Multi-time-step Segment Routing based Traffic Engineering Leveraging Traffic Prediction

Van An Le*†, Tien Thanh Le‡, Phi Le Nguyen‡ Huynh Thi Thanh Binh‡, Yusheng Ji†*
*Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan
†National Institute of Informatics, Tokyo, Japan
‡Hanoi University of Science and Technology, Hanoi, Vietnam
Email: *†{anle, kei}@nii.ac.jp; ‡thanh.ltcb190211@sis.hust.edu.vn; ‡{lenp, binhht}@soict.hust.edu.vn

***Abstract*—Based on the concept of source routing, Segment Routing (SR) allows the source or ingress node to inject a sequence of segment labels into the packet header and specify the routing path. Due to the routing flexibility, SR has been widely used to solve traffic engineering problems such as minimizing the maximum link utilization. However, most of the prior works only solve the problem in a single snapshot without considering network traffic dynamics, resulting in frequent traffic reroute. To cope with this problem, we focus on solving the segment routing based traffic engineering problem by taking into account the future traffic changes. We formulate the multi-time-step segment routing problem and leverage traffic prediction to extend the length of routing cycles. Due to the large search space of multi-time-step segment routing problem, we further propose a heuristic algorithm for incrementally recomputing the segment routing paths in sub-second. Through extensive experiments on real backbone network traffic datasets, we show that our proposal can achieve a near-optimal performance in term of maximum link utilization while significantly reducing the number of routing changes.**

I. Introduction

During the last few years, Segment Routing (SR for short) has gained increasing worldwide attention from both the academic and industrial community as a powerful tool to solve the traffic engineering-related problems. The key idea of Segment Routing is to divide the routing path into segments to control better, and thereby improve network utilization. SR allows the source node (or ingress node) to inject a sequence of segment labels into the packet header to specify the routing path. Due to routing flexibility, SR is widely used to address Traffic Engineering (TE) related problems. According to [1], three different TE objectives have been covered by the prior works, i.e., minimizing network energy, optimizing network congestion, and minimizing the number of rejected requests. In this paper, our objective is to explore the SR to minimize the maximum link utilization (i.e., the TE problem) and thereby, avoid network traffic congestion.

The literature shows that the TE problem has been well studied in both theoretical and approximation approach. In [2], Bhatia, Hao, Kodialam, *et al.* introduced two network scenarios concerning 2-segment routing (i.e., the routing path only contains two segments) in which the TE problem was solved with and without the knowledge of traffic matrix. In [3], authors considered the unexpected traffic fluctuation and link failures problems. They then proposed a local search-based algorithm to solve the targeted problems. Later on, Jadin, Aubry, Schaus, *et al.* in [4] attained a further improvement on the TE problems with SR by fully exploiting the SR architecture (both node and edge segments) and proposed the first Column Generation-based approach. In [5] and [6], the authors used the Segment Routing to enhance the routing management in Software Defined Networks. They proposed efficient routing algorithms that can solve the scalability issues and improve traffic distribution.

Although the TE problem with the Segment Routing issue has been well studied, most of the solutions proposed so far only addressed the local optimization, where they considered the problem in only a single snapshot. Specifically, the primary approach is to formulate the problem under a mixed-integer linear programming model, and then propose heuristic algorithms using various techniques such as local search (**srls**) [3], column generation (**cg4sr**) [4]. This approach usually used the traffic matrix (i.e., traffic demands) of the corresponding snapshot as the input. However, due to the network behavior's dynamic, the traffic matrix often varies over the snapshots; thus, leading to the changes in the routing policy obtained. In some rare work [2], the authors proposed a Traffic Matrix Oblivious Segment Routing which does not require the traffic matrix to be given prior. The proposed routing policy was designed to work well for a wide range of traffic demands. Although, by using a fixed routing policy, this approach can alleviate the overhead caused by routing path change, it may not guarantee the link utilization constraint, especially with the network's dynamic behavior. Besides, it was pointed out in [3] that the Traffic Matrix Oblivious Segment Routing has been shown to only be practical for offline traffic engineering on relatively-small networks.

In this paper, we study the TE problem globally to overcome the aforementioned problems. Specifically, we aim at designing a Segment routing protocol in a long time horizon, i.e., Multi-Time-step Segment Routing (MTSR for short), which minimizes the maximum link utilization. In this paper, instead of considering the full Segment routing algorithm, we consider 2-Segment Routing (2-SR for short). In 2-SR, a routing path is divided into two segments going through an intermediate node. The reasons for choosing 2-SR are two folds. On the one hand, by lessening the number of intermediate nodes, we can

978-3-903176-32-4

reduce the targeted problem's complexity. On the other hand, as pointed out in [1], [2], fully leveraging SR can only obtain a slightly better solution but causing a very high computation cost.

The main idea of our proposed approach is as follows. First, we equally divide a time horizon into time-steps and formulate the MTSR problem with 2-SR. Assume that we can acquire the accurate predicted traffic matrices of the next future T time-steps by leveraging advanced prediction models. Then, instead of using the current traffic matrix as input for solving the TE problem like in the existing approaches, we use the predicted traffic matrices for our MTSR problem. By this way, the routing policy obtained by solving the MTSR problem can work well without changing in the next T time-step. The period of T time-steps is called a routing cycle, and the MTSR problem is repeatedly solved after each cycle to update the routing policy.

It is worth noting that the MTSR problem's performance is significantly affected by the multi-step traffic prediction, which remains a big challenging problem despite tremendous efforts from researchers. Therefore, we design two modified versions of the MTSR problem that reduce the traffic prediction's heavy tasks. In the first version, instead of using predicted traffic matrices of all the next T time-steps, we only use the predicted maximum demand of each flow. In the second version, we combine multiple consecutive cycles into equal periods and then predict each flow's maximum demands in each period.

To further reduce the number of re-routed traffic flows, we propose a heuristic algorithm (called LS2SR), which lessens the difference between adjacent cycles' routing policies. The idea is to utilize the previous cycle's routing paths as the input of the MTSR problem in the next cycle. The heuristic algorithm also helps tackle the scalability issues; thereby, providing a better solution in the context of time-constrained optimization. The contribution of this paper can be summarized as follows:

- To the best of our knowledge, we are the first to consider the routing path change problem in the Traffic Engineering and formulate the multi-time-step segment routing problem with 2-SR as an Integer Linear Programming problem. We propose a heuristic algorithm to solve that problem in the context of time-constrained optimization.
- Although prediction accuracy is the main focus in most traffic matrices prediction studies [7], [8], it is not strongly related to the performance of downstream applications such as TE. In [9], the authors evaluated the impact of prediction error on traffic engineering problems. However, the evaluation scenarios are still simple when only short-term traffic prediction is used as input to solve the simple TE problem. In this work, we are the first to leverage the multi-step traffic prediction results to address the TE problem. We design two modified versions of the TE problem that consider the impact of the prediction accuracy.
- We conduct extensive experiments using different real backbone network datasets to evaluate the performance of our proposed approach.

The rest of the paper is organized as follows: we first introduce the problem of TE with 2-Segment Routing in Section II. Section III presents our formulations of the multi-time-step segment routing problem and some theoretical analysis. Section IV describes the heuristic algorithm to solve the proposed problems. We show extensive experimental results to evaluate our proposed approaches in Section V and conclude the paper in Section VI.

II. Traffic Engineering problem with 2-SR

In this section, we briefly introduce the TE problem with 2-SR. This problem was introduced in [2] as a traffic matrix aware segment routing. Therefore, some notations and figures from [2] are reused in this paper. However, in contrast with their problem, we do not consider that the traffic flows can be arbitrarily split and routed by different paths.

The network is represented by an undirected graph $G = (V, E)$, where V is the set of nodes ($|V| = N$), and E is the set the network's links. Each link $e \in E$ has a capacity $c(e)$. Let $M_t \in \mathbb{R}^{N \times N}$ be the traffic matrix at time-step t and $m_{ij}^t \in M_t$ denote the traffic flow from node i to j (flow ij for short, $i, j \in V$) at time step t. Let α_{ij}^k be the binary variable which $\alpha_{ij}^k = 1$ indicates that flow ij is routed through intermediate node k, and otherwise. Therefore α_{ij}^k can be considered as the routing policy. We assume that the network is controlled by a central controller such as SDN controller [10]. The controller plays an important role in collecting the knowledge of the network (i.e., topology, traffic statistics), predicting future traffic demands, and applying the routing policy to devices via PCEP [11]. However, the implementation of the controller is out-of-scope and will be omitted in this paper.

In 2-SR, we only need to select one intermediate node k for each flow ij. Figure 1 shows the example of 2-segment routing path for flow ij with the intermediate node k. The traffic from i to k and from k to j is routed through the shortest path between them. The intermediate node $k = i$ or $k = j$ means that flow ij is routed by the shortest path from i to j. The problem (called P_0) can be formulated as the following integer linear program. The variable θ represents the maximum link utilization.

P_0 :

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

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

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

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

We have $g_{ij}^k(e) = f_{ik}(e) + f_{kj}(e)$, in which $f_{ik}(e) = 1$ if link e is on the shortest path from i to k of flow ij with intermediate node k and $f_{ik}(e) = 0$, otherwise. Note that in P_0, the maximum link utilization θ can be greater than 1, which means the network may be congested. Equation (2) and (4) ensure that all traffic from i to j is routed and it cannot be

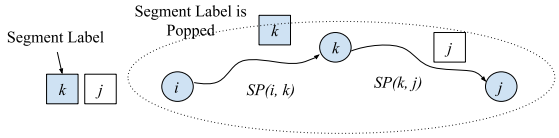


Fig. 1: Illustration of 2-segment routing [2].

split into different paths. Equation (3) depicts that total traffic load on link e is not greater than the link's capacity.

By solving the problem above, we can get a routing policy for a single time-step t. Although we can also reuse the same routing policy for the next T time-step to mitigate the routing path's variation, the link utilization constraint may not be assured due to network traffic's dynamic behavior. Therefore, for future time-steps, we need to resolve the problem and update the routing policy. A trivial approach to minimize the maximum link utilization in the next T time-steps is to solve the problem P_0 at every time-step. As mentioned in Section 1, this approach may lead to a considerable number of re-routed flows. To this end, in the next section, we propose an extension of P_0, which addresses the segment routing problem in multiple steps. We first present the mathematical formulation and then describe some theoretical analysis.

III. The multi-time-step Segment Routing Problem

A. Problem formulations

In this section, we present the formulation of the multi-time-step segment routing problem (called P_1) and its modified versions (called P_2 and P_3). As mentioned in Section I, by solving the problems, we obtain a routing policy that can be applied for the next routing cycle, i.e., next T time-steps.

Assume that $M = [M_1, M_2, ..., M_T]$ are the predicted traffic matrices of the next T future time-steps, which can be acquired by using the state-of-the-art prediction models such as [8], [12]. We extend the problem P_0 for T future steps by adding more constraints related to the traffic demands of each time-step.

P_1 :

$$\text{minimize} \quad \theta \tag{5}$$

$$\sum_{k \in V} \alpha_{ij}^k = 1 \quad \forall i, j \in V \tag{6}$$

$$\sum_{ij} \sum_{k} g_{ij}^k(e) \alpha_{ij}^k m_{ij}^t \leq \theta c(e) \quad \forall e \in E; \forall t \in T \tag{7}$$

$$\alpha_{ij}^k \in \{0, 1\} \quad \forall i, j, k \in V \tag{8}$$

It can be seen that P_1 is different from P_0 by the link utilization constraints (7). Specifically, in P_1, the routing policy α_{ij}^k needs to satisfy each link's capacity constraints at every time-step. Due to the tight constraints, problem P_1 becomes more complicated than P_0. Besides that, P_1 requires to predict all the traffic matrices of the next T time-steps, which remains a significant challenge even with the most state-of-the-art deep learning models. Obviously, the prediction accuracy plays a vital role in the performance of the problem P_1. Unfortunately, the results of some proposed prediction models [8], [12] show that the prediction performance worsens when the prediction step increases. Therefore, to reduce the problem complexity and alleviate the traffic prediction task's burden, we formulate problem P_2 and P_3, which are loose versions of P_1.

P_2 :

$$\text{minimize} \quad \theta \tag{9}$$

$$\sum_{k \in V} \alpha_{ij}^k = 1 \quad \forall i, j \in V \tag{10}$$

$$\sum_{ij} \sum_{k} g_{ij}^k(e) \alpha_{ij}^k \max_{t \in T} m_{ij}^t \leq \theta c(e) \quad \forall e \in E \tag{11}$$

$$\alpha_{ij}^k \in \{0, 1\} \quad \forall i, j, k \in V \tag{12}$$

In problem P_2, we only consider the maximum values of every flow ij over the next T time-steps. By doing that, P_2 has the same number of constraints as P_0, and also, we have to predict only one traffic matrix whose element is the maximum value of each traffic flow.

P_3 :

$$\text{minimize} \quad \theta \tag{13}$$

$$\sum_{k \in V} \alpha_{ij}^k = 1 \quad \forall i, j \in V \tag{14}$$

$$\sum_{ij} \sum_{k} g_{ij}^k(e) \alpha_{ij}^k \max_{t \in T_p} m_{ij}^t \leq \theta c(e) \quad \forall e \in E; \forall T_p \tag{15}$$

$$\alpha_{ij}^k \in \{0, 1\} \quad \forall i, j, k \in V \tag{16}$$

In problem P_3, the routing cycle is divided into P sub-periods in which each of them has T_p time-steps ($T_p < T$). Then, similar to P_2, we formulate the problem with the maximum values of flow ij in each sub-period T_p. Accordingly, to solve P_3, we only need to predict P traffic matrices, which are the maximum traffic of every flow ij in each sub-period T_p. The number of predicted traffic matrices depends on the way we divide the routing cycle.

B. Traffic prediction for multi-time-step segment routing

As mentioned in the previous section, accurately predicted traffic matrices are required as input for all the problem formulations P_1, P_2, and P_3. However, our objective is not to develop a new prediction model but to leverage the existing models' results for addressing the traffic engineering problem. From the literature review, although there are a large number of proposed models for traffic matrices prediction, most of them only focus on improving prediction accuracy [7], [8]. We modify the prediction model to meet the requirements of the 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) [12] as the prediction model. GWN can extract temporal and spatial features in the historical network data, which help achieve better prediction accuracy than other time-series models such as Long Short-Term Memory (LSTM) and AutoRegressive Integrated Moving Average (ARIMA).

C. Theoretical analysis

In this part, we theoretically analyze the performance ratios of P_2, and P_3 to P_1. We assume that (θ_1^*, α_1^*), (θ_2^*, α_2^*), and (θ_3^*, α_3^*) are the optimal solutions obtained by solving P_1, P_2, and P_3, respectively. We denote by $u(t, e, \alpha_p^*)$ 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.

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

Proof. According to (7), we have:

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

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

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

Concerning P_2, we have:

$$\begin{aligned} \theta_2^* &\geq \frac{\sum_{ij} \sum_k g_{ij}^k(e)(\alpha_2^*)_{ij}^k \max_t m_{ij}^t}{c(e)} \quad \forall t, e \\ &\geq \frac{\sum_{ij} \sum_k g_{ij}^k(e)(\alpha_2^*)_{ij}^k m_{ij}^t}{c(e)} \quad \forall t, e \\ \Rightarrow \theta_2^* &\geq \max_{\forall t,e} \frac{\sum_{ij} \sum_k g_{ij}^k(e_2)(\alpha_2^*)_{ij}^k m_{ij}^{t_2}}{c(e_2)} = u(\alpha_2^*) \end{aligned}$$

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:

$$\sum_{ij} \sum_k g_{ij}^k(e)(\alpha_3)_{ij}^k \max_{t \in T_p} m_{ij}^t \leq \theta_3 c(e) \quad \forall T_p, e \tag{17}$$

As $\max_{t \in T_p} m_{ij}^t \geq m_{ij}^t$ $(\forall t \in T_p)$, from (17), we can derive that

$$\sum_{ij} \sum_k g_{ij}^k(e)(\alpha_3)_{ij}^k m_{ij}^t \leq \theta_3 c(e) \quad \forall t, e \tag{18}$$

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:

$$\sum_{ij} \sum_k g_{ij}^k(e)(\alpha_2)_{ij}^k \max_{t \in T} m_{ij}^t \leq \theta_2 c(e) \quad \forall e \tag{19}$$

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 (19), we have:

$$\sum_{ij} \sum_k g_{ij}^k(e)(\alpha_2)_{ij}^k \max_{t \in T_p} m_{ij}^t \leq \theta_2 c(e) \quad \forall T_p, e \tag{20}$$

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

In the following, we will analysis the performance ratio of P_2 and P_3 to P_1. As $\theta_2^* \geq \theta_3^*$, we will derive the upper bound of $\frac{\theta_2^*}{\theta_1^*}$ which will be also the upper bound of $\frac{\theta_3^*}{\theta_1^*}$.

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 is upper bounded by $\lambda = \frac{\sum_{ij} \max_t m_{ij}^t}{\max_t \max_{ij} m_{ij}^t} \frac{c(e^*)}{c(e_2^*)}$.*

Proof. According to (11), we have

$$\sum_{ij} \sum_k g_{ij}^k(e_2^*)(\alpha_2^*)_{ij}^k \max_t m_{ij}^t = \theta_2^* c(e_2^*) \tag{21}$$

We also have

$$\sum_{ij} \sum_k g_{ij}^k(e_2^*)(\alpha_2^*)_{ij}^k \max_t m_{ij}^t \leq \sum_{ij} \max_t m_{ij}^t \tag{22}$$

Therefore, from (21) and (21), we deduce that

$$\theta_2^* c(e_2^*) \leq \sum_{ij} \max_t m_{ij}^t \tag{23}$$

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

$$\theta_1^* c(e^*) \geq \theta_1^* c(e) \geq \sum_{ij} \sum_k g_{ij}^k(e) \alpha_{ij}^k m_{ij}^t \quad \forall t; \forall e \tag{24}$$

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

$$\sum_{ij} \sum_k g_{ij}^k(e) \alpha_{ij}^k m_{ij}^t \geq \max_{ij} m_{ij}^t \tag{25}$$

(25) holds for all time-steps t. Therefore, from (24) and (25), we deduce that

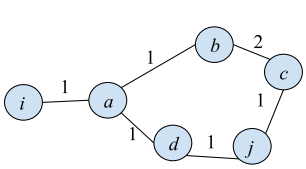
$$\theta_1^* c(e^*) \geq \max_t \max_{ij} m_{ij}^t \tag{26}$$

Finally, from (23) and (26), we have

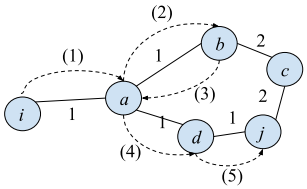
$$\frac{\theta_2^*}{\theta_1^*} \leq \frac{\sum_{ij} \max_t m_{ij}^t c(e^*)}{\max_t \max_{ij} m_{ij}^t c(e_2^*)}$$

$\square$

Suppose that all the network links have the same capacity, then the performance ratio λ largely depends on the current network situation. If all traffic flows in the network have similar demands and stable within the routing cycle, we have $\lambda \gg 1$. However, since most of traffic flows in the network are mice flows and the network traffic is extremely dynamic,



(a) Equivalent paths



(b) Non simple paths

Fig. 2: Examples of redundant paths in 2-segment routing.

we have $\sum_{ij} \max_t m_{ij}^t \approx \max_t \max_{ij} m_{ij}^t$, and then $\lambda \approx 1$. Therefore, by solving the multi-time-step segment routing with formulation P_2, we can reduce the problem complexity while still achieving a good routing policy.

IV. Local search algorithm for solving multi-time-step segment routing problem

The MTSR problem described in Section III is an integer programming problem with a vast search space. Traditional MILP solvers are hardly scaled to large topologies with more than 20 nodes [3]. Thus, using heuristic or meta-heuristic algorithms would be the most practical way to solve this problem rapidly. Gay, Hartert, and Vissicchio proposed a local search algorithm to solve the n-segment routing algorithm. We adapt the algorithm proposed in [3] to propose a new algorithm that effectively solves the multi-step segment routing problem by exploiting the structure of 2-segment routing called Local Search 2 Segment Routing (LS2SR). We also improve LS2SR on solving the MTSR problem by adding a mechanism to refine the routing policy from the previous cycle, thereby lessening the routing policy variation over different cycles.

A. Search space reduction technique

We begin by investigating the search space of the MTSR problem. In Section III, we have mathematically formulated the MTSR problem by using binary variables α_{ij}^k which indicates whether a traffic flow from a source node i to a destination node j goes through an intermediate node k. These formulations contain two major sources of redundancy, namely equivalent paths and non-simple paths.

Figure 2a shows examples of equivalence paths. Consider a flow ij with a routing path $i \to a \to d \to j$ going through two intermediate nodes a and d. In the formulations provided in Section III, the path $i \to a \to d \to j$ is duplicated in the search space as it is counted as a routing path going through a (i.e., $\alpha_{ij}^a = 1$), and also as a routing path going through d (i.e., $\alpha_{ij}^d = 1$).

The second type of redundancy is caused by so-called non-simple paths, i.e., paths that visit a link more than once. Figure 2b illustrates an example of non-simple path. In this example, the routing path from i to j with intermediate node b: $i \to a \to b \to a \to d \to j$ (represented by dotted line) visits link (a, b) twice. Obviously, by pruning the looped route $a \to b \to a$ we obtain more efficient path $i \to a \to d \to j$. Therefore, including such non-simple paths in the search space only causes redundant time complexity but can not improve the solution's goodness. It is obvious that the route constructed by middle point a will cost less than that of middle point b.

In the following, we propose a method to construct the routing path search space. Let $\mathcal{P}$ be the set of 2-segment routing paths of all the source and destination pairs, and $\mathcal{P}_{ij} \in \mathcal{P}$ denotes the set of 2-segment routing paths from node i to j. We construct $\mathcal{P}$ as follows. First, we enumerate all possible 2-segment paths from node i to j (with all possible intermediate nodes k). Then, we remove from $\mathcal{P}_{ij}$ ($\forall i, j \in V$) all the redundant paths defined as above. To reduce the search space, instead of using binary variables α_{ij}^k, we use integer variables x_{ij} which indicates the index of a routing path in $\mathcal{P}_{ij}$, i.e., $x_{ij} \in \{0, 1, ..., |\mathcal{P}_{ij}|\}$. Finally, a routing policy at routing cycle c is defined by the combination of all x_{ij} $\forall i, j \in V$, denoted as $x^c = [x_{ij}]$.

In the following, we first present the link and flow selection strategy in Section IV-B and then describe the details of LS2SR in Section IV-C.

B. Link and flow selection strategy

LS2SR falls into the category of improvement-based heuristic algorithms, which starts from a feasible solution and improves the solution by applying successive changes. In particular, the initial solution of LS2SR is the shortest path routing. Then, at each iteration, LS2SR selects a flow, which is crucial to minimize the maximum link utilization and alters its path. Motivated by the algorithm proposed in [3], we design a critical flow selection strategy in which the link with higher traffic load will have a higher chance of being selected. Specifically, a link e is selected with a probability p_e defined as

$$p_e = \frac{load(e)^\beta}{\sum_{e \in E} load(e)^\beta} \tag{27}$$

where $\beta \in [0, +\infty)$ is the *intensification coefficient*. If β is high, then the selection distribution will be short-tailed. The link selection method will select a small subset of the highly loaded edges. Otherwise, the selections becomes a more diverse. The reason behind this heuristic is that changing the path of flow passing through the highly loaded link will reduce the load in that link, thereby reducing the maximum link utilization of the whole network.

Subsequently, we select a flow passing though the selected link e, by the probability $p_{i,j}$:

$$p_{i,j} = \frac{(m_{ij})^\gamma}{\sum_{ij \in \mathcal{D}(e)} (m_{ij})^\gamma} \tag{28}$$

where $\mathcal{D}(e)$ denotes the set of flows routed on link e, and γ is the *intensification coefficient* for this selection distribution. Therefore, the large flow routed though the highly loaded link e is rerouted.

C. Local search for multi-time-step segment routing

In this part, we present the details of LS2SR. To alleviate the routing path change, in LS2SR, the routing path in the first cycle is initialized by the shortest path; In each subsequent cycle, the initial routing policy is the routing policy obtained

Algorithm 1 LS2SR Algorithm

1: **Input:** network $G(V, E)$; traffic matrix M; initial solution x^{c-1}, set of all the routing paths $\mathcal{P}$
2: **Output:** Routing path x^{*c}
3: $\mathcal{P} = [sort(\mathcal{P}_{ij})] \quad \forall i, j \in V$
4: $x^{*c} = x^{c-1}$; $\theta^* = MLU(G, x^{*c}, M)$
5: **while** not timeout **do**
6: $\quad e = select\_link(G, M, x^{*c})$;
7: $\quad i, j = select\_flow(G, M, e, x^{*c})$
8: $\quad x^c = select\_path(x^{*c}, \mathcal{P}_{ij})$; $\theta = MLU(G, x^c, M)$
9: $\quad$ **if** $\theta^* - \theta > \epsilon$ **then**
10: $\quad\quad x^{*c} = x^c$; $\theta^* = \theta$
11: $\quad$ **end if**
12: **end while**

TABLE I: Details of the real traffic datasets

Dataset	Brain	Abilene	Geant
Number of nodes	9	12	22
Number of links	14	15	36
Granularity (minute)	1	5	15

from the previous cycle. The details of LS2SR are presented in Algorithm 1. We first sort the paths in $\mathcal{P}_{ij}$ by the increasing order of their cost (the sum of all the link cost in the path) and calculate the maximum link utilization corresponding to the initial solution (line 4). The loop from line 5 to 12 is to search for a new routing solution that may reduce the maximum link utilization. In each iteration, we first select the link e and flow ij based on the selection strategies described in Section IV-B. Then, we choose a new path from $\mathcal{P}_{ij}$ for flow ij (line 8). The new path is chosen systematically as follows: as $\mathcal{P}_{ij}$ has been sorted by the paths' cost, we select the path that has the next higher cost than the current path. The rationale behind this path selection operator is to avoid the highly loaded link while trying not to select the path with significantly high cost. If the maximum link utilization of the newly founded solution x^c is significantly lower than the current best solution (i.e. x^{*c}), then we update the current best solution as line 10. After that, the heuristic algorithm continues to find a new possible solution until the time limit has reached. The parameter ϵ is defined to determine whether x^{*c} is updated and help to reduce the number of rerouted flows. We can easily obtain the routing policy α_{ij}^k after getting the final solution x^{*c}.

V. Experimental study

We design three types of experiment to evaluate our approach on MTSR problems in both theoretical and real scenarios with real backbone network datasets. The experiment's results can be reproduced at [13].

A. Datasets

We conduct experiments on three real datasets: Brain, Abilene, and Geant, available at [14]. Due to the variation in the number of traffic matrices in each dataset, we decide to use the first 10000 traffic matrices of each dataset in our experiments. Each dataset is divided into three sets: 70% for training, 10% for validating, and 20% for testing. Table I shows the details about each dataset. Note that, although the Brain network has 161 nodes in total, most of them are regional nodes. Therefore, we only consider the aggregated traffic of the backbone nodes in the Brain network.

B. Performance metrics

We evaluate the proposed approach's performance using the maximum link utilization (MLU) and the number of routing changes (RC). After obtaining the routing policy, the MLU is calculated using the actual traffic matrix from the test set. The routing changes is the total number of flows that are rerouted after each routing cycle.

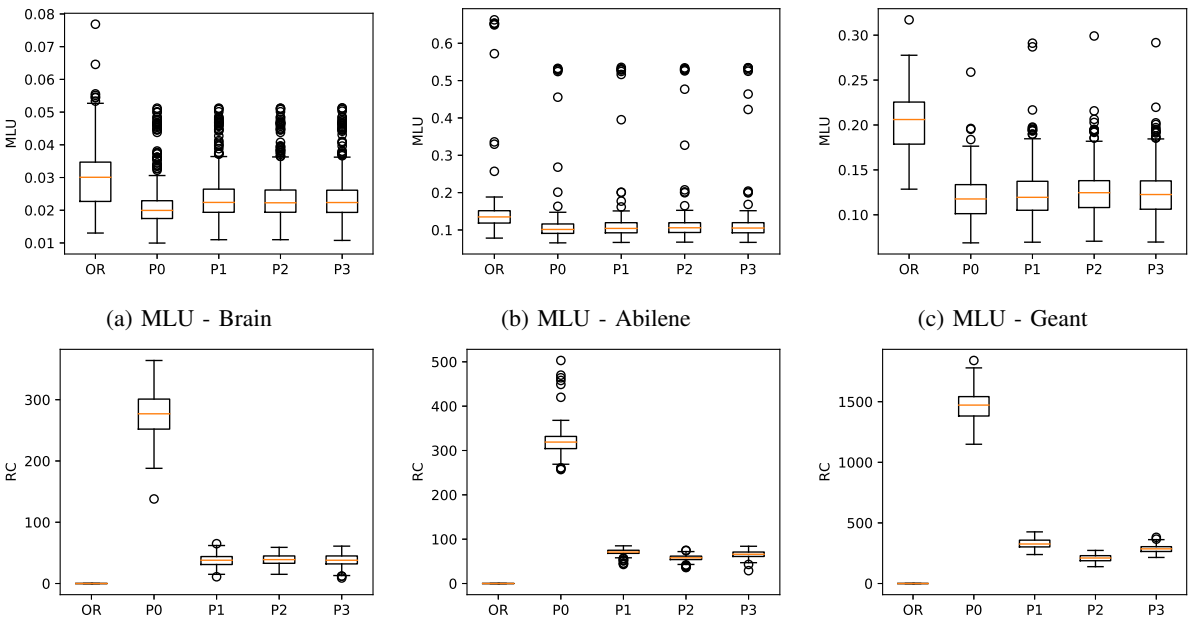
C. Results

1) Experiment 1: In the first experiment, we evaluate the performance of the routing policy acquired from the five problem formulations P_0, P_1, P_2, P_3, and Traffic Matrix Oblivious Segment Routing (OR) [2]. Note that we solve the problem P_0 for every time-steps and only solve the OR problem once. In this experiment, we assume that the future traffic matrices can be accurately predicted, then the inputs for all the problems are the real traffic matrices from the test set. We use the MILP solver from the PuLP library [15] to solve all the problems and obtain the optimal solutions. In all experiments, routing cycle length is fixed as $T = 6$.

Figure 3 shows the results of all approaches on Brain, Abilene, and Geant datasets. Obviously, by solving the TE problem at every time-step, P_0 achieves the best results in terms of MLU on all datasets. However, although P_0 has a slightly better MLU compared to P_1, P_2, and P_3 (e.g., 3.77%, 5.59%, and 4.73% on Geant dataset, respectively), it suffers from significantly high number of routing changes. The Oblivious Routing (OR) shows the lowest performance in terms of MLU on all the datasets. Especially on the Geant network, the average MLU obtained by OR is nearly twice that of other approaches.

As mentioned in the theoretical analysis (Section III-C), among the three formulations of the MTSR problem, P_1 has come first in comparing the MLU, P_3 and P_2 have come second and third. However, the gaps between them are relatively small. Although all of the approaches have almost the same values in terms of routing changes, P_2 shows a slightly better performance in the Geant dataset. According to the results, there are three merits of solving MTSR with P_2 formulation: reducing the problem complexity, alleviating the difficulty in the traffic prediction task, and achieving comparable performance with other approaches. Therefore, in the rest of the paper, the presented results of the MTSR are obtained by solving the problem P_2.

2) Experiment 2: In this experiment, we evaluate the performance of the proposed approach as an online segment routing algorithm. We train a state-of-the-art deep learning model called Graph WaveNet (GWN) [12] and use the trained model for predicting future traffic matrices. The training details are omitted in this paper but can be found at [13]. The prediction model uses historical data of H time-steps to



(a) MLU - Brain (b) MLU - Abilene (c) MLU - Geant

(d) RC - Brain (e) RC - Abilene (f) RC - Geant

Fig. 3: The maximum link utilization and number of routing changes on three datasets

predict future traffic matrices. At the beginning of each routing cycle, after obtaining the predicted traffic matrices, we solve the problems and acquire the routing policy. Then, the actual traffic matrix is used to calculate the maximum link utilization for every time-step. The MTSR problem is solved by both the MILP solver and our heuristic algorithm to evaluate the heuristic algorithm (LS2SR). We compare the performance of our proposed method with three other baselines:

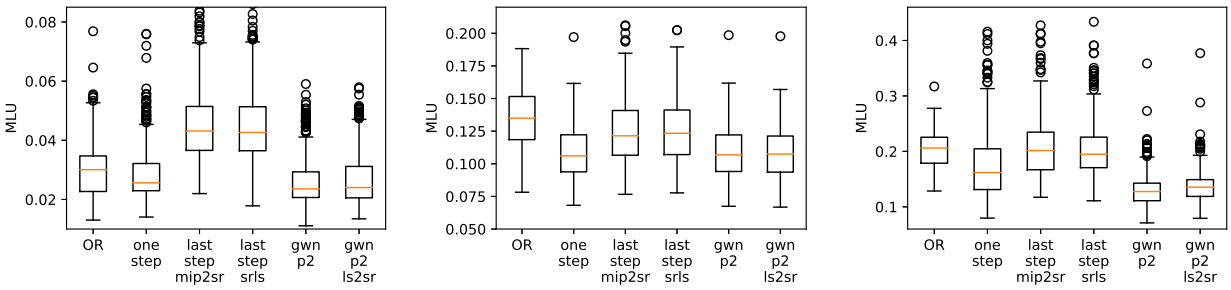
- **Last-step**: the traffic matrix of the last time-step in the routing cycle is used as input for the segment routing problem and then the obtained routing policy is applied for the next cycle. We use two proposed approaches (**mip2sr** [2] and **srls** [3]) to solve the problem. Note that **srls** is a local search-based algorithm that exploits n-segment routing.
- **One-step** prediction: the traffic matrix of the first time-step in the next cycle is predicted by GWN. Then, it is used as input for solving the P_0 and then the obtained routing policy is applied for the next cycle.
- **OR**: the Traffic Matrix Oblivious Segment Routing.

In all experiments, we set the length of the routing cycle as $T = 6$ and the number of historical steps used for the prediction model as $H = 2T$. The time constraint of the MILP solver is set equal to $60s$ while that of the heuristic algorithms is $1s$. In LS2SR algorithm, we set $\beta = 16$ and $\gamma = 1$ which are adopted from [3]. Note that in the **last-step** experiment, we do not use **cg4sr** algorithm [4], which fully exploits segment routing, since **srls** produced better results than **cg4sr** in the experiments with ISPs traffic (showed in [4]).

From Figure 4, we can see that the our approaches (gwn-p2 and gwn-p2-ls2sr) achieve the best results in terms of MLU concerning all datasets. In overall, with traffic prediction (both one-step and multi-step prediction) we can achieve better MLU than the other approaches. For example, in terms of average MLU, our approach improves by 35.47%, 33.19% and 32.46% compared to the last-step mip2sr, srls and OR on the Geant dataset. Similar to Experiment 1, due to the fixed routing policy, OR cannot adapt to the network traffic dynamic and thus suffering from a high MLU. In terms of routing changes, our heuristic algorithm significantly reduces the number of flows that need to be rerouted compared to other heuristic algorithms. The average number of routing changes is close to that of OR (zero routing change), even in the most extensive network topology (i.e., the Geant network). In comparing our heuristic algorithm and the MILP solver (gwn-p2-ls2sr and gwn-p2) on solving the MTSR problem, our heuristic achieves the same performance in terms of MLU while significantly reduces the number of flows needs to be rerouted in a routing cycle. Note that the LS2SR is set to find the solution within $1s$, which shows the algorithm's scalability.

3) Experiment 3: In this part, we investigate the impact of the routing cycle's length (T) and different prediction models on the performance of traffic engineering (MLU). We first use GWN as a traffic matrices predictor and conduct experiments with different routing cycle lengths ($T = \{3, 6, 12\}$), then we fix $T = 6$ and use different prediction models such as LSTM, ARIMA for the traffic matrices prediction task. Note that in all experiments, we obtain the routing policy by solving problem P_2 with the LS2SR algorithm.

The results in Table II show the prediction error in terms of Mean Absolute Error (MAE) and the MLU. It is clear that MAE increases when the prediction model needs to predict traffic matrices in the far future. However, the increasing of the prediction errors do not have a strong relationship



(a) MLU - Brain (b) MLU - Abilene (c) MLU - Geant

(d) RC - Brain (e) RC - Abilene (f) RC - Geant

Fig. 4: The maximum link utilization and number of routing changes with different routing approaches

	Brain		Abilene		Geant	
T	MLU	MAE	MLU	MAE	MLU	MAE
3	0.0272 ± 0.0004	350993.2	0.1115 ± 0.0021	3.1	0.1351 ± 0.0014	16.7
6	0.0271 ± 0.0005	433130.0	0.1122 ± 0.0021	3.7	0.1369 ± 0.0015	19.2
12	0.0270 ± 0.0005	560248.1	0.1145 ± 0.0025	4.7	0.1385 ± 0.0016	25.6

TABLE II: Evaluate the impact of routing cycle's length

	Brain		Abilene		Geant	
Model	MLU	MAE	MLU	MAE	MLU	MAE
ARIMA	0.0331 ± 0.0007	2147543.5	0.1317 ± 0.0026	39.2	0.1930 ± 0.0022	152.9
LSTM	0.0280 ± 0.0005	459269.0	0.1122 ± 0.0021	3.8	0.2045 ± 0.0023	91.6
GWN	0.0271 ± 0.0005	433130.0	0.1122 ± 0.0021	3.7	0.1369 ± 0.0015	19.2

TABLE III: Evaluate the performance of different prediction models

with the MLU. For instance, concerning the Geant dataset, in experiments with $T = 3$ and $T = 12$, while the MAE increases by 53%, the MLU only increases by 2.5%. This phenomenon can be explained as follows. As in P_2, we formulate the MTSR problem by considering the worst case of traffic in the next T time-steps, the obtained routing policy can work well within the routing cycle. Moreover, as mentioned in Section III-A, in P_2, we only need to predict one traffic matrix, which makes the prediction tasks much easier. Therefore, the results support the conclusion that P_2 is suitable for solving the MTSR problem.

Table III shows the performance comparison of three different prediction models in terms of MLU and MAE. The results indicate that deep learning models outperform the traditional statistic model (i.e., ARIMA) in the traffic prediction task. Comparing the two deep learning models, GWN achieves the best results in all experiments and metrics. Although having poor results in terms of MAE, the results of ARIMA on the traffic engineering metric are acceptable on all three datasets.

VI. Conclusion

In this paper, we studied the multi-time-step segment routing problem with traffic prediction. Our objective is to leverage the traffic prediction for doing traffic engineering while minimizing the number of flows that need to be rerouted. We considered the problem complexity and the difficulty of multi-step traffic prediction to formulate three versions of the segment routing problem. We also provided theoretical analysis on the solution of the three formulations. To address the scalability issue, we proposed a heuristic algorithm for quickly solving the multi-step segment routing and further reduce the number of routing changes. Our evaluation on real network datasets showed that the proposed approach can significantly reduce the number of routing changes while still meet the requirements on the maximum link utilization.

Acknowledgment

This research is supported in part by JSPS KAKENHI Grant Numbers JP20H00592, JP18KK0279 and JP19H04093.

REFERENCES

[1] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment routing: A comprehensive survey of research activities, standardization efforts and implementation results," *arXiv preprint arXiv:1904.03471*, 2019.

[2] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2015, pp. 657–665.

[3] S. Gay, R. Hartert, and S. Vissicchio, "Expect the unexpected: Sub-second optimization for segment routing," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, IEEE, 2017, pp. 1–9.

[4] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "Cg4sr: Near optimal traffic engineering for segment routing with column generation," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1333–1341.

[5] M.-C. Lee and J.-P. Sheu, "An efficient routing algorithm based on segment routing in software-defined networking," *Computer Networks*, vol. 103, pp. 44–55, 2016.

[6] V. Pereira, M. Rocha, and P. Sousa, "Traffic engineering with three-segments routing," *IEEE Transactions on Network and Service Management*, 2020.

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

[8] V. A. Le, P. Le Nguyen, and Y. Ji, "Deep convolutional lstm network-based traffic matrix prediction with partial information," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 261–269.

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

[10] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, no. 2-6, p. 11, 2012.

[11] S. Sivabalan, J. Medved, C. Filsfils, V. Lopez, J. Tantsura, W. Henderickx, E. Crabbe, and J. Hardwick, "Pcep extensions for segment routing," *IETF draft-sivabalan-pce-segment-routing-03. txt*, 2014.

[12] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *International Joint Conference on Artificial Intelligence 2019*, International Joint Conferences on Artificial Intelligence, 2019, pp. 1907–1913.

[13] https://github.com/vananle/MS2SR.

[14] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," English, in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, http://sndlib.zib.de, extended version accepted in Networks, 2009., Apr. 2007. [Online]. Available: http : / / www . zib . de / orlowski / Paper / OrlowskiPioroTomaszewskiWessaely2007 - SNDlib - INOC.pdf.gz.

[15] Stuart Mitchell, *Pulp*. [Online]. Available: https://coin-or.github.io/pulp/.