IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 30, 2019, accepted October 27, 2019, date of publication November 4, 2019,
date of current version November 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2951163

A Computation-Efficient Approach for Segment Routing Traffic Engineering

TOSSAPHOL SETTAWATCHARAWANIT[1,2], YI-HAN CHIANG[2], VORAPONG SUPPAKITPAISARN[3], AND YUSHENG JI[1,2]

[1]Department of Informatics, The Graduate University for Advanced Studies, Sokendai, Tokyo 101-8430, Japan
[2]Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-8430, Japan
[3]Department of Computer Science, The University of Tokyo, Tokyo 113-8654, Japan

Corresponding author: Tossaphol Settawatcharawanit (tossaphol@nii.ac.jp)

ABSTRACT The unprecedented growth of network traffic has brought excessive challenges to network operators. To prevent network congestion, network operators conduct traffic engineering (TE) for their routing optimization. In recent years, segment routing traffic engineering (SRTE) has emerged as one of the promising approaches for its high scalability and low control overheads. However, conventional SRTE approaches in large-scale networks are computationally prohibitive, which may lead to delayed system operations and unsatisfactory service qualities. In this paper, we formulate a bi-objective mixed-integer nonlinear program (BOMINLP) to investigate the trade-off between link utilization and computation time in SRTE. Due to the difficulty in solving the original problem directly, we decompose it into two sequential sub-problems. The first sub-problem is to minimize computation time through node selection, and the second one is to minimize maximum link utilization via flow assignment. To this end, we first employ randomized sampling based on stretch bounding to obtain a reduced solution space and then solve a linear program (LP) using existing software tools for the sub-problems. To evaluate our proposed solution, we employ network topologies and traffic matrices from publicly available datasets. Our simulation results show that our proposed solution can effectively reduce computation time while retaining comparable maximum link utilization as compared with several comparison approaches.

INDEX TERMS Segment routing, traffic engineering, bi-objective mixed-integer nonlinear program.

I. INTRODUCTION

The diversification of traffic types and the explosive growth of traffic demands have prompted great research attention to network congestion. By means of traffic engineering (TE) [1]–[5], network flows can be dynamically embedded into physical substrate networks, thereby avoiding network congestion and optimizing routing performance for network operators. In practice, a commonly adopted mechanism for TE is multi-protocol label switching traffic engineering (MPLS-TE) [6], [7], in which network resources along each routing path can be reserved. However, MPLS-TE requires to maintain and distribute network states (such as network topology and bandwidth availability) across the whole network [8], which may result in poor network scalability [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Tang.

Segment routing traffic engineering (SRTE) that leverages centralized controllers to maintain network states has been regarded as a promising solution to cope with the network scalability issues of MPLS-TE. In SRTE, a *segment* indicates the shortest path between any two nodes and a *segment routing path* is an end-to-end path composed of multiple connected segments. Therefore, each segment routing path can be viewed as a logical tunnel from the ingress to the egress. By using intermediate nodes as segment labels in SRTE, the excessive number of concatenated labels in MPLS-TE can be alleviated.

Various works on SRTE have been devoted to network congestion in recent years. In [10], Bhatia *et al.* formulated a generic SRTE problem to minimize maximum link utilization, where all intermediate nodes are used to construct optimal segment routing paths. In [11], Cianfrani *et al.* formulated a mixed-integer linear program for the SRTE problem to minimize the maximum link utilization among

segment routing nodes. In Li and Yeung [13] proposed tunnel training architecture with tunnel limit extension for 2-segment routing that utilizes shortest path routing. In [13], Li *et al.* proposed a mixed-integer linear program to optimize link utilization while limiting the number of segment labels. In [14], we proposed a stretch bounding approach that achieves near-optimal maximum link utilization for SRTE. Although the above works have addressed the network congestion issues, they do not pay much attention to the reduction of computation time.

In literature, some other exiting works on SRTE put their focuses on improving computation time, system throughput, or the use of segment labels. In [15] and [16], Trimponias *et al.* proposed to reduce the number of candidate intermediate nodes based on graph centrality, thereby minimizing computation time at the price of worse link utilization. In [17], Zhong *et al.* proposed an online maximum profit algorithm to solve a segment routing problem in integrated terrestrial-satellite networks. In [18], Gang *et al.* formulated a mixed-integer linear program to maximize throughput in hybrid segment routing networks. In [19], Huang *et al.* proposed an integer linear program that considers the maximum segment label depth and flow entry overhead. In [20], Zhang *et al.* proposed a bandwidth allocation algorithm to maximize user satisfaction as a function of resource allocation in hybrid segment routing networks. In [21], Hartert *et al.* formulated a constraint programming problem for SRTE, and designed a local search algorithm that can be manually terminated to meet a predefined time limit. In [22], Gay *et al.* proposed a local search approach to iteratively improve current TE solutions rather than finding a complete solution based on the assumption that traffic changes are limited. However, none of the above works can guide us on how to strike a balance between link utilization and computation time in SRTE.

In this paper, we aim to investigate the trade-off relationship between link utilization and computation time in SRTE. To this end, we formulate a bi-objective mixed-integer nonlinear program (BOMINLP) to minimize link utilization and computation time. In the light of the two conflicting objective functions and the non-linearity of constraints, we decompose the original problem into two sequential sub-problems (i.e. node selection and flow assignment). Then, we propose a randomized sampling approach for the first sub-problem and then leverage an LP solver for the second one. We employ two publicly available datasets for performance evaluation, and our simulation results demonstrate that the proposed solution can effectively reduce the computation time but also retain comparable maximum link utilization. The contributions of this paper are as follows.

- We investigate the trade-off of link utilization and computation time in SRTE and formulate it as a BOMINLP.
- We decompose the original problem into the sequential sub-problems of node selection and flow assignment.
- We propose a randomized sampling approach and leverage an LP solver for the sub-problems, respectively.
- We show that our proposed solution can reduce computation time enormously while achieving comparable maximum link utilization on publicly available datasets.

The rest of this paper is organized as follows. In Sec. II, we present the network environment, segment routing, flow splitting, and node selection. Sec. III describes the problem formulation and decomposition. Sec. IV presents our proposed solutions for the decomposed sub-problems. Sec. V demonstrates our simulation results. Finally, this paper concludes in Sec. VI.

II. SYSTEM MODEL

A. NETWORK ENVIRONMENT

Consider a general network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, c)$, where $\mathcal{V}$ and $\mathcal{E}$ refer to the sets of vertices (i.e. routers) and undirected edges (i.e. links), respectively, and each edge $e \in \mathcal{E}$ is associated with the weight $w(e)$ and the capacity $c(e)$. In addition, each source-destination pair (i, j) can have a traffic demand t_{ij}, for all $i, j \in \mathcal{V}$. For brevity, we denote by $\mathcal{P}_{uv}$ the set of all shortest paths directed from node u to node v, and by $\bar{\mathcal{E}}_s$ the set of edges along each shortest path $s \in \mathcal{P}_{uv}$, where $u, v \in \mathcal{V}$. For the ease of reading, we list primary symbols that are used throughout the paper in TABLE 1.

B. SEGMENT ROUTING

A *segment* is either a single shortest path or a set of equal-cost shortest paths between any two nodes in the network. Whenever an incoming flow passes through a node, it will be divided into multiple outgoing sub-flows, which can be implemented by the equal-cost multi-path routing (ECMP) [24]. A *segment routing path* is an established end-to-end path that is constituted by a sequence of segments. FIGURE 1 illustrates how segment routing works in practice. In FIGURE 1a, the source node i sends a traffic flow to an intermediate node l with the segment labels l and j. When the intermediate node l receives the first packet of the flow, the segment label of the intermediate node l is popped out. Then, the intermediate node l can reroute the traffic flow to the destination node j with the segment label j. The segment labels that need to be specified in the packet header are the remaining segment labels of rerouting nodes and the destination. In FIGURE 1b, we see that each segment routing path is composed of two segments. The traffic from the source node i to the destination node j passes through the intermediate node l. The first segment is routed on a single shortest path, but the second one is routed on two equal-cost shortest paths.

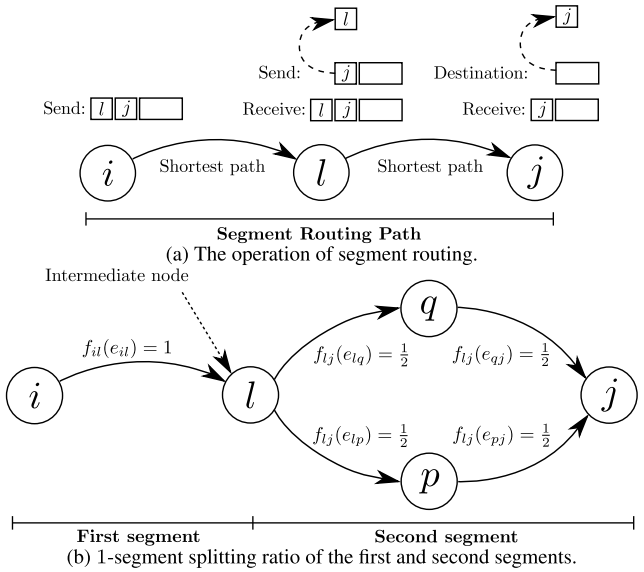
Despite the generality of multi-segment settings, we will focus on the use of two segments. The reasons for choosing the 2-segment setting are two-fold:

- lower elapsed time for processing packet headers, and
- near-optimal maximum link utilization.

The shortest paths can be constructed by the IGP extension protocol for segment routing which has been under standardization [23].

TABLE 1. List of notations.

Notation	Definition
$\mathcal{G}$	Network graph
$\mathcal{G}_{uv}$	Edge-induced directed subgraph of $\mathcal{G}$, formed from $\bar{\mathcal{E}}_s$ directed from u to v for all $u, v \in \mathcal{V}$
$\mathcal{V}$	Set of vertices (nodes)
$\mathcal{V}_{uv}$	Set of vertices (nodes) of $\mathcal{G}_{uv}$ for all $u, v \in \mathcal{V}$
$\mathcal{E}$	Set of undirected edges (links)
$\bar{\mathcal{E}}_s$	Set of undirected edges (links) along each shortest path $s \in \mathcal{P}_{uv}$ for all $u, v \in \mathcal{V}$
$\mathcal{E}_{uv}$	Set of directed edges (links) of $\mathcal{G}_{uv}$, where each edge have a direction according to $\mathcal{P}_{uv}$
$w(e)$	Link weight $e \in \mathcal{E}$
$c(e)$	Link capacity $e \in \mathcal{E}$
t_{ij}	Traffic demand from i to j for all $i, j \in \mathcal{V}$
$f_{uv}(e)$	1-segment splitting ratio from u to v for all $u, v \in \mathcal{V}$ on $e \in \mathcal{E}$, where $f_{uv}(e) \in [0, 1]$
$g_{ilj}(e)$	2-segment splitting ratio from i to j that passes through l for all $i, l, j \in \mathcal{V}$ on link e
$\mathcal{P}_{uv}$	Set of all shortest paths from u to v for all $u, v \in \mathcal{V}$
$\text{in}_{uv}(z)$	Set of incoming edges incident to node $z \in \mathcal{V}_{uv}$ for all $u, v \in \mathcal{V}$
$\text{out}_{uv}(z)$	Set of outgoing edges from node $z \in \mathcal{V}_{uv}$ for all $u, v \in \mathcal{V}$
$\text{str}(i, l, j)$	Stretch of the segment routing path of i, l and j for all $i, l, j \in \mathcal{V}_{ij}$
$v_{ilj}(\alpha)$	Binary indicator of candidate intermediate node l between i and j for all $i, l, j \in \mathcal{V}$, $\alpha \geq 1$
x_{ilj}	Decision variable of the amount of traffic flow from i to j that passes through l for all $i, l, j \in \mathcal{V}$
y_{ilj}	Decision variable of candidate intermediate node l between i and j for all $i, l, j \in \mathcal{V}$
θ	Maximum link utilization
$\ddot{\theta}$	Normalized maximum link utilization
ϕ	Maximum link utilization obtained by using a shortest path algorithm
α	Coefficient for stretch bounding, where $\alpha \geq 1$
β	Regulatory coefficient, where $0 < \beta \ll 1$
ζ	Computation time



(a) The operation of segment routing.

(b) 1-segment splitting ratio of the first and second segments.

FIGURE 1. Segment routing examples.

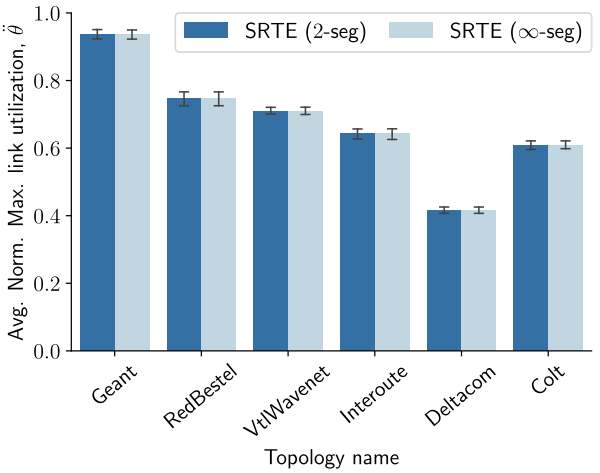


FIGURE 2. A comparison between SRTE (2-seg) and SRTE (∞-seg).

FIGURE 2 compares the 2-segment SRTE setting (2-seg) and unbounded segment SRTE setting (∞-seg). 2-seg uses ECMP, while ∞-seg uses all simple paths available to reach the destination without considering the routing cost. The maximum link utilization performance of 2-seg is identical to ∞-seg. Moreover, the effectiveness of the 2-segment setting has been asserted by [10], [25], [26], therefore we restrict our focus to the 2-segment setting in the following.

C. FLOW SPLITTING

Segment routing leverages ECMP to distribute sub-flows across multiple paths. To quantify the amount of sub-flows (of the source-destination pair) passing through each individual edge, consider a flow that passes through an intermediate node l from the source node i to the destination node j in the 2-segment setting. We denote $f_{il}(e)$ and $f_{lj}(e)$ the 1-segment splitting ratios of the first and second segment, respectively.

Let $\mathcal{G}_{uv} = (\mathcal{V}_{uv}, \mathcal{E}_{uv})$ be an edge-induced directed sub-graph when $\mathcal{E}_{uv} = \bigcup_{s \in \mathcal{P}_{uv}} \bar{\mathcal{E}}_s$, where s is a shortest path from u to v, $s \in \mathcal{P}_{uv}$, and $\mathcal{V}_{uv}$ is a set of nodes incident to an edge in $\mathcal{E}_{uv}$. Note that, while $\mathcal{G}$ is undirected, $\mathcal{G}_{uv}$ is directed as the set $\mathcal{E}_{uv}$ contains a direction from u to v in the shortest path s. To assess $f_{il}(e)$ and $f_{lj}(e)$, suppose that z is a node in $\mathcal{G}_{uv}$, we define the following information:

- $\text{in}_{uv}(z)$: the set of incoming edges incident to node $z \in \mathcal{V}_{uv}$.
- $\text{out}_{uv}(z)$: the set of outgoing edges from node $z \in \mathcal{V}_{uv}$.

Then, we define $f_{uv}(e)$ as

$$f_{uv}(e) = \begin{cases} f'_{uv}(e, z), & \text{if } e \in \mathcal{E}_{uv}, \\ 0, & \text{otherwise,} \end{cases} \quad \forall e \in \mathcal{E}, z \in \mathcal{V}_{uv}, u, v \in \mathcal{V}. \quad (1)$$

With the above information, we can obtain an assignment of $f'_{uv}(e,z)$ which satisfies the following equations:

$$\sum_{e\in \text{out}_{uv}(z)} f'_{uv}(e,z) = \sum_{e\in \text{in}_{uv}(z)} f'_{uv}(e,z), \quad \forall u,v\in\mathcal{V},\ z\in\mathcal{V}_{uv}\setminus\{u,v\}, \tag{2}$$

$$\sum_{e\in \text{out}_{uv}(u)} f'_{uv}(e,u) = 1, \quad \forall u,v\in\mathcal{V}, \tag{3}$$

$$\sum_{e\in \text{in}_{uv}(v)} f'_{uv}(e,v) = 1, \quad \forall u,v\in\mathcal{V}, \tag{4}$$

$$f'_{uv}(e_1,z) = f'_{uv}(e_2,z), \quad \forall e_1,e_2\in \text{out}(z), \quad \forall u,v\in\mathcal{V},\ z\in\mathcal{V}_{uv}, \tag{5}$$

where (2) -(4) refer to flow conservation at an intermediate node, the source node, and the destination node, respectively. (5) ensures outgoing traffic should split equally with ECMP.

By substitute (u,v) in $f_{uv}(e)$ with (i,l) and (l,j), we can obtain $f_{il}(e)$ and $f_{lj}(e)$, respectively. Subsequently, we define the 2-segment splitting ratio (aggregate the splitting ratios of the two segments) $g_{ilj}(e)$ as

$$cg_{ilj}(e) = f_{il}(e) + f_{lj}(e), \quad \forall e\in\mathcal{E}, i,l,j\in\mathcal{V}. \tag{6}$$

D. NODE SELECTION

For each source-destination pair, there is a tremendous number of candidate intermediate nodes, but some of them are too far away from either the source or the destination. Therefore, it is essential in practice to keep the number of candidate intermediate nodes at a reasonable value.

Consider the source node, an intermediate node and the destination node triple (i,l,j), and the shortest paths $s_1\in\mathcal{P}_{il}$, $s_2\in\mathcal{P}_{lj}$ and $s_3\in\mathcal{P}_{ij}$. We define the *stretch* as the ratio of the total weights of the two segments to that of an end-to-end shortest path, which can be expressed as

$$\text{str}(i,l,j) = \frac{\sum_{e\in\bar{\mathcal{E}}_{s_1}} w(e) + \sum_{e\in\bar{\mathcal{E}}_{s_2}} w(e)}{\sum_{e\in\bar{\mathcal{E}}_{s_3}} w(e)}, \quad \forall i,l,j\in\mathcal{V}. \tag{7}$$

By leveraging the stretch concept, the number of candidate intermediate nodes can be greatly reduced through stretch bounding (see FIGURE 3 for an example). To indicate whether node l serves as a candidate intermediate node after applying stretch bounding, we define the indicator as

$$cv_{ilj}(\alpha) = \begin{cases} 1, & \text{str}(i,l,j)\le\alpha, \\ 0, & \text{otherwise}, \end{cases} \quad \forall i,l,j\in\mathcal{V}, \tag{8}$$

which is used to ensure that $\text{str}(i,l,j)$ is not greater than a stretch bounding coefficient $\alpha\ge 1$. Note that $\alpha=1$ states that only the shortest paths are used, and $\alpha=\infty$ represents no stretch bounding takes effect.

The purpose of stretch bounding is to explicitly control the path length for transmission (i.e., stretch), thereby reducing transmission latency [27], [28]. However, it remains unknown from these works how to leverage stretch bounding for link utilization in SRTE.

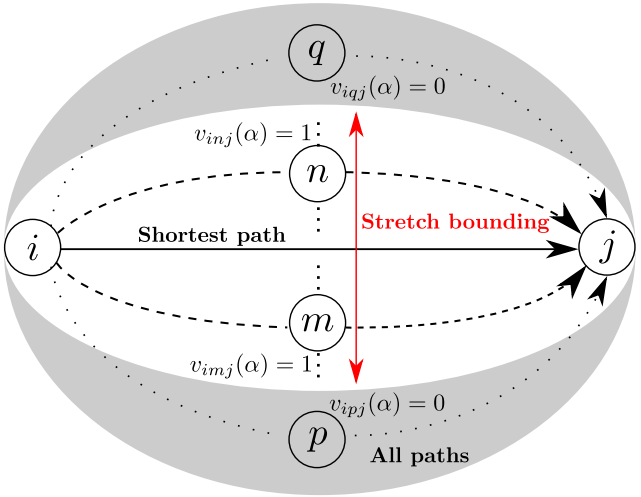


FIGURE 3. Stretch bounding approach.

III. PROBLEM FORMULATION

Generally, network operators have incentives to periodically optimize link utilization and computation time to adapt to network dynamics. For these reasons, we formulate an optimization problem as a bi-objective mixed-integer nonlinear program (BOMINLP) to jointly minimize maximum link utilization θ and computation time ζ via

$$\boldsymbol{x} = \{x_{ilj}, \forall i,l,j\in\mathcal{V}\}, \quad \text{(flow assignment)} \tag{9}$$

$$\boldsymbol{y} = \{y_{ilj}, \forall i,l,j\in\mathcal{V}\}, \quad \text{(node selection)} \tag{10}$$

where x_{ilj} denotes the amount of traffic flow from node i to node j that passes through node l and y_{ilj} determines whether l serves as a candidate intermediate node between i and j for all $i,l,j\in\mathcal{V}$.

Mathematically, we formulate the BOMINLP as

$\mathbb{P}_1$ (**Joint Node Selection and Flow Assignment**):

$$\min_{\boldsymbol{y}}\ \zeta \triangleq h(\boldsymbol{y}), \quad \longleftarrow \text{computation time}$$

$$\min_{\boldsymbol{x}}\ \theta, \quad \longleftarrow \text{maximum link utilization}$$

$$\text{s.t. } \mathbf{C1:} \sum_{l\in\mathcal{V}} x_{ilj}y_{ilj} \ge t_{ij}, \quad \forall i,l,j\in\mathcal{V},$$

$$\mathbf{C2:} \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}\sum_{l\in\mathcal{V}} g_{ilj}(e)x_{ilj}y_{ilj} \le \theta\cdot c(e), \quad \forall e\in\mathcal{E},$$

$$\mathbf{C3:} y_{ilj} = \min\Big(\sum_{l'\in\mathcal{V}} v_{il'j}(\alpha), \lceil\beta|\mathcal{V}|\rceil\Big), \quad \forall i,l,j\in\mathcal{V},$$

$$\mathbf{C4:} y_{ilj}\in\{0,1\}, \quad \forall i,l,j\in\mathcal{V},$$

$$\mathbf{C5:} x_{ilj}\ge 0, \quad \forall i,l,j\in\mathcal{V},$$

where $h:\boldsymbol{y}\to\mathbb{R}_{\ge 0}$ is a monotonically increasing function that maps from candidate intermediate nodes to the corresponding computation time . **C1** represents that the flow assignment should satisfy all traffic demands. **C2** ensures that

The monotonically increasing property can be observed in general SRTE (e.g. [14], [15]).

the flows routed through a link do not exceed the link capacity. **C3** presents the intermediate node restriction by stretch bounding and restricts the size of candidate intermediate nodes through the regulatory coefficient β to prevent loosely bounded stretches. **C4** and **C5** refer to the auxiliary constraints for node selection and flow assignment, respectively.

In essence, $\mathbb{P}_1$ is formulated to characterize the trade-off between maximum link utilization and computation time. However, there is a lack of solution approaches for tackling $\mathbb{P}_1$ directly due to the following reasons.

- *Conflicting objective functions.* To minimize ζ, it is desirable to have less selected nodes, which may give rise to a concentrated flow assignment. To minimize θ, it is intuitive to assign flow uniformly as much as possible, but at the price of more selected intermediate nodes. Evidently, it is not possible to optimize ζ and θ simultaneously.
- *Mixed decision variables.* The decision variables $\boldsymbol{x}$ are non-negative real numbers and $\boldsymbol{y}$ are binary integers. Due to the combinatorial feature of $\boldsymbol{y}$, solving $\mathbb{P}_1$ optimally is in essence NP-hard (as a general ILP [29]).
- *Nonlinear constraints.* **C1** and **C2** involve products between two decision variables. These expressions make $\boldsymbol{x}$ and $\boldsymbol{y}$ involved and difficult to decouple.

The aforementioned reasons explain the difficulties in solving $\mathbb{P}_1$ optimally. Therefore, we are motivated to consider the following two sequential sub-problems:

$\mathbb{P}_2$ (**Node Selection**):	$\mathbb{P}_3$ (**Flow Assignment**):
$\min_{\boldsymbol{y}} \ \zeta,$	$\min_{\boldsymbol{x}\|\boldsymbol{y}} \ \theta,$
s.t. **C3**,**C4**,	s.t. **C1**,**C2**,**C5**.

It is important to note that, if we fix $\boldsymbol{y}$, the BOMINLP stated above will turn to be an LP. In this way, we simply need to look for a subset of candidate intermediate nodes in $\mathbb{P}_2$ through $\boldsymbol{y}$, based on which we find out a flow assignment in $\mathbb{P}_3$ through $\boldsymbol{x}$. Since $\mathbb{P}_2$ and $\mathbb{P}_3$ are an ILP and an LP, respectively, we can then design a computation-efficient algorithm for solving them in practice.

Remark 1: In fact, $\mathbb{P}_2$ and $\mathbb{P}_3$ retain all of the constraints in $\mathbb{P}_1$, therefore the feasible solution space of $\mathbb{P}_1$ can be kept intact after the problem decomposition.

IV. ALGORITHM DESIGN

In this section, we propose a two-phase algorithm (denoted by SRTE$^+$) to address the decomposed sub-problems $\mathbb{P}_2$ and $\mathbb{P}_3$. Our design principle is to select candidate intermediate nodes efficiently and effectively.

In **Phase 1**, we leverage a randomized sampling approach for the node selection. The purpose of the randomized sampling is to distribute the traffic load among a limited number of candidate intermediate nodes so that the congestion at a specific link can be avoided. The LP problem size becomes larger if there are more elements of $\boldsymbol{y}$ being one. Therefore, the number of elements of $\boldsymbol{y}$ being one will determine the size of the LP and the computation time to solve it. In **Phase 2**, we reduce the number of decision variables according to $\boldsymbol{y}$, and then solve $\mathbb{P}_3$. Note that Algorithm 1 leverages α and β as partial information to reduce the computation time, since the exact form of h may not be available.

Algorithm 1 A Computation-Efficient 2-Phase SRTE$^+$ Alg

Input: α, β, $\mathcal{V}$, $\mathcal{E}$, $\{c(e), \forall e \in \mathcal{E}\}$, $\{t_{ij}, \forall i,j \in \mathcal{V}\}$, $\{g_{ilj}(e), \forall i,l,j \in \mathcal{V}, e \in \mathcal{E}\}$.

Output: $\boldsymbol{x}$.

(Phase 1) Randomized Sampling based Node Selection

1: Initialize $\bar{\mathcal{V}}_{ij} \leftarrow \{l | v_{ilj}(\alpha) = 1 \text{ in (8)}, \forall i,l,j \in \mathcal{V}\}$;
2: Initialize $\boldsymbol{y} \leftarrow \{y_{ilj} = 0, \forall i,l,j \in \mathcal{V}\}$;
3: **for** $i \in \mathcal{V}$ **do**
4: **for** $j \in \mathcal{V}$ **do**
5: **if** $|\bar{\mathcal{V}}_{ij}| \leq \lceil \beta |\mathcal{V}| \rceil$ **then**
6: **for** $l \in \bar{\mathcal{V}}_{ij}$ **do**
7: Set $y_{ilj} \leftarrow 1$;
8: **else**
9: Initialize $\mathcal{L} \leftarrow \varnothing$;
10: **while** $|\mathcal{L}| < \lceil \beta |\mathcal{V}| \rceil$ **do**
11: Choose $l \in \bar{\mathcal{V}}_{ij} \setminus \mathcal{L}$ randomly;
12: Set $y_{ilj} \leftarrow 1$ and $\mathcal{L} \leftarrow \mathcal{L} \cup \{l\}$;

(Phase 2) LP-based Flow Assignment

13: Solve $\mathbb{P}_3$ based on $\boldsymbol{y}$;
14: Output $\boldsymbol{x} \leftarrow \{x_{ilj}, \forall l \in \bar{\mathcal{V}}_{ij}, i,j \in \mathcal{V}\}$.

Algorithm 1 presents a computation-efficient 2-phase algorithm for SRTE$^+$ (see FIGURE 4 for an overview).

- Lines 1-2: The algorithm initializes the set of candidate intermediate nodes with $\bar{\mathcal{V}}_{ij}$ and sets the indicators $\boldsymbol{y}$ to zeros.
- Lines 5-7: The algorithm checks the size $|\bar{\mathcal{V}}_{ij}|$ of candidate intermediate nodes. If the size is at least $\lceil \beta |\mathcal{V}| \rceil$, all intermediate nodes in $\bar{\mathcal{V}}_{ij}$ are selected as candidate intermediate nodes by setting y_{ilj} to 1 for all $l \in \bar{\mathcal{V}}_{ij}$.
- Lines 8-12: If the size $|\bar{\mathcal{V}}_{ij}|$ is greater than $\lceil \beta |\mathcal{V}| \rceil$, the algorithm randomly selects candidate intermediate nodes from $\bar{\mathcal{V}}_{ij}$ until the number of candidates intermediate nodes reach $\lceil \beta |\mathcal{V}| \rceil$. Specifically, the algorithm initially sets $\mathcal{L}$ to be the empty set to keep track of the selected candidate intermediate nodes. The algorithm iteratively selects intermediate node l from $\bar{\mathcal{V}}_{ij} \setminus \mathcal{L}$ by setting y_{ilj} to 1, and then adding l into $\mathcal{L}$.
- Lines 13-14: Finally, the algorithm solves $\mathbb{P}_3$ with the reduced number of candidate intermediate nodes in $\boldsymbol{y}$

In practice, one may choose loosely bounded stretches (i.e., large α) to prevent network congestion, but at the price of high computation time since there are many choices of paths. As a complement, β can be used to limit the number of candidate intermediate nodes (e.g. $\beta = 2.5$-7% [15]).

TABLE 2. Datasets, network topologies and traffic matrices.

Dataset name	Topology name	Traffic matrix	# of nodes	# of edges	# of traffic matrices	# of flows
GEANT	GEANT	IntraTM-2005-04-29-09-15	27	38	96	729
REPETITA	RedBestel	RedBestel.0000	84	93	5	5000
REPETITA	VtlWavenet	VtlWavenet2011.0003	92	96	5	5000
REPETITA	Interoute	Interoute.0001	110	148	5	5000
REPETITA	Deltacom	Deltacom.0001	113	161	5	5000
REPETITA	Colt	Colt.0001	153	177	5	5000

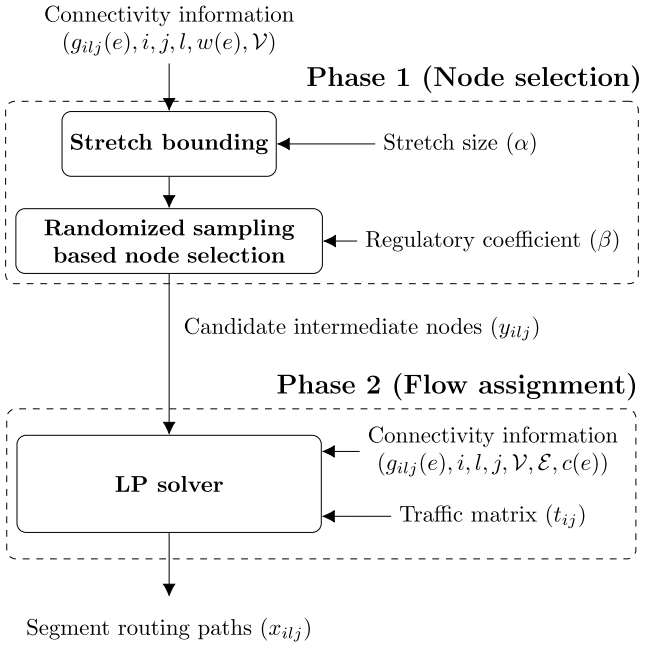


FIGURE 4. The algorithmic overview of SRTE$^+$.

(the number of $\boldsymbol{y}$ being 1 is reduced), which implies the reduced problem size of $\mathbb{P}_3$.

Since LP can be solved in polynomial time using the Karmarkar's interior point algorithm which grows in cubic order of the number of variables [30]. Consequently, the running time of SRTE grows quickly with the number of nodes $|\mathcal{V}|$ and the number of candidate intermediate nodes (the number of elements of $\boldsymbol{y}$ being 1). Such running time would be very slow even for a network of moderate size when all nodes are considered as candidate intermediate nodes. For instance, among the data set that we employ in Sec. V, the number of nodes ranges from 27 to 153, corresponding to $(27)^3$ to $(153)^3$ segment routing path variables, which could lead to the LP solving computationally prohibitive. By means of the regulatory coefficient β, solving the LP can be accelerated by $1-\beta$ (e.g. 96.6% when $\beta = 0.04$). Note that with specific candidate intermediate nodes, the problem $\mathbb{P}_3$ is in essence an LP, which can be solved by various software tools (e.g. CPLEX [31], GUROBI [32] or GLPK [33]).

Remark 2: To implement SRTE$^+$ on real networks, the most common way is to leverage a software-defined network (SDN) architecture . By using the SDN architecture, we can directly place our optimization program (i.e. Algorithm 1) on an SDN controller. After running the optimization program, the SDN controller will configure routers according to the decisions of flow assignment.

V. PERFORMANCE EVALUATION

In this section, we first choose balance stretch sizes for our proposed solution in different network topologies. Then, we compare our proposed solution with various TE approaches. Finally, we demonstrate the impact of routing metrics. Note that all of the following simulation results are averaged over all traffic matrices and yield 95% confidence intervals.

A. SIMULATION SETTINGS

1) SYSTEM SET-UP

We conduct our simulations on a Dell PowerEdge R430 server (composed of an Intel Xeon E5-2640 v3 processor and 64-GB physical memory) with Linux 4.4.0. As to the LP solver, we use IBM ILOG CPLEX version 12.8 [31].

2) DATASETS

To make our simulation results reproducible, we employ the following two public datasets that provide practical network topologies and traffic matrices.

- The GEANT dataset [34] contains one network topology and 10,772 real-world traffic matrices (96 of which will be chosen for our performance evaluation).
- The REPETITA dataset [35] contains 266 real-world network topologies (five of which will be chosen for our performance evaluation), and each network topology is associated with 5 synthetic traffic matrices.

In addition, we restrict the number of flows to 5,000 for each traffic matrix due to the shortage of system memory. More detailed information regarding the network topologies and traffic matrices can be found in TABLE 2.

3) ROUTING METRICS

We consider three different commonly used routing metrics as follows:

- unary: all links have equal weights.
- delay: each link weight is calculated based on the physical link distance.
- inverse: each link weight is set to the inverse of the link capacity.

4) PERFORMANCE METRICS

To evaluate the effectiveness of our proposed solution, we employ the computation time ζ and the normalized maximum link utilization $\ddot{\theta} = \theta/\phi$ as our primary performance metrics, where ϕ is the maximum link utilization obtained by a shortest path algorithm (e.g. Dijkstra algorithm [36] and Bellman-Ford algorithm [37]).

5) COMPARISON SCHEMES

To demonstrate the performance gain achieved by our proposed solution SRTE$^+$, we consider the following TE approaches for comparison.

- SRTE [10]: all of the nodes in $\mathcal{V}$ are chosen as candidate intermediate nodes.
- SRBS [14]: candidate intermediate nodes are chosen similarly as SRTE$^+$, except that SRBS neither regulates the size of candidate intermediate nodes nor performs randomized sampling.
- DEG [15]: all of the nodes are sorted in descending order of their degree centrality, and the first $\lceil \beta|\mathcal{V}| \rceil$ nodes will be chosen.
- BETW [15]: all of the nodes are sorted in descending order of their betweenness centrality, and the first $\lceil \beta|\mathcal{V}| \rceil$ nodes will be chosen.
- RAND [15]: $\lceil \beta|\mathcal{V}| \rceil$ of nodes are chosen uniformly as candidate intermediate nodes.

For simplicity, we focus on the unary routing metric (i.e., all links have equal weights) in Sec. V-B.1 and V-B.2. The impact of various routing metrics will be left to Sec. V-B.3.

B. SIMULATION RESULTS

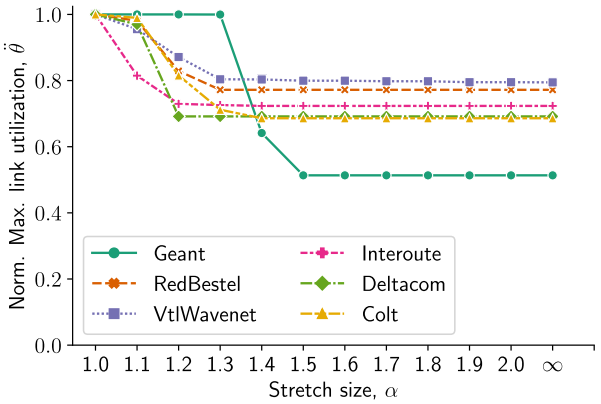
1) THE CHOICES OF STRETCH SIZES

Choosing stretch sizes differently has direct impacts on both of the performance metrics as depicted in FIGURE 5.

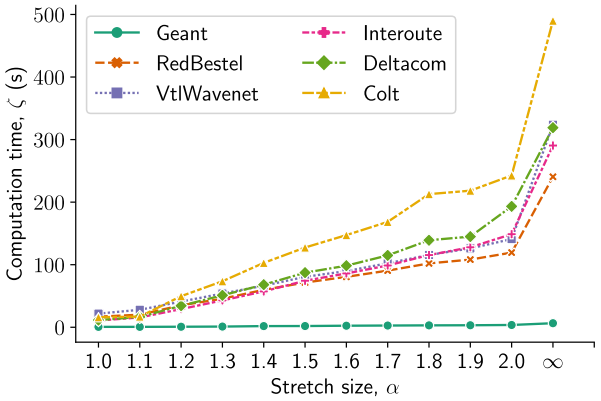
a: LARGE STRETCH SIZES IMPROVE LINK UTILIZATION

In FIGURE 5a, as the stretch size increases, the maximum link utilization tends to decrease for all network topologies. This is because larger stretch sizes can offer more candidate intermediate nodes and more path choices. In addition, we observe that the maximum link utilization remains unchanged when the stretch size becomes large. The reason is that large stretch sizes include many candidate intermediate nodes that are too far away, and therefore they cannot reduce the maximum link utilization any further. Note that how vertices are connected can affect link weights and network connectivity, so the lines in FIGURE 5a can vary greatly with network topologies.

The normalization is to scale the link utilization results such that the link utilization of the shortest path is at 100% utilization. For brevity, we will interchangeably use the terminology normalized maximum link utilization and maximum link utilization.



(a) The maximum link utilization



(b) The computation time

FIGURE 5. The impact of stretch size (unary).

b: LARGE STRETCH SIZES REQUIRE HEAVY COMPUTATION

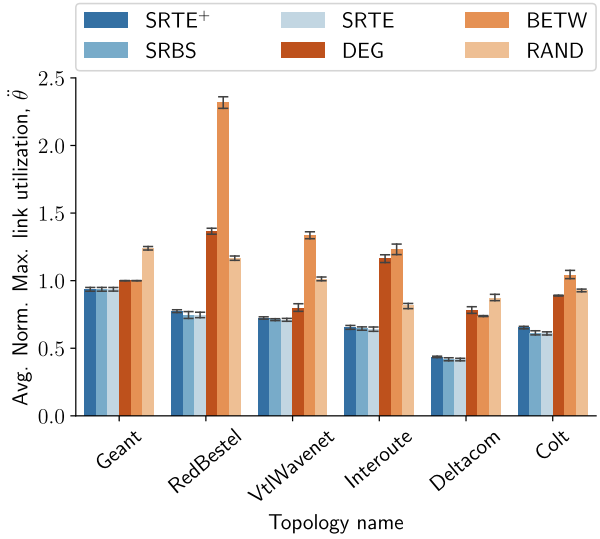
In FIGURE 5b, we see that the computation time is monotonically increasing with the stretch size. The reason is that large stretch sizes will increase the number of candidate intermediate nodes, and hence the problem size of LP and the required computation time significantly increase. In addition, the larger the network topology, the higher the computation time due to the greater problem size.

c: STRETCH SIZES BALANCE THE PERFORMANCE METRICS

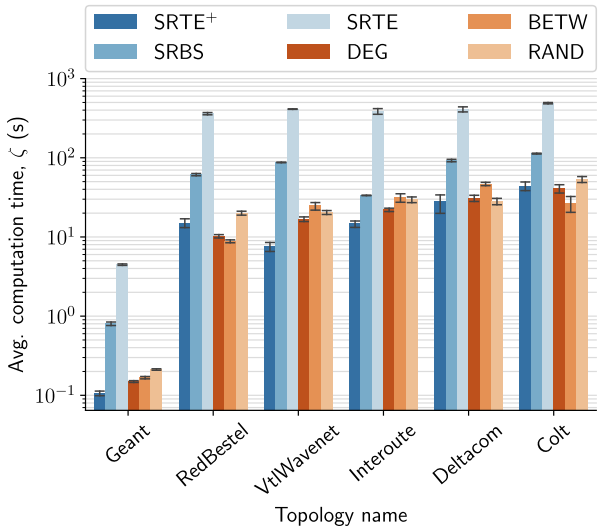
As the maximum link utilization remains unchanged and the computation time increases monotonically, we can certainly choose the smallest stretch size (the saturated point) that corresponds to the lowest maximum link utilization. In other words, increasing the stretch size beyond the saturated point does not help decrease the link utilization, but will incur higher computation time. By looking at FIGURE 5a and FIGURE 5b, we see that there exist stretch sizes that balance the maximum link utilization and computation time on each network topology. Accordingly, we have the balanced stretch sizes for each network topology as shown in TABLE 3, which will be used for the following subsections.

TABLE 3. The list of balanced stretch sizes (the grey-shaded column and row refer to the settings used in FIGURE 6 and 7, respectively).

Dataset name	Topology name	Unary	Delay	Inverse
GEANT	GEANT	1.5	1.4	1.2
REPETITA	RedBestel	1.3	1.3	1.3
REPETITA	VtlWavenet	1.3	1.3	1.3
REPETITA	Interoute	1.2	1.2	1.3
REPETITA	Deltacom	1.2	1.4	1.2
REPETITA	Colt	1.4	1.4	1.3



(a) The maximum link utilization



(b) The computation time (in log scale)

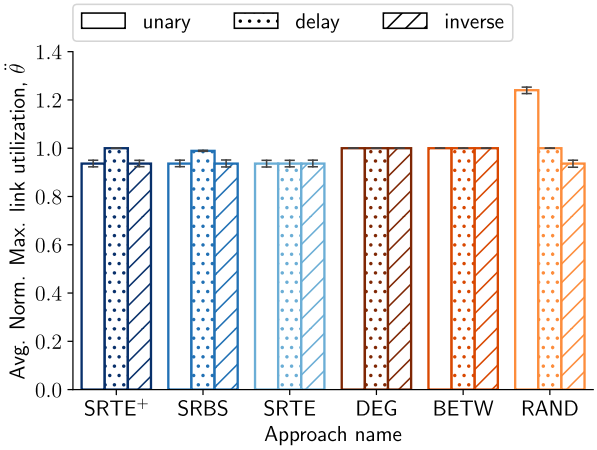
FIGURE 6. Performance comparison among various approaches ($\beta = 0.04$, unary).

2) THE COMPARISON AMONG TE APPROACHES

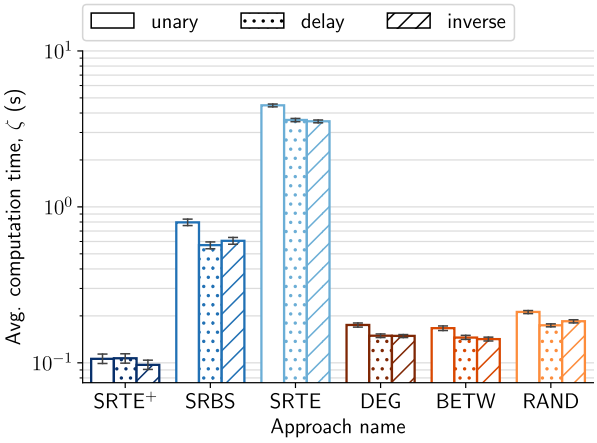
FIGURE 6 illustrates how SRTE$^+$ outperforms other TE approaches in terms of the two performance metrics.

a: SRTE$^+$ IMPROVES LINK UTILIZATION EFFECTIVELY

In FIGURE 6a, SRTE$^+$ outperforms DEG, BETW, and RAND in terms of the maximum link utilization.



(a) The maximum link utilization



(b) The computation time (in log scale)

FIGURE 7. The impact of routing metrics ($\beta = 0.04$, Geant).

The reason is that stretch bounding allows SRTE$^+$ to use diverse candidate intermediate nodes whereas the centrality approaches use static candidate intermediate nodes for all source-destination pairs. SRTE$^+$ has competitive maximum link utilization as compared to SRTE and SRBS. Even though the number of candidate intermediate nodes in SRTE$^+$ is much smaller than those in SRTE and SRBS, the maximum link utilization remains low in SRTE$^+$.

b: SRTE$^+$ REDUCES THE COMPUTATION TIME ENORMOUSLY

In FIGURE 6b (in log scale), SRTE$^+$ outperforms SRTE and SRBS in terms of computation time since the LP problem size in SRTE$^+$ is much smaller than those in SRTE and SRBS. SRTE$^+$ has comparable computation time to DEG, BETW and RAND because the regulatory coefficient ensures that SRTE$^+$ has similar LP problem sizes as the centrality approaches. Note that our simulation results demonstrate that the computation time of SRTE$^+$ (for all network topologies that we have tested) can be kept within one minute, which conforms to the requirement that TE programs are typically

invoked by network operators periodically in short intervals, e.g. 5-10 minutes [2].

c: SRTE$^+$ BALANCES THE TWO OBJECTIVE FUNCTIONS EFFECTIVELY

SRTE serves as a lower bound of SRTE$^+$ with respect to the maximum link utilization as it uses all candidate intermediate nodes to reduce network congestion. Even though SRTE$^+$ sacrifices a small amount of link utilization (θ), the computation time (ζ) can be reduced enormously.

3) THE IMPACT OF ROUTING METRICS

FIGURE 7 shows how SRTE$^+$ varies with other routing metrics and how it outperforms other TE approaches.

a: SRTE$^+$ THRIVES ON VARIOUS ROUTING METRICS

In FIGURE 7a, SRTE$^+$ has comparable maximum link utilization as compared to SRTE and SRBS even SRTE$^+$ has much less number of candidate intermediate nodes. In addition, SRTE$^+$ outperforms DEG, BETW, and RAND in terms of maximum link utilization since SRTE$^+$ can better distribute the traffic load to less congested paths as compared to the centrality approaches. In FIGURE 7b, SRTE$^+$ outperforms all comparison TE approaches across all routing metric in terms of computation time since it has small LP problem size.

VI. CONCLUSION

In this paper, we formulated a BOMINLP to characterize the trade-off between link utilization and computation time in SRTE. Due to the conflicting objective functions, mixed decision variables, and nonlinear constraints of the original problem, we proposed to decompose it into the node selection and flow assignment sub-problems. Then, we designed a computation-efficient two-phase SRTE$^+$ algorithm to solve the sub-problems sequentially: we first proposed randomized sampling to reduce the number of candidate intermediate nodes and then assigned traffic flows by solving LPs with reduced problem sizes. We conducted our simulations based on two publicly available datasets with practical network topologies and traffic matrices. Extensive simulation results show that SRTE$^+$ can reduce computation time enormously with respect to several comparison approaches, and meanwhile the achieved maximum link utilization remains comparable.

ACKNOWLEDGMENT

The authors would like to thank Kalika Suksomboon, Sirod Sirisup, Shigeki Yamada, Kensuke Fukuda, Michihiro Koibuchi, Shunji Abe, and Ying Cui for their valuable comments and suggestions.

REFERENCES

[1] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Oct. 2013.

[2] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, 2013.

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

[4] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Dynamic hybrid routing: Achieve load balancing for changing traffic demands," in *Proc. IEEE IWQoS*, May 2014, pp. 105–110.

[5] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, vol. 2, Mar. 2000, pp. 519–528.

[6] D. Awduche, J. Malcolm, J. Agogbua, and M. O'Dell, and J. McManus, *Requirements for Traffic Engineering Over MPLS*, document RFC 2702, Sep. 1999.

[7] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Commun. Mag.*, vol. 37, no. 12, pp. 42–47, Dec. 1999.

[8] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, *Overview and Principles of Internet Traffic Engineering*. document RFC 3272, May 2002.

[9] I. F. Akyildiz, T. Anjali, L. Chen, J. C. de Oliveira, C. Scoglio, A. Sciuto, J. A. Smith, and G. Uhl, "A new traffic engineering manager for DiffServ/MPLS networks: Design and implementation on an IP QoS testbed," *Comput. Commun.*, vol. 26, no. 4, pp. 388–403, Mar. 2003.

[10] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 657–665.

[11] A. Cianfrani, M. Listanti, and M. Polverini, "Incremental deployment of segment routing into an ISP network: A traffic engineering perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3146–3160, Oct. 2017.

[12] T. Schuller, 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.

[13] X. Li and K. L. Yeung, "Traffic engineering in segment routing using MILP," in *Proc. IEEE ICC*, May 2019, pp. 1–6.

[14] T. Settawatcharawanit, V. Suppakitpaisarn, S. Yamada, and Y. Ji, "Segment routed traffic engineering with bounded stretch in software-defined networks," in *Proc. IEEE LCN*, Oct. 2018, pp. 477–480.

[15] G. Trimponias, Y. Xiao, H. Xu, X. Wu, and Y. Geng, "On traffic engineering with segment routing in SDN based WANs," arXiv, Ithaca, NY, USA, Tech. Rep. arXiv:1703.05907, Mar. 2017. [Online]. Available: https://arxiv.org/abs/1703.05907

[16] G. Trimponias, Y. Xiao, X. Wu, H. Xu, and Y. Geng, "Node-constrained traffic engineering: Theory and applications," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1344–1358, Aug. 2019.

[17] G. Zhong, J. Yan, and L. Kuang, "Improving integrated terrestrial-satellite network utilization using near-optimal segment routing," in *Proc. IEEE/CIC ICCC Wkshps*, Aug. 2018, pp. 64–68.

[18] Y. Gang, P. Zhang, X. Huang, and T. Yang, "Throughput maximization routing in the hybrid segment routing network," in *Proc. Telecomun. Commun. Eng.*, Nov. 2018, pp. 262–267.

[19] L. Huang, Q. Shen, W. Shao, and C. Xiaoyu, "Optimizing segment routing with the maximum SLD constraint using openflow," *IEEE Access*, vol. 6, pp. 30874–30891, 2018.

[20] P. Zhang, Y. Gang, X. Huang, S. Zeng, and K. Xie, "Bandwidth allocation with utility maximization in the hybrid segment routing network," *IEEE Access*, vol. 7, pp. 85253–85261, 2019.

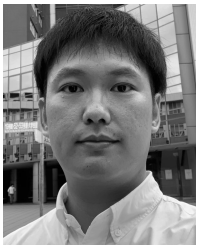
[21] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 15–28, Oct. 2015.

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

[23] P. Psenak and S. Previdi, "OSPFv3 extensions for segment routing," IETF, Fremont, CA, USA, Tech. Rep. draft-ietf-ospf-ospfv3-segment-routing-extensions-23, Jan. 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-ospf-ospfv3-segment-routing-extensions-23

[24] C. E. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, document RFC 2992, Nov. 2000.

[25] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic engineering using segment routing and considering requirements of a carrier IP network," in *Proc. IFIP Netw.*, Jun. 2017, pp. 1–9.
[26] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Predictive traffic engineering with 2-segment routing considering requirements of a carrier IP network," in *Proc. IEEE LCN*, Oct. 2017, pp. 667–675.
[27] K. Huang, C.-C. Ni, R. Sarkar, J. Gao, and J. S. B. Mitchell, "Bounded stretch geographic homotopic routing in sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 979–987.
[28] R. Flury, S. V. Pemmaraju, and R. Wattenhofer, "Greedy routing with bounded stretch," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1737–1745.
[29] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to Theory NP-Completeness*. New York, NY, USA: W. H. Freeman, 1990.
[30] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. ACM STOC*, Dec. 1984, pp. 302–311.
[31] (2014) *IBM ILOG CPLEX Optimization Studio*. [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer
[32] (2019). *Gurobi*. [Online]. Available: https://www.gurobi.com/
[33] A. Makhorin. (Oct. 2008). *GLPK (GNU Linear Programming Kit)*. [Online]. Available: http://www.gnu.org/software/glpk/glpk.html
[34] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, 2006.
[35] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms," Oct. 2017, *arXiv:1710.08665*. [Online]. Available: https://arxiv.org/abs/1710.08665
[36] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[37] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, 1958.



TOSSAPHOL SETTAWATCHARAWANIT received the M.Eng. degree from the Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand. He is currently pursuing the Ph.D. degree with the Department of Informatics, The Graduate University for Advanced Studies and the National Institute of Informatics, Tokyo, Japan. His research interests include segment routing, network optimization, algorithm design and scheduling algorithms for communication networks.

YI-HAN CHIANG received the Ph.D. degree from the Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan, in 2017. He is currently a Project Assistant Professor with the National Institute of Informatics, Tokyo, Japan. His research interests include mobile edge computing, green wireless networking, network optimization and approximation algorithms.

VORAPONG SUPPAKITPAISARN received the Ph.D. degree in information science and technology from the University of Tokyo, in 2012. He was a Project Researcher with the Global Research Center for Big Data Mathematics, National Institute of Informatics and JST ERATO Kawarabayashi Large Graph Project.

Since 2015, he has been an Assistant Professor with the Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo. His research interests include elliptic curve cryptography, computer arithmetic, graph algorithms, and large-scale data analysis.

YUSHENG JI 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, Japan, in 1990. She is currently a Professor with the National Institute of Informatics, The Graduate University for Advanced Studies. Her research interests include network architecture, resource management, and quality of service provisioning in wired and wireless communication networks. She has been an Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, a Symposium Co-Chair of the IEEE GLOBECOM 2012 and the IEEE GLOBECOM 2014, and a Track Co-Chair of the IEEE VTC 2016-Fall and IEEE VTC 2017-Fall.