This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3293098

Prophet: Traffic Engineering-centric Traffic Matrix Prediction

Yuntian Zhang, Ning Han, Tengteng Zhu, Junjie Zhang, *Member, IEEE*,
Minghao Ye, Songshi Dou, and Zehua Guo, *Senior Member, IEEE, Member, ACM*

***Abstract*—Traffic Matrix (TM), which records traffic volumes among network nodes, is important for network operation and management. Due to cost and operation issues, TMs cannot be directly measured and collected in real time. Therefore, many studies work on predicting future TMs based on historical TMs. However, existing works are usually accuracy-centric prediction solutions that mainly focus on improving predicting accuracy of flows' sizes (i.e., values of elements in TMs) without considering the practical application of TMs. In this paper, we propose a novel TM prediction solution called Prophet for Traffic Engineering (TE), a typical application for TMs which takes TMs as input to optimize routing. We identify that the critical property (i.e., ratio among elements) in a TM plays an important role in TE's performance. Based on this analysis, we adopt the matrix normalization to maintain the critical property in TMs and customize a TE-centric angle loss function to introduce scale invariance of TMs for capturing the overall relationship error. Different from the element-wise Mean Squared Error (MSE) loss function in accuracy-centric prediction solutions, our proposed TE-centric angle loss function has a clear geometric interpretation, which confines the angle between predicted TM and real TM to zero. Simulation results show that the predicted TMs from Prophet can improve the performance of link-level TE and path-level TE by up to 45.4% and 52.8%, respectively, compared to existing solutions.**

I. Introduction

Traffic Matrix (TM) is important for network operation and management. A TM usually records traffic volumes between all pairs of Source Destination (SD) nodes in a network for a given period of time slot (e.g., 5 mins). TMs are essential for network operators to optimize network performance. For example, Traffic Engineering (TE) is a typical network application to reduce link congestion probability in a network. Given the input TMs, TE minimizes the link utilization of the most congested links in the network by effectively routing and rerouting flows to achieve load balancing [1]. However, due to cost and operational issues, TMs usually are not directly measured and collected in real time [2].

Existing studies usually work on accurately predicting future TMs based on historical TMs. We call them accuracy-centric prediction solutions. Typically, traffic demands in backbone networks exhibit a stable trend in a long time. Thus, well-known linear prediction methods are used to predict TMs, such as AutoRegressive Moving Average (ARMA) and AutoRegressive Integrated Moving Average (ARIMA) [3].

Yuntian Zhang, Ning Han, Tengteng Zhu, Songshi Dou, and Zehua Guo are with Beijing Institute of Technology, Beijing 100081, China.
Junjie Zhang is with Fortinet, Inc., Sunnyvale, CA 94086 USA.
Minghao Ye is with New York University, New York City, NY 11201 USA.

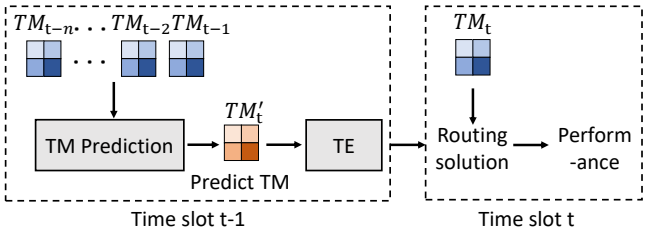


Fig. 1. How to predict TMs and use the predicted TMs for TE. Red squares denote predicted TMs, and blue squares denote real TMs.

However, dynamic traffic fluctuations can also be observed in backbone networks [4]. When TMs change significantly and dynamically, these linear prediction methods do not always work well since they cannot identify and describe some nonlinear features among a series of TMs.

Some recent works use Machine Learning (ML) models as nonlinear prediction methods [5]–[10], but they also cannot precisely predict the exact value of each element[1] in a TM due to random bursty traffic. In some cases, a small difference between a predicted TM and a real TM (e.g., only one element in the predicted TM is different from the one in the real TM) could dramatically degrade TE's performance. For example, a recent work [11] shows that critical flows exist in the networks with a dominant impact on TE's performance, and a small prediction error on critical flows' demand could change the routing in the network significantly and degrade performance.

We argue that focusing on precisely predicting each element's value in TMs does not guarantee to predict TMs well. Instead, predicted TMs should be evaluated based on network performance when using these predicted TMs. Fig. 1 shows how to use predicted TMs for TE, where TM_t denotes the real TM at time slot t, and TM'_t denotes the predicted TM at time slot t. Typically, a TM prediction solution trains a ML model with a loss function. After the training, given a series of historical TMs $\{TM_{t-n}, ..., TM_{t-2}, TM_{t-1}\}$ at time slot $t-1$, TM'_t is generated using the trained ML model and then used by TE to generate a routing solution for flows at time slot t. When TM_t arrives, flows in TM_t are routed following the routing solution, which is previously generated based on predicted TM TM'_t. We can see that the TM prediction is unaware of the downstream TE. If TM_t is different from TM'_t, the prediction error could significantly degrade TE's performance. Thus, the impact of TMs on real application's

[1]In this paper, we use the flow and the element in a TM interchangeably.

performance should be considered when predicting future TMs.

Motivated by the above analysis, in this paper, we propose Prophet, a TM prediction solution for achieving good performance of TE. We find out that the ratio among elements in each TM plays an important role in TE. We call it *critical property*. Critical property reveals that even if the corresponding TM is a n times of the real TM, the performance of downstream TE will stay invariant. In other words, the ML model does not need to be element-wise accurate. Thus, critical property introduces robustness and flexibility in TM prediction. The key of Prophet is to maintain the critical property in each TM when predicting TMs. The details of the critical property are introduced in Section III.

To take advantage of the critical property in TMs, we face two challenges. First, revealing the critical property requires a huge computation overhead due to the nonlinear growth time complexity. A TM usually consists of k^2 elements, where k denotes the number of nodes in a network. We need $\mathcal{O}(k^4)$ calculation to reveal the relationship between any two elements. This is because for each element, its relationship with other (k^2-1) elements should be considered. Since the relationship calculation is required for each iteration during the ML model training, evaluating it could be computationally intractable. Additionally, we have to handle some special cases. For example, some elements could be 0 in a TM. Thus, simply dividing element A by element B to calculate the relationship between A and B does not work if element B is 0. Second, existing accuracy-centric loss functions such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) serve for element-wise error calculation but do not consider the relationship error. The critical property claims the ratio among elements in each TM is important. That is, if a predicted TM is n times of corresponding real TM, it is satisfied by downstream TE. However, MSE and MAE are not sensitive to the ratio among elements. Therefore, design a novel loss function for TM prediction is on the horizon.

Prophet addresses these two challenges with two key techniques: matrix normalization and TE-centric angle loss function design. First, acting as a requisite process in ML, matrix normalization differs from widely used SD pair normalization exampled in Section II. Matrix normalization scales all elements in a TM by dividing the maximum element in this TM, which maintain the ratio among elements. Second, TE-centric angle loss function introduce scale invariance of TMs, which aims to measure the overall relationship error rather than element-wise accuracy-centric error. TE-centric angle loss function gains a clear geometric interpretation, which confines the angle between predicted TM and real TM to zero. Furthermore, scale invariance help improve the robustness in downstream TE, which is observed when compared with conventional methods. The two techniques are detailed in Section IV. Besides, Prophet is a general TM prediction framework, which allows off-the-shelf ML models to plug in easily. Simulation results show that the predicted TMs from Prophet can improve the performance of link-level TE and path-level TE by up to 45.4% and 52.8%, respectively, compared to existing solutions.

To the best of our knowledge, our paper is the first attempt to bridge the gap between TM prediction and downstream application scenarios. It can also inspire other researches to further study the combination of ML and optimization in computer networks. The main contributions of this paper are summarized as follows:

- We propose to predict TMs with consideration of TE's performance and identify the critical property in TMs related to TE's performance.
- We adopt the matrix normalization to maintain the critical property in TMs and customize a TE-centric angle loss function to introduce scale invariance of TMs.

The rest of this paper is organized as follows. Section II introduces the background and motivation of this paper. Section III proposes to identify the critical property in the TM for TE. Section IV describes the design of Prophet. Section V presents and analyzes the simulation results. Section VI discusses some open questions. Section VII recaps the related work, and Section VIII concludes this paper.

II. Background and motivation

In this section, we review previous TM prediction methods and introduce the background of TE. We also illustrate our motivation for the design of Prophet.

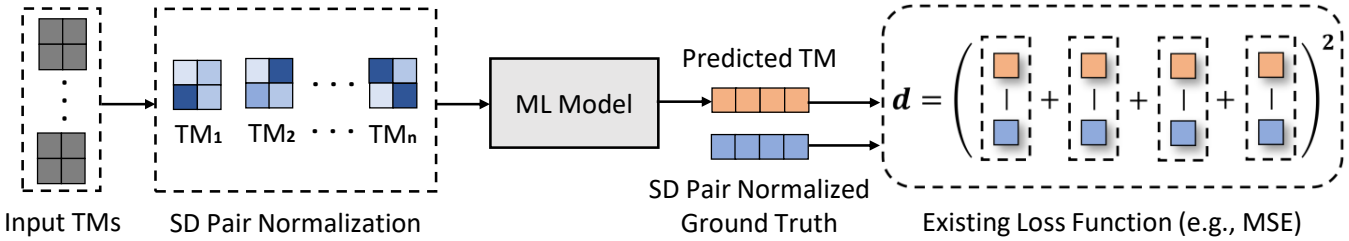
A. TM prediction

Existing works on TM prediction mainly focus on precisely predicting the value of each SD pair in the TM based on historical TMs. Linear prediction models (e.g., ARIMA [3], [12]) can model network traffic, but they cannot tackle the nonlinear feature of network traffic well. Many ML techniques are used to model nonlinear features among a series of TMs and improve prediction accuracy. NeuTM [6] employs Long Short-Term Memory (LSTM) for TM prediction. Troia et al. [7] investigate GRU, a particular type of Recurrent Neural Network (RNN), to predict TMs. A recent work called ACRNN [10] uses Convolution Neural Network (CNN) to capture the inter-flow correlations and RNN to capture the intra-flow dependencies coupled with attention mechanism. These ML-based solutions improve the overall prediction accuracy of TMs. However, precisely predicting each element in a TM remains very difficult. Some elements in the predicted TM could also exhibit great difference from the elements of the same locations in the real TM. Besides, previous methods do not take consideration of the impact on TE optimization into TM prediction.

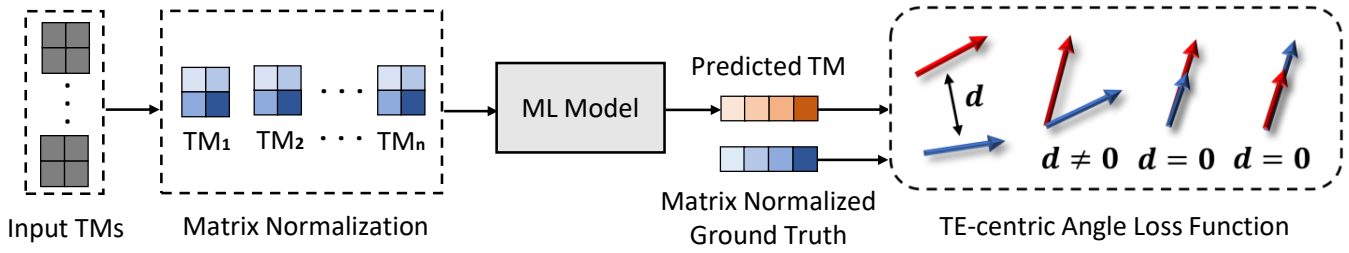
B. TE

Predicted TMs are used for specific network downstream applications to maintain or improve network performance. TE is a major application for TMs and focuses on mitigating congestion of a network by effectively routing and rerouting flows to minimize the Maximum Link Utilization (MLU) in the network. TE is typically formulated as a Linear Programming (LP) problem (e.g., Multi-commodity flow, MCF [13]) and takes TMs as input. The optimization problem can be

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3293098



(a) The training procedure in TM prediction of existing accuracy-centric solutions. In SD Pair Normalization module, each SD pair is divided by the corresponding maximum value of SD pair among all the TMs. Shallower color indicates smaller element value. For instance, the (1,1) element's (i.e., SD pair in first row, first column) maximum value is in TM_n. Loss Function module employs existing loss function (e.g., MSE), which is element-wise and accuracy-centric. Notation d represents loss function.



(b) The training procedure in TM prediction of our proposed Prophet. In Matrix Normalization module, all elements in a TM is scaled by dividing the maximum element in this TM. Besides, a TE-centric angle loss function is customized to introduce scale invariance. Notation d represents loss function. We provide three examples in order. The first example indicates that if there is an angle between flattened predicted TM and real TM, loss function is not equal to zero. Two other examples show that loss function is not sensitive to ratio scale of two TMs, even if they are element-wise different. Both cases $d = 0$ provide a geometric interpretation that our novel loss function introduces scale invariance.

Fig. 2. Comparison of existing accuracy-centric prediction solutions and our proposed Prophet.

optimally solved by existing solvers (e.g., Gurobi Optimizer [14]) or efficiently solved by heuristic algorithms based on the problem's complexity.

TE can be briefly divided into two types: link-level TE by solving MCF problem for arbitrary paths and path-level TE by solving MCF problem for preconfigured paths. Link-level TE presents optimal performance at the cost of high computation complexity. Path-level TE reduces computation complexity and can provide near-optimal performance when preconfigured paths are selected rationally [15]. The formal formulation of the MCF problem for arbitrary paths and for preconfigured paths are detailed in Section III-A.

C. Motivation

Fig. 2 shows the difference of training procedure between existing TM prediction solutions and our proposed Prophet. Fig. 2(a) introduces the framework of existing solutions. Generally, the input TMs are firstly normalized for each SD pair. For example, if a TM contains k SD pairs, then each SD pair is divided by the corresponding maximum value of SD pair among all the TMs. Then, a ML model produces the predicted TM. Finally, a loss function, typically the accuracy-centric MSE is applied to calculate the error between the predicted TM and the ground truth, and the error is backpropagated to train the parameters in the ML model.

Existing solutions mainly focus on designing growing exquisite and complex ML model to precisely predict elements of SD pairs. The normalization and loss function calculation operations are used to predict each SD pair accurately to gain an ideal TM. However, these two operations do not consider the critical property, and thus making the TM prediction unaware of the downstream TE.

In Fig. 2(b), Prophet takes advantage of the critical property in TMs in matrix normalization and a novel TE-centric angle loss function design. First, as the critical property requires ratio scaling of the TM, we adopt the matrix normalization technique. Matrix normalization scales all elements in a TM to $[0, 1]$ by dividing the maximum element in this TM. The normalization process favourably maintains the critical property and provides a requisite part for ML. Second, a TE-centric angle loss function is designed to introduce scale invariance for ML model when predicting TMs. Thus, the overall relationship error could be well measured instead of element-wise accuracy-centric error. What is more, angle loss function has a clear geometric interpretation, which confines the angle between flattened predicted TM and real TM to zero.

To briefly summarize, our general TM prediction framework Prophet covers the shortage of existing solutions by two key techniques, matrix normalization and TE-centric angle loss function. Therefore, Prophet maintains the critical property well when predicting TMs. We also point out that designed matrix normalization techniques and TE-centric angle loss function are simple and easy to deploy in the real world. Additionally, to further improve the performance, Prophet can easily plug in off-the-shelf or other advanced ML models.

III. Identifying the critical property in TMs for TE

In this section, we introduce two typical problem formulation for TE, and analyze how to identify the critical property in TMs for TE.

A. TE problem formulation

1) MCF problem for arbitrary paths: A network is $G = (V, E)$, where V is the set of switches and E is the set of links between switches. T is the traffic matrix. The utilization of link (i, j) ($(i, j) \in E$) cannot exceed its upper bound capacity $C_{i,j}$. We use $x_{i,j}^{s,d}$ to denote the percentage of traffic demand from source s to destination d routed on link (i, j). $v_{s,d}$ is used to denote the traffic demand from source s to destination d in traffic matrix T. u denotes the MLU. The problem P_l is formulated as follows:

$$P_l: \quad \min_{x,u} \quad u \tag{1}$$

$$s.t.$$

$$\sum_{k:(k,i)\in E} x_{k,i}^{s,d} - \sum_{k:(i,k)\in E} x_{i,k}^{s,d} = \begin{cases} -1, \text{ if } i = s \\ 1, \text{ if } i = d \\ 0, \text{ otherwise} \end{cases}, \tag{1a}$$

$$load_{i,j} = \sum_{s,d\in V} x_{i,j}^{s,d} \cdot v_{s,d}, \ \forall (i, j) \in E, \tag{1b}$$

$$load_{i,j} \leq C_{i,j} \cdot u, \ \forall (i, j) \in E, \tag{1c}$$

$$x_{i,j}^{s,d} \in [0, 1], \ \forall s, d \in V, \ \forall (i, j) \in E, \tag{1d}$$

where $\{x_{i,j}^{s,d}\}$ and u are continuous decision variables, $\{T\}$, $\{v_{s,d}\}$ and $\{C_{i,j}\}$ are given constants. Eq. (1) is the objective function that aims at minimizing the MLU. Eq. (1a) ensures that no flow gets lost at any node. Eq. (1b) denotes the traffic load on link (i, j) under traffic matrix T. Eq. (1c) is used to ensure the link capacity constraints and get the MLU u.

2) MCF problem for preconfigured paths: A network is $G = (V, E)$, where V is the set of switches and E is the set of links between switches. T is the traffic matrix. The utilization of link (i, j) ($(i, j) \in E$) cannot exceed its upper bound capacity $C_{i,j}$. The preconfigured path-set for node pair (s, d) is P_{sd}. We use $x_p^{s,d}$ to denote the percentage of traffic demand from source s to destination d routed on path p. And $v_{s,d}$ is used to denote the traffic demand from source s to destination d in traffic matrix T. $\delta_{p,i,j}^{s,d}$ is an indicator constant which denotes if link (i, j) is on path p of pair (s, d). Let u denote the MLU. The problem P_p is formulated as follows:

$$P_p: \quad \min_{x,u} \quad u \tag{2}$$

$$s.t.$$

$$\sum_{p\in P_{sd}} x_p^{s,d} = 1, \ \forall s, d \in V, \tag{2a}$$

$$load_{i,j} = \sum_{s,d\in V} \sum_{p\in P_{sd}} \delta_{p,i,j}^{s,d} \cdot x_p^{s,d} \cdot v_{s,d}, \ \forall (i, j) \in E, \tag{2b}$$

$$load_{i,j} \leq C_{i,j} \cdot u, \ \forall (i, j) \in E, \tag{2c}$$

$$x_p^{s,d} \in [0, 1], \ \forall s, d \in V, \ \forall (i, j) \in E, \tag{2d}$$

where $\{x_p^{s,d}\}$ and u are continuous decision variables, $\{T\}$, $\{v_{s,d}\}$ and $\{C_{i,j}\}$ are given constants. Eq. (2) is the objective function that aims to minimizing the MLU. Eq. (2a) guarantees that the sum of the ratio on each path from one source-destination node pair s, d is equal to 1. Eq. (2b) denotes the traffic load on link (i, j) under traffic matrix T. Eq. (2c) is used to ensure the link capacity constraints and get the MLU u. Preconfigured paths come from SMORE [15]. SMORE is a leading TE solution that achieves load balancing by intelligently splitting traffic on selected paths. These paths are calculated by oblivious routing algorithm, and SMORE deploys a centralized controller to dynamically adjust the split ratios of all flows for each TM based on these preconfigure paths. SMORE utilizes diverse paths for robustness. We employ the SMORE preconfigured paths for the Abilene and Cernet topologies for our work.

To summarize, this paper includes two types of standard TE problem formulations: Multi-commodity flow (MCF) for arbitrary paths, and preconfigured paths. They are used as the downstream optimization tasks. Modern optimization solvers (e.g., Gurobi Optimizer [14]) can already efficiently handle most large scale optimization problems. In our experiments, Gurobi is used to solve the above two LP problems given predicted TMs as input.

B. Metric

TE aims to minimize the MLU in the network by splitting flows among links across the network. Once we know the traffic demand $v_{s,d}$ from source s to destination d in T, the corresponding allocated traffic demand on link (i, j) becomes $\sum_{s,d\in V} x_{i,j}^{s,d} v_{s,d}$. Thus, generating a routing solution for traffic flows is equivalent to determining a set of decision variables $x_{i,j}^{s,d}$ under the previous TE formulation.

We introduce the metric called performance ratio (PR). Denote $\boldsymbol{R}$ as a routing solution on TM $\boldsymbol{T}$. Then

$$PR\{\boldsymbol{R}, \boldsymbol{T}\} = \frac{U_{pred}}{U_{optimal}} \tag{3}$$

where $U_{optimal}$ denotes the MLU of an optimal TE solution with real TMs. U_{pred} denotes the MLU when deploying the routing solution on real TMs, which is generated by the predicted TM. Since U_{pred} is larger than $U_{optimal}$ theoretically, lower PR towards the value 1 means better performance.

C. Identifying the critical property in TMs for TE

Existing works always tightly couple good performance of TE with high prediction accuracy of element values in TMs. However, as we explained before, precisely predicting each element value in TMs is very difficult. Thus, these works lead to bad performance in TE. *Can we find another way to realize good performance for TE without accurately predicting each element in TMs?*

We therefore consider that the TMs may exist some critical properties which play important roles in TE's performance. We apply an analytical method to explore the impact of the critical property in TMs on TE.

Theorem 1 *The performance ratio of the optimal routing solution (i.e., the solution of link-level MCF problem for arbitrary paths) remains invariant for two TMs if the value ratio of each element at the same location of the two TMs is unchanged.*

Proof:

$$PR\{\boldsymbol{R},\boldsymbol{T}\} = \frac{U_{pred}}{U_{optimal}} = \frac{\max\limits_{(i,j)\in E}\sum\limits_{s,d\in V}\frac{x_{i,j}^{s,d}v_{s,d}^{t}}{C_{i,j}}}{\min\limits_{\boldsymbol{R_{best}}\in R_{set}}\max\limits_{(i,j)\in E}\sum\limits_{s,d\in V}\frac{x_{i,j}^{s,d}v_{s,d}^{t}}{C_{i,j}}} \quad (4)$$

where R_{set} indicates all the possible routing solutions, $\boldsymbol{R_{best}}$ is the optimal routing solution corresponding to the optimal MLU.

Once a routing solution is solved, $x_{i,j}^{s,d}$ and $C_{i,j}$ remains invariant. If we scale all the $v_{s,d}^{t}$ in T_t under the same ratio, the PR will stay invariant. Thus, if the corresponding TM is a n times of the real TM, the performance ratio will stay invariant with that of optimal routing related with the real TM. This completes our proof. □

Following the similar method, we can prove that the theorem under MCF problem for preconfigured paths still holds.

To summarize, two typical problem formulation for TE is introduced and the critical property is identified. However, how to maintain the critical property in the ML schemes remains unsolved, which is detailed in the next section.

IV. Design of Prophet

In this section, we discuss design details of the proposed Prophet, a TE-centric TM prediction framework.

A. Design challenges

We face two challenges when making attempts to take advantage of the critical property in TMs.

First, revealing the critical property requires a dense computation overhead due to $\mathcal{O}(k^4)$ time complexity, where k denotes the number of nodes in a network. Evaluating it could be computationally intractable since the calculation is required for each iteration during the ML model training.

Second, existing accuracy-centric loss functions such as MSE and MAE compute for element-wise error calculation but do not consider the overall relationship error. Therefore, they are not sensitive to the critical property (i.e., ratio among elements). Designing a novel loss function which could introduce scale invariance and provide some geometric interpretation is on the horizon.

B. Matrix normalization

Normalization is an requisite process in ML, where the data is either scaled or transformed to make an equal contribution of each feature [16] and help the training algorithms converge faster. Existing accuracy-centric prediction solutions employ SD pair normalization, since they focus on predicting each SD pair accurately to gain an ideal TM. However, SD pair normalization do not consider the critical property, thus making the TM prediction unaware of the downstream TE.

In this paper, we adopt matrix normalization, which scales all elements in a TM to $[0, 1]$ by dividing the maximum element in the TM. Matrix normalization in fact scale TMs proportionally in order to maintain the critical property.

C. TE-centric angle loss function

The critical property reveals the *ratio among elements*. Thus, instead of precisely predicting each element in a TM which is often computationally intractable, we consider to maintain the critical property (i.e., ratio scaling) of the TM. A TE-centric angle loss function is designed due to this motivation.

Suppose we have a vector space (M, d) of TM. (M, d) is an ordered pair where M is a set (i.e., a set of TMs) and d is a metric on M, i.e., a function

$$d : M \times M \to \mathbb{R}$$

such that for any $x, y \in M$, and $\alpha{>}0$, the followings hold:

1) $d(x, y) \geq 0$
2) $d(x, y) = 0 \iff x = \alpha y$
3) $d(\alpha x, y) = d(x, y)$
4) $d(x, \alpha y) = d(x, y)$

where 1) holds for non-negativity. 2) introduces the identity of indiscernibles. Different from traditional metric space, the metric d equals to 0 so long as $x = \alpha y$ (i.e., stretch a vector). However, $d \neq 0$ if there exists a angle between the predicted vector and ground truth vector. 3) and 4) indicates that ratio scalings of a vector do not affect the value of d.

Thus, we can design a few explicit functions which satisfies the above conditions. We select an intuitive and concise one for the training of Prophet, called TE-centric angle loss function. The loss function f_{loss} is defined as

$$f_{loss}(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^T \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2} \quad (5)$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ represent the predicted TM and the corresponding normalized ground truth TM, respectively. $\|\cdot\|_2$ denotes the ℓ^2 norm, also denoted as Euclidean norm. The second term in Eq. (5) is $cos\theta$ in fact, where θ denotes the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Theorem 2 *The proposed TE-centric angle loss function Eq.* (5) *satisfies the following conditions:*

1) $d(x, y) \geq 0$
2) $d(x, y) = 0 \iff x = \alpha y$
3) $d(\alpha x, y) = d(x, y)$
4) $d(x, \alpha y) = d(x, y)$

Proof: We prove these conditions in order.

1) The second term in Eq. (5) is $cos\theta$ (i.e., θ denotes the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$), which ranges from 0 to 1. Thus, $f_{loss}(\boldsymbol{x}, \boldsymbol{y})$ is in the interval of $[0, 1]$.

2) $f_{loss}(\boldsymbol{x}, \boldsymbol{y}) = 0$ derives $cos\theta = 1$. Thus, $\theta = 2k\pi, k \in \mathbb{Z}$, which indicates $\boldsymbol{x}$ and $\boldsymbol{y}$ has the same direction. In other words, stretching one of the vector with a positive ratio α is allowed.

3) Suppose $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ is an n-dimensional TM vector, which flattens a TM by concatenating each row, then

$$\begin{aligned} f_{loss}(\alpha\boldsymbol{x}, \boldsymbol{y}) &= 1 - \frac{\alpha\boldsymbol{x}^T \cdot \boldsymbol{y}}{\|\alpha\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2} \\ &= 1 - \frac{\alpha(x_1, x_2, \ldots, x_n) \cdot \boldsymbol{y}}{\sqrt{(\alpha x_1)^2 + (\alpha x_2)^2 + \cdots + (\alpha x_n)^2}\|\boldsymbol{y}\|_2} \\ &= 1 - \frac{\alpha(x_1, x_2, \ldots, x_n) \cdot \boldsymbol{y}}{\alpha\sqrt{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}\|\boldsymbol{y}\|_2} \\ &= 1 - \frac{\boldsymbol{x}^T \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2} \\ &= f_{loss}(\boldsymbol{x}, \boldsymbol{y}) \end{aligned} \tag{6}$$

4) The proof method is the same as 3). This completes our proof. □

To summarize, the proposed TE-centric angle loss function successfully introduces scale invariance of TMs and provides a good geometric interpretation due to its angle design. Therefore, the loss function maintains the critical property well which leads to good performance for TE.

D. Training and inference

We describe the training and inference of the ML model in Prophet. The training takes the normalized TM dataset as input. Slide windows [5] are used to construct training pairs $(\boldsymbol{\xi}, \boldsymbol{\theta})$. $\boldsymbol{\xi}$ denotes several historical TMs, and $\boldsymbol{\theta}$ denotes the corresponding ground truth. Thereafter, the Back-propagation Through Time (BPTT) algorithm [17] is applied to update the parameters in the ML model with a mini-batch loss. The loss is calculated by our proposed TE-centric angle loss function. Algorithm 1 shows the pseudo-code for the training algorithm.

The inference takes several historical TMs as the input of the trained model, and generate the predicted TM $\boldsymbol{T}$ for TE optimization. We solve the optimization problem in Section III-A with $\boldsymbol{T}$ to get the routing solution $\boldsymbol{R}$. We could also calculate PR for evaluation. After collecting real incoming TM TM_{real}, we deploy the routing solution $\boldsymbol{R}$ on TM_{real} and calculate MLU U_{pred}. Then, we solve the optimization problem in Section III-A again with TM_{real} and achieve MLU $U_{optimal}$. The PR is calculated as $PR\{\boldsymbol{R}, \boldsymbol{T}\} = \frac{U_{pred}}{U_{optimal}}$. We note that inversed normalization of the predicted TM is unnecessary in our research. In the downstream TE optimization, we normalize the capacity of all links with the maximum capacity value. Algorithm 2 shows the pseudo-code for the inference algorithm.

E. More details of Prophet

Rich time-relevant information can be mined among TMs, thus temporal modeling is important. RNN is widely used to extract time-relevant information due to its cyclic connections. However, RNN is not suitable for long-term temporal prediction because of its gradient exploding and vanishing problem [18]. LSTM and GRU are designed to fix these problem. Since GRU can achieve similar performance but with less computation cost compared to LSTM, we choose a one-layer GRU for the temporal modeling.

Algorithm 1: Training Algorithm of Prophet

Input: Matrix normalized TM dataset D, the number of samples in the dataset N, learning rate a, batch size s, the number of historical TMs p
Output: ML model Φ_ψ with parameter ψ

Construct $D\{(\boldsymbol{\xi_i}, \boldsymbol{\theta_i})\}_{i=1}^N$ using slide windows as [5];
Sample mini-batch $D_k \sim D\{(\boldsymbol{\xi_i}, \boldsymbol{\theta_i})\}_{i=1}^N$;
for *each* D_k **do**
  $loss = 0$;
  for $i \leftarrow 1$ **to** N **do**
    // Estimate $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}$ with the ML model Φ_ψ
    $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}} \leftarrow \Phi_\psi(\boldsymbol{\xi_i})$;
    // Calculate angle loss with $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}$ and $\boldsymbol{\theta_i}$
    $loss_i = 1 - \frac{\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}^T \cdot \boldsymbol{\theta_i}}{\|\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}\|_2\|\boldsymbol{\theta_i}\|_2}$;
    $loss \leftarrow loss + loss_i$;
  end
  $loss \leftarrow \frac{1}{s} \times loss$;
  // Update ψ utilizing BPTT algorithms
  $\psi \leftarrow BPTT(\psi; a; loss)$;
end

Algorithm 2: Inference Algorithm of Prophet

Input: ML model Φ_ψ with parameter ψ, p historical TMs $\boldsymbol{\xi}$
Output: Routing solution $\boldsymbol{R}$, PR

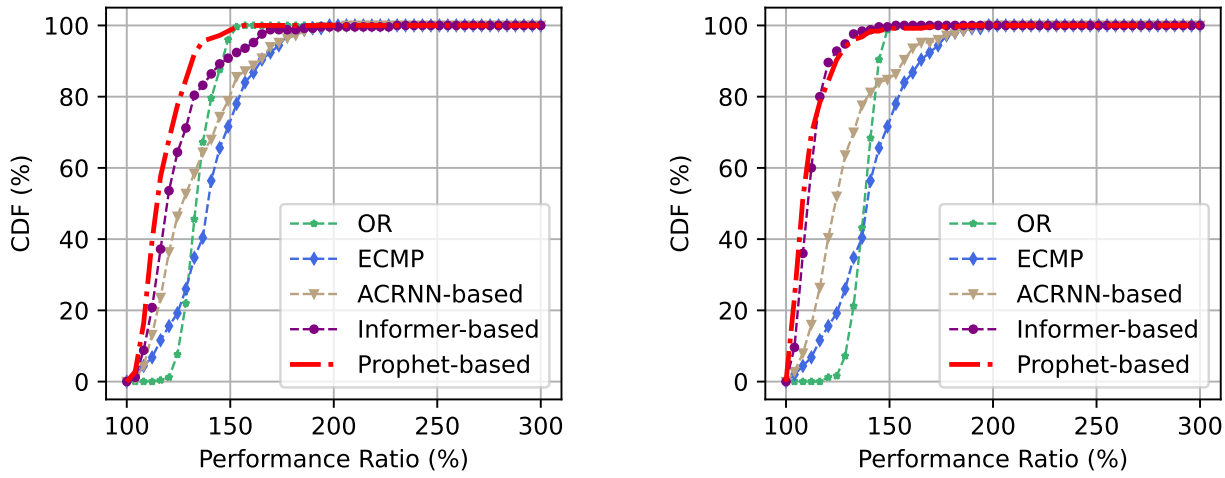
// Take historical TMs as input, and infer incoming TM $\boldsymbol{T}$
$\boldsymbol{T} = \Phi_\psi(\boldsymbol{\xi})$;
// Solve routing solution $\boldsymbol{R}$
Solve optimization problem in Section III-A with $\boldsymbol{T}$ to get the routing solution $\boldsymbol{R}$;
// Calculate PR metric for further evaluation
After collecting real incoming TM TM_{real}, deploying the routing solution $\boldsymbol{R}$ on TM_{real} and calculate MLU U_{pred};
Solve optimization problem in Section III-A with TM_{real} and achieve MLU $U_{optimal}$;
$PR\{\boldsymbol{R}, \boldsymbol{T}\} = \frac{U_{pred}}{U_{optimal}}$;

We define the flattened vector for the historical TM i as x_i. These vectors flatten the corresponding TMs by concatenating each row. Then, x_i is forwarded into the GRU unit to generate the corresponding hidden state vector h_i:

$$h_i = f\left(x_i, h_{i-1}; \Theta\right) \tag{7}$$

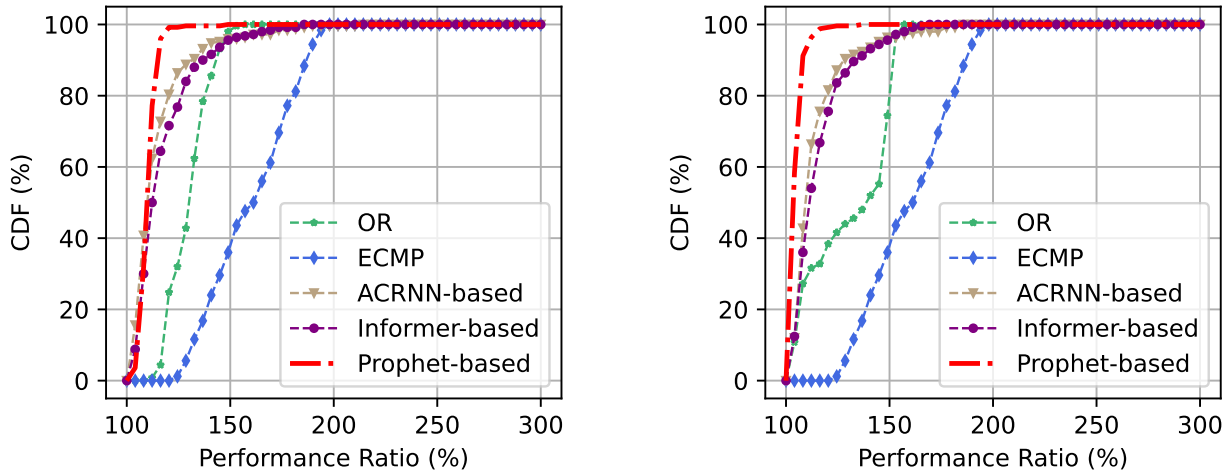
where the hidden state of the last time step is outputted as the predicted TM. The hidden state of the GRU is initialized with all zero elements. Θ denotes the parameters of GRU. The f here is a nonlinear function. We set f as Rectified Linear Unit (ReLU) in our simulation. The output elements pass a ReLU function to keep the values positive.

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3293098



(a) Abilene Link-level TE (b) Abilene Path-level TE

Fig. 3. Performance ratio for 250 TMs of Abilene under link-level TE and path-level TE.



(a) CERNET Link-level TE (b) CERNET Path-level TE

Fig. 4. Performance ratio for 250 TMs of CERNET under link-level TE and path-level TE.

TABLE I
COMPARISON ON THE AVERAGE OF PERFORMANCE RATIO.

Topology	Formulation	Prophet-based	Informer-based	ACRNN-based	ECMP	OR
Abilene	path-level TE	111.2%	112.0%	128.3%	139.9%	137.5%
	link-level TE	117.2%	124.2%	132.5%	139.9%	134.3%
CERNET	path-level TE	104.4%	115.4%	113.9%	159.5%	130.7%
	link-level TE	110.0%	117.2%	114.5%	159.5%	129.7%

V. EVALUATION

In this section, we present and analyze Prophet's performance with extensive simulations from three aspects:

A1 We show Prophet outperforms existing TE solutions under two TE scenarios. We note that Prophet with a simple ML model (i.e., a one-layer GRU) performs better than the exquisite ML models under conventional schemes and achieves lower computational cost. Thus, we verify the value of the critical property.

A2 We conduct experiments to compare and reveal that Prophet can efficiently enhance off-the-shelf ML models. This verify that Prophet is a general TM prediction framework allowing off-the-shelf ML models to plug in easily.

A3 It is both interesting and important to visualize the critical property in Prophet. Thus, we finish a complete process of identifying, maintaining and unveiling the critical property.

A. Simulation Settings

1) Datasets and data preparations: Two real-world WAN traffic datasets are used in our simulation: Abilene [19] and the China Education and Research Network (Cernet). Abilene topology has 12 nodes and 30 links, and its TMs are collected every 5 minutes and last for 24 weeks. Cernet topology has 14 nodes and 32 links, and its TMs are collected every 5 minutes and lasts for 5 weeks. We split all the instances into training instances, validation instances and test instances with the ratio of 6 : 2 : 2.

2) Configurations of the experiments: Prophet is implemented in PyTorch [20] and Gurobi [14]. In the simulation, we employ a one-layer GRU. The hidden size of GRU is set as 144 in Abilene topology and 196 in Cernet topology. The number of historical TMs is set as 5. The optimizer is Adam [21] with learning rate set as 0.01, and the batch size is 64. The training process runs for 100 epochs. All experiments are done on a computer with an Intel Xeon E5-2620 CPU, one Nvidia TITAN X GPU and 16GB memory.

B. Comparison Schemes

We evaluate Prophet's predicted TMs under dynamic TE solutions, and compare the TE's performance with four TE solutions under two TE scenarios. The details of the compared methods are as follows.

1) TE scenarios:

- **Link-level TE**: it solves MCF problem for arbitrary paths in Eq. (1) to decide the split ratio of each flow on links and generate the MLU.
- **Path-level TE**: it solves MCF problem for preconfigured paths in Eq. (2) to decide the split ratio of each flow on preconfigured paths and generate the MLU. Preconfigured paths come from SMORE [15].

2) TM prediction + Dynamic TE solutions: Dynamic TE solutions solves an optimization problem, which requires the predicted TM as the input. Thus, we introduce three TM prediction schemes in the experiment.

- **Prophet-based**: our proposed Prophet is a TE-centric TM prediction solution. It notes the matrix normalization and customize a TE-centric angle loss function to maintain the critical property of TMs, which helps achieve good performance for TE.
- **ACRNN-based** [10]: ACRNN is a state-of-the-art (SOTA) work for TM prediction. It focuses on spatial-temporal prediction which uses CNN to capture the inter-flow correlations and RNN to capture the intra-flow dependencies coupled with the attention mechanism. Parameters of compared ACRNN are well-tuned to achieve the performance reported in [10].
- **Informer-based** [22]: Informer firstly solves Long Sequence Time-series Forecasting (LSTF) problem with Transformer and three efficient techniques: ProbSparse self-attention mechanism, self-attention distilling, and generative style decoder. Informer provides a prediction model with high capacity, enabling to capture precise long-range dependency coupling among TMs.

3) Static TE solutions:

- **Equal-Cost Multi-Path (ECMP)**: ECMP reduces the congestion probability by equally splitting each flow on its shortest path(s).
- **Oblivious Routing (OR)**: OR computes a routing scheme with respect to all possible TMs to guarantee the worst-case performance without knowledge of actual traffic demands. We employ optimal OR [23] for link-level TE and Räcke's OR [24] for path-level TE, respectively.

TABLE II
CLOCK TIME AND MODEL PARAMETERS.

Topology	ML model	Clock time	Parameters
Abilene	Prophet	0.35 h	125 k
	ACRNN	1.20 h	1,644 k
	Informer	0.10 h	3,682 k
CERNET	Prophet	0.27 h	232 k
	ACRNN	0.95 h	1,823 k
	Informer	0.08 h	5,253 k

C. Performance Comparison (*A1*)

For both datasets, we split all the data as Section V-A1 mentions. Without loss of generality, we take the first 250 matrices in the test set to conduct the simulations. The comparison among Prophet-based, Informer-based, ACRNN-based, ECMP and OR is shown in Figs. 3, 4, and Tables I, II.

The performance of TE is evaluated using performance ratio (PR) $PR = \frac{U_{pred}}{U_{optimal}}$ introduced in section III-B. $U_{optimal}$ denotes the MLU of an optimal TE solution with real TMs. U_{pred} denotes the MLU when deploying routing solutions on real TMs, which are generated using predicted TMs or a static TE solution. If $PR = 1$, routing performance using predicted TMs is as good as using real TMs. Lower PR means better performance.

Fig. 3 presents the PR under link-level TE and path-level TE of Abilene. For both scenarios, Prophet-based solution outperforms Informer-based solution and ACRNN-based solution, since it maintains the critical property for TE. In Fig. 3(a), Prophet improves the performance of ECMP and OR by 19.4% and 14.6%, respectively. Though OR can guarantee the worst-case performance, Prophet-based solution performs better between the PR value 100% and 150% (i.e., common-case performance). In Fig. 3(b), Prophet-based solution outperforms the other four solutions, since we can observe that about 90% PR values of Prophet-based solutions are lower than 125%.

Fig. 4 shows the PR under link-level TE and path-level TE of Cernet. Prophet-based solution outperforms the ECMP and OR in both TE scenarios. We note that Informer-based and ACRNN-based solutions achieve competitive results compared to ECMP and OR, but are dominated by Prophet-based solution in both figures. Furthermore, nearly all of the PR values in Prophet-based solution are lower than 125%. These impressive results again reflect the important role of the critical property in TE and verify the effectiveness of the design of the Prophet.

What is more, Table II compares the cost of Prophet, ACRNN and Informer via training clock time and ML model

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3293098

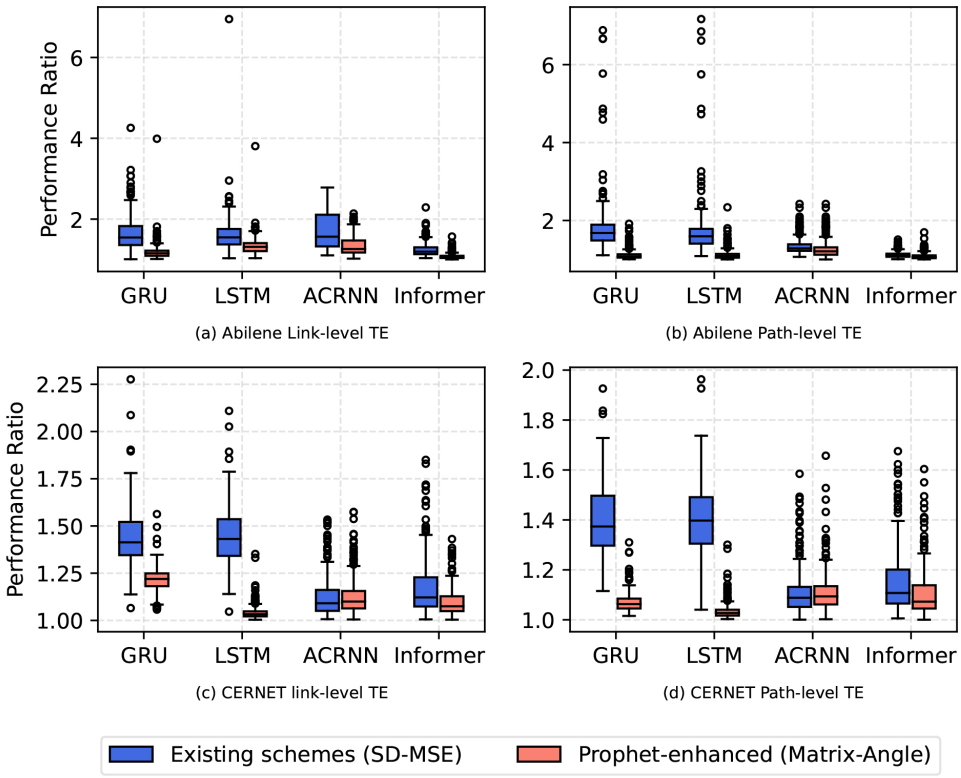


Fig. 5. Comparison of existing schemes (SD-MSE) and prophet-enhanced (Matrix-Angle).

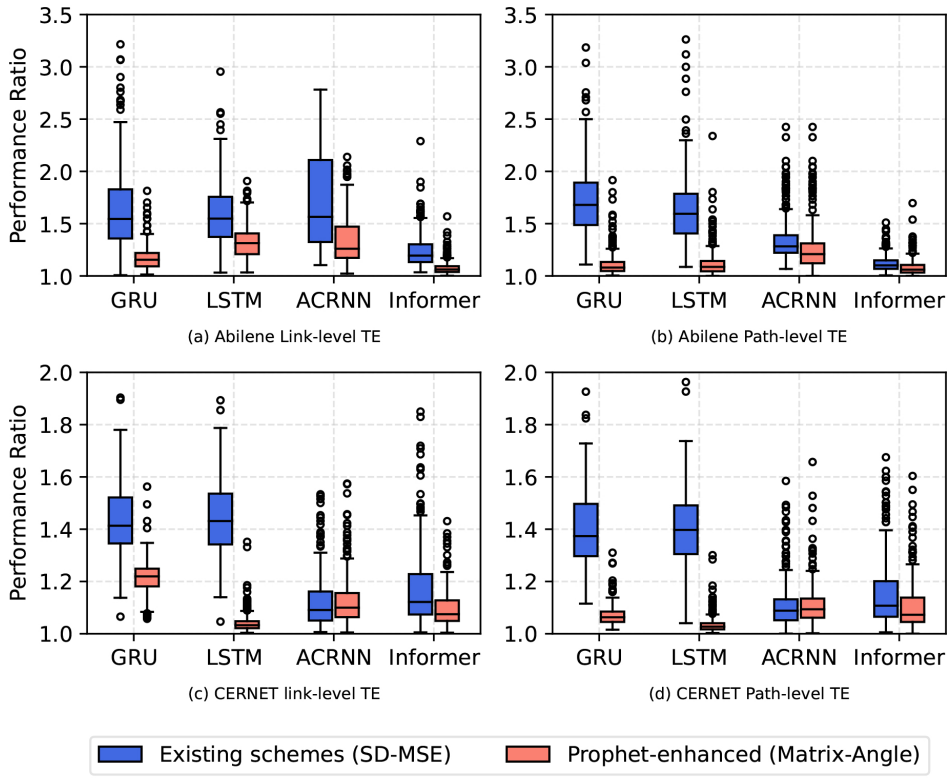


Fig. 6. Comparison of existing schemes (SD-MSE) and prophet-enhanced (Matrix-Angle). The performance ratio is truncated at 3.5 in Abilene and 2.0 in CERNET for clarity.

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3293098

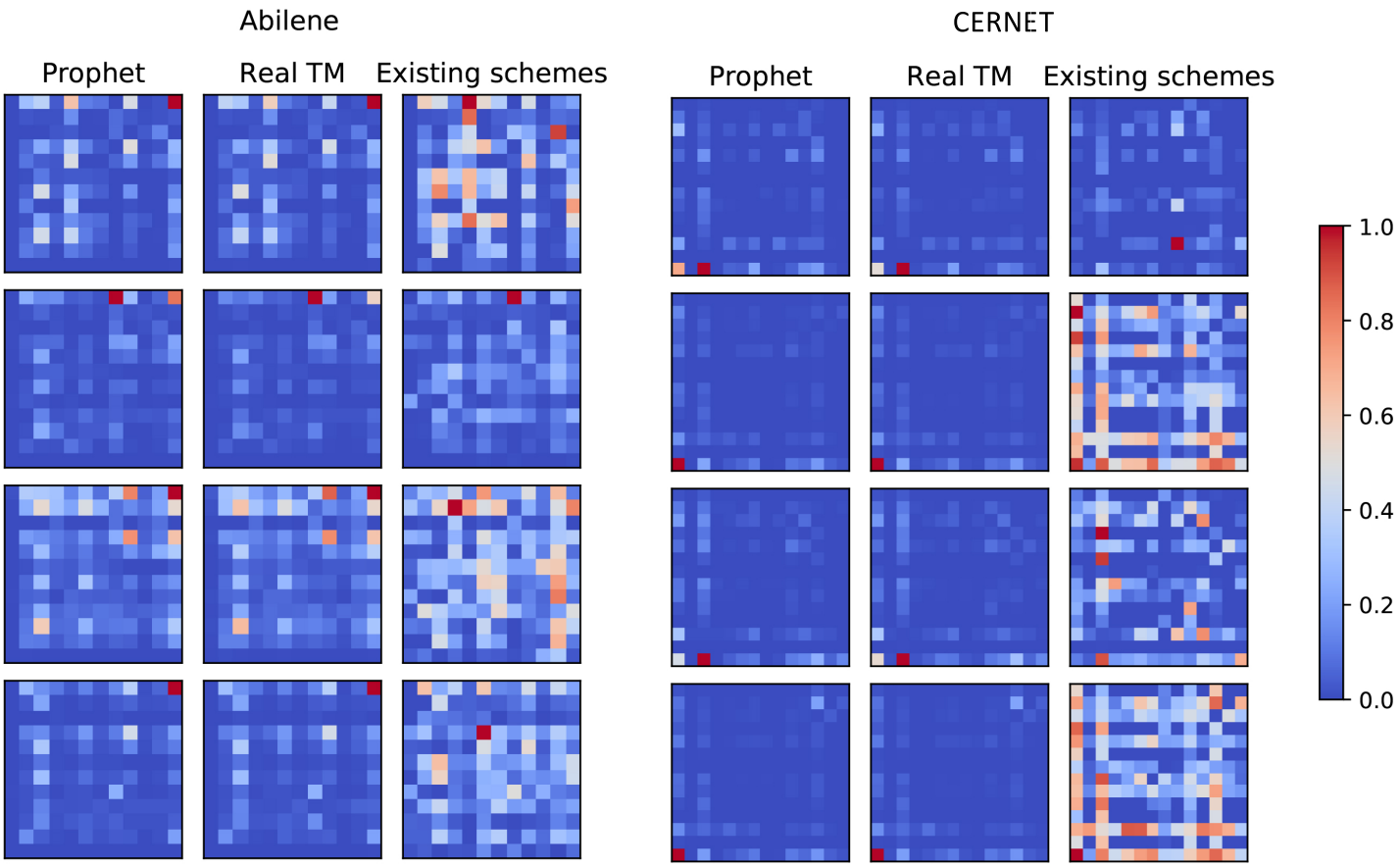


Fig. 7. Heatmaps of TMs. Each heatmap is a $k \times k$ matrix representing $k \times k$ SD pairs in a TM. The color of each grid represents the value of each SD pair normalized by the maximum value in the TM. Thus, the value of each grid is between 0 and 1. In each row, the middle, left, and right figures represent heatmap of real TM, predicted TM using Prophet, and predicted TM using existing schemes, respectively.

parameters, all evaluated on one Nvidia TITAN X GPU. We can observe that under both topologies, Prophet saves much more clock time and model parameters than ACRNN. Since the above discussions have shown Prophet's outstanding performance, we conclude that it is important and valuable to maintain the critical property. We also remark that the training clock time of Informer is less than Prophet because its transformer-based [25] architecture can benefit more from parallel acceleration compared with the backbone GRU utilized in Prophet.

*D. Enhance off-the-shelf ML model (**A2**)*

We re-emphasize that Prophet aims to maintain the critical property in TE and acts as a general framework shown in Fig. 2. That is, existing off-the-shelf TM prediction solutions in TE can be enhanced by Prophet. In this section, extensive simulations are conducted to show that Prophet indeed enhances off-the-shelf ML model.

- **Existing schemes (SD-MSE)**: in this section, existing TM prediction schemes corresponds to Fig. 2(a). To be more specific, a ML model is coupled with SD pair normalization + MSE loss function (SD-MSE).
- **Prophet-enhanced (Matrix-Angle)**: in this section, Prophet-enhanced corresponds to Fig. 2(b). That is, a ML model is coupled with matrix normalization + TE-centric angle loss function (Matrix-Angle).

As [6], [9], [10] reported, GRU and LSTM shows relatively dominant results in TM prediction. In the simulations, we choose GRU, LSTM, ACRNN and Informer as the ML models. For each model, we compare their corresponding TE performance under existing schemes (SD-MSE) and Prophet-enhanced (Matrix-Angle). Simulations results are shown in Fig. 5 and 6.

Fig. 5 shows the boxplot TE results of existing schemes (SD-MSE) and prophet-enhanced (Matrix-Angle). In two network topologies under two TE scenarios, Matrix-Angle significantly overcomes SD-MSE, which again indicates the effectiveness of maintaining the critical property. Moreover, the results verify that Prophet can efficiently enhance off-the-shelf ML models.

It is worth mentioning that in the Abilene topology, we observe a bit of performance drop under unexpected scenarios. That is, a few values of the performance ratio are over 4.0. Firstly, we point out that Matrix-Angle has significantly controlled these unexpected scenarios since there is more performance drop under SD-MSE. Secondly, we have found that Prophet shows promising results in the common-case performance.

E. Unveil the Critical Feature (*A3*)

It is both interesting and important to visualize the critical property in Prophet. Thus, we finish a complete process of identifying the critical property, maintaining it, and unveiling it.

Fig. 7 reveals the heatmaps of TMs. The critical property claims that ratio among elements is important for TE. The heatmap stays unchanged if we ratio scale the corresponding TM. Thus, if heatmaps of real TM and Prophet are similar, we can indicate the critical property is held.

Fig. 7 shows that in all 8 randomly selected cases (i.e., 4 of Abilene, and 4 of Cernet), the heatmaps of real TM and Prophet are quite similar. On the contrary, the heatmaps of existing schemes seems unsatisfactory and often fails to capture the hot-spots. Though exisitng schemes may achieve higher element-wise prediction accuracy under traditional loss such as MSE and MAE, they cannot maintain the critical property well.

VI. Discussion

Advanced ML model: In the experiment A1, we only adopts a one-layer GRU as the ML model. There are two reasons. Firstly, design of complex and exquisite ML model is not the focus of this paper. Secondly, a simple ML model can give prominence to the effectiveness and importance of the critical property.

Despite this, a well-designed ML model is also valuable. Fortunately, emerging ML techniques provide us new opportunities to further improve the performance of common-cases in TE. Encoder-decoder framework is naturally suitable for sequence modelling and has witnessed great success in tasks such as NLP [26] and time series prediction [27]. Moreover, rising Transformer [25] dominants not only the NLP tasks, but also many vision tasks [28]–[30]. We will consider introducing the exquisite attention mechanism in Transformer to better maintain the critical property and serve for larger network. To tackle some practical issues of resources, the trade-off between performance and computation cost should also be studied.

Cope with COPE: The epoch-making Common-case Optimization with Penalty Envelope (COPE) [4] studies how to cope with dynamic and unpredictable changes in traffic demand. COPE points out that prediction-based TE is good at optimizing common-case performance, but suffers from large performance penalty when traffic demands change significantly. Besides, OR can guarantee the worst-case performance but is weak in common-case. Motivated by above analysis, COPE design TE algorithms that optimize for the expected scenarios while providing a worst-case guarantee for unexpected scenarios.

Though focusing on maintaining the critical property, our proposed Prophet belongs to prediction-based TE solution. Prophet significantly goes a step further and promotes common-case performance as shown in the evaluation. However, due to the inherent feature of prediction-based schemes, we observe a bit performance drop under unexpected scenarios in Fig. 5. Thus, we desire to further guarantee the worst-case for these unexpected scenarios. To some disappointment, COPE is theoretically derived under convex hull of traffic demands, since prediction methods at that time essentially estimate the TM of the next interval as a convex combination of the previously seen TMs (e.g., exponential moving average). In recent years, NN-based solutions, which are highly nonlinear, claim to cope with COPE and provide theoretical guarantee for a class of highly nonlinear solutions.

Retraining: In this paper, we mainly describe the training process of Prophet as an offline task, similar to prior works. However, in practice, the TMs tend to be seasonal and bursty, making online deployment challenging. Such online deployment can employ a rolling-based slide window to reorganize the training instances. Prophet utilizes the new training instances to train a new ML model, in order to capture new trends and features of incoming TMs. Therefore, techniques to determine the frequency to retrain and which TM data should be included or excluded from the training instances are required to be carefully investigated.

VII. Related Work

In the past decade, TM prediction has been widely studied. The existing TM prediction methods can be divided into two categories: linear method and nonlinear method. For linear method, Autoregressive Integrated Moving Average (ARIMA) is used in [3] to model network traffic. [12] employs SARIMA, a generalization of the ARIMA model to predict TM while taking account of the short-term traffic variation in order to accommodate prediction uncertainty incurred by temporal traffic changes and prediction errors. [31] proposes three TM prediction methods in the IP backbone network: Independent Node Prediction (INP), Total Matrix Prediction with Key Element Correction (TMP-KEC) and Principle Component Prediction with Fluctuation Component Correction (PCP-FCC). It uses ARIMA and PCA as main algorithms. However, these linear methods cannot handle the nonlinear nature of network traffic.

With the rapid development of Neural Network (NN) in recent years, NN based methods have been used in TM prediction as nonlinear solutions, which are proved to achieve better performance than linear methods. [5] propose a network traffic estimation method utilizing the Deep Belief Network (DBN) via link counts and routing information in large-sacle IP backbone networks. [6] presents NeuTM, a framework for network TM prediction based on LSTM. [7] investigates a particular type of RNN, the GRU to make prediction of traffic matrices, which is able to achieve great accuracy. [8] investigates RNN, GRU and LSTM to solve the network traffic prediction problem. Zhao et al. [9] propose a novel TM prediction approach based on deep LSTM and a linear regression model. Evaluation shows it can achieve great TM prediction performance on Abilene. A recent work called ACRNN [10] is designed for TM prediction, which uses CNN to capture the inter-flow correlations and RNN to capture the intra-flow dependencies.

VIII. Conclusion

In this paper, we propose Prophet, a novel TE-centric TM prediction solution. We claim to consider TE's performance

while predicting TMs and identify the critical property in the TM. Inspired by the analysis, we adopt the matrix normalization to maintain the critical property and customize a TE-centric angle loss function to introduce scale invariance of TMs. Simulation results show that Prophet can achieve supreme performance in TE compared to existing solutions. Extensive evaluations also reveal that Prophet can efficiently enhance off-the-shelf ML models.

REFERENCES

[1] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Load balancing for multiple traffic matrices using sdn hybrid routing," in *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2014, pp. 44–49.

[2] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic matrices: balancing measurements, inference and modeling," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005, pp. 362–373.

[3] W. Jin, "A process level network traffic prediction algorithm based on ARIMA model in smart substation," in *IEEE International Conference on Signal Processing*, 2013.

[4] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2006, pp. 99–110.

[5] L. Nie, D. Jiang, L. Guo, and S. Yu, "Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks," *Journal of Network and Computer Applications*, vol. 76, pp. 16–22, 2016.

[6] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in SDN," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–5.

[7] S. Troia, R. Alvizu, Y. Zhou, G. Maier, and A. Pattavina, "Deep learning-based traffic prediction for network optimization," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, 2018, pp. 1–4.

[8] N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 187–193.

[9] J. Zhao, H. Qu, J. Zhao, and D. Jiang, "Towards traffic matrix prediction with LSTM recurrent neural networks," *Electronics Letters*, vol. 54, no. 9, pp. 566–568, 2018.

[10] K. Gao, D. Li, L. Chen, J. Geng, F. Gui, Y. Cheng, and Y. Gu, "Incorporating intra-flow dependencies and inter-flow correlations for traffic matrix prediction," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–10.

[11] 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.

[12] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, and K. Shiomoto, "Traffic prediction for dynamic traffic engineering," *Computer Networks*, vol. 85, no. jul.5, pp. 36–50, 2015.

[13] T. C. Hu, "Multi-commodity network flows," *Operations research*, vol. 11, no. 3, pp. 344–360, 1963.

[14] "L. Gurobi Optimization, "Gurobi optimizer reference manual", 2019. [Online]." Available: http://www.gurobi.com.

[15] 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.

[16] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.

[17] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in neural information processing systems (NIPS'19)*, 2019.

[21] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR'15)*, 2015.

[22] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[23] D. Applegate and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03, 2003.

[24] 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

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[26] K. Cho, B. van Merrienboer, Çaglar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[27] H. Hu and X. He, "Sets2Sets: Learning from Sequential Sets with Neural Networks," in *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2019, pp. 1491–1499.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[29] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck transformers for visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 16 519–16 529.

[30] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 558–567.

[31] L. O. W. D. W. Liu, A. Hong and G. Zhang, "Prediction and Correction of Traffic Matrix in an IP Backbone Network," in *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC).*, 2014.