in the network. Compared to SMORE, FlexDATE can reduce the average and maximum network disturbance



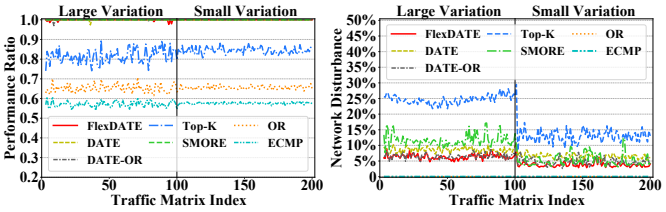
(a) Abilene network (K_{ub} = 40). (b) CERNET network (K_{ub} = 36).

Fig. 8. Number of critical flows K selected by FlexDATE for each test TM in the second week of the Abilene and CERNET networks.

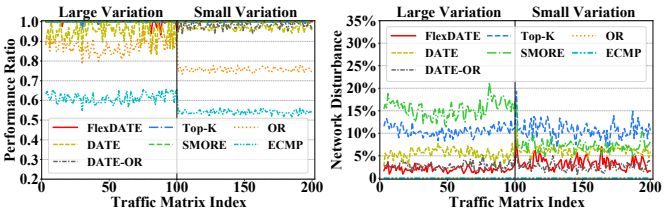
by 2.9%-9.1% and 19.2%-29.6% in the three networks listed in Fig. 7, respectively. Thus, rerouting a small set of critical flows is a good choice to trade off network performance and disturbance. As for the static routing strategies, their performance ratios are not very promising since there are no routing updates when traffic changes. However, OR can provide a better worst-case performance guarantee compared to ECMP. By forwarding non-critical flows with OR, the worst-case performance of FlexDATE and DATE-OR becomes much better than DATE. Therefore, OR is suitable to serve as the default routing strategy for non-critical flows in FlexDATE.

Generalization over different traffic variations. To validate the capability of FlexDATE in handling different traffic fluctuation scenarios in large networks, we conduct experiments in the Sprintlink and Tiscali networks with synthesized TMs as described in Section VI-A. Fig. 9 shows the performance and disturbance comparison on each synthesized test TM with different traffic variations, where the first 100 TMs represent dynamic traffic scenarios and the remaining 100 TMs are relatively stable in terms of traffic variation. When it comes to the comparison of different traffic fluctuation scenarios, we can observe more frequent performance degradation and higher network disturbance for several TE solutions in the dynamic TMs compared to the stable TMs. For instance, the performance ratio of DATE could be lower than OR with dynamic TMs in the Tiscali network, as illustrated in Fig. 9(b). Due to drastic traffic changes, the routing optimized based on previous traffic conditions would be less effective in dynamic traffic scenarios, which requires more routing changes to maintain good performance with higher disturbance.

Unlike DATE, FlexDATE can achieve the performance target $PR_{tg} = 0.9$ for all TMs with a worst-case performance ratio of 97.6% and 93.2% in the Sprintlink and Tiscali networks, respectively. As shown in Fig. 10, FlexDATE tends to select different numbers of critical flows in dynamic scenarios to accommodate drastic traffic changes, while the K values in stable scenarios are relatively constant. Moreover, the average performance ratio of FlexDATE is 99.9% in the two networks while rerouting less than 5.1% of the total network traffic on average, which demonstrates the capability of FlexDATE to accommodate different traffic fluctuation scenarios. One interesting finding is that FlexDATE would incur higher disturbance in the stable TMs compared to the dynamic TMs in the Tiscali network. This observation also holds for other approaches that reroute a small set of flows, such as Top-K and DATE. One possible reason is that the performance of OR and

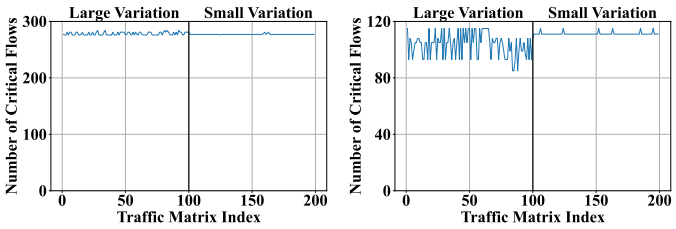


(a) Sprintlink network.



(b) Tiscali network.

Fig. 9. Comparison of performance ratio and network disturbance on each test TM of the Sprintlink and Tiscali networks with different traffic variations. The first 100 TMs represent dynamic traffic scenarios with large variations, while the remaining 100 TMs are relatively stable with small variations.



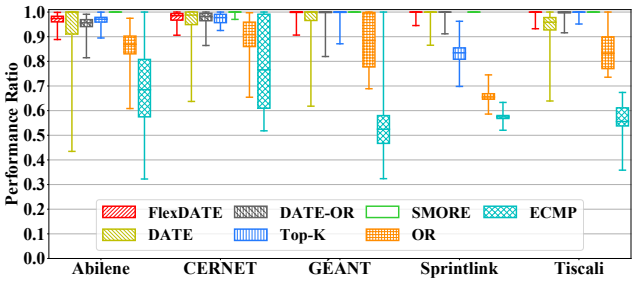
(a) Sprintlink network (K_{ub} = 284). (b) Tiscali network (K_{ub} = 118).

Fig. 10. Number of critical flows K selected by FlexDATE for each test TM with different traffic variations in the Sprintlink and Tiscali networks.

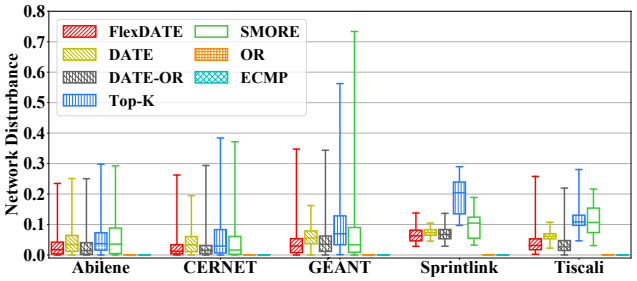
ECMP are also lower in stable traffic scenarios. Since these TE solutions use OR or ECMP as the default routing strategy for non-critical flows, they might need to reroute flows with larger traffic demands to maintain good performance at the cost of incurring higher network disturbance. Another observation is that DATE-OR can achieve comparable performance and disturbance as FlexDATE. As depicted in Fig. 9, DATE-OR achieves similar performance with slightly higher disturbance than FlexDATE in the Sprintlink network. For the Tiscali network, DATE-OR's performance in stable traffic scenarios is worse than FlexDATE, but the corresponding disturbance is also lower than FlexDATE. Since the performance of the original DATE is relatively well in these two networks, it is reasonable for DATE-OR to achieve good performance by employing the robust OR for non-critical flows. Overall, the simulation results show that FlexDATE can generalize to different traffic scenarios with near-optimal performance and low disturbance by selecting and rerouting flexible numbers of critical flows.

D. Resilience

For TE operations, it is important to maintain good network performance in the presence of link failures. When link failure happens, FlexDATE is able to perform TE-level resilience and adapt to different link failure scenarios. This is because the GNN architecture adopted by FlexDATE can effectively model network topology and perform message exchanges between neighboring nodes according to network



(a) Network performance with different single link failures.



(b) Network disturbance with different single link failures.

Fig. 11. Performance ratio and network disturbance comparison in random single link failure scenarios. The upper and lower whiskers are the highest and lowest value achieved on the entire test set, respectively. The box is drawn from the 25th percentile to the 75th percentile value with a horizontal line in the box representing the median value.

connectivity. When there is a link failure, such information can be fed into FlexDATE through the updated network CM. Then, the direct information exchange between the network nodes connected to the failed link would be temporarily disabled in GNN. Therefore, FlexDATE can effectively absorb link failure information when selecting and rerouting a new set of critical flows. To reveal the potential of FlexDATE's GNN architecture in handling link failures, we evaluate the resilient routing provided by FlexDATE under various single link failure scenarios[4]. Note that FlexDATE is trained without any knowledge of link failures. For the Abilene, CERNET, and GÉANT networks, we randomly fail one link every hour, which means that twelve Abilene/CERNET TMs and four GÉANT TMs are evaluated before switching to the next link failure scenario. As for the Sprintlink and Tiscali networks, we randomly fail a unique link every 2 TMs. If some of the flows are affected by link failure (e.g., some preconfigured paths become unavailable), they must be rerouted to prevent traffic loss. In this scenario, the rerouted traffic of these affected flows should not be counted towards network disturbance measurement of FlexDATE and the baseline methods.

Fig. 11 presents the performance ratio and network disturbance of different TE solutions under various single link failure scenarios. We can see that FlexDATE performs the best all the time. As shown in Fig. 11(a), FlexDATE can adapt to different link failure scenarios and achieve above 90% of optimal performance for almost all test TMs in the five networks. The only exceptions are two Abilene TMs with a performance ratio of 88.9% and 89.7%, respectively. In terms of network disturbance, the results in Fig. 11(b) illustrate

[4]For single-degree nodes, failures on their inbound/outbound links are excluded from the link failure scenarios since any failure on these links will result in traffic loss.

that FlexDATE has the lowest median network disturbance among all dynamic TE solutions except DATE-OR in the five networks, which matches the objective of FlexDATE to achieve near-optimal network performance and mitigate network disturbance. Meanwhile, we can find the limitations of the baseline methods through observations. For example, DATE would incur low disturbance in the extreme cases with restrictions on network disturbance, but at the cost of achieving bad performance in some link failure scenarios. Compared to DATE, FlexDATE achieves 7.99%-45.4% worst-case performance improvement in the five networks. Even though DATE-OR's performance is much better than DATE with similar disturbance as FlexDATE, FlexDATE can still provide better worst-case performance than DATE-OR in the five networks with up to 8.7% performance improvement. As for Top-K, it can achieve good performance in most cases by rerouting a proper number of elephant flows. However, Top-K would experience performance degradation in the Sprintlink network and also incur high network disturbance. For SMORE, 73.3% of total traffic could be rerouted with severe service disruption in the GÉANT network to maintain network performance under link failures, while FlexDATE would cause 38.6% less network disturbance in the worst case compared to SMORE. As a result, FlexDATE can utilize the capability of RL and GNN with a reasonable reward function design to achieve good generalization in unseen single link failure scenarios.

E. Execution Time

To catch up with dynamic traffic changes and responsively react to unexpected link failures, it is important for FlexDATE to achieve high efficiency in computation during online deployment. Therefore, we measure the execution time of FlexDATE in the five networks and list the measurement results in Table IV, including the RL inference time for identifying critical flows and the LP solving time for obtaining the optimal path split ratios of the critical flows. Note that all the time measurements are performed on a Linux server with a 4-core Intel 3.4 GHz CPU and 16 GB memory, where Gurobi optimizer v9.1.1 [35] is configured as an LP solver.

From Table IV, we can see that the overall execution time of FlexDATE is relatively low. Considering all traffic scenarios in the five networks, FlexDATE can infer critical flows and solve for optimal routing in less than 500 ms. Thus, FlexDATE is able to catch up with dynamic traffic changes with efficient RL inference and LP solving. There are several reasons for FlexDATE's high efficiency in computation. First, unlike traditional optimal explicit routing that computes the optimal traffic split ratio on each link, we adopt a path-based LP formulation with fewer routing variables and lower computation overhead. Second, FlexDATE's LP only needs to optimize the path split ratios of a set of critical flows instead of all flows, which also reduces the time complexity of LP.

Besides, there are some interesting findings from Table IV. For example, the Tiscali network is slightly larger than the Sprintlink network (see Table II), but the RL inference time and LP execution time in the Tiscali network are smaller than that of the Sprintlink network. To analyze the possible causes, we list the average number of critical flows K for each network and the corresponding upper bound value K_{ub} in Table IV. On the one hand, a larger K_{ub} leads to the higher complexity of the K-net policy network in RL, and thus increases the RL inference time in the Sprintlink network. On the other hand, FlexDATE selects more critical flows in the Sprintlink network, which means there are more routing variables to be solved by LP with higher computation complexity. Thus, the computation overhead of FlexDATE in the Sprintlink network is larger than that of the Tiscali network. From the above analysis, we can see that the execution time of FlexDATE is not only affected by topology size but also the number of critical flows selected, which further demonstrates the advantages of critical flow rerouting in FlexDATE.

TABLE IV
EXECUTION TIME IN DIFFERENT NETWORKS

Topology	Avg. K	K_{ub}	RL Inference (ms)		LP Solving (ms)	
			Avg.	Range	Avg.	Range
Abilene	39.0	40	1.4	1.2-5.1	28.4	19.6-116.0
CERNET	34.8	36	2.1	1.5-4.1	25.8	16.7-124.6
GÉANT	47.2	51	2.6	2.2-4.0	38.4	28.9-121.9
Sprintlink	259.0	284	5.3	4.3-10.3	266.5	258.9-431.7
Tiscali	94.0	118	3.9	3.6-5.6	88.5	70.4-98.5

VIII. RELATED WORKS

A. Traditional TE Solutions

Multiple network techniques are used for traditional TE design. For instance, [1] and [2] use Multi-Protocol Label Switching (MPLS) and route flows by solving an optimization problem to obtain the flows' explicit paths. [3], [49], and [50] leverage Open Shortest Path First (OSPF) and ECMP to dynamically select ECMP paths to balance link utilization. OSPF-OMP [51] exchanges special traffic-load control messages to adaptively allocate traffic among multiple equal-cost paths. Weighted ECMP [4] extends ECMP by splitting traffic based on carefully designed weights at each node to significantly improve performance. Two-phase routing [43], [52] chooses a set of intermediate nodes and tunes the traffic split ratios to the nodes to optimize routing performance.

Another line of work focus on oblivious routing that provides strong performance guarantees for all possible traffic demands. Räcke's oblivious routing [20] constructs diverse forwarding paths based on tree-structured overlays and computes a probability distribution on these paths to forward the traffic without any knowledge of traffic demands. Applegate and Cohen [27] formulate an LP problem for optimal oblivious routing that is optimized with respect to all possible TMs, which can provide promising worst-case performance bounds. However, these methods are still far from optimal.

B. SDN-based TE Solutions

SDN-based TE is widely applied to optimize network performance with a global view of the network. Dynamic hybrid routing [41] relies on a flexible routing policy from SDN and dynamically re-balances traffic to accommodate traffic fluctuations for better load balancing. Agarwal et al. [53] consider a network with partially deployed SDN switches

and improve network utilization by strategically placing SDN switches. Guo et al. [54] design a TE solution named SOTE to achieve load balancing in an SDN/OSPF hybrid network. Xu et al. [55] focus on real-time routing update issues by jointly optimizing route selection and update scheduling to reduce the route update delay. SMORE [21] generates a set of paths using an oblivious routing algorithm [20] and then utilizes LP to dynamically adapt the split ratios of all flows based on these preconfigured paths. Additionally, SDN-based TE solutions have been deployed in the industry to achieve high utilization in inter-data center WANs, including Google's B4 [56] and Microsoft's SWAN [57]. However, none of the above works takes mitigating the impact of flow rerouting into account when designing TE solutions.

C. Machine Learning-based TE Solutions

Machine learning has been used to design different TE solutions. To minimize signaling delay in large SDNs, Lin et al. [58] propose QoS-aware adaptive routing, which employs RL for designing a distributed three-level control plane architecture. Xu et al. [59] use RL to optimize performance metrics (i.e., throughput and delay) in backbone networks. For multi-region networks, MRTE [60] utilizes multi-agent RL to model each network region as individual RL agents, where each region can control terminal and outgoing traffic to make routing decisions in a distributed manner. To reduce routing update overhead, SmartEntry [61] and FlexEntry [62] leverage RL to identify critical entries for destination-based routing updates with considerable entry update savings.

As an efficient graph representation learning framework, GNN is recently applied in TE solutions to model network topology with good generalization. For distributed TE in multi-region networks, FedTe [24] develops a two-layer GNN architecture in accordance with different levels of network abstraction and uses Supervised Learning (SL) to predict the optimal cross-region traffic distribution for local routing optimization. Bernárdez et al. [25] combine multi-agent RL and GNN to perform message exchanges between neighboring links, which results in effective link weight settings for OSPF to reduce network congestion. However, these existing works do not consider the impact of flow rerouting on existing services or take mitigating network disturbance as an objective.

IX. 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 a flexible and disturbance-aware TE solution named FlexDATE, which leverages RL and GNN to learn a flexible critical flow selection policy that can accommodate dynamic network traffic and different link failure scenarios. For each given TM, FlexDATE smartly selects a small set of critical flows with RL and reroutes them with LP to balance link utilization of the network while mitigating network disturbance. Extensive evaluations show that FlexDATE is able to achieve near-optimal load balancing performance and effectively mitigate network disturbance in dynamic traffic conditions as well as single link failure scenarios.

References

[1] Y. Wang and Z. Wang, "Explicit routing algorithms for internet traffic engineering," in *IEEE ICCCN*, 1999, pp. 582–588.

[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*, vol. 17, no. 3, pp. 778–788, 2009.

[4] J. Zhang, K. Xi, L. Zhang, and H. J. Chao, "Optimizing network performance using weighted multipath routing," in *IEEE ICCCN*, 2012, pp. 1–7.

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

[6] R. Sommer and A. Feldmann, "Netflow: Information loss or win?" in *ACM SIGCOMM IMW*, 2002, pp. 173–174.

[7] "Tiktok," 2021. [Online]. Available: https://www.tiktok.com/

[8] "Kuaishou," 2021. [Online]. Available: https://www.kuaishou.com/en

[9] "Vimeo," 2021. [Online]. Available: https://www.vimeo.com/

[10] "Google lens," 2021. [Online]. Available: https://lens.google.com/

[11] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp. 87–95, 2014.

[12] "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/

[13] Y. Li *et al.*, "Hpcc: High precision congestion control," in *ACM SIGCOMM*, 2019, pp. 44–58.

[14] C. Kim *et al.*, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, vol. 15, 2015.

[15] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[16] N. Van Tu, J. Hyun, and J. W.-K. Hong, "Towards onos-based sdn monitoring using in-band network telemetry," in *APNOMS*, 2017, pp. 76–81.

[17] J. Hyun, N. Van Tu, and J. W.-K. Hong, "Towards knowledge-defined networking using in-band network telemetry," in *IEEE/IFIP NOMS*, 2018, pp. 1–7.

[18] 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*, vol. 50, no. 2, pp. 11–23, 2020.

[19] R. Carpa, M. D. de AssunçÃo, O. Glück, L. LefÈvre, and J.-C. Mignot, "Evaluating the impact of sdn-induced frequent route changes on tcp flows," in *IEEE CNSM*, 2017, pp. 1–9.

[20] H. Räcke, "Optimal hierarchical decompositions for congestion minimization in networks," in *ACM STOC*, 2008, p. 255–264.

[21] P. Kumar *et al.*, "Semi-oblivious traffic engineering: The road not taken," in *USENIX NSDI*, 2018, pp. 157–170.

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

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

[24] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Federated traffic engineering with supervised learning in multi-region networks," in *IEEE ICNP*, 2021, pp. 1–12.

[25] G. Bernárdez *et al.*, "Is machine learning ready for traffic engineering optimization?" in *IEEE ICNP*, 2021, pp. 1–11.

[26] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Date: Disturbance-aware traffic engineering with reinforcement learning in software-defined networks," in *IEEE/ACM IWQoS*, 2021, pp. 1–10.

[27] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *ACM SIGCOMM*, 2003, p. 313–324.

[28] J. Zhang, M. Ye, Z. Guo, C. Y. Yen, and H. J. Chao, "Cfr-rl: Traffic engineering with reinforcement learning in sdn," *IEEE JSAC*, vol. 38, no. 10, pp. 2249–2259, 2020.

[29] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *ACM HotNets*, 2016, pp. 50–56.

[30] W. Kool *et al.*, "Attention, learn to solve routing problems!" in *ICLR*, 2018.

[31] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.

[32] A. Vaswani *et al.*, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

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

[35] Gurobi Optimization LLC, "Gurobi optimizer reference manual," 2021. [Online]. Available: https://www.gurobi.com/

[36] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM CCR*, vol. 32, no. 4, pp. 133–145, 2002.

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

[38] B. Zhang, J. Bi, J. Wu, and F. Baker, "Cte: Cost-effective intra-domain traffic engineering," *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 115–116, 2014.

[39] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM CCR*, vol. 36, no. 1, pp. 83–86, 2006.

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

[41] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Dynamic hybrid routing: Achieve load balancing for changing traffic demands," in *IEEE/ACM IWQoS*, 2014, pp. 105–110.

[42] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Load balancing for multiple traffic matrices using sdn hybrid routing," in *IEEE HPSR*, 2014, pp. 44–49.

[43] M. Kodialam, T. Lakshman, J. B. Orlin, and S. Sengupta, "Oblivious routing of highly variable traffic in service overlays and ip backbones," *IEEE/ACM ToN*, vol. 17, no. 2, pp. 459–472, 2008.

[44] TMgen: Traffic Matrix Generation Tool. [Online]. Available: https://tmgen.readthedocs.io/en/latest/

[45] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," in *ACM SIGCOMM CCR*, vol. 45, no. 4, 2015, pp. 579–592.

[46] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *USENIX OSDI*, 2016, pp. 265–283.

[47] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[48] D. Thaler and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," *IETF RFC 2991*, November 2000.

[49] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE JSAC*, vol. 20, no. 4, pp. 756–767, 2002.

[50] K. Holmberg and D. Yuan, "Optimization of internet protocol network design and routing," *Networks: An International Journal*, vol. 43, no. 1, pp. 39–53, 2004.

[51] C. Villamizar, "Ospf optimized multipath (ospf-omp)," *IETF Internet-Draft, draft-ietf-ospf-omp-03.txt*, 1999. [Online]. Available: https://ci.nii.ac.jp/naid/10026755527/en/

[52] M. Antic, N. Maksic, P. Knezevic, and A. Smiljanic, "Two phase load balanced routing using ospf," *IEEE JSAC*, vol. 28, no. 1, pp. 51–59, 2009.

[53] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *IEEE INFOCOM*, 2013, pp. 2211–2219.

[54] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in sdn/ospf hybrid network," in *IEEE ICNP*, 2014, pp. 563–568.

[55] H. Xu, Z. Yu, X.-Y. Li, C. Qian, L. Huang, and T. Jung, "Real-time update with joint optimization of route selection and update scheduling for sdns," in *IEEE ICNP*, 2016, pp. 1–10.

[56] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 3–14, 2013.

[57] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven wan," in *ACM SIGCOMM*, 2013, pp. 15–26.

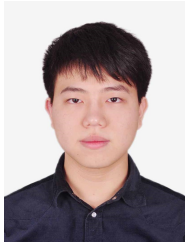
[58] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *IEEE SCC*, 2016, pp. 25–33.

[59] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM*, 2018, pp. 1871–1879.

[60] N. Geng, T. Lan, V. Aggarwal, Y. Yang, and M. Xu, "A multi-agent reinforcement learning perspective on distributed traffic engineering," in *IEEE ICNP*, 2020, pp. 1–11.

[61] J. Zhang, Z. Guo, M. Ye, and H. J. Chao, "Smartentry: Mitigating routing update overhead with reinforcement learning for traffic engineering," in *ACM SIGCOMM NetAI*, 2020, pp. 1–7.

[62] M. Ye, Y. Hu, J. Zhang, Z. Guo, and H. J. Chao, "Mitigating routing update overhead for traffic engineering by combining destination-based routing with reinforcement learning," *IEEE JSAC*, vol. 40, no. 9, pp. 2662–2677, 2022.



Minghao Ye (Graduate Student Member, IEEE) received the B.Eng. degree in microelectronic science and engineering from Sun Yat-sen University, Guangzhou, China, the B.Eng. degree (Hons.) in electronic engineering from The Hong Kong Polytechnic University, Hong Kong, in 2017, and the M.S. degree in electrical engineering from New York University, New York, NY, USA, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include traffic engineering, network optimization, software-defined networking, and machine learning for networking.

Junjie Zhang (Member, IEEE) received the B.S. degree in computer science from Nanjing University of Posts & Telecommunications, China, in 2006, the M.S. degree in computer science and the Ph.D. degree in electrical engineering from New York University, New York, NY, USA, in 2010 and 2015, respectively.

He has been with Fortinet, Inc., Sunnyvale, CA, USA, since 2015. He holds two US patents in the area of computer networking. His research interests include network optimization, traffic engineering, machine learning, and network security.

Zehua Guo (Senior Member, IEEE) received B.S. degree from Northwestern Polytechnical University, Xi'an, China, M.S. degree from Xidian University, Xi'an, China, and Ph.D. degree from Northwestern Polytechnical University. He was a Research Fellow at the Department of Electrical and Computer Engineering, New York University Tandon School of Engineering, New York, NY, USA, and a Research Associate at the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA. His research interests include programmable networks (e.g., software-defined networking, network function virtualization), machine learning, and network security. Dr. Guo is an Associate Editor for IEEE Systems Journal and the EURASIP Journal on Wireless Communications and Networking (Springer), and an Editor for the KSII Transactions on Internet and Information Systems. He is serving as the TPC of several journals and conferences (e.g., Elsevier Computer Communications, AAAI, IWQoS, ICC, ICCCN, ICA3PP). He is a Senior Member of IEEE, China Computer Federation, China Institute of Communications, and Chinese Institute of Electronics, and a Member of ACM.

H. Jonathan Chao (Life Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering from National Chiao Tung University, Taiwan, in 1977 and 1980, respectively, and the Ph.D. degree in electrical engineering from The Ohio State University, Columbus, OH, USA, in 1985. He was the Head of the Electrical and Computer Engineering (ECE) Department at New York University (NYU) from 2004 to 2014. He has been doing research in the areas of machine learning for networking, real-time communications, datacenter networks, high-speed packet scheduling/switching/routing, software-defined networking, network function virtualization, and network security. During 2000-2001, he was the Co-Founder and a CTO of Coree Networks, Tinton Falls, NJ, USA. From 1985 to 1992, he was a Member of Technical Staff at Bellcore, Piscataway, NJ, USA, where he was involved in transport and switching system architecture designs and application-specified integrated circuit implementations, such as the world's first SONET-like framer chip, ATM layer chip, sequencer chip (the first chip handling packet scheduling), and ATM switch chip. He is currently a Professor of ECE at NYU, New York City, NY, USA. He is also the Director of the High-Speed Networking Lab. He has co-authored three networking books, *Broadband Packet Switching Technologies-A Practical Guide to ATM Switches and IP Routers* (New York: Wiley, 2001), *Quality of Service Control in High-Speed Networks* (New York: Wiley, 2001), and *High-Performance Switches and Routers* (New York: Wiley, 2007). He holds 63 patents and has published more than 280 journal and conference papers. He is a fellow of the IEEE and the National Academy of Inventors. He was a recipient of the Bellcore Excellence Award in 1987. He was a co-recipient of the 2001 Best Paper Award from the IEEE TRANSACTION ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.