Achieving Multi-Time-Step Segment Routing via Traffic Prediction and Compressive Sensing Techniques

Van An Le, Yusheng Ji, *Fellow, IEEE*, Huu Huy Tran, Phi Le Nguyen, *Member, IEEE*, and John C. S. Lui, *Fellow, IEEE*

***Abstract*—Traffic engineering (TE) is one of the most critical issues in networking, as it enables efficient and reliable network operations. With the advent of Machine Learning (ML) techniques, many ML-based TE methods have emerged in recent years, especially those employing Deep Neural Networks for future traffic prediction to enhance the performance of traditional approaches. However, current methods suffer from two major issues. Firstly, most prior works only solve the TE problem based on short-term traffic prediction, neglecting the network traffic dynamics over an extended time period. This oversight results in high network disturbance when numerous traffic flows need to be rerouted to adapt to traffic changes. Secondly, although traffic prediction models rely on historical traffic data to perform future prediction, ML-based TE studies often ignore the high overhead for network traffic monitoring. To address these issues, we propose a traffic prediction-based routing algorithm in which the routing rules can be applied to multiple time-steps without requiring changes, ultimately leading to reduced network disturbance. We employ the segment routing (SR) technique as the routing algorithm and formulate the multi-time-step segment routing method that incorporates future traffic prediction. To address the high monitoring overhead, we present an approach that combines partial traffic prediction and compressive sensing techniques to estimate unmeasured data. Through extensive experiments on real backbone network traffic datasets, we demonstrate that our proposal can achieve more than 80% of the optimal performance in reducing maximum link utilization while significantly reducing the number of routing changes and traffic monitoring cost.**

***Index Terms*—Traffic engineering, segment routing, traffic prediction, graph neural network, network monitoring, compressive sensing.**

Manuscript received 19 April 2023; revised 28 August 2023; accepted 12 November 2023. Date of publication 4 December 2023; date of current version 15 April 2024. This research is partially supported by JSPS KAKENHI Grants with the numbers JP20H00592 and JP21H03424. The associate editor coordinating the review of this article and approving it for publication was T. Inoue. *(Corresponding author: Yusheng Ji.)*

Van An Le was with the Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-0003, Japan. He is now with National Institute of Advanced Industrial Science and Technology, Tokyo, Japan (e-mail: anle@nii.ac.jp).

Yusheng Ji is with the Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-0003, Japan (e-mail: kei@nii.ac.jp).

Huu Huy Tran and Phi Le Nguyen are with the School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 11615, Vietnam (e-mail: huy.th183557@sis.hust.edu.vn; lenp@soict.hust.edu.vn).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TNSM.2023.3338622

I. Introduction

THE CISCO Annual Internet Report [1] predicts that there will be 5.3 billion Internet users by 2023, an increase from 3.9 billion in 2018. Due to the high demand for Internet services such as video streaming, and VoIP, backbone network traffic has experienced exponential growth. As a result, traffic engineering (TE) tasks like optimizing traffic routing and network monitoring face significant challenges.

Recently, many studies have leveraged Machine Learning (ML) and Deep Neural Network (DNN) techniques, combined with traditional TE solutions, to address network problems. ML/DNN techniques can be used to predict future traffic demands or directly generate routing rules [2]. However, these approaches often suffer from two main problems. Firstly, there is a problem of high network disturbance or a large number of rerouting flows, leading to a degradation in the overall network's Quality of Service (QoS) [3]. Most of the proposed solutions only address the routing problem in a single snapshot (which is called a "time-step" in this paper) or use a short-term traffic prediction to calculate the routing rules without considering the long time horizon [4], [5], [6], [7]. Due to the dynamic behavior of the network traffic, the traffic matrix often varies over time, and the network controller may need to reroute many flows to balance the traffic loads, leading to significant network disturbance and service disruption. Depending on the traffic fluctuation, network optimization, and traffic rerouting can be performed with high frequency (e.g., at every minute). Secondly, the current ML-based TE solutions impose high network monitoring overhead. Many prior works only focus on solving routing problems and assuming the network statistics such as traffic matrix or link utilization are available. However, with the explosion of traffic and the expansion of the physical network, obtaining all the network statistics imposes a high monitoring overhead. In addition, applying ML/DNN into networking also needs a huge amount of monitored data for training/predicting processes. Although the quality of the measurements may have a huge impact on the performance of the TE solution [8], there are only a few studies that consider the joint problem of network monitoring and traffic engineering. Moreover, the scalability issue of applying ML/DNN techniques is often omitted in many ML-based TE studies.

To address these issues, this paper proposes a new approach that uses segment routing to optimize routing over a long time horizon and mitigates the need for frequent routing changes. A graph-based deep neural network is used to accurately predict future traffic demand, and based on the predicted values, a long-term routing method is proposed to optimize the routing rules. In this regard, we employ the segment routing (SR) technique as the routing algorithm and formulate the multi-time-step segment routing method (called MTSR) that incorporates future traffic prediction. To reduce network monitoring overhead, a partial traffic prediction approach is combined with the compressive sensing technique. Specifically, a partial future traffic matrix is predicted using a small amount of observed traffic, and the compressive sensing technique is used to reconstruct the entire traffic matrix. This approach reduces monitoring overhead while still achieving high performance in network routing. This work is an extension of a previous study [9].

The paper outlines several significant contributions related to traffic engineering in the face of high network demand and disturbance and high monitoring overhead.

- We address the issue of network disturbance by introducing the multi-time-step segment routing (MTSR) method, which utilizes an Integer Linear Programming formulation and advanced Deep Neural Network models for multi-step prediction to minimize the number of rerouting flows over a long time horizon.
- To account for prediction accuracy, we present three versions of the MTSR and provide a theoretical analysis of their performance.
- To enhance the practicality of MTSR by reducing network monitoring cost, we propose an extended approach called MTSR-CS that combines partial network traffic prediction with compressive sensing technique.
- We conduct extensive experiments using different real backbone network datasets to evaluate the performance of our proposed methods and compare them to state-of-the-art approaches.

The remainder of the paper is organized as follows: Section II provides an overview of related work and problem discussion, Section III-C presents our proposed method MTSR to address the high network disturbance problem, Section IV presents an extended of MTSR (called MTSR-CS) which can reduce the network monitoring overhead, Section V provides extensive experimental results, and finally, we conclude the paper in Section VI.

II. Background and Existing Works

This section first provides a brief overview of traffic engineering (TE) and related works. Then, we discuss the two problems which are addressed in this paper.

A. Traditional Traffic Engineering (TE)

The minimization of congestion is widely considered one of the most significant objectives in traffic engineering. Typically, achieving this objective involves reducing the maximum link utilization (MLU) in a network. In theoretical works, the Multi-Commodity Flow (MCF) problem is often used to obtain fractional solutions in which traffic flows are split and directed through various paths. However, in practice, many ISP networks rely on shortest-path routing techniques, such as OSPF and IS-IS, due to their simplicity. By adjusting link weights in a distributed manner, shortest path routing can compute near-optimal forwarding paths. Nonetheless, this method has drawbacks, such as lengthy re-convergence time and poor performance when network topology or traffic demands change. Recent advancements in routing techniques, such as MPLS and RSVP, have offered increased flexibility and improved traffic engineering performance by enabling explicit routing paths. However, MPLS-TE solutions are known to have long convergence times and a high cost of maintaining the TE tunnels. Another routing approach, known as oblivious routing and described in [10] and [11], performs traffic routing based solely on network topology without any knowledge of current network traffic. This method is easier to implement and does not cause the rerouting problem associated with adaptive routing techniques.

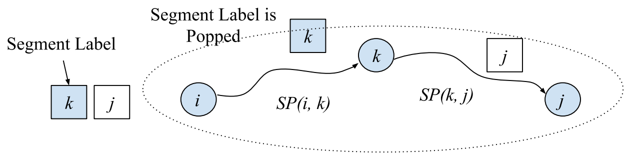


Fig. 1. Illustration of 2-segment routing [12].

B. Traffic Engineering With Segment Routing

Segment routing is a routing paradigm that operates on the basis of source routing. It allows the source node (or ingress node) to embed a list of Segment Identifiers (SIDs) in the packet header. This segment list serves as a set of instructions (SR policy) to direct the packet through the network devices. While SIDs can distinguish both nodes and links in the network, this paper focuses solely on the node-segment for simplicity. Packets originating from the source node must traverse all nodes in the segment list before being forwarded to the destination. Shortest path routing techniques, such as OSPF, are used to route the packet within the segment. Figure 1 demonstrates an instance of 2-segment routing, wherein a traffic flow originating from node i and destined for node j is directed through two segments, specifically, the $i - k$ and $k - j$ segments.

Due to its flexible routing capabilities, SR has been extensively investigated both theoretically and practically. Bhatia et al. [12] formulated the TE problem with SR as a linear programming problem, where only two segments were considered. Subsequent studies have focused on utilizing more than two segments. For instance, 3-SR [13] proposed an optimization model that uses three segments and combines both node and edge segments. Jadin et al. [14] further improved the TE problem with SR by fully exploiting n-SR (n-segment routing with both node and edge segments) and proposed the first Column Generation-based approach. In practical approaches, the authors in [15] considered unexpected traffic fluctuation and link failure problems and proposed a local search-based algorithm to solve the targeted problems under a sub-second constraint. Recently, studies by SR Tunnel [16] and [17] extended the work in [12] by proposing an optimization model to minimize the number of deployed SR policies.

C. The High Network Disturbance Problem

The majority of proposed traffic engineering solutions tackle the routing optimization problem by considering a snapshot of the traffic demand at the current time-step. Although optimizing network routing at each time-step may effectively fulfill the traffic engineering objective, such as minimizing the peak link utilization, it gives rise to a notable predicament. This predicament manifests as a considerable volume of data flows being rerouted to accommodate fluctuations in traffic demands, subsequently resulting in pronounced disturbances across the network (e.g., service disruption). The impact of the network disturbance problem has been examined in a study conducted by [3]. The findings from this study indicate that the substantial redirection of flow traffic can lead to a substantial decline of up to 50% of total network throughput. Furthermore, the rerouting can cause out-of-order problems in certain flows. The utilization of oblivious routing techniques [10], [11], [12] presents a viable method for mitigating the problem of network disturbance. By calculating the routing rules in advance only using the network topology (without prior knowledge of the traffic demand), these methods do not require changing the path of the network flows. While this approach can reduce the overhead caused by routing path changes, it may not perform well under dynamic network behaviors. Additionally, the oblivious routing method such as Traffic Matrix Oblivious Segment Routing in [12] is only practical for offline traffic engineering on relatively small networks as mentioned in [15].

Recently, several studies [18], [19], [20] utilized machine learning techniques to mitigate this problem. For example, within the work presented in [18], an approach founded upon Reinforcement Learning principles was introduced. This approach involves the identification of a subset of flows termed as the "critical flow set" which subsequently becomes the exclusive focus for rerouting. Consequently, the necessity to reroute flows is constrained, ensuring that the count of such flows remains within the confines of $k\%$ of the total number of flows within the network (e.g., $k = 10\%$). Further exploration of this method can be found in studies such as [19] and [20]. They introduced the integration of a graph neural network, thereby facilitating the generalization of the proposed approach to encompass previously untrained network scenarios. However, this method still results in a large number of rerouted flows over the time horizon when frequently executed.

D. The High Network Monitoring Cost of ML-Based Approach

Recently, machine learning (ML) techniques have been increasingly utilized to address various networking issues, including traffic routing, demand prediction, and anomaly detection. In traffic engineering, ML can be employed to forecast future traffic demands, enabling proactive calculation of routing rules to adapt to anticipated changes in traffic patterns. Previous studies [4], [6], [7], [21] have utilized diverse Deep Neural Network models, such as Convolutional Neural Network and Long Short-Term Memory, for traffic prediction and have obtained promising outcomes compared to traditional methods. In general, applying ML techniques issues a significant amount of data for the training and prediction processes. In the majority of ML-based TE solutions, there exists an implicit assumption that the necessary data, such as historical traffic demands, is readily accessible and the cost of network monitoring is frequently overlooked. To enhance the practicality of ML-based TE methods, there are some studies that focus on elevating network monitoring overhead.

For instance, the authors in [7] reduced the monitoring cost by only measuring a subset of network flows and proposed a method that exploits the forward and backward ConvLSTM layers to correct the input data. More recently, VAE (Variational AutoEncoders) models have been used by Kakkavas et al. in [22] to learn the distribution from traffic during the training phase. In the testing phase, the trained decoder of the VAE model is used to reconstruct the end-to-end traffic matrix from the links' load. This approach can be applied to reduce monitoring overhead since the number of links is typically less than the number of traffic flows, and link-level measurements are easier to obtain than flow-level measurements. However, this approach has not been evaluated for addressing traffic engineering issues.

III. Reducing Network Disturbance Leveraging Traffic Prediction

In this section, we first provide a concise overview of the network model and the traffic engineering problem using segment routing. Then, we present a traffic prediction-based routing algorithm named Multi-Time-Step Segment Routing (MTSR) to reduce network disturbance. We formulate the MTSR problem using three different traffic prediction approaches, each of which is based on the complexities inherent in the traffic prediction methods. Finally, we conduct a theoretical analysis of the three approaches. It is essential to note that within this section, we assume that historical traffic data is readily available for utilization in the prediction tasks. In practical scenarios, this data is typically acquired through network monitoring modules, which can cause substantial monitoring overhead. Consequently, to enhance the practicality of MTSR, we introduce the MTSR-CS approach in Section IV.

A. Network Model

The traffic engineering-based segment routing was first introduced in [12] as a traffic matrix-aware segment routing method. Hence, several notations and figures from [12] have been incorporated in this paper. The network is represented as a directed graph $G = (V, E)$, where V is the set of nodes $(|V| = N)$ and E is the set of links with each link $e \in E$ having a capacity of $c(e)$. The traffic matrix at time-step t is denoted as $M_t \in \mathbb{R}^{N\times N}$, where $m^t_{ij} \in M_t$ signifies the traffic flow from node i to j at time-step t. The term "flow ij" represents the total traffic that enters the network at node

i and exits at node j. We define the binary variable α_{ij}^k as the routing policy, where $\alpha_{ij}^k = 1$ indicates that flow ij is routed through the intermediate node k, and $\alpha_{ij}^k = 0$ otherwise. It is assumed that a central controller, such as an SDN controller [23], manages the network by collecting information about the network, predicting future traffic demands, and applying the routing policy to devices through PCEP [24]. However, the implementation of the controller is beyond the scope of this paper.

B. Single Time-Step Segment Routing

The 2-SR technique requires selecting a single intermediate node k for each flow ij, as shown in Figure 1. The traffic between nodes i and k and between nodes k and j is directed through the shortest path connecting them. If k is equal to i or j, the flow ij follows the shortest path from i to j. The problem of segment routing for a single time-step (P_0) can be expressed as an integer linear program [12], where the variable $\theta \in \mathbb{R}_+$ denotes the maximum link utilization in the network. Although, in practice, θ cannot be larger than 1 as the total traffic on a link cannot exceed its capacity, the problem formulation allows θ to exceed 1, indicating an over-congested network.

P_0: single time-step segment routing

$$\text{minimize}\, \theta \tag{1}$$

$$\sum_{k\in V} \alpha_{ij}^k = 1 \quad \forall i,j \in V \tag{2}$$

$$\sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k m_{ij}^t \le \theta c(e) \quad \forall e \in E \tag{3}$$

$$\alpha_{ij}^k \in \{0,1\} \quad \forall i,j,k \in V \tag{4}$$

The equation $g_{ij}^k(e) = f_{ik}(e) + f_{kj}(e)$ is used to calculate the total traffic load on a link e for flow ij with intermediate node k. Here, $f_{ik}(e) = 1$ if link e is part of the shortest path from i to k of flow ij with intermediate node k, and $f_{ik}(e) = 0$ otherwise. Eq. (2) and (4) ensure that all traffic from i to j is routed and it cannot be distributed across multiple paths. Eq. (3) ensures that the total traffic load on link e is less than or equal to its capacity.

C. Multi-Time-Step Segment Routing

The problem P_0 provides a routing policy for a single time-step t. Therefore, we need to resolve problem P_0 and update the routing policy every time-step. This approach may lead to a considerable number of rerouting flows and high network disturbance. To this end, we propose an extension of P_0, which addresses the segment routing problem by applying multi-time-step traffic prediction. Our strategy for mitigating network disturbance revolves around reducing the frequency of path alterations for traffic flows. This approach involves solving the TE problem taking into consideration the anticipated values of forthcoming traffic demand. Figure 2 illustrates the process within a routing cycle including three tasks. At the beginning of the cycle (denoted by time-step t), we use the historical traffic data collected from the previous cycle to estimate future demands. Subsequently, these forecasted

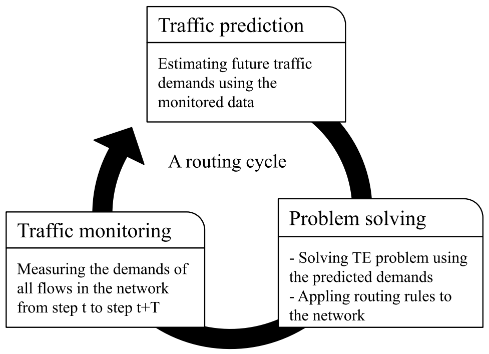

Traffic prediction
Estimating future traffic demands using the monitored data
A routing cycle
Traffic monitoring
Measuring the demands of all flows in the network from step t to step t+T
Problem solving
- Solving TE problem using the predicted demands
- Appling routing rules to the network

Fig. 2. The tasks within a routing cycle.

demand values are utilized for solving the TE problem. The routing rules derived from this approach can be applied for routing the traffic in the network over T time-steps of the cycle without necessitating frequent updates while retaining the adaptability required to accommodate dynamic demand fluctuations. Consequently, the traffic data is measured from time-step t through time-step $t + T$ and will be used in the next routing cycle.

In this context, we compare three different prediction schemes aimed at estimating the future traffic demand for the subsequent T time-steps. First, we provide some notations regarding the traffic prediction used in this paper. The proposed approaches are subsequently utilized to formulate the multi-time-step segment routing problem in three variants, denoted as P_1, P_2, and P_3, respectively. Each variant is based on a specific traffic prediction approach, and their corresponding theoretical analyses are presented. We use a prediction model $\hat{y} = f(x, \omega)$ to estimate the traffic demands (i.e., traffic matrices) of the next T time-steps using historical traffic data. Let t be the current time-step. The input of the prediction model is denoted as $x = [M_{t-1}, .., M_{t-H}]$, which contains the previous H traffic matrices. The predicted values and the model parameters are represented by $\hat{y}$ and ω, respectively. We describe the three traffic prediction approaches below:

- *Full prediction:* This approach involves estimating the traffic matrices of all network flows at every time-step in the next routing cycle. Therefore, we have $\hat{y} = [M_{t+1}, .., M_{t+T}]$.
- *Max prediction:* In this approach, we only predict the largest traffic demand for each traffic flow in the next T steps. The predicted traffic demand matrix is denoted by $\hat{y} = [M^*]$ which $m_{ij}^* \in M^*$ is the maximum value of flow ij $(i, j \in V)$ in the next routing cycle.
- *Period-max prediction:* This approach involves dividing the routing cycle into P sub-periods, each of which has T_p time-steps. We then estimate the maximum values of flow ij in each sub-period. Accordingly, we have $\hat{y} = [M_p^*](p = 1, .., P)$

An example of these three prediction approaches is shown in Figure 3. While the *full prediction* approach provides more information to the routing algorithm about future traffic fluctuation, it may suffer from low prediction accuracy. For example, the performance of some proposed prediction models [7], [25]

TABLE I
THE DIFFERENCES OF THREE PROBLEM FORMULATIONS

Problem	Routing cycle	Required number of predicted traffic matrices	Problem complexity
P_0	1 time-step	No traffic prediction	Low ($O(\|V\|+\|E\|)$ constraints)
P_1	T time-steps	T traffic matrices of every time-step in the next cycle	High ($O(\|V\|+T\times\|E\|)$ constraints)
P_2	T time-steps	One traffic matrix of maximum demands in the next cycle	Low ($O(\|V\|+\|E\|)$ constraints)
P_3	T time-steps	P matrices of maximum demands of every sub-periods in the next cycle	Medium ($O(\|V\|+P\times\|E\|)$ constraints)

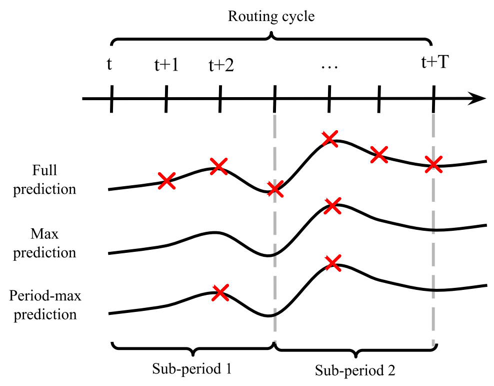


Fig. 3. An illustration of three traffic prediction approaches on one traffic flow. The red crosses indicate the values that need to be predicted corresponding to each method.

deteriorates as the number of predicted values increases. On the other hand, the *max prediction* approach reduces the prediction task's complexity by only estimating the worst-case scenario of network demands. The third approach can be seen as a trade-off between the first two methods.

1) P_1 (MTSR With Full Prediction): We present the problem formulations of the multi-time-step segment routing, denoted by P_1, P_2, and P_3, which are based on three traffic prediction approaches. First, we provide the problem formulation for P_1 using the output of the *full prediction* approach. Let $M = [M_1, M_2, \ldots, M_T]$ be the predicted traffic matrices for the next T time-steps, where the index t for the current time-step is omitted for simplicity. To formulate P_1, we extend the problem P_0 for T future steps by introducing additional constraints that correspond to the traffic demands at each step.

$$\text{minimize}\,\theta \tag{5}$$
$$\sum_{k\in V}\alpha_{ij}^k = 1 \quad \forall i,j\in V \tag{6}$$
$$\sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k m_{ij}^t \le \theta c(e) \quad \forall e\in E; \forall t\in T \tag{7}$$
$$\alpha_{ij}^k\in\{0,1\} \quad \forall i,j,k\in V \tag{8}$$

It can be observed that P_1 differs from P_0 due to the link capacity constraints (7). More specifically, in P_1, the routing policy α_{ij}^k must satisfy the link capacity constraints for each time-step in the future. The increased number of constraints makes P_1 more intricate than P_0. Additionally, P_1 necessitates predicting all the traffic matrices for the next T time-steps, which remains a significant challenge even with the latest deep learning models. Notably, prediction accuracy plays a critical role in the performance of P_1. However, this approach might exhibit low prediction accuracy due to the large number of values that need to be predicted. Therefore, to mitigate the burden on traffic prediction tasks and reduce the problem complexity, we formulate P_2 and P_3 using the *max prediction* and *period-max prediction* methods. P_2 and P_3 can be considered as the relaxed versions of P_1.

2) P_2 (MTSR With Max Prediction):

$$\text{minimize}\,\theta \tag{9}$$
$$\sum_{k\in V}\alpha_{ij}^k = 1 \quad \forall i,j\in V \tag{10}$$
$$\sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k \max_{t\in T} m_{ij}^t \le \theta c(e) \quad \forall e\in E \tag{11}$$
$$\alpha_{ij}^k\in\{0,1\} \quad \forall i,j,k\in V \tag{12}$$

In problem P_2, the formulation only takes into account the maximum values of each flow ij over the next T time-steps. This approach results in the same number of constraints as that of P_0. Furthermore, it only requires the prediction of a single traffic matrix with elements representing the maximum value of each traffic flow.

3) P_3 (MTSR With Period-Max Traffic Prediction):

$$\text{minimize}\,\theta \tag{13}$$
$$\sum_{k\in V}\alpha_{ij}^k = 1 \quad \forall i,j\in V \tag{14}$$
$$\sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k \max_{t\in T_p} m_{ij}^t \le \theta c(e) \ \forall e\in E; \forall T_p \tag{15}$$
$$\alpha_{ij}^k\in\{0,1\} \quad \forall i,j,k\in V \tag{16}$$

Similar to P_2, we formulate P_3 using the maximum values of flow ij in each sub-period T_p. To solve P_3, we only need to predict P traffic matrices, which represent the maximum traffic of every flow ij in each sub-period T_p. The differences between the proposed problem formulations are summarized in Table I. In general, the number of required predicted traffic matrices depends on the method used to estimate future traffic.

D. Theoretical Analysis

In this part, we theoretically analyze the performance ratios of P_2, and P_3 to P_1. Denote (θ_1^*, α_1^*), (θ_2^*, α_2^*), and (θ_3^*, α_3^*) as the optimal solutions obtained by solving P_1, P_2, and P_3, respectively. We denote $u(t, e, \alpha_p^*)$ the utilization of link e when we apply routing policy α_p^* in routing cycle t. Then, the maximum link utilization of the network when applying routing policy (α_p^*), denoted as $u(\alpha_p^*)$, is defined as follows:

$$u(\alpha_p^*) = \max_{\forall t,e} u(t,e,\alpha_p^*) = \max_{\forall t,e} \frac{\sum_{ij}\sum_k g_{ij}^k(e)(\alpha_p^*)_{ij}^k m_{ij}^t}{c(e)}$$

Theorem 1:

- $\theta_1^* = u(\alpha_1^*)$; $\theta_2^* \geq u(\alpha_2^*)$; $\theta_3^* \geq u(\alpha_3^*)$
- $\theta_1^* \leq \theta_3^* \leq \theta_2^*$

Theorem 2: Let e^* be the link with the largest capacity, and e_2^* be the link where the equal sign of constraint (11) in P_2 holds; Then, the performance ratio of P_2 to P_1 (i.e., $\frac{\theta_2^*}{\theta_1^*}$) is upper bounded by λ, which is calculated as follows.

$$\lambda = \frac{\sum_{ij}\max_t m_{ij}^t}{\max_t \max_{ij} m_{ij}^t}\frac{c(e^*)}{c(e_2^*)} \tag{17}$$

The proofs are presented in Appendix A. According to Theorem 1, when applying the routing policies obtained from solving the MTSR problem, the actual maximum link utilization of the network is less than its theoretical θ^*. In addition, since the performance of P_3 is bounded by P_1 and P_2 (i.e., $\theta_1^* \leq \theta_3^* \leq \theta_2^*$), we will derive the upper bound of $\frac{\theta_2^*}{\theta_1^*}$ which will also be the upper bound of $\frac{\theta_3^*}{\theta_1^*}$. Assume that all the network links have the same capacity, the performance ratio λ largely depends on the current network situation in the routing cycle. If all flows in the network have similar demands and are stable within the routing cycle, the value of λ could be much greater than 1. However, the network traffic is usually unbalanced where a small number of elephant flows accounted for a large portion of total network traffic, especially in ISP and data center networks [26], [27], [28]. Due to the unbalanced traffic flows, we may have $\sum_{ij}\max_t m_{ij}^t \approx \max_t \max_{ij} m_{ij}^t$. Therefore, the more unbalanced and dynamic the traffic flows, the lower the performance ratio λ is.

According to Theorem 2, by solving the MTSR problem with formulation P_2, we can significantly reduce the problem complexity while still achieving a good performance for the routing policy. In addition, P_2 only requires the predicted values of the maximum demand instead of the predicted demands of every time-steps in the next routing cycle. By doing so, P_2 alleviates the difficulty in the traffic prediction task. Therefore, we formulate the MTSR problem using P_2 formulation. In addition, by using the maximum traffic prediction, we can easily extend the method to n-segment routing. There are several n-segment routing algorithms (e.g., SRLS [15]) that take the traffic matrix as input to compute routing rules. Hence, we can use the maximum traffic matrix which is obtained from the prediction model as the input for these n-segment routing algorithms. In the remaining part of this paper, when mentioning the MTSR without any specification, we mean the MTSR with formulation P_2.

E. Traffic Prediction-Based Graph Neural Network

As mentioned in the previous section, accurately predicted traffic matrices are required as input for all the problem formulations, except in the case we don't consider traffic

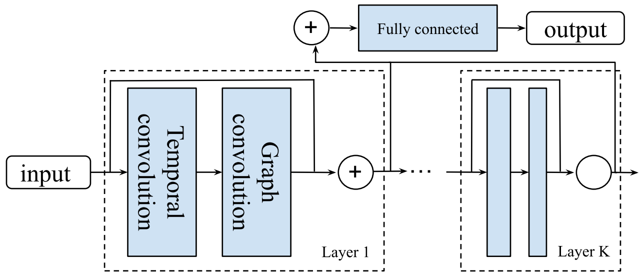


Fig. 4. The structure of GWN model.

prediction for P_0, i.e., using current traffic for TE. Note that, our objective is not to develop a new prediction model but to leverage the existing models for addressing the traffic engineering problem. There is a large number of proposed models for traffic matrices prediction such as in [5], [7]. Here we adopt the prediction model to meet the requirements of our problem formulations. For example, in problem P_1, at the beginning of each routing cycle, the prediction model uses the historical data of the last H time-steps to predict the traffic of the next T time-steps. In P_2, the prediction model only needs to infer the maximum demand for each traffic flow.

In this paper, we use Graph WaveNet (GWN) [25] as the prediction model. Motivated by [29], GWN adopts stacked dilated casual convolutions to extract temporal features in the historical traffic data. Its structure is depicted in Figure 4. The model comprises multiple layers, with each layer consisting of two principal modules: the temporal convolution module and the graph convolution module. The temporal convolution module utilizes dilated causal convolution to capture a node's temporal trends, while the graph convolution module extracts a node's features based on its structural information. The final output is obtained by combining the outputs of all layers through a fully connected layer. It should be noted that each layer may include multiple blocks, each comprising the two modules. GWN model was originally developed for predicting transportation traffic. In order to adapt GWN for network traffic prediction, we treat each traffic flow (i.e., source-destination flow *ij*) as a node in the graph, with the traffic volume of the flow serving as the attribute of the node. Since we do not have an explicit graph that represents the relationships among traffic flows, we employ the self-adjacency matrix module described in [25]. This module includes two learnable vectors that can dynamically learn the relationships among flows based on the current input data.

By combining the temporal convolution and the graph convolution modules, GWN is able to handle spatial-temporal data (e.g., traffic flows) and achieve better prediction accuracy than other time-series models such as Long Short-Term Memory (LSTM) and Auto-Regressive Integrated Moving Average (ARIMA). The implementation of GWN for traffic prediction used in this work can be found at [30].

F. Obtaining Routing Rules

After getting the predicted traffic demand, we need to solve the optimization problem (e.g., P_2) to obtain the routing rules. To solve the MTSR problem, we can use optimization solvers or heuristic algorithms. However, MTSR is an integer programming problem with a vast search space, thus traditional optimization solvers are hardly scaled to large topologies with more than 20 nodes [15]. Thus, using heuristic or meta-heuristic algorithms would be a practical way to solve this problem rapidly. Gay, Hartert, and Vissicchio proposed a local search algorithm called SRLS [15] to solve the n-segment routing algorithm. We adapted the algorithm SRLS [15] and proposed an algorithm called Local Search 2 Segment Routing (LS2SR) that can effectively solve the MTSR problem (with P_2 formulation) by exploiting the intrinsic structure of 2-segment routing. We also improved LS2SR by adding a mechanism to refine the routing policy from the previous cycle, thereby reducing the routing policy variation over different routing cycles. The details of this method have been presented in our prior study [9]. We have evaluated the scalability of LS2SR with different network sizes (see Appendix B) to show that LS2SR can work on large-scale networks and achieve comparable performance with SRLS.

IV. Partial Traffic Prediction and Compressive Sensing

In the previous sections, we have presented the multi-time-steps segment routing (MTSR) strategy to address the network disturbance by utilizing future traffic prediction. The performance of MTSR relies on the accuracy of the traffic prediction model (i.e., GWN) which requires a large amount of data for the training and predicting processes. Note that, in Section III-C, we assume the data for traffic prediction tasks are available. However, collecting the historical data of all flows causes the problem of high network monitoring cost. As mentioned in Section II-D, this problem is often omitted in the prior ML-based TE studies. To enhance the practicality of our proposed method, we introduce an extension of MTSR called MTSR-CS which combines the MTSR and the compressive sensing technique. In MTSR-CS, network measurements and traffic predictions are selectively conducted solely for a subset of flows. Subsequently, a comprehensive matrix is reconstructed from the partial traffic predictions using compressive sensing before being used to calculate the routing rules.

A. Compressive Sensing-Based Network Traffic Reconstruction

According to compressive sensing theory, the signal can be reconstructed or recovered from a few samples by exploiting the sparsity characteristic of the original signal. Therefore, compressive sensing can be used in reconstructing the network traffic from a few measurement data as follows.

$$Z = \Phi X \tag{18}$$

where $X \in \mathbb{R}^{F \times 1}$ is a vector that contains the traffic volume of all flows. $F = N \times N$ is the total number of source-destination flows in the network (N is the number of nodes). Note that, instead of using a $N \times N$ matrix to represent the network traffic, it is represented by a vector that has F elements). $Z \in \mathbb{R}^{L \times 1}$ represents the measured traffic volumes. L is the number of flows that are measured ($L < F$). $\Phi \in \{0,1\}^{L \times F}$ is a binary matrix to indicate the measured flows. Figure 5 shows an example of partial traffic measurement.

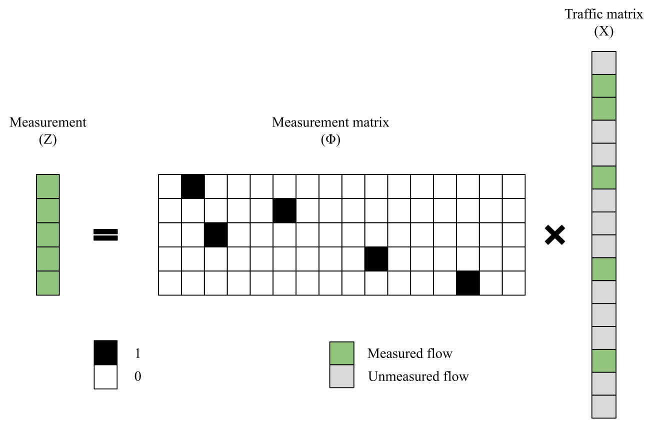
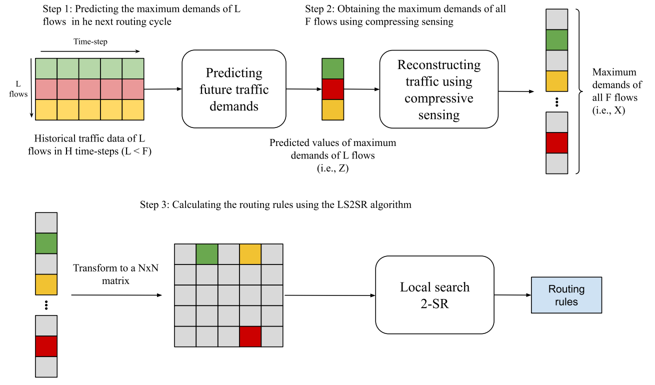


Fig. 5. Partial traffic measurement. The traffic matrix X is represented as a vector.



Step 1: Predicting the maximum demands of L flows in he next routing cycle
Time-step
L flows
Historical traffic data of L flows in H time-steps (L < F)
Predicting future traffic demands
Predicted values of maximum demands of L flows (i.e., Z)
Step 2: Obtaining the maximum demands of all F flows using compressing sensing
Reconstructing traffic using compressive sensing
Maximum demands of all F flows (i.e., X)
Step 3: Calculating the routing rules using the LS2SR algorithm
Transform to a NxN matrix
Local search 2-SR
Routing rules

Fig. 6. Illustration of partial traffic prediction and matrix reconstruction using compressive sensing.

However, since network traffic X is not sparse in practice, compressive sensing cannot be directly applied to reconstruct X from Z. To overcome this problem, the authors in [31] use a transformation matrix $D \in \mathbb{R}^{F \times F}$ to sparsely project X in the transformation domain D as:

$$X = DS \tag{19}$$

where $S \in \mathbb{R}^{F \times 1}$ is a sparse projection of X. From Eq. (18)–(19), we have:

$$Z = \Phi DS \tag{20}$$

Since S is a sparse vector, we can apply compressive sensing to reconstruct S from the measurement Z. Then the full network traffic X can be obtained by using Eq. (19).

B. Reconstruct the Traffic Matrix From the Partial Traffic Prediction

In contrast to the approach proposed in [31], our methodology does not utilize compressive sensing to reconstruct the network traffic itself. Rather, we leverage it to reconstruct the maximum traffic demand from a partial traffic prediction. The conceptual framework behind this is presented in Figure 6. The routing algorithm (MTSR) takes a traffic

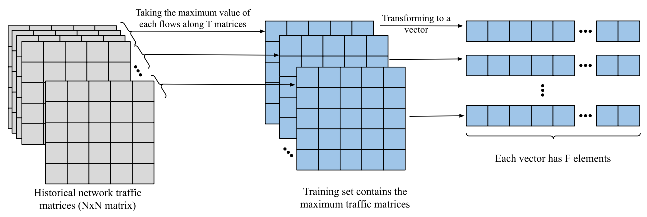

Taking the maximum value of each flows along T matrices
Transforming to a vector
Historical network traffic matrices (NxN matrix)
Training set contains the maximum traffic matrices
Each vector has F elements

Fig. 7. Generating the training dataset from historical traffic matrices.

matrix that contains the anticipated values of the maximum demand of all traffic flows in the next routing cycle. Our approach reduces the monitoring overhead by reconstructing this matrix from a partial traffic prediction. To this end, we first estimate the maximum demands of L flows via a prediction model. Subsequently, compressive sensing is applied to obtain the maximum traffic demands of all flows. Finally, the routing rules are calculated via the LS2SR algorithm [9]. Thus, instead of monitoring and predicting the entire traffic matrix, we monitor and predict the future demands of a subset of traffic flows, thereby decreasing the monitoring overhead.

In order to implement the methodology outlined in [31], it is necessary to define the following terms: Let Z (as defined in Eq. (18)) be a vector composed of elements representing the predicted values of the maximum demands of L monitored flows. Let X denote the vector comprising the predicted maximum demands of all F flows. The proposed technique is composed of two phases, namely the training phase and the testing phase. In the training phase, a training dataset is utilized to calculate the transformation matrix D, which is subsequently employed to train the prediction model. In the testing phase, partial traffic prediction of a subset of flows (Z) is carried out at the onset of each routing cycle. The maximum demand of other flows X is then reconstructed from Z using compressive sensing and the transformation matrix D. The routing rules are subsequently computed utilizing the LS2SR algorithm. The specifics of the process will be elucidated in the ensuing sections.

C. Training Phase

The training phase comprises two fundamental tasks, namely the acquisition of the transformation matrix D and the training of the prediction model. The training dataset encompasses the historical measured traffic volume of all flows. Nonetheless, as the focus of the *max prediciton* is centered on the maximum demand during a routing cycle, the original dataset is converted into the set of maximum traffic matrices, as depicted in Figure 7. The maximum matrix is constructed by computing the maximum values of each flow over T time-steps within each routing cycle. Subsequently, the resulting matrix is flattened into a vector that comprises F elements. Thereafter, the generated data is employed to train the prediction model and acquire the transformation matrix D.

The transformation matrix D can be acquired by solving the optimization problem:

$$\text{minimize} \quad \left\| \hat{X} - D\hat{S} \right\|^2 \tag{21}$$

$$\left\| \hat{S}(i) \right\|_0 \leq K \tag{22}$$

$$\hat{S}(i) \geq 0 \tag{23}$$

$$i = 1, 2, .., T \tag{24}$$

where $D \in \mathbb{R}^{F \times F}$ and $\hat{S} = (\hat{S}(1), \hat{S}(2), \ldots, \hat{S}(T)) \in \mathbb{R}^{F \times T}$ are two unknown variables. Furthermore, $\hat{X} = (\hat{X}(1), \hat{X}(2), \ldots, \hat{X}(T)) \in \mathbb{R}^{F \times T}$ denotes T vectors within the training set, and each column $\hat{S}(i) \subset \hat{S}$ is a sparse representation of the traffic vector $\hat{X}(i)$. The parameter K is an upper bound on the number of non-zero entries in the sparse representation.

As reported in [31], the Alternative Least Square (ALS) algorithm can be employed to obtain the solution to the aforementioned problem. Firstly, we randomly assign values to the transformation matrix D. Subsequently, given D, ALS is utilized to determine $\hat{S}$. Based on the obtained $\hat{S}$, we can then update the values of D. The iterative updates of $\hat{S}$ and D continue until certain termination criteria (e.g., maximum iteration) are met. The updating process of D can be described as follows. Initially, one column of the transformation matrix D is updated at a time. Let d_k be the k^{th} column of D, and $\hat{s}k$ be the k^{th} row of $\hat{S}$ (where $k = 1, 2, \ldots, F$). The multiplication $D\hat{S}$ is decomposed into the sum of M rank-1 matrices: $D\hat{S} = \sum_{j=1}^{M} d_j \hat{s}_j$. We have:

$$\hat{X} - D\hat{S} = \hat{X} - \sum_{j=1}^{M} d_j \hat{s}_j \tag{25}$$

$$= \left(\hat{X} - \sum_{j \neq k}^{M} d_j \hat{s}_j \right) - d_k \hat{s}_k \tag{26}$$

$$= E_k - d_k \hat{s}_k \tag{27}$$

Then, to update the k^{th} column of D, we solve the following optimization problem:

$$\text{minimize} \quad \| E_k - d_k \hat{s}_k \|^2 \tag{28}$$

$$\left\| \hat{S}(i) \right\|_0 \leq K \tag{29}$$

$$\hat{s}_k \geq 0 \tag{30}$$

$$i = 1, 2, .., T; k = 1, 2, \ldots, F \tag{31}$$

We use Singular Value Decomposition (SVD) to update d_k and $\hat{s}_k$. For the details of solving the above problem, please refer to [31]. Then, we repeat the process above to update other columns of D.

D. Testing Phase

1) Traffic Reconstruction: During the testing phase, there are three main tasks to be accomplished: predicting the future traffic (i.e., Z), reconstructing vector X, and calculating network routing rules. At the beginning of each routing cycle, we estimate the maximum future demands of L flows using the monitored data obtained from the last cycle. After that, we reconstruct the vector X from the partial prediction. Let $Z \in \mathbb{R}^{L \times 1}$ be the predicted traffic values. We solve an optimization problem to find the sparse representation S using the transformation matrix D, the measurement matrix Φ, and the predicted traffic vector Z.

$$S = \text{argmin} \, \|S\|_0 \tag{32}$$

$$Z = \Phi D S \tag{33}$$

$$S \geq 0 \tag{34}$$

Then, we get the reconstruction results of X using Eq. (19).

2) Selecting Monitored Flows: After applying the routing rules, we determine which flows (a set of L flows) will be monitored in the routing cycle. We measure the demand of these flows within the routing cycle (from time-step t to time-step $t + T$. The monitored data is used as input to perform the *max prediction* in the next routing cycle. Intuitively, the large flows in traffic volume tend to have a significant impact on the routing performance, However, accurately monitoring the L largest flows is not feasible since all traffic flows need to be measured beforehand. Therefore, we propose a simple monitoring scheme based on the traffic volumes of L monitored flows from the previous cycle. The new set of selected flows is a combination of φL largest flows from the previous set and a randomly selected $(1-\varphi)L$ flows from other traffic flows, where $0 < \varphi < 1$. Note that when performing *max prediction*, we estimate the maximum traffic demand of L flows that have been measured in the preceding routing cycle. Therefore, the traffic data for executing *max prediction* is available.

V. Evaluation

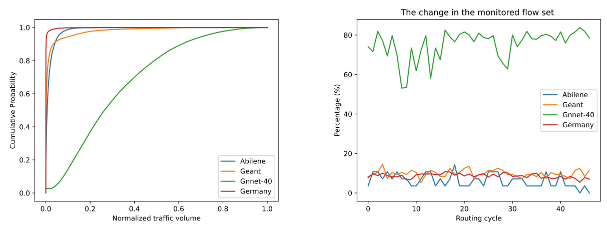
We design four different experiments to evaluate our approach in both hypothetical and practical scenarios with different backbone networks including real and synthetic datasets. The experiment's results can be reproduced at [30]. In the initial two experiments, we evaluate the performance of the MTSR method under the assumption that data pertaining to all traffic flows can be accurately measured. Subsequently, we examine the performance of the MTSR-CS method with different numbers of monitored flow (e.g., L). In the last experiment, we study the impact of several factors including: (1) different prediction models, and (2) the routing cycle length T. In addition, other ablation studies such as the scalability assessment of the routing algorithm LS2SR and the time dedicated to traffic prediction/reconstruction are presented in Appendix B.

A. Datasets and Performance Metrics

We conducted experiments on four datasets: Abilene, Geant, Germany, and Gnnet-40, available at [32], [33]. Abilene and Geant are well-known public datasets that are used for evaluating traffic routing algorithms in many studies such as [9], [18], [34]. Among these datasets, Gnnet-40 is one of the synthetic network datasets used in the Graph Neural Networking challenge 2021 [33]. Details of the datasets are presented in Table II. Figure 8 illustrates the differences in traffic among all the datasets. Figure 8(a) is a cumulative distribution function (CDF) of the traffic demands in the network, normalized to the range [0, 1]. Among these datasets, Gnnet-40 exhibits a uniform distribution in traffic demands, while

TABLE II
Details of the Real Traffic Datasets

Dataset	No. nodes	No. links	Monitoring granularity
Abilene	12	30	5 (min)
Geant	22	72	15 (min)
Gnnet-40	40	72	5 (min)
Germany	50	72	5 (min)



(a) Traffic distribution (CDF plot). (b) Traffic dynamicity.

Fig. 8. The distribution and dynamicity of all traffic datasets.

the others demonstrate an "elephant-mice" flow distribution, where more than 80% of flows are small flows. Next, we visualize the differences in the top L largest flow sets over routing cycles in Figure 8(b) (e.g., $L = 20\%$ of the total flows, $T = 6$ time-steps). The figure reveals that in Gnnet-40, the top L largest flow set changes rapidly over the routing cycle. More than 80% of the top L largest flows in the current cycle are replaced by other flows in the next cycle. The traffic data is divided into three sets, namely train, validation, and test, based on the time horizon, which corresponds to 70%, 10%, and 20% of the complete dataset, respectively. Although the monitoring granularity of the datasets is relatively large, our problem formulation and proposed approach can be generally applied to the network systems that have finer monitoring granularity.

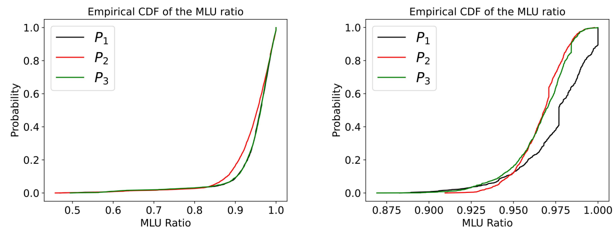
We evaluate the performance of our proposed approach using two main metrics: the MLU ratio (r_{mlu}) and the rerouting disturbance (RD) adopted from [18]. The MLU ratio is calculated using Eq. (35) where θ^* is the maximum link utilization of the network obtained using P_0. Since in P_0, the routing rule is computed for every time-step using the current traffic demand, the MLU of P_0 can be considered as optimal results. Therefore, $r_{mlu} = 1$ means that the routing algorithm achieves as good as the routing rule as the optimal routing.

$$r_{mlu} = \frac{\theta^*}{\theta} \tag{35}$$

The rerouting disturbance (RD) is defined in Eq. (36) as the ratio of the number of rerouting flows to total traffic flows in a time-step. In Eq. (36), f_r is the number of rerouting flows per time-step, and $F = N \times N$ is the total number of traffic flows.

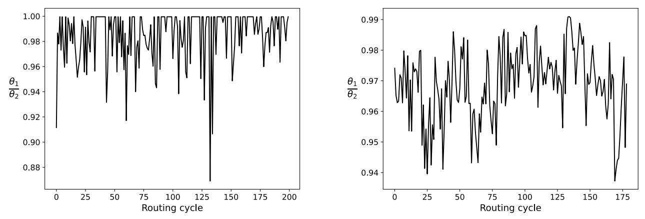
$$RD = \frac{f_r}{F} \tag{36}$$

We employ the Mean Absolute Error (MAE) as a metric for assessing the accuracy of predictive and reconstructive values in the tasks of future traffic forecasting and traffic matrix reconstruction. The MAE is calculated using Eq. (37)



(a) Abilene network (b) Geant network

Fig. 9. The empirical CDF MLU ratio of three MTSR methods.



(a) Abilene network (b) Geant network

Fig. 10. The MLU ratio θ_1/θ_2 over routing cycles. θ_1 and θ_2 are the maximum link utilization of the network obtained using methods P_1 and P_2, respectively.

where Γ is the total number of predicted/reconstructed values, $\hat{y}_i$ denotes the anticipated or reconstructed value, while y_i signifies the ground-truth value.

$$MAE = \frac{1}{\Gamma}\sum_{i=0}^{\Gamma}|\hat{y}_i - y_i| \tag{37}$$

B. Experiment 1: Performance Comparison of Three MTSR Approaches

In the first experiment, we assess the performance of the multi-time-step segment routing with different traffic prediction methodologies, denoted as P_1, P_2, and P_3, respectively. These approaches are segment routing algorithms that employ *full, max, and period-max traffic prediction* techniques, as described in Section III-C. The objective of this experiment is to determine the best MTSR approach in the absence of prediction errors. For this purpose, we assume that future traffic matrices can be accurately predicted by utilizing the actual traffic matrices from the test set as input. We use the solver from the PuLP library [35] to solve all the problems and obtain the optimal solutions. In all experiments, the routing cycle length T for MTSR schemes is set to 12.

Figure 9 presents the empirical cumulative distribution function (CDF) of the MLU ratio (r_{mlu}) on the Abilene and Geant datasets. The results demonstrate that all MTSR approaches exhibit strong performance, achieving approximately 90% of the optimal results. Among the three approaches P_1 yields the highest performance, followed by P_3 and P_2, respectively. However, the differences between their performances are marginal. Figure 10 illustrates the performance ratio $\frac{\theta_1}{\theta_2}$ across different routing cycles. As evidenced by the results, P_2, which uses the *max prediction* approach, can attain over 90% the performance of P_1 with full prediction in both datasets.

TABLE III
THE PARAMETERS OF THE GWN MODEL AND THE EXPERIMENT SETTING

Model parameters			
Number of layers	3	Hidden units	32
Number of blocks per layer	2	Batch size	256
Convolution's kernel size	2	Learning rate	0.001
Convolution's stride	2	Drop out	0.3
Training parameters			
Number of input step H	15	Number of train epoch	300
Routing cycle length T	6	Early stop condition (when model does not improve after a number of training epoch)	50

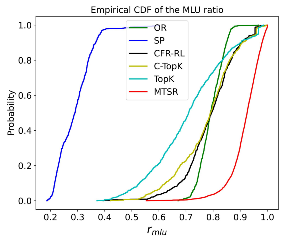
The observations reveal three key benefits of addressing the MTSR problem by adopting the *max prediction* approach (P_2): (a) mitigating the complexity of the problem, (b) alleviating the challenges associated with traffic prediction, and (c) attaining performance that is comparable to other approaches. Consequently, for the rest of this paper, we refer to MTSR as P_2 utilizing the *max prediction* approach.

C. Experiment 2: Performance Comparison of Different Routing Algorithms

We conduct an experimental evaluation to assess the performance of the proposed MTSR approach, which utilizes the *max prediction* approach (P_2), and compare its performance with other routing algorithms. To predict future traffic matrices, we train a Graph WaveNet (GWN) model [25]. The prediction model leverages data from the last H time-steps to predict the maximum demand matrices of each flow in the next routing cycle. The implementation of the Graph WaveNet model utilized in this study is adopted from [36]. Table III displays the parameters of the GWN model and the experiment settings employed in the training and testing phase.

At the beginning of each cycle, we utilize the predicted traffic matrices to solve the problem and obtain the routing policy. Subsequently, the actual traffic matrix is utilized to calculate the maximum link utilization for each time-step. We adopt the proposed LS2SR algorithm [7] to solve the MTSR problem, with the predicted traffic serving as the input. We set the routing cycle length to $T = 6$ and the number of historical steps utilized for the prediction model to $H = 15$. We set the values of $\beta = 16$ and $\gamma = 1$, which are adopted from [15].

We compare the performance of our proposed MTSR approach against several other routing algorithms, including ***P*₀** (same as in Experiment 1), Traffic Matrix Oblivious Segment Routing (**OR**) [12], shortest path routing based on link's weight (**SP**), critical flow rerouting using Deep Reinforcement Learning (**CFR-RL**), TopK Critical (**C-TopK**), TopK, and our proposed MTSR algorithm. In order to have a fair comparison, we set the value of k% to 10% for CFR-RL, TopK, and TopK Critical, which were adopted from [18]. Note that, in [18], new routing paths for the critical flows are obtained by solving the MCF problem, but we solve it using 2-segment routing. Since the network traffic is measured every five minutes (e.g., Abilene), in this paper, we set a limited time


(a) Abilene network

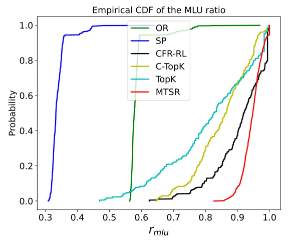
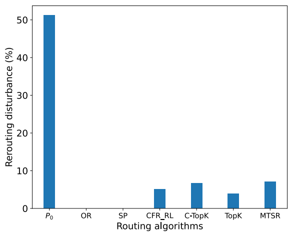

(b) Geant network

Fig. 11. The empirical CDF of MLU ratio.


(a) Abilene

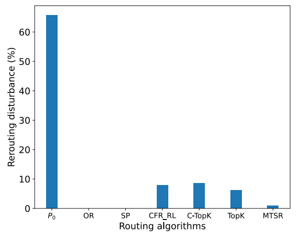
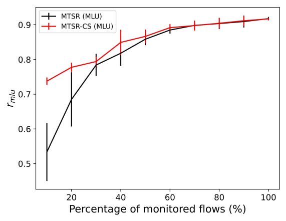

(b) Geant

Fig. 12. The comparison in the rerouting disturbance.


(a) MLU on Abilene dataset

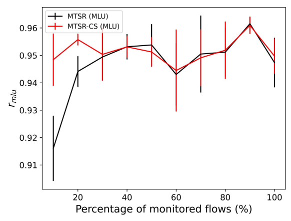

(b) MLU on Geant dataset

Fig. 13. The MLU of MTSR and MTSR-CS with different percentages of monitored flows.

for solving the routing rules as 60 seconds for all the routing algorithms. The time for running the traffic prediction task is negligible as shown in Appendix B.

Figures 11 and 12 present the empirical CDF of the MLU ratio and the rerouting disturbance, respectively. The results demonstrate that MTSR can significantly reduce network disturbance while achieving over 80% of the optimal routing performance in both datasets. Additionally, MTSR outperforms other routing algorithms in terms of MLU ratio. Regarding rerouting disturbance, CFR-RL can keep the disturbance at a similar value to our method by setting a maximum number of flows that can be rerouted per step (10%). However, in larger networks such as the Geant network, the rerouting disturbance of CFR-RL is considerably higher than our method. Shortest path routing and Oblivious Segment Routing cause no rerouting disturbance but are not adaptive to dynamic network demands, often resulting in high MLU.

D. Experiment 3: Performance Evaluation of MTSR-CS

The objective of this experiment is to assess the effectiveness of the MTSR-CS technique, which has been developed with the purpose of minimizing the monitoring overhead, while simultaneously maintaining the routing performance. The degree of monitoring overhead is quantified by the number of monitored flows, denoted by L. It is intuitive that as the value of L increases, the monitoring overhead also increases. Thus, in this experiment, we vary the value of L from 10% to 100% of total number of flows and measure the MLU ratio (r_{mlu}). Regarding our proposed monitoring scheme (Section IV-D2), the new set of selected flows is a combination of φL largest flows from the set of previous routing cycle and a randomly selected $(1-\varphi)L$ flows from other traffic flows. In this experiment, we set $\varphi = 0.5$. It is important to note that in the case of $L = 100\%$, we carry out the MTSR method in the same manner as in the second experiment, i.e., without the utilization of compressive sensing.

As depicted in Figure 13, our results indicate that the use of compressive sensing leads to a significant improvement in r_{mlu}, particularly when $L < 30\%$ total flows. In comparison with the conventional approach of full traffic monitoring ($L = 100\%$ total flows), the MTSR-CS method successfully reduces the monitoring overhead while still achieving comparable routing performance. Notably, in the Geant network, MTSR-CS is able to obtain the same performance result by monitoring only 10% to 20% of the total number of flows. This observation can be attributed to the uneven distribution of traffic in the Geant network, where a mere 10% of the flows (i.e., 48/484 flows) correspond to more than 80% of the total network traffic. By monitoring only 10% of the flows, we can effectively capture the traffic data of all the large flows within the network. Consequently, MTSR-CS with $L = 10\%$ of total flows can achieve similar routing performance to the case of full network monitoring. Note that in this experiment, MTSR only performs traffic prediction for the selected flows and refrains from employing compressive sensing techniques to reconstruct the entire traffic matrix. As a result, its performance does not exceed that of MTSR-CS, which incorporates compressive sensing methodologies.

Subsequently, we conduct the experiment using all four datasets to evaluate the impact of different traffic distributions on the matrix reconstruction and routing tasks. Here, we compare the MTSR-CS approach using different monitoring schemes and visualize the reconstruction error (MAE) and the routing performance (MLU ratio) in Figure 14 and Figure 15 respectively. In total, we have three monitoring schemes: Random, TopK, and our proposed monitoring scheme (see Section IV-D2). In the TopK monitoring scheme, we assume that the data of the L largest flows can be measured during each routing cycle. On the other hand, in the Random scheme, L flows are randomly selected for monitoring in each routing cycle.

Generally, it is observed that a lower reconstruction error of the traffic matrix corresponds to an enhanced routing performance. Considering Abilene, Geant, and Germany datasets, our proposed scheme yields similar results to the TopK scheme in terms of reconstruction error, thereby achieving the same routing performance as the TopK approach, while significantly outperforming the Random monitoring scheme. The proposed monitoring scheme is considered a trade-off

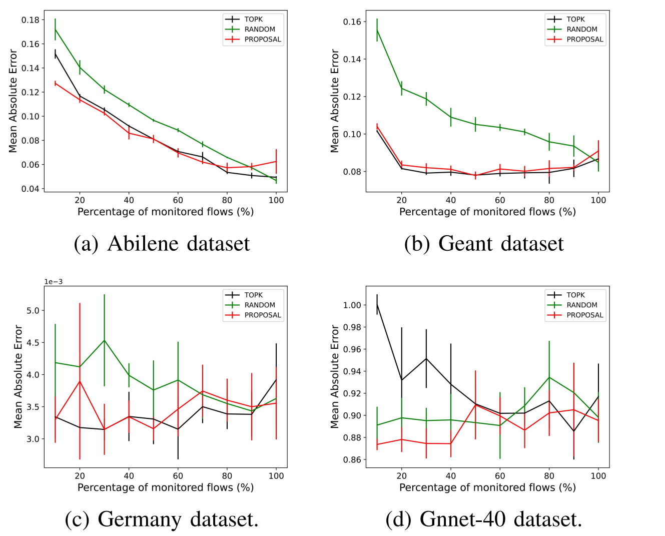

(a) Abilene dataset (b) Geant dataset (c) Germany dataset. (d) Gnnet-40 dataset.

Fig. 14. The MAE of reconstructed matrices using different monitoring schemes.

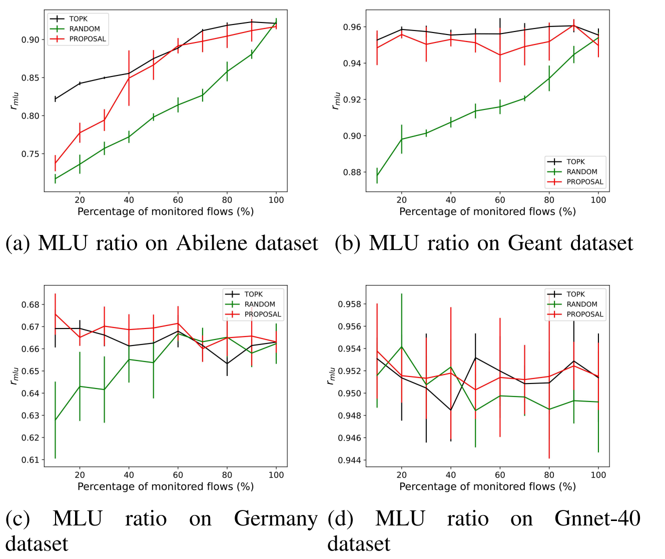

(a) MLU ratio on Abilene dataset (b) MLU ratio on Geant dataset (c) MLU ratio on Germany dataset (d) MLU ratio on Gnnet-40 dataset

Fig. 15. The routing performance (MLU) of MTSR-CS using different monitoring schemes.

method between the Random and TopK monitoring schemes. Although the TopK monitoring scheme yields the best results, it is deemed impractical due to the strong assumption that the L largest flows can be identified prior to monitoring.

In the case of Gnnet-40, which has a uniform traffic distribution, distinct outcomes are observed. Notably, the TopK scheme exhibits the poorest performance among the monitoring schemes. This decline in the performance of TopK can be attributed to the rapid alteration of the monitored flow set throughout the routing cycle, as discussed in Section V-A. In the TopK scheme, measurements are conducted on the L largest flows, and this data is subsequently employed in the next routing cycle. Given that the majority of these flows will not constitute the largest flows in the following routing cycle, utilizing data from the prior cycle results in high reconstruction

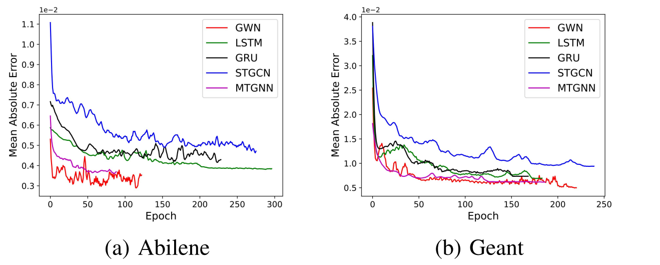

(a) Abilene (b) Geant

Fig. 16. The MAE of different prediction models in case $H = 15$ and $T = 6$.

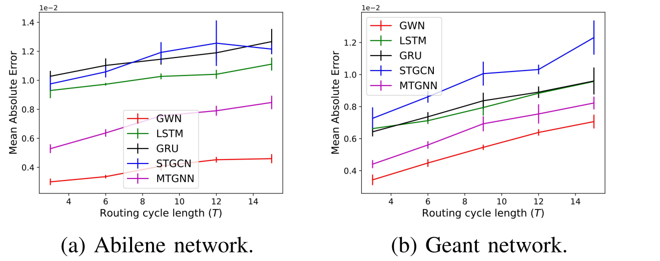

(a) Abilene network. (b) Geant network.

Fig. 17. The prediction errors of different models with T varying from 3 to 15.

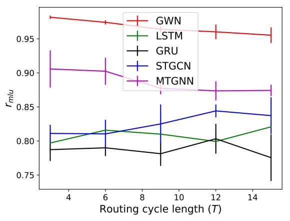
errors and consequently lower MLU ratios. In conclusion, our proposed scheme not only demonstrates greater practicality compared to the TopK scheme but also exhibits the capacity to attain commendable performance across varying network sizes and traffic patterns.

E. Ablation Studies

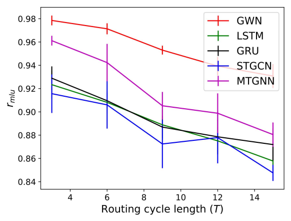
In this section, we perform ablation studies to evaluate the performance of different prediction models and investigate the impact of the routing cycle length, and traffic distribution on the proposed routing algorithm.

1) The Performance of Different DNN-Based Models: We demonstrate the effectiveness of the Graph WaveNet (GWN) model for long-term traffic prediction by comparing its prediction error with that of other deep neural network (DNN) models, namely LSTM [37], GRU [38], STGCN [39], and MTGNN [40]. We conduct the maximum traffic prediction with varying prediction steps T, which represent the length of the routing cycle, and investigate their impact on both the prediction error (measured as Mean Absolute Error) and the routing performance (measured as MLU ratio). We consider routing cycles of length 3 to 15 steps and perform each experiment five times to obtain the average results. Figure 16 illustrates the training processes of five different DNN models. Graph-based models including GWN, STGCN, and MTGNN can outperform RNN-based methods in the training process. Among them, GWN demonstrates superior performance in reducing the prediction error.

2) The Impact of the Routing Cycle Length: The relationship between the number of prediction steps and the error is illustrated in Figure 17. Overall, the GWN model outperforms the other prediction models in terms of MAE. In all cases, GWN achieves a reduction in prediction error ranging from 40% to 70% compared to the other models. Additionally,



(a) Abilene network.



(b) Geant network.

Fig. 18. The routing performance of different models with T varying from 3 to 15.

Figure 17 shows that increasing the length of the routing cycle leads to an increase in the prediction error. Figure 18 displays the MLU ratio of the network when using the predicted traffic demands from different prediction models. The results reveal that having the smallest error in traffic prediction, as achieved by the GWN model, helps to improve the MLU ratio. Furthermore, an increase in the length of the routing cycle leads to a degradation in network routing performance.

VI. Conclusion and Discussion

A. Conclusion

This paper presents a study on the use of multi-time-step segment routing (MTSR) with long-term traffic prediction to address the high network rerouting disturbance and high traffic monitoring overhead. To achieve this, we proposed a solution that utilizes traffic prediction to perform traffic engineering while reducing the number of flows that need to be rerouted. Given the complexity of the problem and the difficulty of multi-step traffic prediction, we formulated three versions of the MTSR problem and provided theoretical MTSR. We introduce an extension called MTSR-CS which combines the MTSR and the compressive sensing technique. In MTSR-CS, we only monitor and predict a subset of flows and use compressive sensing technique to reconstruct the full matrix from the partially predicted values, and hence, reduce the network monitoring overhead.

Our evaluation of different network datasets showed that our proposed approach can significantly reduce network disturbance and meet the requirements for minimizing the maximum link utilization. The experimental results on MTSR-CS demonstrated that the monitoring cost can be significantly reduced while maintaining routing performance comparable to full network monitoring.

B. Discussion and Future Directions

There are two main differences in our network model compared to other studies that use segment routing as the traffic routing scheme. First, we only use 2-segment routing (2-SR) [12] instead of utilizing n-segment routing (n-SR) [15]. As pointed out in [12], [41], [42], [43], [44], using 2-SR can achieve a close performance to n-segment routing while significantly reducing the problem complexity. In addition, the set of possible paths between two nodes in n-SR is generally larger than that of 2-SR, which increases the probability of path changes and network disturbance when solving the routing problem. Therefore, using 2-SR can naturally reduce the network disturbance. However, even when using 2-SR, the network disturbance remains high, as demonstrated by the experimental results presented in Section V. Second, since our main targets are to reduce the number of flow rerouting and the network monitoring overhead, we will not consider a network in which a traffic flow can be arbitrarily split and routed into different paths or ECMP routing. However, our problem formulation can easily be applied to the network system with arbitrarily split flows or ECMP routing by considering the variable α_{ij}^k in our problem formulation as a real number (i.e., $\alpha_{ij}^k \in [0,1]$ representing the split ratio of flow *ij*). It is important to emphasize that this adjustment exclusively pertains to the "problem solving" task (in Fig. 2) of our approach, and there exists no disparity in the training process of the deep neural network for traffic prediction.

Currently, our methods rely on a centralized controller for executing multiple tasks including collecting network traffic, performing traffic prediction, solving the routing problem, and distributing the routing rules. Nevertheless, when applied to large-scale networks, this approach brings forth several challenges, notably the risk of single-point failure, heightened network monitoring overhead, and substantial delays in the distribution of routing rules. Therefore, we intend to confront the aforementioned issues specific to large-scale networks through the implementation of an ML-based distributed routing algorithm levering the Multi-agent Reinforcement Learning technique [45].

Appendix A
Theoretical Analysis

Theorem 3:
- $\theta_1^* = u(\alpha_1^*)$; $\theta_2^* \geq u(\alpha_2^*)$; $\theta_3^* \geq u(\alpha_3^*)$
- $\theta_1^* \leq \theta_3^* \leq \theta_2^*$

Proof: According to (7), we have:

$$\theta_1^* \geq \frac{\sum_{ij}\sum_k g_{ij}^k(e)\big(\alpha_1^*\big)_{ij}^k m_{ij}^t}{c(e)} = u\big(t, e, \alpha_p^*\big)\ \forall t, e$$

As θ_1^* is the optimal solution of P_1, the following equation holds:

$$\theta_1^* = \max_{\forall t, e} \frac{\sum_{ij}\sum_k g_{ij}^k(e)\big(\alpha_1^*\big)_{ij}^k m_{ij}^{t_1}}{c(e)} = u(\alpha_1^*)$$

Concerning P_2, we have:

$$\begin{aligned}\theta_2^* &\geq \frac{\sum_{ij}\sum_k g_{ij}^k(e)\big(\alpha_2^*\big)_{ij}^k \max_t m_{ij}^t}{c(e)}\ \forall t, e\\ &\geq \frac{\sum_{ij}\sum_k g_{ij}^k(e)\big(\alpha_2^*\big)_{ij}^k m_{ij}^t}{c(e)}\ \forall t, e\\ \Rightarrow \theta_2^* &\geq \max_{\forall t, e} \frac{\sum_{ij}\sum_k g_{ij}^k(e)\big(\alpha_2^*\big)_{ij}^k m_{ij}^{t_2}}{c(e)} = u(\alpha_2^*)\end{aligned}$$

Similarly, we have: $\theta_3^* \geq u(\alpha_3^*)$. Now, we are going to prove that $\theta_1^* \leq \theta_3^* \leq \theta_2^*$.

First, we will prove the following hypothesis: *"If (α_3, θ_3) is a feasible solution of P_3, then it is also a feasible solution of P_1; if (α_2, θ_2) is a feasible solution of P_2, then it is also a feasible solution of P_3"*. According to this hypothesis, we derive that $\theta_1^* \leq \theta_3^*$ and $\theta_3^* \leq \theta_2^*$.

According to (15), we have:

$$\sum_{ij}\sum_{k} g_{ij}^k(e)(\alpha_3)_{ij}^k \max_{t\in T_p} m_{ij}^t \leq \theta_3 c(e) \ \forall T_p, e \tag{38}$$

As $\max_{t\in T_p} m_{ij}^t \geq m_{ij}^t$ $(\forall t \in T_p)$, from (38), we can derive that

$$\sum_{ij}\sum_{k} g_{ij}^k(e)(\alpha_3)_{ij}^k m_{ij}^t \leq \theta_3 c(e) \ \forall t, e \tag{39}$$

It means that (α_3, θ_3) satisfies constraint (7), thus it is a feasible solution of P_1.

Similarly, according to (11), we have:

$$\sum_{ij}\sum_{k} g_{ij}^k(e)(\alpha_2)_{ij}^k \max_{t\in T} m_{ij}^t \leq \theta_2 c(e) \ \forall e \tag{40}$$

Let T_p is an arbitrary sub-period of T, then $\max_{t\in T_p} m_{ij}^t \leq \max_{t\in T} m_{ij}^t$. Therefore, from (40), we have:

$$\sum_{ij}\sum_{k} g_{ij}^k(e)(\alpha_2)_{ij}^k \max_{t\in T_p} m_{ij}^t \leq \theta_2 c(e) \ \forall T_p, e \tag{41}$$

It means that (α_2, θ_2) satisfies constraint (15), thus it is a feasible solution of P_3 ■

Theorem 4: Let e^* be the link with the largest capacity, and e_2^* be the link where the equal sign of constraint (11) in P_2 holds; Then, the performance ratio of P_2 to P_1 is upper bounded by λ, which is calculated as follows.

$$\lambda = \frac{\sum_{ij} \max_t m_{ij}^t}{\max_t \max_{ij} m_{ij}^t} \frac{c(e^*)}{c(e_2^*)} \tag{42}$$

Proof: According to (11), we have

$$\sum_{ij}\sum_{k} g_{ij}^k(e_2^*)(\alpha_2^*)_{ij}^k \max_{t} m_{ij}^t = \theta_2^* c(e_2^*) \tag{43}$$

We also have

$$\sum_{ij}\sum_{k} g_{ij}^k(e_2^*)(\alpha_2^*)_{ij}^k \max_{t} m_{ij}^t \leq \sum_{ij} \max_t m_{ij}^t \tag{44}$$

Therefore, from (43) and (44), we deduce that

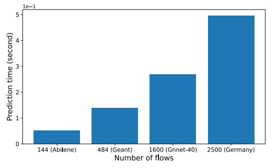
$$\theta_2^* c(e_2^*) \leq \sum_{ij} \max_t m_{ij}^t \tag{45}$$

Concerning P_1, from constraint (7) and the assumption that e^* has the largest capacity, we have

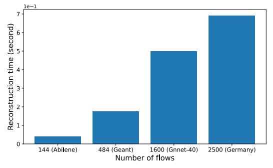
$$\theta_1^* c(e^*) \geq \theta_1^* c(e) \geq \sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k m_{ij}^t \quad \forall t; \forall e \tag{46}$$

As $\sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k m_{ij}^t$ is the total amount of traffic routed through link e at time-step t, it should be greater than or equal to the traffic of any pair ij routed through e, it means that

$$\sum_{ij}\sum_{k} g_{ij}^k(e)\alpha_{ij}^k m_{ij}^t \geq \max_{ij} m_{ij}^t \tag{47}$$



(a) The run time of traffic prediction.



(b) The run time of traffic reconstruction

Fig. 19. The run time of traffic prediction and traffic reconstruction tasks.

Eq. (47) holds for all time-steps t. Therefore, from (46) and (47), we deduce that

$$\theta_1^* c(e^*) \geq \max_t \max_{ij} m_{ij}^t \tag{48}$$

Finally, from (45) and (48), we have

$$\frac{\theta_2^*}{\theta_1^*} \leq \frac{\sum_{ij} \max_t m_{ij}^t c(e^*)}{\max_t \max_{ij} m_{ij}^t c(e_2^*)} \tag{49}$$

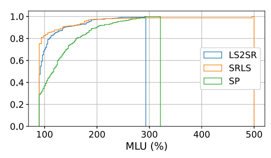
■

APPENDIX B
ABLATION STUDIES

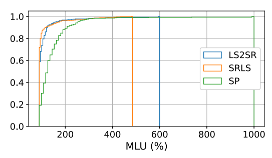
1) Traffic Prediction and Reconstruction Time: In this part, we measure the running time of the traffic prediction and matrix reconstruction tasks in the proposed MTSR and MTSR-CS methods (Fig. 19). All of the experiments are conducted in a single machine with 40 cores of Intel Xeon Silver 4210R CPU @ 2.40GHz, and an NVIDIA GeForce RTX 3090 card (cuda version 12.1). According to Fig. 19, the prediction and reconstruction times of MTSR and MTSR-CS are less than 0.5 seconds even with the large network. Therefore, the time for estimating future traffic demand is negligible. In addition, the prediction time in MTSR-CS can be smaller since we only consider L traffic flows.

2) The Scalability of the Proposed Routing Algorithm: To evaluate the performance of our proposed methods in cases of large-scaled networks, we use REPETITA dataset [46]. The dataset contains more than 200 topologies of the real backbone network, whose number of nodes varies from 4 to more than 100. For each topology, five traffic matrices were generated and adjusted so that the optimal values of the MLU (obtained from solving the MCF problem) are around 90%. This is the same dataset that was used to evaluate the scalability of the SRLS approach in [15]. The dataset was divided into three groups based on the number of nodes in the network topology. Group 1 comprised networks with less than 20 nodes, Group 2 comprised networks with 20 to 40 nodes, and Group 3 contained large networks with more than 40 nodes. We compared the performance of our approach with SRLS [15] and the Shortest Path (SP) routing approach. It is worth noting that SRLS used n-segment routing for solving the TE problem while LS2SR only used 2-segment routing.

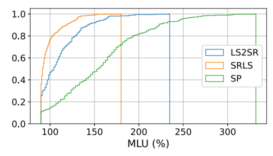
We conducted the experiment using five synthetic traffic matrices of each topology and calculated the MLU, the number of rerouting flows per time-step (RC), and the average delay as



(a) Small networks

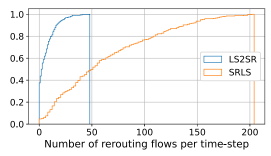


(b) Medium networks

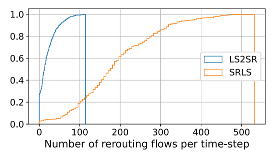


(c) Large networks

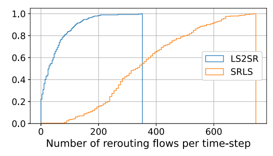
Fig. 20. The maximum link utilization of different groups of network topology.



(a) Small networks

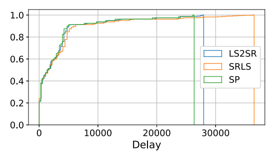


(b) Medium networks

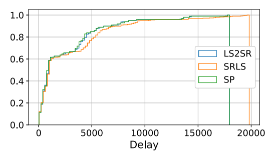


(c) Large networks

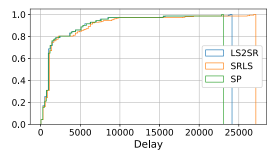
Fig. 21. The number of rerouting flows per time-step of different groups of network topology.



(a) Small networks



(b) Medium networks



(c) Large networks

Fig. 22. The average delay per time-step of different groups of network topology.

performance metrics. The delay was computed by averaging the delay of all the traffic flows, which was calculated as the sum of link delays in its path. The link delays provided in the REPETITA dataset represented the geometrical distance between two nodes. Note that since SP has zero rerouting flow, its results were not shown in Fig. 21. The results of the experiment, shown in Fig. 20, Fig. 21, and Fig. 22, indicate that our proposed approach, LS2SR, achieved the same performance in terms of MLU as SRLS. However, LS2SR had the best results in reducing the number of rerouting flows and had similar results compared to SP, consistently outperforming SRLS regarding the average delay metric.

References

[1] "Cisco annual Internet report (2018–2023)," Cisco Syst., Inc., San Jose, CA, USA, White Paper, 2020.

[2] Y. Xiao, J. Liu, J. Wu, and N. Ansari, "Leveraging deep reinforcement learning for traffic engineering: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2064–2097, 4th Quart., 2021.

[3] R. Cârpa, 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 *Proc. 13th Int. Conf. Netw. Serv. Manag. (CNSM)*, 2017, pp. 1–9.

[4] J. Wang et al., "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[5] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in SDN," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, 2018, pp. 1–5.

[6] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, and W. Zhang, "Interactive temporal recurrent convolution network for traffic prediction in data centers," *IEEE Access*, vol. 6, pp. 5276–5289, 2018.

[7] V. A. Le, P. Le Nguyen, and Y. Ji, "Deep convolutional LSTM network-based traffic matrix prediction with partial information," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manag. (IM)*, 2019, pp. 261–269.

[8] V. A. Le, T. T. Le, P. L. Nguyen, H. T. T. Binh, R. Akerkar, and Y. Ji, "GCRINT: Network traffic imputation using graph convolutional recurrent neural network," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[9] V. A. Le, T. T. Le, P. L. Nguyen, H. T. T. Binh, and Y. Ji, "Multi-time-step segment routing based traffic engineering leveraging traffic prediction," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, 2021, pp. 125–133.

[10] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, "Optimal oblivious routing in polynomial time," in *Proc. 35th Ann. ACM Symp. Theory Comput.*, 2003, pp. 383–388.

[11] H. Räcke, "Optimal hierarchical decompositions for congestion minimization in networks," in *Proc. 14th Annu. ACM Symp. Theory Comput.*, 2008, pp. 255–264.

[12] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 657–665.

[13] V. Pereira, M. Rocha, and P. Sousa, "Traffic engineering with three-segments routing," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1896–1909, Sep. 2020.

[14] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "CG4SR: Near optimal traffic engineering for segment routing with column generation," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1333–1341.

[15] S. Gay, R. Hartert, and S. Vissicchio, "Expect the unexpected: Sub-second optimization for segment routing," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[16] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic engineering using segment routing and considering requirements of a carrier IP network," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1851–1864, Aug. 2018.

[17] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, "Failure resiliency with only a few tunnels—Enabling segment routing for traffic engineering," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 262–274, Feb. 2021.

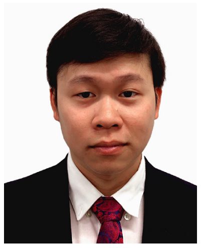
[18] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2249–2259, Oct. 2020.

[19] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "FlexDATE: Flexible and disturbance-aware traffic engineering with reinforcement learning in software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 4, pp. 1433–1448, Aug. 2023.

[20] M. Ye, Y. Hu, J. Zhang, Z. Guo, and H. J. Chao, "Reinforcement learning-based traffic engineering for QoS provisioning and load balancing," in *Proc. IEEE/ACM 31st Int. Symp. Qual. Serv. (IWQoS)*, 2023, pp. 1–10.

[21] L. Nie, D. Jiang, L. Guo, and S. Yu, "Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks," *J. Netw. Comput. Appl.*, vol. 76, pp. 16–22, Dec. 2016.

[22] G. Kakkavas, M. Kalntis, V. Karyotis, and S. Papavassiliou, "Future network traffic matrix synthesis and estimation based on deep generative models," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2021, pp. 1–8.
[23] "Software-defined networking: The new norm for networks," ON Foundation, Luzern, Switzerland, White Paper, 2012.
[24] S. Sivabalan et al., "PCEP extensions for segment routing," Internet Eng. Task Force, RFC 8664, 2014.
[25] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.
[26] L. A. Dias Knob, R. P. Esteves, L. Z. Granville, and L. M. Rockenbach Tarouco, "Mitigating elephant flows in SDN-based IXP networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2017, pp. 1352–1359.
[27] F. Tang, H. Zhang, L. T. Yang, and L. Chen, "Elephant flow detection and load-balanced routing with efficient sampling and classification," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1022–1036, Jul.–Sep. 2021.
[28] W. Wang, Y. Sun, K. Salamatian, and Z. Li, "Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 1, pp. 5–18, Mar. 2016.
[29] A. v. D. Oord et al., "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.
[30] github.com. [Online]. Available: https://github.com/vananle/TNSM2023
[31] D. Jiang, W. Wang, L. Shi, and H. Song, "A compressive sensing-based approach to end-to-end network traffic reconstruction," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 507–519, Jan.–Mar. 2020.
[32] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0—Survivable network design library," in *Proc. 3rd Int. Netw. Optim. Conf. (INOC 2007)*, 2007, pp. 276–286. [Online]. Available: https://onlinelibrary.wiley.com/action/showCitFormats?doi=10.1002%2Fnet.20371
[33] (Barcelona Neural Netw. Center, Barcelona, Spain). *Graph Neural Networking Challenge 2021 Creating a Scalable Network Digital Twin*. 2021. [Online]. Available: https://bnn.upc.edu/challenge/gnnet2021/
[34] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "DATE: Disturbance-aware traffic engineering with reinforcement learning in software-defined networks," in *Proc. IEEE/ACM 29th Int. Symp. Qual. Serv. (IWQOS)*, 2021, pp. 1–10.
[35] S. Mitchell. "Pulp 2.7.0." github.com. 2022. [Online]. Available: https://coin-or.github.io/pulp/
[36] github.com. 2019. [Online]. Available: https://github.com/nnzhan/Graph-WaveNet
[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[38] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
[39] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3634–3640.
[40] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2020, pp. 753–763.
[41] F. Hao, M. Kodialam, and T. V. Lakshman, "Optimizing restoration with segment routing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[42] T. Settawatcharawanit, Y.-H. Chiang, V. Suppakitpaisarn, and Y. Ji, "A computation-efficient approach for segment routing traffic engineering," *IEEE Access*, vol. 7, pp. 160408–160417, 2019.
[43] X. Li and K. L. Yeung, "Traffic engineering in segment routing using MILP," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.
[44] P. L. Ventre et al., "Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 182–221, 1st Quart., 2021.
[45] T. Li et al., "Applications of multi-agent reinforcement learning in future Internet: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 1240–1279, 2nd Quart., 2022.
[46] S. Gay, P. Schaus, and S. Vissicchio, "Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017, *arXiv:1710.08665*.



Van An Le received the B.E. degree in computer engineering from the University of Technology, Ho Chi Minh City, Vietnam, in 2016, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan, in 2022. He is currently a Postdoctoral Researcher with the National Institute of Advanced Industrial Science and Technology, Japan. His research interests include machine learning, network resource management, and mobile-edge computing.

Yusheng Ji (Fellow, IEEE) received the B.E., M.E., and D.E. degrees in electrical engineering from the University of Tokyo. She joined the National Center for Science Information Systems, Tokyo, Japan, in 1990. She is currently a Professor with the National Institute of Informatics, Tokyo, and the Graduate University for Advanced Studies, SOKENDAI, Japan. Her research interests include network resource management and mobile computing. She is an Associate Editor of *IEEE Vehicular Technology Magazine*, served as an Editor for IEEE TRANSACTIONS OF VEHICULAR TECHNOLOGY, a TPC Co-Chair of INFOCOM 2023, a General Co-Chair of ICT-DM 2018, MSN2020, BigCom2023, a Symposium Co-Chair of IEEE GLOBECOM 2012, 2014, and ICC 2020, and a Track Co-Chair of IEEE VTC 2016 Fall and 2017 Fall. She is a Distinguished Lecturer of IEEE Vehicular Technology Society.

Huu Huy Tran received the bachelor's degree from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2023, where he is currently a Research Assistant with the School of Information and Communication Technology. His research interests include optimization, time-series prediction, machine learning, and deep learning.

Phi Le Nguyen (Member, IEEE) received the B.E. and M.S. degrees from the University of Tokyo in 2007 and 2010, respectively, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan, in 2019. She is currently a Lecturer with the School of Information and Communication, Hanoi University of Science and Technology, Vietnam. Her research interests include network architecture, optimization, and artificial intelligence.

John C. S. Lui (Fellow, IEEE) received the Ph.D. degree in computer science from UCLA. He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests are in online learning algorithms and applications (e.g., multi-armed bandits and reinforcement learning), quantum Internet, machine learning on network sciences and networking systems, network economics, large scale storage systems, and performance evaluation theory. He is an active consultant to industry, believing that it is an effective way to do technology transfer and a wonderful way to learn about real and relevant research problems. He is an Elected Member of the IFIP WG 7.3, a Fellow of ACM and Hong Kong Academy of Engineering Sciences, a Senior Research Fellow of the Croucher Foundation, and was the past Chair of the ACM SIGMETRICS from 2011 to 2015.