LARRI: Learning-based Adaptive Range Routing for Highly Dynamic Traffic in WANs

Minghao Ye[†], Junjie Zhang[‡], Zehua Guo[§*], H. Jonathan Chao[†]
[†]New York University [‡]Fortinet, Inc. [§]Beijing Institute of Technology

***Abstract*—Traffic Engineering (TE) has been widely used by network operators to improve network performance and provide better service quality to users. One major challenge for TE is how to generate good routing strategies adaptive to highly dynamic future traffic scenarios. Unfortunately, existing works could either experience severe performance degradation under unexpected traffic fluctuations or sacrifice performance optimality for guaranteeing the worst-case performance when traffic is relatively stable. In this paper, we propose LARRI, a learning-based TE to predict adaptive routing strategies for future unknown traffic scenarios. By learning and predicting a routing to handle an appropriate range of future possible traffic matrices, LARRI can effectively realize a trade-off between performance optimality and worst-case performance guarantee. This is done by integrating the prediction of future demand range and the imitation of optimal range routing into one step. Moreover, LARRI employs a scalable graph neural network architecture to greatly facilitate training and inference. Extensive simulation results on six real-world network topologies and traffic traces show that LARRI achieves near-optimal load balancing performance in future traffic scenarios with up to 43.3% worst-case performance improvement over state-of-the-art baselines, and also provides the lowest end-to-end delay under dynamic traffic fluctuations.**

***Index Terms*—traffic engineering, range routing, supervised learning, graph neural networks, routing prediction**

I. Introduction

Traffic Engineering (TE) aims to optimize network performance and resource utilization by configuring the routing across Wide Area Networks (WANs) to control traffic distribution. Traditional TE solutions [1]–[9] usually formulate and solve a routing optimization problem to balance the load on network links for given traffic demands. Although traffic demands among network nodes are relatively stable most of the time in a WAN, highly dynamic traffic fluctuations have been observed in real life [10]. A typical way to deal with unexpected traffic fluctuations is to update routing periodically based on the most recent traffic demands between all node pairs, which are known as Traffic Matrices (TMs), to improve network performance [11]–[14]. The TMs can be periodically measured and collected during the past time interval (e.g., every 5 minutes) using network measurement techniques, such as SNMP [15] and NetFlow [16]. However, once the future traffic demands deviate considerably from previously measured TMs, the current routing strategy might become *incompatible* since it is optimized based on previous TMs. In this situation, the network performance could degrade dramatically[1].

*Corresponding author.

[1]In terms of load balancing performance, which is verified through experiment results with real-world network topologies and TMs in Section II.

To accommodate highly dynamic traffic and address the routing incompatibility issue, one straightforward approach is to frequently update routing to catch up with traffic changes. However, it could introduce severe network service disruption [17], [18] and high routing update overhead [19], [20]. Another way to work around this issue is to adopt oblivious routing approaches [21], [22], which provide a strong worst-case performance guarantee for all TMs without routing updates. Although oblivious routing can handle unexpected traffic changes, it tends to be too conservative and performs sub-optimal on each TM [22]. To balance performance optimality and worst-case performance guarantee, one interesting idea is to periodically compute an optimal routing targeting a range of TMs instead of all TMs [10], [22]. However, it is very difficult, if not impossible, to obtain an accurate range of future traffic demands in advance. As a result, an inaccurate demand range may compromise the performance of these methods.

Recently, emerging Machine Learning (ML) techniques provide new opportunities to improve network performance under future traffic variations. Existing works usually predict future TMs based on historical traffic demand measurements [23]–[26], but a predicted TM only records the average traffic demands within a future time period and cannot effectively capture fine-grained future traffic variations. In this situation, ML could be applied to predict the future demand range for the above-mentioned range routing approaches [10], [22] to optimize routing with consideration of future traffic variations. However, there are two main challenges. First, good demand range predictions do not necessarily result in promising routing performance. Since demand range prediction is decoupled from range routing optimization, small prediction errors might result in undesired routing decisions. As shown in [24], [25], a relatively accurate demand prediction could still lead to poor routing performance under actual future traffic demands. Second, the computation complexity of range routing is prohibitively high. Even though the demand range is assumed to be accurately predicted, it would be very time-consuming to compute a range routing strategy. As we verified in Section V-F, more than 24 hours are required for computing range routing [10], [22] in the Google Cloud network [27] with 42 nodes and 156 links, which inevitably limits timely routing updates and thus cannot accommodate future traffic variations.

In this paper, we propose **L**earning-based **A**daptive **R**ange **R**out**I**ng (LARRI), which enables time-efficient and adaptive routing prediction to accommodate highly dynamic future traffic scenarios. By integrating demand range prediction and range routing imitation into one step, LARRI directly predicts

a routing based on historically measured traffic demands to handle an appropriate range of possible TMs in terms of future traffic fluctuations. During the offline training, LARRI learns from the target routing strategies provided by our proposed path-based range routing Linear Programming (LP) model (simplified and reformulated based on [22]), where the target future demand range can be obtained from a pre-collected TM dataset for training purposes. In the online deployment, future traffic demands are unknown beforehand when making routing decisions. Instead of expensively solving for range routing [10], [22] with an inaccurate demand range, a well-trained LARRI model can predict a routing for large networks in seconds to accommodate future traffic variations. When future traffic dramatically fluctuates, LARRI would predict a robust routing to handle a wide range of future possible TMs to avoid severe performance degradation. When future traffic is predicted to be relatively stable, a routing targeting a narrow range of future possible TMs would be generated by LARRI to ensure performance optimality. Such routing strategies can be deployed through existing Software-Defined Networking (SDN) [28] techniques to facilitate flexible and timely routing updates. Moreover, we utilize graph representation learning techniques and message passing frameworks offered by Graph Neural Networks (GNNs) [29], [30] to design a scalable routing prediction model with unique advantages in modeling network topologies and characterizing traffic demands.

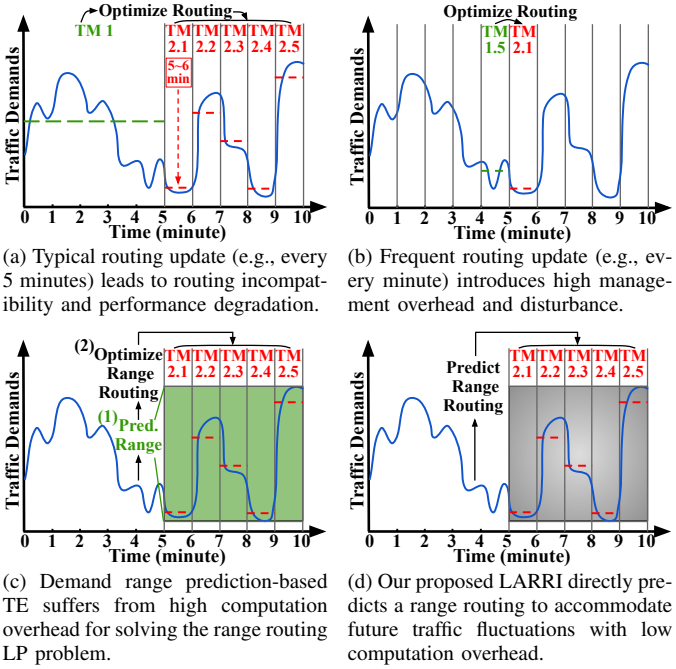
The contributions of this paper are summarized as follows:

- We propose an ML-based routing scheme called LARRI that for the first time directly predicts appropriate routing strategies for future unknown traffic scenarios in WANs.
- We formulate a path-based range routing LP model to provide good performance for a range of TMs with reduced computation complexity, which is used to generate target routing strategies for training LARRI.
- We customize a scalable GNN-based architecture to reduce the complexity of the prediction model and accelerate the training and inference processes.
- Extensive simulation results in six real-world network topologies and traffic traces show that LARRI can improve the worst-case load balancing performance by up to 43.3% compared to state-of-the-art baselines, and also provides the lowest end-to-end delay in extreme cases to ensure good service quality.

The rest of the paper is organized as follows. Section II explains our motivation with experiment results. Section III introduces our proposed path-based range routing LP model. Section IV details the design of LARRI's routing prediction model. Section V evaluates LARRI's performance. Section VI lists related work, and Section VII concludes this paper.

II. Motivation

Routing incompatibility. Existing TE solutions usually optimize and update routing every 5 minutes based on the recently measured 5-minute TM [11]–[14]. To verify the routing incompatibility issue and its impact on network performance, we evaluate typical routing update methods on the BRAIN network [31], [32] with real traffic traces. Fig. 1(a) provides an illustrative example of typical TE solutions, which optimize routing every 5 minutes (e.g., at time = 5) based on recent TMs (e.g., TM 1). Here, TM 1 records the average traffic demands of all source-destination pairs in the past 5 minutes (e.g., time = 0-5), and the routing is optimized by solving a Multi-Commodity Flow (MCF) problem [33] to minimize the Maximum Link Utilization (MLU), which is a common objective for load balancing (i.e., $\min \max_{e \in E}(l_e/c_e)$, see Table I). Then, the routing optimized for TM 1 would be evaluated in the next 5 minutes in terms of MLU. Each of the future TMs (e.g., TM 2.1-2.5) is a 1-minute TM representing fine-grained future traffic variations. With the above settings, we conduct experiments on the entire TM dataset and present the evaluation results in Fig. 2, which demonstrates that the load balancing performance could significantly degrade in many future traffic scenarios. For example, we observe more than 30% performance degradation[2] in 28.7% of the future TMs. In other words, the network could experience significant performance degradation for around 7 hours per day (24 hours $*$ 28.7%) on average. Moreover, the routing incompatibility issue would cause even more than 70% performance degradation in extreme cases when traffic fluctuates dramatically (e.g., unexpected traffic spikes). As a result, such incompatible routing could overload WAN links and lead to severe network congestion with increased delay and packet loss rate [10].



(a) Typical routing update (e.g., every 5 minutes) leads to routing incompatibility and performance degradation.

(b) Frequent routing update (e.g., every minute) introduces high management overhead and disturbance.

(c) Demand range prediction-based TE suffers from high computation overhead for solving the range routing LP problem.

(d) Our proposed LARRI directly predicts a range routing to accommodate future traffic fluctuations with low computation overhead.

Fig. 1: Different TE solutions for addressing the routing incompatibility issue under highly dynamic future traffic fluctuations. The inputs for routing optimization and the future TMs for routing evaluation are marked in green and red colors, respectively.

Limitation of existing solutions. One possible way to work

[2]Compared to the optimal load balancing performance (MLU), where the routing is optimized and applied on the exact same future TM.

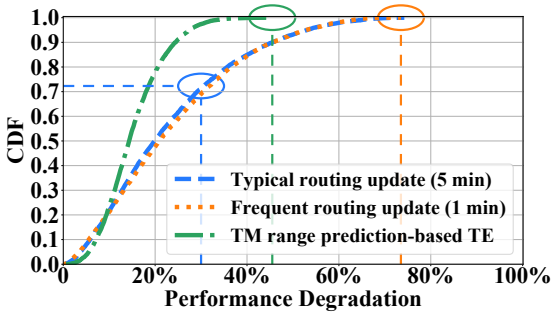


Fig. 2: Routing performance degradation under dynamic traffic fluctuations in the BRAIN network with different TE solutions.

around the routing incompatibility issue is to update routing more frequently (e.g., in a per-minute level) in the hope that the routing could closely adhere to traffic changes, as shown in Fig. 1(b). Unfortunately, frequent routing updates cannot fundamentally solve the routing incompatibility issue under highly dynamic traffic fluctuations. As shown in Fig. 2, the performance improvement of frequent 1-minute updates over typical 5-minute updates is negligible, and the network could still experience significant performance degradation. Besides, there are several potential drawbacks: (1) leads to significant network disturbance and service disruption [17], [18]; (2) introduces additional routing update overhead [19], [20]. We also evaluated other TE solutions (e.g., oblivious routing, TM prediction-based TE) and presented the results in Section V, which verify their limitations as described in Section I.

Our insight. To fundamentally solve the routing incompatibility issue, TE should take future traffic variations into consideration and provide an appropriate routing to handle dynamic traffic fluctuations during the future time interval (e.g., next 5 minutes). We initially considered a promising solution as shown in Fig. 1(c), which can be divided into two steps: (1) predicts a demand range for the future time interval; (2) optimizes a range routing with LP [22] based on the predicted demand range. As a result, the routing should be more robust to dynamic traffic variations in the future. Unfortunately, the computation complexity of the original range routing LP model [22] is prohibitively high, as we mentioned in Section I. Therefore, we simplify and reformulate the original LP model with a few diverse preconfigured paths to reduce the computation cost. The promising results in Fig. 2 demonstrate the effectiveness of our proposed path-based LP model under the assumption that the demand range can be accurately predicted. However, it is still computationally expensive to solve the path-based LP model for practical large topologies during online deployment, as later shown in Section V-F. Besides, it is impossible to predict the future demand range without any error in practice. In this situation, demand range prediction-based TE is prone to prediction errors since the prediction and optimization are separated. Thus, we integrate demand range prediction and range routing imitation into one step and propose LARRI. As depicted in Fig. 1(d), LARRI can directly learn and predict a range routing to accommodate future traffic fluctuations with low computation overhead.

TABLE I: Notations

$G(V,E)$	network topology with nodes V and directed links E ($\|V\|=N, \|E\|=M$).
$D^{s,d}$	the traffic demand from source s to destination d ($s,d \in V$).
$P^{s,d}$	the set of preconfigured paths from source s to destination d ($s,d \in V$, $s \neq d$, $\|P^{s,d}\| \leq K$).
$\sigma_p^{s,d}$	the percentage of traffic demand from source s to destination d routed on 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 path p; 0, otherwise ($p \in P^{s,d}$, $e \in E$).
l_e	the traffic load on link e ($e \in E$).
c_e	the capacity of link e ($e \in E$).

III. Path-based Range Routing

In this section, we introduce the range routing strategy used as the training target for LARRI and detail its LP formulation. Table I shows the notations used in this section.

A. Overview

With consideration of management overhead and network disturbance, LARRI performs routing updates at 5-minute intervals, which is a common routing update frequency adopted by other TE solutions [11]–[14]. Given a sequence of past measured traffic traces, LARRI is trained to predict a target routing strategy that is optimized to accommodate an appropriate range of possible TMs within the future routing update interval, as shown in Fig. 1(d). Here, a range of possible TMs is a set of possible TMs with demand range restrictions. Let $D_+^{s,d}$ and $D_-^{s,d}$ denote the upper and lower demand bounds for a source-destination pair $\langle s,d \rangle$, respectively. Then, a range of possible TMs can be represented as follows.

$$\{\text{TM}\}_{\text{range}} = \{\text{TM} \,|\, D^{s,d} \in \text{TM},\ D_-^{s,d} \leq D^{s,d} \leq D_+^{s,d},\ s,d \in V\}.$$

Assume that a continuous period of traffic traces is available, then $D_+^{s,d}$ and $D_-^{s,d}$ can be determined according to the traffic fluctuations within the routing update interval (details in Section V-A2). Given the demand range $D_+^{s,d}$ and $D_-^{s,d}$ for each source-destination pair $\langle s,d \rangle$, a target routing strategy is obtained by solving a range routing optimization problem described in Section III-B. We leverage an oblivious routing algorithm [21], [34] to compute and preconfigure a set of diverse forwarding paths $\{P^{s,d}\}$ for each source-destination pair, where the path budget K is usually set to 4 [4]. It has been shown that the combination of diverse preconfigured forwarding paths and flexible path split ratios is good enough to achieve close-to-optimal performance in real-world topologies and TMs [4]. In the following subsection, we will focus on the range routing problem with preconfigured paths.

B. Range Routing Optimization Problem

Given a single TM and a routing R, the performance ratio PR is defined to measure how far R is from being optimal:

$$PR_{\text{R}}(\text{TM}) = U_{\text{R}}(\text{TM}) / U_{\text{opt}}(\text{TM}), \quad (1)$$

where $U_{\text{R}}(\text{TM})$ is the MLU achieved by R, and $U_{\text{opt}}(\text{TM})$ is the MLU achieved by an optimal routing on the given TM. Note that a lower performance ratio indicates that the performance of R is closer to the optimal performance.

$PR_{\text{R}}(\text{TM}) = 1$ means that R achieves optimal performance. By extending $PR_{\text{R}}(\text{TM})$ to be $PR_{\text{R}}(\{\text{TM}\})$ with respect to a set of TMs $\{\text{TM}\}$, PR_{R} can be defined as

$$PR_{\text{R}}(\{\text{TM}\}) = \max_{\text{TM}\in\{\text{TM}\}} PR_{\text{R}}(\text{TM}). \tag{2}$$

A routing R is optimal for the set of TMs $\{\text{TM}\}$, if and only if $PR_{\text{R}}(\{\text{TM}\})$ is minimal. $PR_{\text{R}}(\{\text{TM}\})$ is always at least 1. Since a routing R that achieves optimal performance for all TMs in a set may not exist, the minimum $PR_{\text{R}}(\{\text{TM}\})$ can be strictly larger than 1 [22]. Note that when $\{\text{TM}\}$ includes all possible TMs, the performance ratio PR_{R} is referred to as the oblivious performance ratio.

Given a topology $G(V, E)$ with a set of preconfigured paths $\{P^{s,d}\}$ for each source-destination pair and a range of possible TMs $\{\text{TM}\}_{\text{range}}$, the objective of the range routing problem is to obtain the optimal path split ratios $\{\sigma_p^{s,d}\}$ such that the performance ratio $PR_{\text{R}}(\{\text{TM}\}_{\text{range}})$ is minimized. The range routing optimization problem can be formulated as follows.

$$\min PR \tag{3a}$$
$$\text{subject to} \quad \sigma_p^{s,d} \geq 0 \qquad p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{3b}$$
$$\sum_{p\in P^{s,d}} \sigma_p^{s,d} = 1 \qquad s, d \in V,\ s \neq d \tag{3c}$$
$$\sum_{s,d\in V, s\neq d} \sum_{p\in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} \cdot D^{s,d}/c_e \leq PR \cdot U_{\text{opt}}(\text{TM}) \qquad e \in E,\ D^{s,d} \in \text{TM}, \text{TM} \in \{\text{TM}\}_{\text{range}} \tag{3d}$$

where (3b) and (3c) are path split ratio constraints, and (3d) are performance ratio constraints. There will be an infinite number of constraints (3d), given a range of possible TMs.

By scaling $D^{s,d}$, we can replace (3d) as follows:

$$\sum_{s,d\in V, s\neq d} \sum_{p\in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} \cdot D^{s,d}/c_e \leq PR \qquad e \in E,\ D^{s,d} \in \text{TM}/U_{\text{opt}}(\text{TM}), \text{TM} \in \{\text{TM}\}_{\text{range}}. \tag{4}$$

Given a routing $\{\sigma_p^{s,d}\}$, the constraints (4) can be verified by solving an auxiliary LP for each link e and check whether the objective is $\leq PR$ or not. Let $l_p^{s,d}$ denotes the traffic demand from source s to destination d routed on path p ($p \in P^{s,d}$, $s, d \in V, s \neq d$). Then, the auxiliary LP for each link e can be formulated as follows.

$$\max \sum_{s,d\in V, s\neq d} \sum_{p\in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} \cdot D^{s,d}/c_e \tag{5a}$$
$$\text{subject to} \quad l_p^{s,d} \geq 0 \qquad p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{5b}$$
$$\sum_{p\in P^{s,d}} l_p^{s,d} = D^{s,d} \qquad s, d \in V,\ s \neq d \tag{5c}$$
$$\sum_{s,d\in V, s\neq d} \sum_{p\in P^{s,d}} \delta_{p,e}^{s,d} \cdot l_p^{s,d} \leq c_e \tag{5d}$$
$$D^{s,d} \geq \eta \cdot D_-^{s,d} \qquad s, d \in V,\ s \neq d \tag{5e}$$
$$D^{s,d} \leq \eta \cdot D_+^{s,d} \qquad s, d \in V,\ s \neq d \tag{5f}$$
$$\eta > 0 \tag{5g}$$

where (5b) and (5c) are path load constraints, and (5e) and (5f) are demand range constraints. η is a demand multiplier to make sure the constraints (5d) are met. Then, the dual of (5) for each link e can be formulated as follows.

$$\min \sum_{e^*\in E} c_{e^*} \cdot \pi(e, e^*) \tag{6a}$$
$$\text{subject to} \sum_{s,d\in V, s\neq d} (D_-^{s,d} \cdot s_-^{s,d}(e) - D_+^{s,d} \cdot s_+^{s,d}(e)) \geq 0 \tag{6b}$$
$$\sum_{p\in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} = c_e \cdot (\varphi^{s,d}(e) + s_+^{s,d}(e) - s_-^{s,d}(e)) \qquad s, d \in V,\ s \neq d \tag{6c}$$
$$\sum_{e^*\in p} \pi(e, e^*) - \varphi^{s,d}(e) \geq 0 \quad p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{6d}$$
$$\pi(e, e^*) \geq 0 \qquad e^* \in E \tag{6e}$$
$$\varphi^{s,d}(e) \geq 0 \qquad s, d \in V,\ s \neq d \tag{6f}$$
$$s_-^{s,d}(e) \geq 0 \qquad s, d \in V,\ s \neq d \tag{6g}$$
$$s_+^{s,d}(e) \geq 0 \qquad s, d \in V,\ s \neq d \tag{6h}$$

Then, by combining (3) with (6), the range routing optimization problem can be formulated as follows.

$$\min PR \tag{7a}$$
$$\text{subject to} \quad \sigma_p^{s,d} \geq 0 \qquad p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{7b}$$
$$\sum_{p\in P^{s,d}} \sigma_p^{s,d} = 1 \qquad s, d \in V,\ s \neq d \tag{7c}$$
$$\sum_{e^*\in E} c_{e^*} \cdot \pi(e, e^*) \leq PR \qquad e \in E \tag{7d}$$
$$\sum_{s,d\in V, s\neq d} (D_-^{s,d} \cdot s_-^{s,d}(e) - D_+^{s,d} \cdot s_+^{s,d}(e)) \geq 0 \quad e \in E \tag{7e}$$
$$\sum_{p\in P^{s,d}} \delta_{p,e}^{s,d} \cdot \sigma_p^{s,d} = c_e \cdot (\varphi^{s,d}(e) + s_+^{s,d}(e) - s_-^{s,d}(e)) \qquad e \in E,\ s, d \in V,\ s \neq d \tag{7f}$$
$$\sum_{e^*\in p} \pi(e, e^*) - \varphi^{s,d}(e) \geq 0 \qquad e \in E,\ p \in P^{s,d},\ s, d \in V,\ s \neq d \tag{7g}$$
$$\pi(e, e^*) \geq 0 \qquad e, e^* \in E \tag{7h}$$
$$\varphi^{s,d}(e) \geq 0 \qquad e \in E,\ s, d \in V,\ s \neq d \tag{7i}$$
$$s_-^{s,d}(e) \geq 0 \qquad e \in E,\ s, d \in V,\ s \neq d \tag{7j}$$
$$s_+^{s,d}(e) \geq 0 \qquad e \in E,\ s, d \in V,\ s \neq d \tag{7k}$$

where (7d)-(7k) are the constraints for range routing with demand restrictions. By solving problem (7) with the future demand range derived from training samples, we can obtain a set of optimal path split ratios $\{\sigma_p^{s,d}\}$, which is the target routing strategy to be learned by LARRI.

IV. Routing Prediction Model

To design a scalable routing prediction model, we leverage GNNs [29], [30] to model network topologies and characterize traffic demands. Unlike traditional neural network architectures where the size of the prediction model exponentially expands as the number of network nodes increases, each module/layer of LARRI is shareable and reused by each node in parallel, which greatly simplifies model complexity and substantially reduces training and inference time (shown in Section V-F). Fig. 3 shows an example of LARRI. It consists of an encoder that performs per-node embedding and message exchange, and a decoder that interprets desired path split ratios from each encoded and updated node embedding.

Encoder. The inputs to the encoder are node features and a node connectivity matrix. The features of a given network node are a series of demands originating from that node, and the connectivity matrix indicates the neighbors of each node.

At first, the encoder computes an initial embedding for each node using a shared Long Short-Term Memory (LSTM) [35]

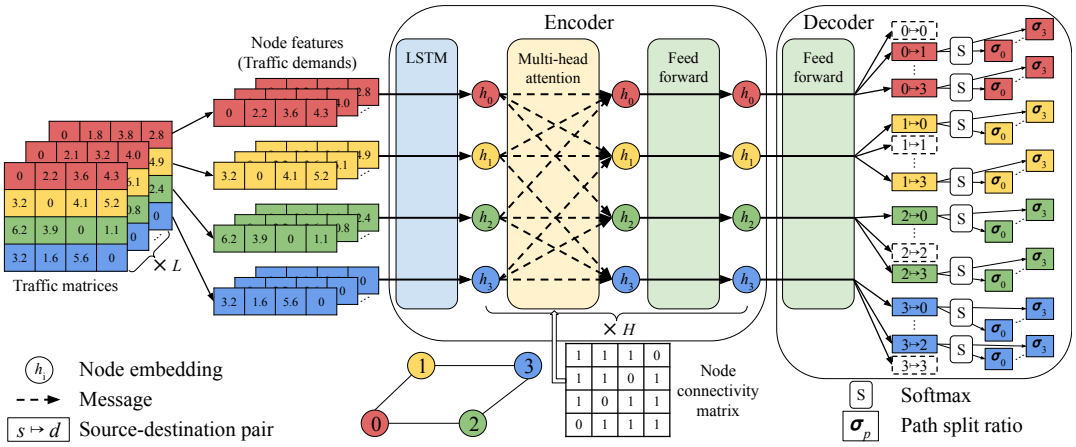


Fig. 3: Example of encoder and decoder in LARRI's GNN-based routing prediction model.

layer. Then, each node's embedding is updated by exchanging messages with its neighbors. The embedding update module consists of a stack of H identical attention layers [36], [37], and each attention layer is composed of two sub-layers: (1) a Multi-Head Attention (MHA) layer that performs message exchange between neighboring nodes, and (2) a node-wise fully connected Feed-Forward (FF) layer that performs a nonlinear transformation. A skip connection [38] and layer normalization [39] are applied to each sub-layer to facilitate training. Let h_v denote the node embedding for a given node v. It is updated iteratively by aggregating the messages passed from its neighbors with a learnable message function $M(\cdot)$:

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

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 the embedding update process, and each iteration can be considered as a feature propagation. After H iterations, each node's embedding would include the information of H hops away neighbors. Thus, when H equals the number of max hops in the network, it would be sufficient for each node to capture the complete information of the whole network (i.e., TMs and topology information).

Decoder. The decoder serves as a readout function R, which consists of a node-wise fully connected FF layer and a pair-wise softmax layer. It interprets each node's final embedding h_v^H as the corresponding path split ratios, i.e.,

$$\{\sigma_p^{v,d} | p \in P^{v,d}, d \in V\} = R(h_v^H, \theta_R), v \in V.$$

Given that $\sigma_p^{v,v}$ is an invalid source-destination pair, we will not interpret its path split ratios, as shown in Fig. 3.

V. Evaluation

In this section, a series of simulations are conducted using real-world network topologies and real traffic traces to evaluate the performance of LARRI and demonstrate its effectiveness by comparing it with different baseline methods.

A. Evaluation Setup

Six real-world network topologies are used in our evaluation, and the number of nodes and directed links is listed in Table II. All the single-degree nodes are removed since they have no effect on routing performance evaluation [40].

TABLE II: Real-world network topologies used in evaluation

Topology	Nodes	Links	TM Dataset	TM Interval
BRAIN	9	28	7 days (2013)	1 minute
Abilene	11	28	24 weeks (2004)	5 minutes
CERNET	14	32	5 weeks (2014)	5 minutes
GÉANT	22	70	16 weeks (2005)	15 minutes
Tiscali	30	134	Synthetic	N/A
Google Cloud	42	156	Synthetic	N/A

*1) **Dataset**:* The first four networks in Table II are equipped with real traffic traces measured at different time intervals. The BRAIN network is a research network in Berlin with topology connectivity and real-life traffic data available at SNDlib [31], [32]. The BRAIN TMs are collected at 1-minute intervals for a continuous period of 7 days in 2013. The Abilene [41], CERNET [42], and GÉANT [43], [44] networks are the research and education networks in the United States, China, and Europe, respectively. The real TMs of the Abilene, CERNET, and GÉANT networks are measured at longer time intervals (e.g., 5 or 15 minutes), as shown in Table II. For each of the four networks, we split the TM dataset into 80% of training samples and 20% of testing samples. For example, we train a LARRI model with the CERNET TMs in the first 4 weeks and then evaluate LARRI on all TMs in week 5 that are unseen before. Note that the last two networks using synthetic traffic traces would be detailed in Section V-E.

*2) **Training**:* As discussed in Section III, LARRI would predict a routing to appropriately handle traffic fluctuations within the future routing update interval. Since LARRI needs to learn from target range routing strategies, it can be formulated as a Supervised Learning (SL) problem. Given a training dataset that includes a continuous period of TMs, we can generate training samples for the above problem. A training sample consists of an input and a target output, where an input includes a sequence of TMs in the past 5 intervals and a topology-specific node connectivity matrix, and the output is a target range routing strategy obtained by solving the LP problem (7), whose demand range is configured according to the TMs in the next interval. Generally, LARRI performs routing updates every 5 minutes in all networks except for the GÉANT network. Since the GÉANT TMs are measured every

15 minutes, LARRI's routing update interval should be set to 15 minutes in the GÉANT network to match the TM interval.

After generating the training samples, we randomly select 80% of the total training samples as the training sample set, and the remaining 20% is considered as the standalone validation sample set, which is used for early stopping to avoid overfitting [45]. LARRI is trained on the training sample set using stochastic gradient descent [46] with the objective of minimizing 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 routing and the target routing as well as an L2 regularization loss [47]. For the encoder of the prediction model, we set the embedding dimension to 128 and the number of attention heads to 8. The dimension of the FF sub-layer is set to 256. For the decoder, the output dimensions of the FF layer are $N \times K$, which correspond to the preconfigured paths for each destination node. A constant learning rate $\alpha = 10^{-4}$ is used for training, and the batch size is set to 512. To avoid overfitting, we also apply dropout [48] to the output of each layer of the encoder with a rate of $\rho_{drop} = 0.1$, and L2 regularization [47] to each layer of the encoder and decoder with regularization parameter $\lambda = 0.001$. Once the training is done, we test LARRI in the test dataset with future TMs that are unseen before.

*3) **Simulation Environment**:* In our evaluation, The proposed GNN-based model is implemented using TensorFlow [49], and the Gurobi optimizer [50] is applied as an LP solver to compute target routing strategies. All the training tasks are conducted in a high-performance computing cluster with a single GPU Tesla V100. Once the training is done, we conduct all simulation tests on a Linux server with a 4-core Intel 2.9 GHz CPU and 16 GB memory. We use NetworkX [51] to simulate network environments based on the real-world network topologies listed in Table II. Given the TM datasets, the traffic flows with different demand volumes are fed into the networks to simulate dynamic traffic variations.

*4) **Performance Metrics**:* For each test TM, we use the MLU performance ratio PR in Eq. (1) to evaluate the load balancing performance of different TE solutions. A lower PR indicates better routing performance. Besides, we also evaluate the end-to-end delay $\Omega = \sum_{e \in E} l_e/(c_e - l_e)$ of each TE solution as described in [52] (see Table I). Given a TM and a routing R, we define a delay performance ratio DR as follows:

$$DR_{\mathrm{R}}(\mathrm{TM}) = \Omega_{\mathrm{R}}(\mathrm{TM})/\Omega_{\mathrm{opt}}(\mathrm{TM}), \tag{8}$$

where $\Omega_{\mathrm{R}}(\mathrm{TM})$ is the end-to-end delay achieved by R, and $\Omega_{\mathrm{opt}}(\mathrm{TM})$ is the end-to-end delay achieved by an optimal routing on the given TM with an objective to minimize Ω. Similarly, a lower DR indicates better end-to-end delay performance. Note that the routing strategies being evaluated are still optimized for load balancing performance (i.e., minimize MLU) instead of minimizing end-to-end delay.

B. Baseline Methods

We evaluate the following baseline methods for comparison.

MCF [33]: formulates a Multi-Commodity Flow (MCF) problem with an objective of minimizing MLU to obtain an optimal routing based on the TM in the previous interval (e.g., TM_{t-1}), and then applies the routing in the future time interval.

SMORE [4]: selects 4 preconfigured paths from [21], [34] for each source-destination pair, obtains the optimal path split ratios based on the TM in the previous interval (e.g., TM_{t-1}), and then applies the routing in the future time interval.

TM Prediction-based TE (TMP) [25]: exploits LSTM to predict the next TM based on a sequence of past TMs, obtains the optimal path split ratios based on the predicted TM, and then applies the routing in the future time interval.

COPE [10]: optimizes performance ratio over a convex hull of a set of past TMs and adopts a penalty envelope to bound the worst-case performance.

Räcke's Oblivious Routing (ROR) [21], [34]: computes a probability distribution on diverse forwarding paths based on decomposition trees, and then routes traffic according to the distribution without the knowledge of actual demands.

Optimal Oblivious Routing (OOR) [22]: optimizes a routing with respect to all possible TMs using an LP formulation presented in [22], and then distributes traffic according to the optimal oblivious routing regardless of actual demands.

Equal-Cost Multipath (ECMP) [53]: evenly distributes traffic among available next hops along the shortest paths.

Note that LARRI, SMORE, and TMP use the same set of preconfigured paths. COPE performs routing updates once a day based on a convex hull constructed from the TMs in the previous day [10]. To bound the worst-case performance, we set the penalty envelope of COPE to 2.0 in the Abilene network (as the original setting in [10]), and 3.0 for the other networks.

C. Performance Evaluation in BRAIN

Fig. 4 shows the MLU performance ratio that each scheme achieved on the entire test set of the BRAIN network versus the percentage of time intervals sorted by MLU performance ratio. We can observe that the MLU performance ratio of MCF and SMORE quickly ramps up and the worst-case MLU performance ratio is higher than 3, which indicates that they suffer from severe routing performance degradation under dynamic traffic fluctuations. Following the typical routing update method discussed in Fig. 1(a), MCF and SMORE optimize routing every 5 minutes based on the previous 5-minute average TM and then apply the routing in the future 5-minute interval. Since they rely on a past TM to perform routing optimization, the resulting routing becomes incompatible once the future TM deviates from the previously measured TM. To cope with future unknown traffic scenarios, TMP predicts a future 5-minute average TM for routing optimization instead of relying on the previously measured TM. However, the performance of TMP is not better than SMORE and MCF. There are several reasons that might explain this result: (1) The prediction errors on traffic demands might result in undesired routing decisions. (2) A single predicted TM cannot represent fine-grained future traffic variations.

To reveal the consequences of frequent routing updates, we also evaluate SMORE, MCF, and TMP in a more re-

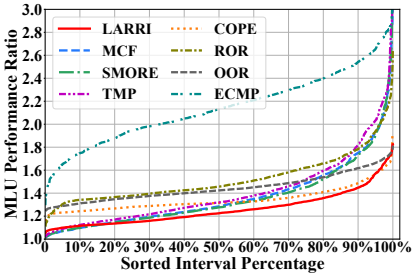


Fig. 4: Load balancing performance comparison in the BRAIN network (truncated at 3.0). The lower the MLU performance ratio, the better the load balancing performance.

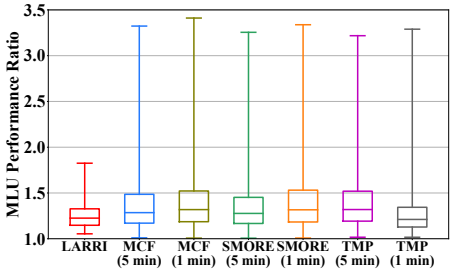


Fig. 5: Load balancing performance comparison with different routing update intervals in the BRAIN network. The whiskers represent the highest/lowest performance ratio on the test set.

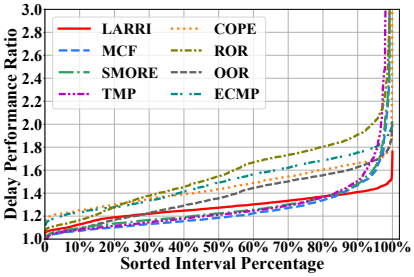


Fig. 6: End-to-end delay performance comparison in the BRAIN network (truncated at 3.0). The lower the delay performance ratio, the better the end-to-end delay performance.

sponsive manner, i.e., perform routing update/TM prediction every minute. Fig. 5 shows the corresponding results in the BRAIN network. Unfortunately, frequent routing update does not improve the performance of SMORE and MCF. Given that the traffic is highly dynamic, the traffic variations between two smaller intervals are still significant enough to cause the routing incompatibility issue. By predicting future TMs at 1-minute intervals, the performance of TMP is improved by 9.3% on average compared to the previous 5-minute prediction, but its worst-case MLU performance ratio could still be as high as 3.29. This observation demonstrates that TMP is quite sensitive to TM prediction errors. In contrast, LARRI can provide stable and robust performance against unexpected traffic fluctuations and outperform all other schemes in almost all time intervals. For example, LARRI can outperform the best-performing baseline in Fig. 5 by 43.3% in the worst case.

For oblivious routing approaches, they are aiming at providing bounded performance for all possible TMs. As shown in Fig. 4, both ROR and OOR can achieve better worst-case performance compared to SMORE, MCF, and TMP. However, they have to sacrifice performance optimality when traffic is relatively stable, which results in sub-optimal average performance. To mitigate this issue, COPE adopts a combination of convex hull and penalty envelope to handle future traffic variations. From Fig. 4, we can observe that COPE achieves better performance than OOR in most of the time intervals. However, COPE still appears to be too conservative since it has no knowledge of future traffic demands. In contrast, LARRI outperforms COPE in 95% of TMs with a 6.4% average performance improvement, and the worst-case MLU performance ratio of LARRI is bounded at 1.83, which is 4.6% better than COPE.

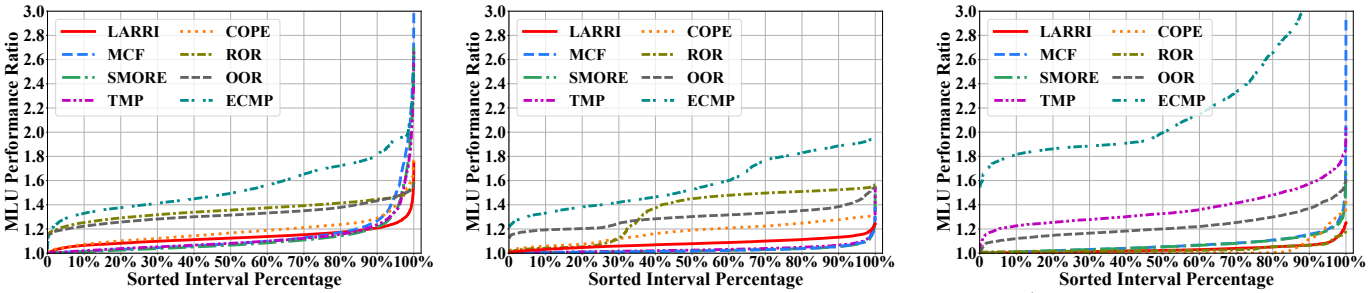
Although LARRI and the baselines are targeting load balancing performance, it is worth evaluating their routing strategies with different performance metrics (e.g., end-to-end delay) to verify their robustness in dynamic traffic scenarios. Fig. 6 shows the delay performance ratio of different TE solutions in the BRAIN network. As we expected, the baseline methods perform poorly since their routing strategies are not optimized for minimizing delay. For example, the delay performance ratio of COPE can reach 3.6 in extreme cases. Surprisingly, LARRI is still able to achieve good end-to-end delay performance with an average performance ratio of 1.28 and a worst-case performance ratio of 1.76, which outperforms all other baseline methods. As a result, LARRI can effectively trade off performance optimality and worst-case performance guarantee to accommodate future traffic variations.

D. Performance Comparison in Other Networks

Fig. 7 shows the MLU performance ratio that each scheme achieves in the other three networks with real traces. In the Abilene network, LARRI performs slightly worse than ROR and OOR in extreme cases, as shown in Fig. 7(a). One possible reason is that LARRI makes inaccurate routing predictions in a few unexpected traffic scenarios that are unseen before. However, LARRI is still capable of providing a strong worst-case performance guarantee. For example, LARRI improves the worst-case performance by at least 34.9% compared to single TM routing optimization methods (i.e., MCF, SMORE, and TMP). Besides, LARRI can still achieve considerable load balancing performance improvement over ROR and OOR in most of the future traffic scenarios.

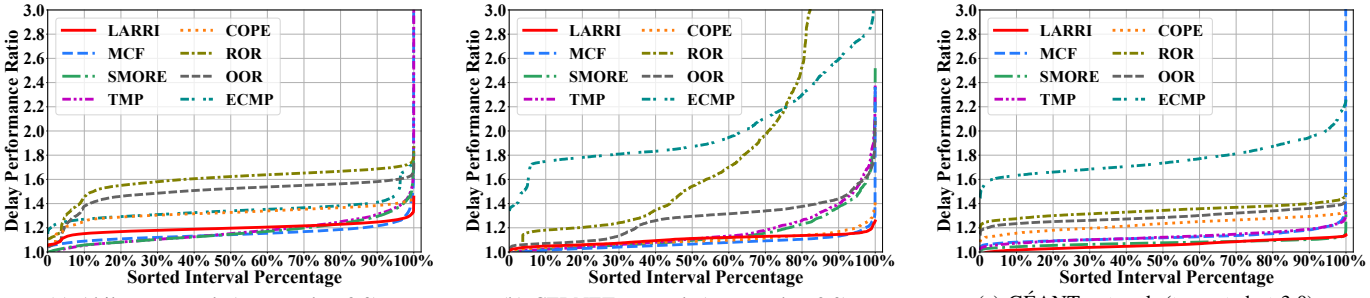
The traffic pattern in the CERNET network is relatively stable most of the time. As shown in Fig. 7(b), MCF, SMORE, and TMP achieve good performance on average, but their worst-case MLU performance ratios are not promising due to their limitations on handling dynamic traffic fluctuations. To accommodate a range of traffic demands, LARRI proactively sacrifices performance optimality to some extent and appears to perform slightly worse than MCF, SMORE, and TMP when traffic is stable. However, LARRI is still able to provide the best worst-case performance guarantee and outperform these baselines by 10.3%-21.4% in extreme cases, which can effectively alleviate the routing incompatibility issue under dynamic traffic scenarios.

In the GÉANT network, the performance of TMP is unstable and only outperforms ECMP with an average MLU performance ratio of 1.37, as shown in Fig. 7(c). Given that TMP is applying a two-step paradigm of prediction + optimization, the minimization of TM prediction errors does not necessarily lead to better routing performance. In contrast, LARRI learns from target routing strategies and directly predicts a routing with an objective to improve network performance, which turns out to be more decision-focused and outperforms TMP. Another interesting finding in the GÉANT network is that ROR performs surprisingly well. One possible reason is that



(a) Abilene network. (b) CERNET network. (c) GÉANT network (truncated at 3.0).

Fig. 7: Load balancing performance comparison in the three networks with real TMs.



(a) Abilene network (truncated at 3.0). (b) CERNET network (truncated at 3.0). (c) GÉANT network (truncated at 3.0).

Fig. 8: End-to-end delay performance comparison in the three networks with real TMs.

ROR adopts the full set of preconfigured paths for each source-destination pair (up to 78 paths per pair), and the traffic in the GÉANT network happens to be well balanced among these paths and results in better performance.

In addition to the load balancing performance, we also evaluate the end-to-end delay performance that each scheme achieves in the three networks, as depicted in Fig. 8. LARRI performs well in the three networks with the lowest worst-case delay performance ratio ranging from 1.14 to 1.45, while most of the baseline methods are experiencing an unacceptable delay in extreme cases. From Fig. 8(b), we can see that COPE performs well in the CERNET network, but its worst-case delay performance ratio is worse than LARRI by 12.8%. Moreover, LARRI can outperform COPE in the Abilene and GÉANT networks with 9.3% and 13.6% average delay performance improvement, respectively. Overall, the range routing strategy predicted by LARRI can provide promising load balancing performance as well as low end-to-end delay to ensure good service quality in dynamic future traffic scenarios.

E. Generalization and Scalability Analysis

To further demonstrate the generalization capability and scalability of LARRI, we introduce two large real-world network topologies for evaluation, i.e., the Tiscali network [54] and the Google Cloud network [27]. Their topology sizes are listed in Table II. Since there is no measured TM for these two networks, we use the Modulated Gravity Model (MGM) [55], [56] to synthesize spatiotemporal TMs with different traffic variations by adjusting MGM parameters (e.g., peak-to-mean ratio and spatial variance). In the training and test dataset, there are 100 dynamic TMs with large traffic variations and 100 stable TMs with small traffic variations. For simplicity, we omit the configuration details and only present the load balancing performance for analysis.

Fig. 9 shows the MLU performance ratio achieved in each future time interval of the Tiscali and Google Cloud networks.[3] The results are split into two parts according to traffic variations. When traffic is stable, all schemes except static baseline methods perform well. However, when traffic is dramatically fluctuating, most of the baseline methods are experiencing performance degradation due to the routing incompatibility issue, especially for MCF and SMORE. Meanwhile, it is also more difficult for TMP to accurately predict future TMs in dynamic traffic scenarios. As illustrated in Fig. 9(a), the worst-case performance ratio of TMP is higher than 4 in the Tiscali network with significant performance degradation. In contrast, with the exact same set of training samples, LARRI performs consistently well and is able to generalize to different traffic scenarios in these two networks, which reveals the advantages of routing prediction over TM prediction. Thanks to the scalable GNN-based architecture design, the range routing strategies offered by LARRI can effectively improve routing robustness against dynamic traffic fluctuations in large networks to avoid severe network congestion.

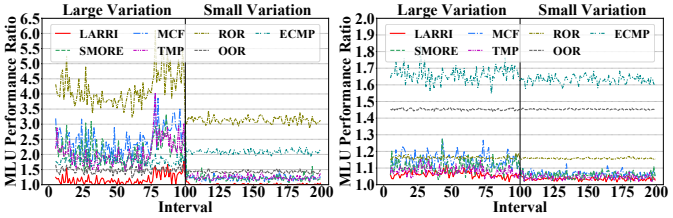
F. Training and Inference Time

In our evaluation, the training time of LARRI depends on the size of the training sample set and the size of the network topology. Before training LARRI, we need to produce a labeled dataset by computing the target routing strategies with our proposed path-based range routing LP. As shown in Table III, it could require more than 30 minutes to solve the target LP in the Google Cloud network. Since all training costs are incurred offline, we can utilize powerful machines and compute these targets in parallel to accelerate the process.

[3]Due to the high computation complexity, COPE is unable to compute a routing within a reasonable timescale, which is omitted from the results.

TABLE III: Computation time of different TE solutions

Topology	LARRI	MCF	SMORE	COPE	TMP	Target LP	Original
BRAIN	0.5s	0.2s	0.1s	2.3m	0.1s	2.1s	2.2s
Abilene	1.0s	0.3s	0.2s	34.6s	0.3s	1.6s	2.3s
CERNET	1.1s	0.4s	0.1s	1.3m	0.2s	1.7s	2.7s
GÉANT	1.2s	1.5s	0.3s	6.9h	0.5s	8.9s	4.5m
Tiscali	1.6s	5.6s	0.5s	>24h	2.3s	18.9m	6.7h
Google	1.9s	16.5s	1.3s	>24h	2.3s	36.5m	>24h



(a) Tiscali network. (b) Google Cloud network.

Fig. 9: Load balancing performance comparison in the Tiscali and Google Cloud networks. COPE is omitted due to scalability issues.

Once the labeled dataset is generated, we can train a LARRI model and use early stopping [45] to halt the training process when the prediction loss stops improving on the standalone validation set. For each of the six networks, it takes less than 40 minutes to train a LARRI model with a single GPU Tesla V100. As a result, our scalable GNN-based model design has greatly facilitated the training process.

For online deployment, Table III shows the computation costs of LARRI and baseline methods in all six networks. The running time is measured on a Linux server with a 4-core Intel 2.9 GHz CPU and the Gurobi optimizer [50], as described in Section V-A3. Note that the static baseline methods are not included since they do not perform routing updates. For LARRI, the routing prediction (inference) time in all six networks is less than 2 seconds. Other schemes can also perform routing updates within a reasonable timescale in large networks, except COPE and the original range routing LP model. As network size increases, the number of variables and constraints in COPE's LP formulation grows exponentially. Given the high computation overhead, COPE cannot compute a routing within 24 hours for the two large networks in our evaluation. The original range routing LP model proposed in [22] also suffers from scalability issues. For example, it takes 6.7 hours to solve a routing optimization problem in the Tiscali network, and its computation time for the Google Cloud network is even longer than 24 hours. By formulating the range routing problem with preconfigured paths, our proposed path-based LP model in (7) can greatly reduce the computation time to 18.9 minutes and 36.5 minutes in the Tiscali and Google Cloud networks, respectively. However, such routing computation overhead is still a major obstacle to directly applying this LP model in large networks with 5-minute routing update intervals. In contrast, LARRI is able to predict a routing strategy for large networks at second-level to accommodate future traffic variations.

VI. Related Work

There has been a large body of literature on TE. Quite a lot of traditional TE solutions [1], [2], [57] rely on Interior Gateway Protocols (IGPs) to route network traffic. With the emergence of SDN [28], TE can deploy flexible routing policies and improve network performance in a responsive manner. Google [11] and Microsoft [12] design and deploy SDN-based centralized TE systems to achieve high utilization in their inter-datacenter WANs, where TE operations are usually performed at 5-minute intervals to cope with traffic changes [18]. However, these solutions might suffer from routing performance degradation due to highly dynamic traffic fluctuations within the routing update intervals.

One way to handle unexpected traffic fluctuations is adopting oblivious routing [21], [22], where the routing is oblivious to the actual traffic demands. Applegate and Cohen [22] formulate an LP problem for optimal oblivious routing to provide promising performance bounds for all possible TMs. These solutions usually do not require routing updates while providing a strong worst-case performance guarantee, but their performance would be compromised when traffic is stable [22]. To address this issue, COPE [10] primarily optimizes routing for a convex hull of past TMs and bounds the worst-case performance for unexpected traffic deviations. However, without the knowledge of future traffic variations, COPE has to expand the convex hull with sufficient past TMs and inevitably sacrifice performance optimality for robustness.

By leveraging the agility of ML techniques, quite a few TM prediction methods [23]–[26] have been proposed in recent years to handle future traffic variations. Given a series of past TMs, they predict the next TM by capturing the temporal and spatial relations in the traffic traces over the time periods. Based on the predicted TM, routing decisions are made to accommodate future traffic variations. However, as we discussed before, a single predicted TM is insufficient to represent fine-grained traffic variations in a given time interval. In addition, TM prediction-based TE could be sensitive to TM prediction errors with unstable performance [24], [25].

VII. Conclusion

In this paper, we propose LARRI, a learning-based TE that is trained to predict appropriate range routing strategies for accommodating dynamic future traffic scenarios. To simplify the complexity of the prediction model, we customized a scalable GNN-based architecture for LARRI to model network topologies and characterize traffic demands, which substantially reduces the time for training and inference. Extensive experiments show that LARRI significantly improves the routing robustness and performs consistently well under different traffic scenarios in terms of load balancing performance and end-to-end delay performance.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2021YFB1714800, National Natural Science Foundation of China under Grant 62002019, Zhejiang Lab Open Research Project under Grant K2022QA0AB02, SongShan Laboratory Fund under Grant YYJC022022009, and Beijing Institute of Technology Research Fund Program for Young Scholars.

REFERENCES

[1] 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.

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

[3] 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.

[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 *USENIX NSDI*, 2018, pp. 157–170.

[5] E. D. Osborne and A. Simha, *Traffic engineering with MPLS*. Cisco Press, 2003.

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

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

[8] 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.

[9] Z. Guo, W. Chen, Y.-F. Liu, Y. Xu, and Z.-L. Zhang, "Joint switch upgrade and controller deployment in hybrid software-defined networks," *IEEE JSAC*, vol. 37, no. 5, pp. 1012–1028, 2019.

[10] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "Cope: Traffic engineering in dynamic networks," *ACM SIGCOMM CCR*, vol. 36, no. 4, p. 99–110, Aug. 2006.

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

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

[13] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. Maltz, "Latency inflation with mpls-based traffic engineering," in *ACM IMC*, 2011, pp. 463–472.

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

[15] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple network management protocol (snmp)," *IETF RFC 1157*, 1990.

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

[17] 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.

[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] S. Tomovic and I. Radusinovic, "Ro-ro: Routing optimality-reconfiguration overhead balance in software-defined isp networks," *IEEE JSAC*, vol. 37, no. 5, pp. 997–1011, 2019.

[20] 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.

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

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

[23] A. Azzouni and G. Pujolle, "Neutm: A neural network-based framework for traffic matrix prediction in sdn," in *IEEE/IFIP NOMS*, 2018, pp. 1–5.

[24] K. Gao *et al.*, "Incorporating intra-flow dependencies and inter-flow correlations for traffic matrix prediction," in *IEEE/ACM IWQoS*, 2020, pp. 1–10.

[25] Z. Liu *et al.*, "Traffic matrix prediction based on deep learning for dynamic traffic engineering," in *IEEE ISCC*, 2019, pp. 1–7.

[26] S. Troia, R. Alvizu, Y. Zhou, G. Maier, and A. Pattavina, "Deep learning-based traffic prediction for network optimization," in *ICTON*, 2018, pp. 1–4.

[27] Global locations of google cloud topology. [Online]. Available: https://cloud.google.com/about/locations/

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

[29] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, vol. 30, 2017.

[30] P. Veličković *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[31] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[32] SNDlib. [Online]. Available: http://sndlib.zib.de/home.action

[33] D. Mitra and K. G. Ramakrishnan, "A case study of multiservice, multipriority traffic engineering design for data networks," in *IEEE GLOBECOM*, vol. 1b, 1999, pp. 1077–1083.

[34] H. Racke, "Minimizing congestion in general networks," in *IEEE FOCS*, 2002, pp. 43–52.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[37] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, vol. 30, 2017.

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

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

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

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

[42] 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.

[43] 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.

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

[45] L. Prechelt, "Early stopping — but when?" in *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 53–67.

[46] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[47] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *NeurIPS*, 1991, p. 950–957.

[48] 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.

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

[50] Gurobi. [Online]. Available: https://www.gurobi.com/

[51] NetworkX. [Online]. Available: https://networkx.org/

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

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

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

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

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

[57] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current ospf/is-is networks," in *IEEE INFOCOM*, vol. 2, 2003, pp. 1167–1177.