Federated Traffic Engineering with Supervised Learning in Multi-region Networks

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

***Abstract*—Network operators usually adopt Traffic Engineering (TE) to configure the routing in their networks to achieve good load balancing performance and high resource utilization. While centralized TE can effectively improve network performance with a global view of the network, distributed TE has been considered as an alternative to manage large-scale networks that are usually partitioned into multiple regions. However, it is challenging for distributed TE to reach a global optimal performance since each region can make its local routing decisions only based on partially observed network states. In this paper, we propose a novel distributed TE scheme called *FedTe*, which leverages supervised learning coupled with a collaborative approach to improve the overall load balancing performance for multi-region networks. FedTe learns from the global optimal routing strategy in a centralized offline manner and predicts the optimal distribution of cross-region traffic among different regions through distributed deployment in real time. The predicted cross-region traffic distribution is integrated with measured local traffic to construct each region's *optimal regional traffic matrix*, which is used to perform intra-region TE optimization. FedTe can also handle dynamic traffic variation and link failures with a 2-layer hierarchical graph neural network architecture. To validate the effectiveness of the proposed scheme, we evaluate FedTe with two real-world network topologies and a large-scale synthetic topology. Extensive evaluation results show that FedTe can achieve near-optimal load balancing performance and outperform state-of-the-art distributed TE approaches by up to 28.9% on average.**

I. Introduction

Traffic Engineering (TE) is an important network operation to optimize network performance and resource utilization by controlling traffic distribution and configuring traffic routing in a network [1]. Network operators aim to optimize the routing in their networks to provide good end-to-end performance for users while achieving efficient resource usage. Therefore, TE is widely applied to improve network performance with an optimization objective, such as minimizing the Maximum Link Utilization (MLU) in the network to reduce network congestion and achieve load balancing. Due to dynamic traffic fluctuations in the network, it is desired to collect network states and update routing periodically to maintain good network performance. Based on the network topology and each measured Traffic Matrix (TM) representing the traffic demands of all flows[1], TE usually formulates and solves a routing optimization problem to optimally redistribute the traffic across the network accordingly.

Generally speaking, existing TE approaches can be divided into two categories: centralized TE and distributed TE. In a fully-controlled and fully-observable network, centralized TE is widely adopted to improve network performance [2]–[9]. Specifically, a centralized controller is deployed in the network to collect network topology and TM periodically. A TE module in the controller is responsible for computing routing paths and redistributing traffic for flows based on observed network states. By leveraging existing Multi-Protocol Label Switching (MPLS) technique [10], a network operator can set up tunnels (i.e., Label Switched Paths (LSPs)) between different network nodes and adjust the traffic split ratios among the tunnels to accommodate the traffic changes [2]–[4]. With emerging Software-Defined Networking (SDN) [11], centralized TE can be deployed in a responsive way [12]–[14], where an SDN controller can redistribute traffic by installing/updating flow table entries at the underlying SDN switches. The SDN-based centralized TE has been deployed in today's Wide Area Networks (WANs) as industrial solutions (e.g., Google's B4 [3] and Microsoft's SWAN [5]).

However, centralized TE suffers from high overhead and long reaction time to redistribute traffic of flows. Typically, centralized TE operates at minute-level intervals to accommodate traffic changes [3], [5]. As the network size grows, the computation complexity of TE algorithms significantly increases since the number of control variables in the routing optimization problems increases dramatically [15]. Upon traffic changes, the centralized controller must update all the underlying switches in real time, which may incur high routing update overhead for large-scale networks [16]. When link failure happens, centralized TE can recompute the optimal routing to bypass the failed links. However, in a geographically distributed network (e.g., Google Cloud [17]), notifying the centralized controller of topology changes and waiting for routing updates is very time-consuming and may result in temporary performance degradation and potential traffic loss [18], [19].

To achieve scalable and efficient network management, a large network is usually divided into multiple geographical regions, and each of the regions is controlled by a regional controller [19]–[21]. In this way, distributed TE can be applied to reduce computation complexity and routing update

The corresponding author is Zehua Guo.
[1]In this paper, a flow is defined as a source-destination pair.

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

overhead, and to handle link failures promptly and efficiently. For each network region, local routing decisions are made by a regional controller to distribute traffic inside the region based on measured network statistics. When link failure occurs, local recovery mechanisms can be performed to reroute some flows in a responsive manner to relieve network congestion and mitigate traffic loss. However, it is very challenging for distributed TE to achieve a global optimal performance with partially-observed network states. The existing heuristic method like hot-potato routing [22] is far from global optima since it only statically directs outgoing traffic to the closest border router without considering other regions' preferences. Several distributed TE approaches [23]–[25] are proposed to facilitate cooperation and collaboration among different regions to find a better local routing strategy. However, they could introduce high communication overhead to exchange information for multiple iterations during online deployment and thus lead to sub-optimal performance under dynamic traffic demands [26].

To improve the performance of multi-region networks in a distributed fashion with low communication overhead, it is important for each region to obtain traffic and routing information from other regions through efficient interactions. This is because local routing decisions for cross-region traffic would affect other regions' network performance as well. For example, directing the outgoing traffic to neighboring regions through different border routers would result in a change of adjacent regions' traffic volumes and potentially lead to unexpected congestion. Therefore, one of the promising approaches is to aggregate the knowledge of different regions and collaboratively work out a good strategy for each individual region to allocate cross-region traffic at its border routers. According to the strategy, each region knows how to direct outgoing traffic through different egress border routers and how to distribute incoming traffic at different ingress border routers. The desired cross-region traffic distributions and local traffic inside a region can be integrated into an *optimal regional TM*, which represents the regional traffic volumes under the collaborative strategy and is then used for intra-region TE to optimize local routing decisions. However, the main challenge is how to efficiently obtain a good strategy for each region through a collaborative approach. Intuitively, the best strategies should be derived from the global optimal routing solutions to achieve global TE objectives. However, given the dynamic TM and network conditions, it would be very difficult, if not impossible, to obtain the global optimal routing strategies without a global view of a network.

To address the aforementioned challenges, we propose a Supervised Learning (SL)-based framework called *FedTe* (Federated Traffic Engineering) for distributed TE in multi-region networks, which enables each network region to optimize local routing decisions and achieve near-optimal performance with low computation/communication overhead. FedTe learns from a global optimal routing strategy as the ground truth through centralized offline training, and then deploys the trained model in each regional controller to predict the optimal distribution of cross-region traffic in a distributed manner. After integrating the prediction results and measured local traffic into an optimal regional TM, each region can solve and obtain the optimal local routing strategy independently using Linear Programming (LP) to improve the overall network performance. To effectively handle traffic dynamics and link failures, we leverage the graph representation learning techniques from Graph Neural Network (GNN) [27], [28]. Since network topologies are naturally represented as graphs, GNN has unique advantages over traditional neural network architectures to model network topologies and imitate the information exchange process among multiple network nodes/regions. To further facilitate training and minimize inter-region communication overhead, we design a 2-layer GNN architecture in accordance with different levels of network abstractions. Specifically, the first layer is responsible for modeling the traffic and topology information inside each network region, while the second layer summarizes the regional information and exchanges GNN encoded messages among multiple regions to share encrypted regional information with privacy protection in a collaborative manner. Such light-weight messages can be fit into a single packet for transmission, and the message exchange process is only performed once when routing updates are required due to traffic changes or link failures. To the best of our knowledge, this is the first paper that adopts SL for distributed TE optimization problem.

The main contributions of this paper are summarized as follows:

- We propose an SL-based TE that predicts the optimal distribution of cross-region traffic for each network region to perform local TE optimization in a distributed manner, while the offline centralized learning is guided by the global optimal routing solutions.
- We design a 2-layer scalable GNN-based neural network architecture to handle dynamic traffic variations and link failures, which can greatly simplify the prediction model complexity and accelerate training/inference processes.
- We evaluate and compare FedTe to the existing distributed TE methods by conducting extensive experiments on different network topologies, where FedTe can outperform the state-of-the-art distributed TE methods by up to 28.9% in terms of average load balancing performance.

The rest of the paper is organized as follows. Section II lists the related works. Section III provides an overview of FedTe's design. Section IV explains how to predict the optimal distribution of cross-region traffic using FedTe's hierarchical GNN architecture. Section V describes the implementation details of FedTe. Section VI evaluates the performance of FedTe. Section VII concludes this paper.

II. Related Works

Existing TE solutions can be generally categorized into two classes: centralized TE and distributed TE.

A. Centralized TE Solutions

Centralized TE usually routes/reroutes flows periodically to balance the load on links by formulating an optimization

problem and solving the problem with optimization solvers or heuristic solutions. To achieve robust network performance, SMORE [4] uses a set of paths computed by an oblivious routing algorithm [29] and dynamically adjusts path split ratios for all flows with a centralized controller. Many centralized TE solutions have also been deployed in the industry, such as B4 for Google [3] and SWAN for Microsoft [5], to achieve high utilization in inter-Data Center (DC) WANs. Recently, some of the academic works leverage the emerging Machine Learning (ML) techniques to improve the performance of centralized TE. Valadarsky et al. [6] design an ML-based solution to generate good routing configurations with consideration of future traffic demands. DRL-TE [7] develops an experience-driven approach based on Reinforcement Learning (RL) to enable model-free routing control. CFR-RL [8] uses RL to select critical flows and reroutes these critical flows to balance link utilization with mitigated network disturbance.

However, these solutions either suffer from the scalability issue or are not designed for multi-region TE. SMORE [4] can update routing responsively with a fixed set of preconfigured paths, but it may still suffer from the scalability issue due to the increasing complexity of solving the optimization problem for large-scale networks, as shown in Section VI-C. For today's industrial solutions [3], [5], they are originally designed for cloud providers to optimize inter-DC traffic allocation in an abstracted site-level topology with tens of sites. However, since the inter-region/intra-region traffic allocations are separately optimized, these TE methods may lead to sub-optimal network performance. For the learning-based approaches, DRL-TE [7] only works for 20 flows with pre-computed paths, while CFR-RL [8] is designed for a fully-observable network with global control from a centralized SDN controller.

B. Distributed TE Solutions

Some large networks are partitioned into multiple regions in accordance with geographic locations, and the control scalability issue becomes critical for these large-scale multi-region networks since the centralized control introduces considerable computation and communication overhead. To address the above issue, distributed TE is introduced as an alternative solution. However, most existing distributed TE solutions are heuristic and unable to adapt to the changes of TMs and network dynamics. For example, hot-potato routing [22] allows one network region to send its outgoing traffic to other regions via the nearest border router, but it may introduce unexpected congestion since the preferences of other regions are not considered. In recent years, some distributed TE solutions based on distributed optimization methods are proposed. They aim at exchanging necessary information among regions to improve overall performance collaboratively [23]. Duchi et al. [24] develop distributed algorithms based on dual subgradient averaging and provide sharp bounds on their convergence rates. Srivastava et al. [25] provide a distributed stochastic gradient algorithm for agents to compute an optimal decision variable that minimizes the worst-case loss incurred by any agent in the network. However, these approaches usually require multiple iterations of message exchanges during online deployment, which incurs high communication overhead. On the contrary, FedTe only performs a light-weight information exchange once when a routing update is needed.

To achieve efficient collaborations among multiple network regions, several data-driven frameworks are proposed for distributed TE. Geng et al. [26] adopt distributed RL agents to get the local link utilization statistics within their regions, where each agent learns a good local routing strategy over a set of preconfigured paths by interacting with other agents and exchanging congestion-related reward signals. Pinyoanuntapong et al. [30] propose a distributed TE framework that adopts multi-agent RL for distributed control to cope with network dynamics. However, they need to take considerable training time to learn and converge to a good policy, since RL agents have to interact with the network environment to obtain action feedbacks and explore the large action space through trail-and-error. In contrast, guided by the feasible global optimal routing solutions, FedTe can be efficiently trained using SL to handle various traffic and link failure scenarios and generalize well to unseen traffic scenarios, as later shown in Section VI.

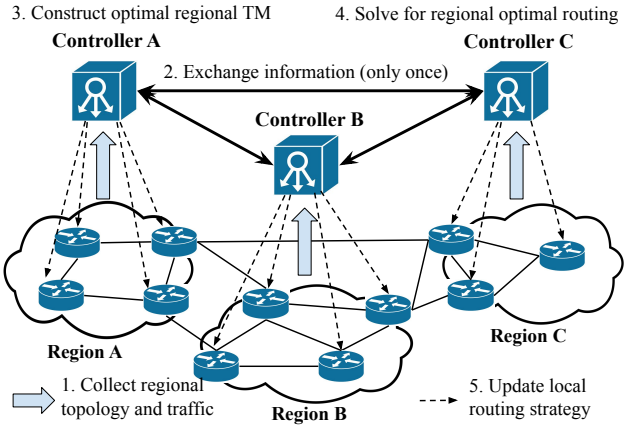
III. System Design

In this section, we introduce the system design of FedTe, which is an SL-based TE solution for distributed TE in multi-region networks. Broadly speaking, FedTe is deployed in the controller of each region. With very limited message exchange among regional controllers, FedTe predicts optimal cross-region traffic distribution for each individual region. Cross-region traffic distribution can serve as an instruction on how to optimally direct outgoing traffic and estimate incoming traffic distribution at border routers. The above setting is based on the assumption that all regions will cooperate together to achieve the global optimal network performance. For a large-scale network or multiple inter-connected Autonomous Systems (ASes) managed by a single network operator, the assumption holds since network operators have full control of the entire network and they are willing to optimize their network performance. Therefore, under the guidance of cross-region traffic distribution from FedTe, each network region can derive the corresponding optimal regional TM and optimize the local routing decision accordingly to achieve global optimal load balancing performance with low computation and communication overhead. Figure 1(a) shows the routing update workflow of FedTe. When a routing update is required, each regional controller running FedTe collects local network states and exchanges a light-weight GNN encoded message with other regional controllers only once to synchronize necessary information. Then, FedTe constructs an optimal regional TM based on predicted cross-region traffic distribution and optimizes the regional routing decision, which would be directly applied in the local network region. It is worth noting that the above routing update procedures can be repeated periodically (e.g., every 5 minutes) or upon topology/traffic changes to improve network performance, and the local region routing solutions can be either flow-based or destination-based. They

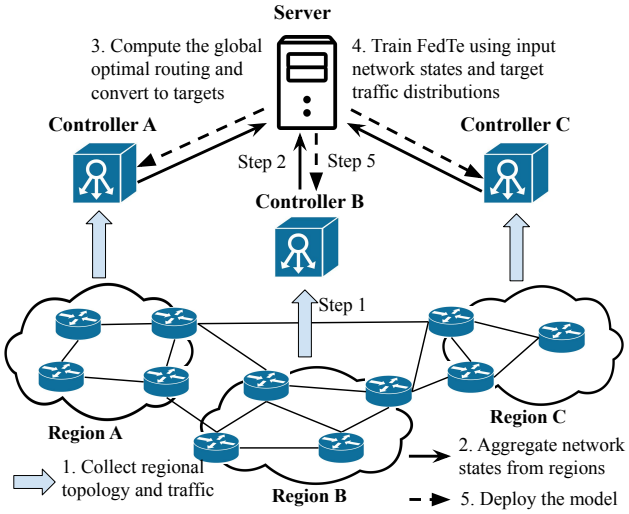
can be easily obtained by formulating and solving a routing optimization problem [1] using LP.

FedTe is trained through a centralized offline training procedure as shown in Fig. 1(b). At first, regional network states are periodically collected and aggregated to a server to compute the global optimal routing solutions, which are converted to the corresponding cross-region traffic distributions. During the training phase, these cross-region traffic distributions are used as the training targets to guide the learning of FedTe. Given the input network states and training targets, the offline training can be performed in a server without the intervention of the real network environment. All the network components are constructed virtually using GNN and information exchange process is imitated by leveraging the graph representation learning techniques of GNNs. In particular, the multi-region network is modeled as a 2-layer hierarchical GNN architecture (details in Section IV). The first layer is responsible for modeling and encoding the node-level topology and traffic information within each network region. Then, the complete information of each network region is summarized and encoded in the second layer and exchanges with other network regions. Updating with the encoded information/messages from other regions, each network region obtains a global view of the network. At the final step, the predicted distributions of incoming and outgoing traffic are decoded from the encoded and updated information of each region, which are compared against the training targets to compute the loss and update GNN parameters (see Section V-B). Although it seems contradictory for a distributed TE scheme to perform centralized offline training, it is important to ensure the optimality of the training targets such that FedTe can learn from the best routing strategies. Moreover, once training is done, the identical FedTe model would be deployed in each regional controller of the real network environment. Then, multiple FedTe models perform in a distributed and collaborative manner to quickly obtain efficient local routing decisions towards global optimal performance.

To generate desired cross-region traffic distribution training targets, we leverage a set of preconfigured paths computed by an oblivious routing algorithm [29] with some customizations and solve a modified Multi-Commodity Flow (MCF) problem with a path budget $K = 4$ as reported in SMORE [4]. To avoid traffic looping among multiple regions, we delete some *invalid paths* from the set of preconfigured paths in advance. Take Figure 2(a) as an example, node 3 wants to send some traffic to node 10. Based on the existing preconfigured paths of flow $\langle 3, 10 \rangle$, some of the traffic are directed to border node 5 according to the red solid path. Once traffic is forwarded to region C1, node 5 needs to forward the traffic to node 10 under controller C1's control. However, if region C1 adopts destination-based routing (i.e., does not distinguish source addresses), it is possible that node 5 will send some traffic back to node 3 according to the preconfigured path from node 5 to node 10 (blue dash-dotted path). As a result, there is a forwarding loop between node 3 and node 5, which may cause severe network problems and resource waste. To avoid



(a) Workflow of FedTe's routing update after distributed deployment.

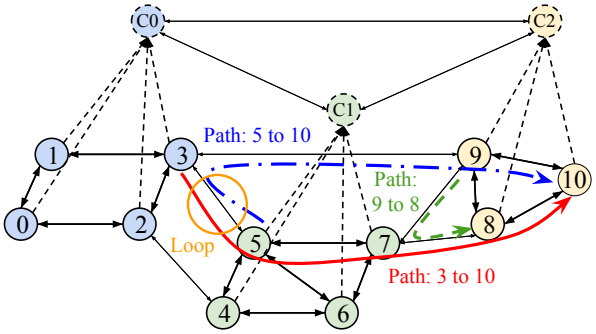


(b) Procedure of centralized offline training.

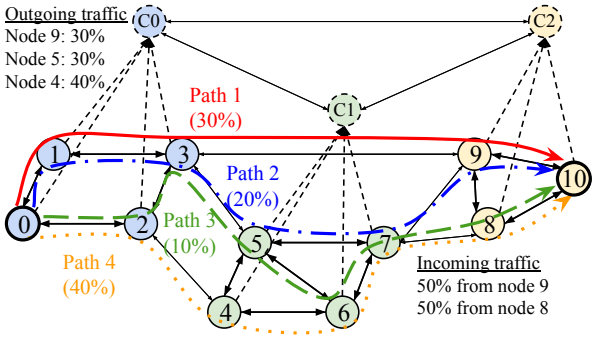
Fig. 1. Overview of FedTe's system design.

the conflict, one of these two paths should be regarded as invalid path and removed from the path set. Another case of invalid path is the green dashed path from node 9 to node 8. Since there is no need to direct local traffic to other regions, such path should be deleted and only the intra-region paths are allowed for local traffic. Therefore, it is essential to remove these invalid paths before solving the modified MCF problem. In addition, the number of invalid paths can be taken as a metric to evaluate whether the network region partition is good or not. For instance, a bad regional partitioning strategy may result in the removal of a large number of invalid paths, such that TE performance would be degraded with limited path sets. For simplicity, we divide network regions based on the geographical distribution of network nodes, which is an effective strategy since we only need to remove a few invalid paths without performance loss.

Once the global optimal routing solution is obtained, it should be converted to the cross-region traffic distribution that serves as the guidance for training FedTe. Recall that the optimal regional TM combines the local traffic and the optimal distribution of incoming and outgoing traffic. Given the global optimal routing strategy, we can extract useful information



(a) An example of invalid paths.



(b) Convert global optimal routing solution to training targets.

Fig. 2. An illustrative example of multi-region networks.

from each region's perspective: (1) How to properly distribute outgoing traffic to available egress border routers with the consideration of global optimality; (2) What are the distributions of incoming traffic at the ingress border routers when all the regions are following global optimal routing policy. To explain the above ideas, an example is illustrated in Fig. 2(b). According to a global optimal routing solution, flow $\langle 0, 10\rangle$ is distributed among 4 valid paths with different split ratios. From the perspective of region C0, the outgoing traffic from node 0 to node 10 should be forwarded to node 4, node 5 and node 9 with split ratios 40%, 30%, and 30%, respectively. From the perspective of region C2, traffic is injected from node 8 and node 9 and destined to node 10. Assume that node 0 wants to send 10 units of traffic to node 10, which means there are 5 units of incoming traffic from node 8 to node 10 and another 5 units of traffic destined to node 10 would be coming from node 9. For the transit traffic traversing region C1, it would be considered as two outgoing traffic flows. One is originated from node 4, the other is originated from node 5, and they are both destined to node 10. For flow $\langle 4, 10\rangle$, all the outgoing traffic would be forwarded to node 8. In contrast, the traffic of flow $\langle 5, 10\rangle$ would be forwarded to node 8 and node 9 with split ratios 33.3% and 66.6%, respectively. The above incoming and outgoing traffic distribution for each region would be used as training targets for FedTe.

There are two reasons why we adopt GNN to predict cross-region traffic distributions and then use LP to optimize local routing. On the one hand, to achieve a global optimization objective, both intra-region and inter-region routing decisions should be optimized simultaneously. A hierarchical TE approach, which finds the optimal inter-region traffic allocation for the abstracted region-level network and then optimizes local routing in each region separately, may lead to sub-optimal performance due to the limited collaboration between inter-region and intra-region routing decisions. For example, the inter-region traffic distribution may overwhelm some network regions with huge traffic loads, even though such distribution can effectively balance the load of inter-region links in the region-level network. Instead, we use GNN to directly learn from the global optimal routing strategy and predict the optimal cross-region traffic allocation with consideration of both intra-region and inter-region routing performance. On the other hand, LP is an efficient and optimal approach to solve for local routing strategy in small-scale network region once the optimal regional TM is obtained. There is no need to further adopt GNN for intra-region routing optimization with the concern about model complexity.

IV. Proposed Model

To model network topologies and message exchange procedures, we leverage graph representation learning techniques and message passing frameworks [27], [28] offered by GNNs. FedTe consists of an intra-region encoder that performs intra-region message exchange, an inter-region encoder performs inter-region message exchange, and a decoder interprets desired cross-region traffic distribution from updated and encoded node features.

A. Intra-Region Encoder

For each region, an intra-region encoder is applied to model and encode the regional topology and traffic information. The inputs to the intra-region encoder are node features and a node connectivity matrix, where the features for a given network node are a series of demands originated from that node, and the connectivity matrix indicates the neighbors of each node. The regional controller is represented as a virtual node with node features initialized as all ones, and all region nodes are assumed to be directly connected to this virtual node. Figure 3(a) shows a multi-region topology example, and the example of an intra-region encoder applied in one region is shown in Fig. 3(b). The encoder computes an initial node embedding using a shared feed-forward layer, and then each node's embedding is updated by exchanging messages with its neighbors. Similar to the transformer model presented in [31], [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 layer that performs message exchanging between neighboring nodes, and the second is a node-wise fully connected feed-forward layer that performs a nonlinear transformation. In addition, a skip connection [33] and layer normalization [34] are applied to each sub-layer. Let h_v denote the node embedding for a given node v, it is updated iteratively by aggregating the messages passed from its neighbors:

$$h_v^{l+1} = \sum_{w \in \chi_v} M(h_v^l, h_w^l, \theta_M^l), \tag{1}$$

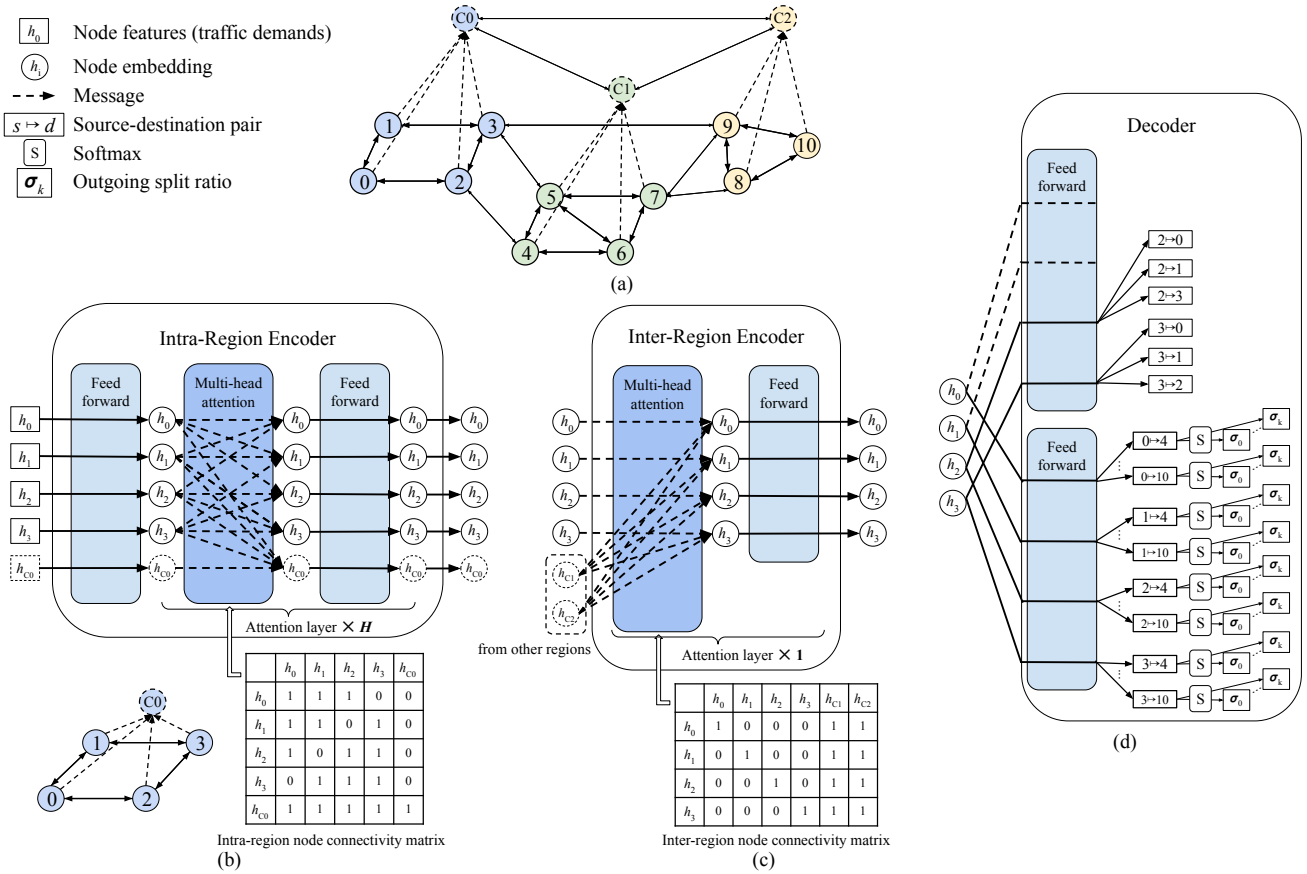


Fig. 3. The hierarchical graph neural network model for FedTe. (a) A multi-region topology example, where C0, C1 and C2 are virtual nodes that represent the network controller for each region. (b) An intra-region encoder computes the initial node embeddings using a shared fully connected Feed-Forward (FF) layer with each corresponding node's features. Then, per-node embeddings are updated using H attention layers according to the connectivity matrix of each given regional topology. Each attention layer consists of a Multi-Head Attention (MHA) layer and a node-wise fully connected FF layer. (c) An inter-region encoder updates per-node embeddings with the virtual node embeddings of controllers in other regions. Note that there is only one attention layer in the inter-region encoder. In other words, each region only exchanges messages once. (d) A decoder interprets each node's embedding as incoming traffic and outgoing split ratios for each corresponding source-destination pair. Note that the traffic incoming from other regions must traverse border nodes. Thus, the incoming traffic is only interpreted from border nodes' embeddings by the decoder with a node-wise fully connected FF layer. Similarly, a node-wise fully connected FF layer and a pair-wise softmax layer are used to generate outgoing split ratios for each corresponding pair.

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

Employing H attention layers can be interpreted as executing H iterations of embedding update process, and one iteration of embedding update process can be essentially considered as a feature propagation. After H iterations, each node's embedding would include H hops away neighbors' information. Thus, when H equals to the number of max hops between any two nodes in the network, it would be enough for each node to capture the complete information of the whole network region. Once the intra-region message exchange phase is done, the updated virtual node's embedding is then used to represent the regional graph embedding and exchanged with other regions.

B. Inter-Region Encoder

To learn the traffic and topology information of other regions, each node's embedding would be updated with the graph embeddings (i.e., virtual node embeddings) from other regions during the inter-region message exchange phase. To limit the communication overhead among regional controllers in the real deployment, each region would only exchange message once for each routing update. Thus, as shown in Fig. 3(c), the inter-region encoder consists of a single attention layer and performs embedding update only once.

C. Decoder

Once the message exchange phases are done, each node's embedding includes the information of the whole network. As shown in Fig. 3(d), a decoder consists of two readout functions R_i and R_o, which are used to decode the corresponding incoming and outgoing traffic distribution from these node embeddings.

R_i interprets the updated per-node embedding as corresponding incoming traffic distribution, i.e.,

$$\{t_i^{v,d} | d \in V^n\} = R_i(h_v, \theta_{R_i}), v \in B_i^n, \tag{2}$$

where $t_i^{v,d}$ is the incoming traffic from node v to node d, B_i^n is the incoming border node set of region n, V^n is the node set of region n, θ_{R_i} is the learnable readout function parameters. Note that the traffic incoming from other regions

must traverse border nodes. Thus, the incoming traffic is only interpreted from border nodes' embeddings. In addition, the incoming traffic $t^{v,v}$ would not affect the regional TM. Thus, we do not interpret it from the node embedding.

R_o interprets the per-node embedding as corresponding outgoing traffic distribution, i.e.,

$$\{\sigma_k^{v,d} | d \in \bar{V}^n, k \in B_o^{v,d}\} = R_o(h_v, \theta_{R_o}), v \in V^n, \quad (3)$$

where $\sigma_k^{v,d}$ is the outgoing split ratio for traffic from node v to node d, $B_o^{v,d}$ is the outgoing border node set of pair $\langle v, d \rangle$, $\bar{V}^n$ is the set of nodes in other regions.

It is worth noting that the encoders and decoder are shared and reused by each node, which greatly simplifies model complexity and substantially reduces training and inference time (shown in Section VI-C). It also allows the distributed deployment of FedTe.

V. Implementation

In this section, we describe the implementation details of FedTe and our experiment setup.

A. Dataset

TABLE I
Network Topologies Used in Evaluation

Topology	Nodes	Directed Links	Regions
Telstra (Australia)	38	152	5
Google Cloud	44	160	3
BRITE	204	964	16

To evaluate the proposed FedTe scheme, we partition two real-world network topologies and a large-scale synthetic topology into different numbers of network regions. The numbers of nodes, directed links, and network regions are shown in Table I. The Telstra network is an ISP network collected by Rocketfuel [35] where the network nodes are scattered across Australia. Thus, we manually partition the Telstra network into five network regions in accordance with the geographic distributions of nodes (i.e., different states of Australia). Note that the single-degree nodes in the Telstra topology are removed since they have no influence on routing performance evaluations [36]. The second real-world network topology is obtained from Google Cloud [17], and the nodes are distributed around the world. We divide this global network into three regions based on different continents: Asia, North America, and Europe. For the large-scale synthetic network topology generated from BRITE [37], there are 204 nodes and 964 directed links in the 4x4 grid region-level topology. As a result, the whole network is partitioned into 16 regions such that each region contains 10-15 nodes and 40-60 links.

In addition to the network connectivity information, it is essential to assign proper link weights and capacities to the three networks. As suggested by [26], all the link weights are set to 10 while each link's capacity is determined by the degree of the two nodes connected to that link. Specifically, the link capacity is set to 10 Gbps if either of the two directly connected nodes' degrees is larger than three; otherwise, we only assign 5 Gbps capacity to the link. Since the measured TMs are not available for the three networks, we generate a sequence of synthesized spatiotemporal TMs based on the Modulated Gravity Model (MGM) [38], [39]. To emulate the characteristics of real traffic, MGM uses gravity-model-like constraints to construct spatial properties and utilizes a sinusoid to reflect the cyclical nature of traffic. Therefore, the generated TMs can expose diurnal patterns that are commonly seen in the networks to reflect traffic changes. Moreover, we need to consider dynamic traffic fluctuations in real traffic scenarios, such as unexpected traffic spikes. There are several parameters in MGM that can be tuned to adjust traffic variations (e.g., spatial variance and Peak-to-Mean (PM) ratio), and we also use an exponential model [38] to introduce additional traffic fluctuations. In our evaluation, we synthesize 200 TMs for training FedTe and another 200 TMs for testing. In both of the training set and testing set, there are 100 dynamic TMs (i.e., large variation MGM + exponential model) and 100 stable TMs (i.e., small variation MGM only). For dynamic TMs, we introduce large variations for hourly traffic with a spatial variance $\sigma^2 = 3$, while the PM ratios of daily and hourly traffic are separately configured as 5 and 1.5 to simulate extreme traffic conditions. As for stable TMs, the spatial variance σ^2 of hourly traffic is only 1.5, and we set the daily and hourly PM ratios to 1.1 and 1.05, respectively.

B. Training

Based on the TMs in the training set, we generate a series of training samples consists of different input network states and output target distributions. For each training sample, the inputs are the network-wide TM and intra-region/inter-region topology connectivity matrices, while the output is the target distribution of cross-region traffic that is converted from the global optimal routing solution based on the given inputs. For each iteration, FedTe collects a batch of training samples to train the neural networks using stochastic gradient descent [40]. The objective of the training process is to minimize a customized loss, which is a linear combination of the Kullback–Leibler Divergence (KLD) loss and the Mean Absolute Error (MAE) loss between the predicted cross-region traffic distribution and the target distribution as well as an L2 regularization loss [41]. FedTe is trained on a Tesla V100 GPU with 16GB memory. All the training costs are incurred offline and we apply early stopping to avoid overfitting [42]. Specifically, we use a standalone validation set that is different from the training/testing set to monitor the loss value every 1,000 iterations. Once the prediction loss cannot further improve on the validation set, we terminate the training process such that the trained model can generalize well to different scenarios. It is worth mentioning that the offline training process can be performed periodically to update the FedTe model with newly collected data or upon major topology changes.

C. Hyperparameters

For the encoder of the prediction model, we set the embedding dimension d_h to 128 and the number of attention heads

M to 8. A GNN encoded message exchanged among regions is represented by an embedding vector. Given that each element in the embedding vector is expressed in 32 bits, the size of one encoded message is $d_h * 32 = 4096$ bits. The dimension of the feed-forward sub-layer in each attention layer d_f is set to 256. Additionally, the number of attention layers H is set to the number of max hops between any two nodes in the three networks to ensure complete information exchanges. Note that the training speed would be slightly increased with fewer attention layers, but the performance of FedTe would also be degraded due to potential information loss. For the decoder, the output dimension of readout function R_i is the maximum size of regional node set among all regions, i.e., $\max_n |V^n|$. The output dimension of readout function R_o is the maximum size of outgoing pairs times the maximum size of outgoing border nodes, i.e., $\max_v((|\bar{V}^n| * |B_o^{v,d}|)|v \in V^n, d \in \bar{V}^n)$. A constant learning rate $\alpha = 10^{-4}$ is used for training. The batch size is set to 64 for all the three networks. To avoid overfitting, we also apply dropout [43] to the output of each layer of encoder with a rate of $\rho_{drop} = 0.1$, and L2 regularization [41] to each layer of encoder and decoder with regularization parameter $\lambda = 0.001$. We fix all these hyperparameters throughout our experiments since FedTe can achieve good performance on different topologies with a single set of fixed hyperparameters as shown in Section VI.

D. Baselines

We compare FedTe to the following baseline methods:

1) **Multi-Region Traffic Engineering (MRTE)** [26]: leverages multi-agent RL to model each network region as individual RL agents, such that the agents can learn from interacting with neighboring regions' agents and make better regional routing decisions towards global optimal performance.
2) **Hot-Potato Routing (HPR)** [22]: directs the outgoing traffic to the closest border router and computes the optimal local routing strategy for each region using LP.
3) **Equal-Cost Multi-Path (ECMP)** [44]: distributes traffic evenly among available next hops along the shortest paths. The link cost setting is discussed in Section V-A.

E. Evaluation Metric

To demonstrate the load balancing performance achieved by FedTe for each TM, we use a metric called performance ratio throughout the evaluation. It is defined as follows:

$$PR = \frac{U_{\text{optimal}}}{U_{\text{FedTe}}}, \tag{4}$$

where U_{optimal} is the MLU achieved by a global optimal routing obtained by solving an MCF problem [1], and U_{FedTe} represents the MLU in the entire network achieved by FedTe's distributed regional routing strategies. A higher PR value indicates that FedTe's load balancing performance is closer to that of global optimal routing. When PR = 1, FedTe is able to achieve the global optimal performance in a distributed manner. For comparison, we also compute the performance ratio of the three baseline methods according to Eq. (4).

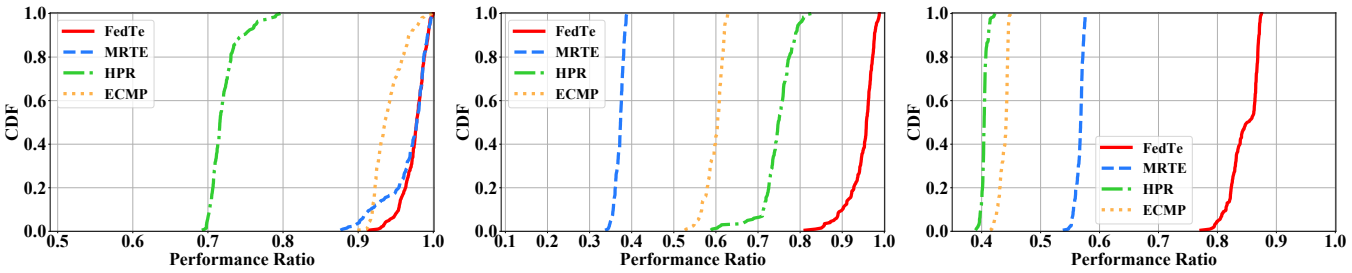
VI. Evaluation

In this section, we present the experiment results in two real-world network topologies and a large-scale synthetic topology to evaluate the performance and overhead of FedTe through extensive comparison against the baseline methods.

A. Performance Comparison

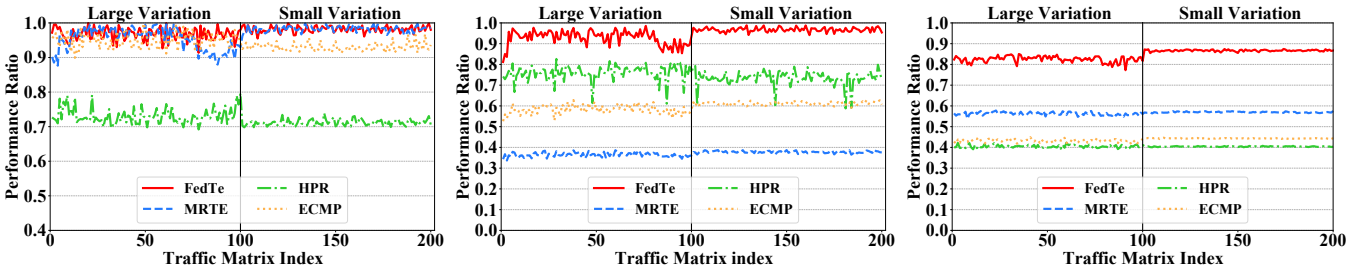
Figure 4 shows the performance ratios achieved by FedTe, MRTE, HPR, and ECMP in Cumulative Distribution Function (CDF) on the entire test set of the three networks, and Figure 5 provides the performance comparison on each synthesized test TM with different traffic variations. As shown in Fig. 4, FedTe performs consistently well with an average performance ratio of 97.5%, 94.8% and 84.6% achieved in the Telstra network, Google Cloud network, and BRITE network, respectively. FedTe can also prevent from extreme performance loss since the local routing strategies are optimized by LP with accurately predicted regional optimal TMs. For example, FedTe is able to achieve a performance ratio higher than 91.3% for all Telstra TMs, while at least 81.1% and 77.2% of performance ratio can be guaranteed for Google Cloud network and BRITE network. When it comes to different traffic scenarios, FedTe can achieve promising performance in both dynamic and stable traffic scenarios and outperform other TE methods. One interesting observation is that the performance of FedTe on large variation TMs are slightly lower than that on small variation TMs as illustrated in Fig. 5. One possible reason is that the exponential model used for large variation synthetic TMs would introduce more uncertainty in the traffic patterns.

For the baseline methods, HPR can only achieve sub-optimal performance in the three networks. While HPR has an average of 74.7% performance in the Google Cloud network as illustrated in Fig. 5(b), it becomes the worst-performing scheme in the other two networks. This is because HPR is a heuristic method which only directs the outgoing traffic to the closest border router without consideration of global optimality. Therefore, HPR may over-utilize some of the links and lead to unexpected congestion in other regions. For ECMP, the performance of static multi-path routing is heavily dependent on the nature of different network topologies without performance guarantee. As shown in Fig. 5, ECMP can achieve good load balancing performance (i.e., $PR \geq 90\%$) for all the test TMs in the Telstra network, but this is not the case for the Google Cloud network and BRITE network with only 59.8% and 43.8% of average performance, respectively. Although HPR and ECMP have the advantage of simplicity, they cannot achieve promising performance due to lack of collaboration. As for MRTE, it can also achieve near-optimal performance in the Telstra network, but the performance is not stable enough for other networks. For example, in the Google Cloud network and BRITE network, MRTE only achieves 37.1% and 56.6% performance on average. The main reason is that RL agents can hardly converge during the training stage when traffic dynamically changes with spatial and temporal patterns. Besides, since MRTE takes link load measurements as the input to RL agents, there might be potential information loss since different



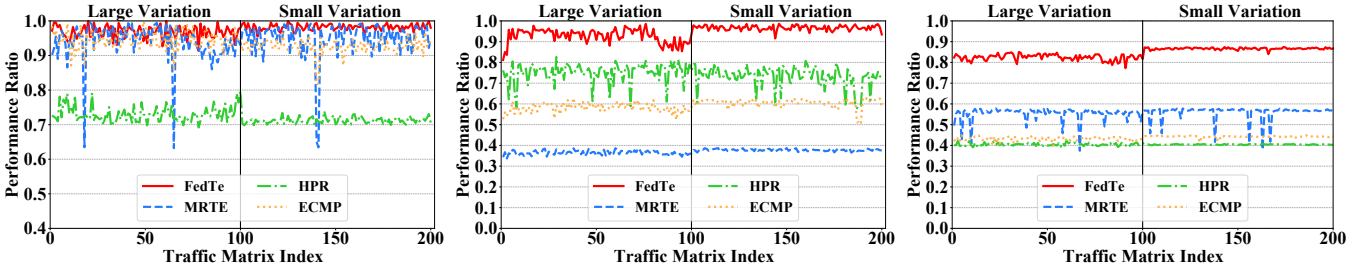
(a) Telstra network. (b) Google Cloud network. (c) BRITE network.

Fig. 4. Comparison of performance ratio in CDF among FedTe and the baseline methods on the entire test set of the three networks.



(a) Telstra network. (b) Google Cloud network. (c) BRITE network.

Fig. 5. Comparison of performance ratio on each test TM of the three networks with two different traffic fluctuation scenarios. The first 100 TMs represents dynamic traffic scenario with large variation, while the remaining 100 TMs are relatively stable with small variation.



(a) Telstra network. (b) Google Cloud network. (c) BRITE network.

Fig. 6. Comparison of performance ratio on each test TM of the three networks with random single link failure scenarios.

combinations of TMs and routing decisions can lead to the same link load observations. Overall, the experiment results indicate that our proposed FedTe can outperform the baseline methods and effectively improve load balancing performance in different traffic scenarios with good generalization.

B. Resilience

An important aspect of TE is to enable the network to be resilient against link failure situations, which means that TE should maintain good network performance in the presence of link failures. In multi-region networks, it is less likely to have multiple link failures in each network region. Therefore, we consider single link failure scenarios where a random link is broken for each given TM. When link failure happens, we exclude the failed link from LP formulation and solve LP for FedTe to obtain the updated regional routing strategies based on the predicted regional optimal TMs.

Figure 6 presents the comparison of performance ratio among FedTe and the baseline methods with random single link failures. In the three networks, FedTe is able to maintain good performance with negligible performance loss and generalize well to various single link failure scenarios. As for the baseline methods, we can observe performance degradations in several link failure scenarios compared to the normal operation without link failure. For instance, both of HPR and ECMP do not work well in the Google Cloud network as depicted in Fig. 6(b), while MRTE is experiencing severe congestion and performance loss in specific link failure scenarios. As illustrated in Fig. 6(a), the performance ratio of MRTE can be degraded to around 65% in the Telstra network for several test TMs, which is even worse than HPR's performance. In the Google Cloud network and BRITE network, the load balancing performance of MRTE is also unsatisfactory in the link failure scenarios since it cannot reach convergence in these two networks. From Fig. 6(c), we can find frequent performance degradation for MRTE in the BRITE network. In terms of worst-case performance, MRTE's performance ratio is even lower than 40%. This observation indicates that MRTE is

TABLE II
COMPARISON OF TRAINING TIME

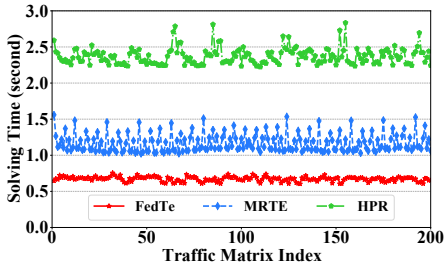
Network Topology	FedTe	MRTE
Telstra (Australia)	12 minutes	18 hours
Google Cloud	6 minutes	14 hours
BRITE	40 minutes	24 days

not capable of generalizing over all single link failure scenarios with a multi-agent RL design. In contrast, FedTe can improve the network performance by 20.1% and 28.9% on average compared to the best performing baseline methods in the Google Cloud network and BRITE network, respectively. As a result, FedTe can guarantee robust TE performance against different single link failure scenarios.

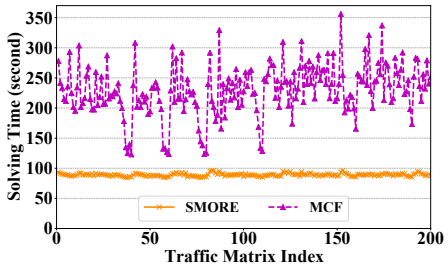
C. Training Time and Computation Costs

Table II compares the training time of the two learning-based TE solutions in the three networks. For FedTe, the training time and number of iterations required for convergence are dependent on the size of network topology as well as the number of network regions. Thanks to the scalable 2-layer GNN architecture design, the training process of FedTe is greatly facilitated. In the Telstra network and Google Cloud network, only several minutes of training are required for FedTe to reach convergence with 18,000 and 7,000 iterations, respectively. For the large-scale BRITE network, it takes less than one hour to train a FedTe model with 29,000 iterations. In contrast, the training cost of the RL-based MRTE approach is much higher compared to FedTe. As instructed in [26], MRTE's training is performed on a 4-core Intel 3.4GHz CPU and 32GB memory. In the two real-world network topologies, it takes more than 12 hours of training for MRTE with 3,000 iterations. Moreover, MRTE requires several weeks of training with 8,000 iterations in the large-scale BRITE network, which makes it difficult to achieve convergence in a reasonable timescale. From the comparison of training time, we can see that the training efficiency of FedTe is greatly improved by leveraging SL and hierarchical GNN architecture.

To evaluate the computation overhead of different TE approaches, we run an experiment to measure the solving time for updating the routing solution in accordance with each given test TM in the large-scale BRITE network. All the computation costs are measured with the same 4-core Intel 3.4GHz CPU, where Gurobi optimizer [45] is applied as an LP solver to compute the optimal routing solution. For comparison, we also measure the computation overhead of two centralized TE scheme, including SMORE [4] and MCF [1]. Note that we do not evaluate the computation cost for ECMP since it is a static routing solution. As shown in Figure 7(a), all the three distributed TE approaches can solve for local optimal routing strategies in less than 3 seconds, which demonstrates the advantages of distributed TE to quickly react to traffic changes. Since the inference time of FedTe's GNN architecture is only tens of milliseconds, FedTe is able to predict the optimal cross-region traffic distribution and optimize local routing strategies in less than one second. This is reasonable since distributed TE divides a large global routing problem into



(a) Computation costs of distributed TE.



(b) Computation costs of centralized TE.

Fig. 7. Solving time of different TE approaches to compute an updated routing solution for each of the 200 test TMs in the large-scale BRITE network.

a few small local routing subproblems for network regions to achieve better scalability. In contrast, due to the large problem size, it takes several minutes for centralized TE to solve for global optimal routing as depicted in Fig. 7(b). Even though centralized TE can achieve optimal or close-to-optimal load balancing performance with a global view of a network, they cannot accommodate dynamic traffic in a responsive manner when network size becomes larger.

VII. CONCLUSION

With consideration of computation and management overhead, large-scale networks are often partitioned into multiple regions in accordance with geographic locations. To achieve global TE objectives in multi-region networks with low computation/communication overhead, we propose FedTe, a SL-based distributed TE scheme that learns from the global optimal routing solution and predicts the optimal distribution of cross-region traffic for each network region to perform local routing optimization. FedTe leverages a scalable hierarchical GNN architecture to accelerate the training and inference processes through efficient message exchange among network nodes and regions. Extensive evaluations show that the proposed FedTe scheme can achieve near-optimal load balancing performance in a distributed manner and enable the network to be resilient against various single link failure scenarios.

ACKNOWLEDGMENTS

This paper was supported by the Key-Area Research and Development Program of Guangdong Province under Grant 2021B0101400001, the National Natural Science Foundation of China under Grant 62002019, and the Beijing Institute of Technology Research Fund Program for Young Scholars. We thank anonymous reviewers for their valuable feedbacks and Nan Geng from Tsinghua University for sharing his codes.

REFERENCES

[1] J. Zhang, K. Xi, and H. J. Chao, "Load balancing in ip networks using generalized destination-based multipath routing," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 6, pp. 1959–1969, 2015.

[2] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "Mate: Mpls adaptive traffic engineering," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3. IEEE, 2001, pp. 1300–1309.

[3] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[4] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-oblivious traffic engineering: The road not taken," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 157–170.

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

[6] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM workshop on hot topics in networks*, 2017, pp. 185–191.

[7] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1871–1879.

[8] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2249–2259, 2020.

[9] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Date: Disturbance-aware traffic engineering with reinforcement learning in software-defined networks," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021, pp. 1–10.

[10] A. Viswanathan, E. C. Rosen, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001. [Online]. Available: https://rfc-editor.org/rfc/rfc3031.txt

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

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

[13] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in sdn/ospf hybrid network," in *2014 IEEE 22nd International Conference on Network Protocols*. IEEE, 2014, pp. 563–568.

[14] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Dynamic hybrid routing: Achieve load balancing for changing traffic demands," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*. IEEE, 2014, pp. 105–110.

[15] M. Moradi, Y. Zhang, Z. Morley Mao, and R. Manghirmalani, "Dragon: Scalable, flexible, and efficient traffic engineering in software defined isp networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2744–2756, 2018.

[16] J. Zhang, Z. Guo, M. Ye, and H. J. Chao, "Smartentry: Mitigating routing update overhead with reinforcement learning for traffic engineering," in *Proceedings of the Workshop on Network Meets AI & ML*, 2020, pp. 1–7.

[17] "Global locations of google cloud topology," 2021. [Online]. Available: https://cloud.google.com/about/locations/

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

[19] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–8.

[20] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 19–24.

[21] W. Reda, K. Bogdanov, A. Milolidakis, H. Ghasemirahni, M. Chiesa, G. Q. Maguire Jr, and D. Kostić, "Path persistence in the cloud: A study of the effects of inter-region traffic engineering in a large cloud provider's network," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 2, pp. 11–23, 2020.

[22] U. Feige and P. Raghavan, "Exact analysis of hot-potato routing," in *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1992, pp. 553–562.

[23] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[24] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic control*, vol. 57, no. 3, pp. 592–606, 2011.

[25] K. Srivastava, A. Nedić, and D. Stipanović, "Distributed min-max optimization in networks," in *2011 17th International Conference on Digital Signal Processing (DSP)*. IEEE, 2011, pp. 1–8.

[26] N. Geng, T. Lan, V. Aggarwal, Y. Yang, and M. Xu, "A multi-agent reinforcement learning perspective on distributed traffic engineering," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–11.

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

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2018.

[29] H. Räcke, "Optimal hierarchical decompositions for congestion minimization in networks," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, ser. STOC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 255–264. [Online]. Available: https://doi.org/10.1145/1374376.1374415

[30] P. Pinyoanuntapong, M. Lee, and P. Wang, "Delay-optimal traffic engineering through multi-agent reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 435–442.

[31] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=ByxBFsRqYm

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

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

[36] D. Applegate and E. Cohen, "Making routing robust to changing traffic demands: Algorithms and evaluation," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1193–1206, 2006.

[37] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2001, pp. 346–353.

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

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

[40] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," 2018.

[41] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proceedings of the 4th International Conference on Neural Information Processing Systems*, ser. NIPS'91. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, p. 950–957.

[42] L. Prechelt, "Early stopping - but when?" in *Neural Networks: Tricks of the Trade - Second Edition*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K. Müller, Eds. Springer, 2012, vol.

7700, pp. 53–67. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_5

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

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

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