A Stretch Bounding Approach for Segment Routing Traffic Engineering

by

Tossaphol Settawatcharawanit

submitted to the Department of Informatics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

SOKENDAI

The Graduate University for Advanced Studies, SOKENDAI

March 2020

A Stretch Bounding Approach for Segment Routing Traffic Engineering

A Stretch Bounding Approach for Segment Routing Traffic Engineering

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 addition, network operators are motivated to optimize their resource utilization by the highly competitive nature of communication network industries and also paramount costs of operating a sheer volume of network resources. However, it is not trivial to perform traffic engineering in dynamic systems since there are many involved processes (e.g. traffic monitoring and traffic approximation) and also requirements (e.g. computation interval for TE).

In recent years, segment routing traffic engineering (SRTE) has emerged as one of the promising approaches for traffic engineering because of its high scalability and low control overheads. By leveraging the source routing paradigm, SRTE embeds routing paths into packet headers in which simplifying the control plane of TE. However, many studies have shown that conventional SRTE approaches in large-scale networks are computationally prohibitive, which may lead to delayed system operations and unsatisfactory service qualities. This contradicts the practice that, generally, network operators require to recompute their TE program within a predefined time limit (e.g. 5-10 minutes). Many studies have tried to find solutions to meet this time requirement but at the expense of sacrificing network link utilization.

To make the SRTE problem practical, we first consider an approach to reduce the problem size of SRTE based on bounding the routing stretch. Even though the stretch bounding approaches have been studied in wireless sensor networks as an approach to restricting routing length, it remains unknown how the stretch bounding approach can be applied to SRTE. We formulate an SRTE problem based on stretch bounding with an objective to minimize network congestion (i.e. minimize the maximum link utilization). Since optimization is usually done systematically in general SRTE approaches, this motivated us to approach the problem differently. In other words, in the general SRTE, the same set of intermediate node is used for every flows in the network, which results in poor link utilization. In our scheme, we look into the details of the problem and cut the search space of the problem by means of stretch bounding. This allows us to construct different candidate intermediate node set for each source-destination pair by which help to balance the congestion efficiently.

The stretch bounding approach helps limit the number of candidate for intermediate nodes; the intermediate node that is too far away—if the intermediate node is used, it will occupy the link capacity along with every link on the path—will not be selected since they are not very useful for the solution.

We then proposed an algorithm design for the problem in which candidate intermediate nodes sets can be selected efficiently, thus reducing the problem size. As a result, the computation time is significantly reduced as compared to the State of the Art approach. Simulation results show that the computation time can be reduced significantly, while maximum link utilization can be kept near-optimal.

However, if network operators desire to control the computation time further, the problem will become more complicated. To this end, we proposed to study a trade-off relationship between computation time and link utilization in SRTE. As for SRTE problem, one can reduce the problem size by mean of reducing candidate intermediate nodes at the cost of higher link utilization. On the other hand, the link utilization can be reduced by means of having more nodes as candidate intermediate nodes sets. These two objectives can be seen as conflicting objective functions. Thus, we are motivated to formulate SRTE problem to characterize this trade-off relationship as a bi-objective mixed-integer nonlinear program (BOMINLP).

Due to the hardness of the BOMINLP (i.e. conflicting objective function, mixed decision variables, and nonlinear constraints), there is no solution approach to solve the problem directly. To this end, we are motivated to decompose the problem 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. For node selection, we leverage the monotonically increasing relationship of the number of candidate intermediate node and computation time in general SRTE to model the problem. Then, we solve the problem with randomized sampling based on the stretch bounding approach in which we leverage partial information regarding the relationship of the number of candidate intermediate node and computation time through the regulatory coefficient. Next, we eliminate candidate intermediate node based on the result of the node selection problem, and we solve a linear program (LP) using existing software tools for the sub-problems, respectively.

We then evaluate the proposed approach based on real traffic matrices and real network topologies from publicly available datasets. The simulation results show that our proposed approach help balance the two performance metrics effectively and efficiently as compared with several comparison approaches. In addition, the results show that the computation time on all tested network topologies can be kept within network operators' predefined time limit.

In brief, we formulate traffic engineering problems under two different objectives in SRTE, and we give practical and efficient solutions. From a theoretical standpoint, we leverage linear programming and problem decomposition to keep the solution space of the decomposed sub-problems intact after problem decomposition. Simulation results show that our approaches outperform other comparison approaches and have abilities to provide better performance and also shorter computation intervals for SRTE problem.

To my parents, for their unbounded love.

Contents

List of Figures

List of Tables

List of Symbols

The next list describes primary symbols that will be later used throughout the body of dissertation.

Graph-related Symbols

$\mathcal{G}$	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 $e \in \mathcal{E}$
$\mathcal{P}_{uv}$	Set of all shortest paths from u to v for all $u, v \in \mathcal{V}$
$\mathrm{in}_{uv}(z)$	Set of incoming edges incident to node $z \in \mathcal{V}_{uv}$ for all $u, v \in \mathcal{V}$
$\mathrm{out}_{uv}(z)$	Set of outgoing edges from node $z \in \mathcal{V}_{uv}$ for all $u, v \in \mathcal{V}$
$\mathrm{str}(i, l, j)$	Stretch of the segment routing path of i, l and j for all $i, l, j \in \mathcal{V}_{ij}$

Number Sets

$\mathbb{R}$	Real Numbers
$\mathbb{Z}$	Integer Numbers

Other Symbols

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 $f_{uv}(e) \in [0, 1]$
$w(e)$	2-segment splitting ratio from i to j that passes through l for all $i, l, j \in \mathcal{V}$ on link e
$v_{ilj}(\alpha)$	Binary indicator of candidate intermediate node l between i and j
$v_{ilj}(\alpha)$	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

Acknowledgements

First of all, I am grateful to my advisor, Yusheng Ji, for being patience, sharp, and reliable. She always gives excellent advice and suggestions. I will always cherish her support during the tough times of my Ph.D. studies. In addition, I would like to thank her for giving me the opportunity to do a Ph.D. I think it is a good life lesson for me.

I want to thank Shigeki Yamada for his great support and advice during my early years of studies. Also, I would like to express my sincere gratitude to Shunji Abe, Kensuke Fukuda, Michihiro Koibuchi, and Yongbing Zhang for their participation as the dissertation committee members. They gave me insightful and constructive advice that help improve this dissertation.

Yi-Han Chaing has been such a great friend and an excellent mentor. He has influenced me a lot in various aspects of my studies (e.g. his concise way of writing, and also his thought process on research). I often think about the discussions I had with him.

I am thankful to Vorapong Suppakitpaisarn. He gives me various advice throughout my studies and also his support. I am grateful for an opportunity to study in his classes (e.g. Approximation Algorithms, and Network Optimization). I am thankful to Kalika Sooksomboon for her comments and advice. She is an excellent reviewer. I also appreciate all the lab members for all their help on various occasions. Also, I would like to thank NII professors who really pays attention to students in their classes.

Last but not least, I would like to express my deepest gratitude to my parents, Termsak Setavatcharavanit and Sukanya Athabowornpisal, for giving me their endless support and understanding. My mother is a wise woman and an excellent household pundit. All my achievements have been possible with her advice in the background. My father is very supportive in my pursuing a PhD. I am deeply indebted for his countless sacrifices and patience. Indeed, he would be the proudest person of my achievements. I also would like to thank Sasikarn Jiviriyawat for her support during my difficult time.

Tossaphol Settawatcharawanit

Tokyo, Japan

March, 2020

Chapter 1

Introduction

This chapter provides a brief overview of segment routed traffic engineering (SRTE), motivation, related work, and highlights the contribution of this dissertation.

1.1 Overview and Motivation

The rapid growth of internet services (e.g. video streaming, social networking, and online games) has brought a great challenge to network operators, as these services operate on operators' infrastructure, which leads to the notorious congestion issue. In addition, the markets of communication networks are highly competitive; as a result, network operators have to adapt their network management tool to utilize network resources effectively.

Unfortunately, traditional internet protocol (IP) based networks generally use shortest path routing, where shortest paths are calculated with shortest path algorithms (e.g. Dijkstra algorithm [24] and Bellman-Ford algorithm [16]). Obviously, there are two main issues of shortest path routing that can be problematic. First, only one shortest path is used for each source-destination pair and thereby limiting network throughput. Second, shortest path routing has limited capability to adapt to dynamic traffic patterns as adjusting link weight based on the level of congestion may create oscillations [17].

The congestion problem is increasingly important as the amount of traffic on operators' networks grow. Consequently, network operators leverage traffic engineering (TE) to manage network resources, by means of mapping network traffic to links, reconfiguring the mapping according to traffic changes, such that the performance and reliability of the network can be ensured.

Traditionally, network operators employ the multi-protocol label switching traffic engineering (MPLS-TE) or IP based TE (adjusting link weight) for their routing optimization. However, these TE approaches have their own shortcomings.

- IP based TE: Since the invent of the internet, network operators have relied on traditional IP routing that leverage shortest path algorithms (e.g. open shortest path first (OSPF) and intermediate system to intermediate system (IS-IS)) to map traffic flows on physical communication links. However, TE concept was hard to realized for IP networks because the limited function of traditional IP mainly in three ways [10]. First, traffic monitoring was inaccurate and inadequate in IP networks. Second, IP networks use additive routing metrics for route selection). More importantly, IP based TE does not have the flexibility to adapt to traffic changes, as modifying link weights may affect overall traffic flows and also has limited path selection diversity. Third, it is NP-hard to set-up optimal link weights [22].

- MPLS-TE was introduce in [11, 12] to address TE problem, as traffic steering can be performed with multi-protocol label switching (MPLS) tunnelling mechanism. MPLS-TE with LDP and RSVP-TE was considered to be the state of the art approach for traffic engineering for many years. Though, they are known to incur high overhead because resource states have to be maintained along routing paths. Fundamental challenges of MPLS-TE are reported in [29]. One is the notorious scalability issues: MPLS-TE leverages resource reservation protocol traffic engineering (RSVP-TE) to maintain network states at each network node, thus incurs high control overhead. Second challenge is that to leverage path diversity of the network, MPLS-TE must maintain a sheer number of RSVP-TE tunnels; this places a considerable amount of works to network operators. Third challenge for MPLS-TE is that RSVP-TE architecture is tied to distributed architecture, which is non-trivial to achieve optimal resource utilization and hard to re-optimize.

Recently, SRTE has attracted a sheer amount of attention from both academia and industry due to its simplicity and efficiency that help improve TE in various ways. First, SRTE leverages source routing paradigm, simplifying network state distribution, as routing information is embedded into packet headers. Second, SRTE uses a PCE or SDN architecture [26, 29], which has been proven benefits network management in recent years [21, 55]; network operators can simply place theirs optimization programs onto a logically centralized controller [2]. The role of software define network (SDN) controller is to monitor network states, perform optimization, and then execute path selection based on the optimization programs.

It is worth to note that, segment routing has wide range of applications (e.g. traffic engineering, network monitoring [9, 51], and service function chaining (SFC) [54, 1, 57]). Nowadays, communication networks are increasing in size and also complexity. Network operators are motivated to come up with simple and efficient approaches to manage network resources. Many network operators consider SRTE as a critical part of TE because of its simplicity and functionality. However, solving SRTE problem is a non-trivial task due to the complexity of SRTE problem. This dissertation aims to investigate the usefulness of SRTE in practice.

1.2 Related Work

There are various works on SRTE that have been devoted to network congestion in recent years.

In [18], to the best of our knowledge, Bhatia *et al.* is the first to formulate a generic SRTE problem to minimize maximum link utilization, where all intermediate nodes are used to construct optimal segment routing paths. Thus, their optimization program required tremendous computation time to solve, especially for realistic network topologies. In [23], Cianfrani *et al.* formulated a mixed-integer linear program for the SRTE problem to minimize the maximum link utilization among segment routing nodes in hybrid segment routed networks, where some nodes are segment routing-capable and the rests are legacy nodes. In [60], Schüller *et al.* proposed tunnel training architecture with tunnel limit extension for 2-segment routing that utilizes shortest path routing; the results were evaluated with proprietary European backbone network datasets. In [52], Li *et al.* proposed a mixed-integer linear program to optimize link utilization while limiting the number of segment labels to a constant number. 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 [69] and [68], Trimponias *et al.* proposed to reduce the number of candidate intermediate nodes based on graph centrality (e.g. degree centrality, betweenness centrality), thereby minimizing computation time at the price of worse link utilization. To the best of our knowledge, these works are the first to reveal the computation complexity of SRTE and propose to reduce the problem size. Unfortunately, their approaches leverage static graph centrality, which leads to static intermediate nodes selection. Hence, this increases the maximum link utilization when a small amount of candidate intermediate nodes are selected. In [77], Zhong *et al.* proposed an online maximum profit algorithm to solve a segment routing problem in integrated terrestrial-satellite networks. In [35], Gang *et al.* formulated a mixed integer linear program (MILP) to maximize throughput in hybrid segment routing networks. In [44], Huang *et al.* proposed an integer linear program (ILP) that considers the maximum segment label depth and flow entry overhead. In [76], Zhang *et al.* proposed a bandwidth allocation algorithm to maximize user satisfaction as a function of resource allocation in hybrid segment routing networks. In [40], 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 [37], Gay *et al.* proposed a local search approach to improve current TE solutions iteratively 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.

1.3 Contributions

In this dissertation, our goal is to study the practicality of SRTE in real networks. Our contribution can be seen as two scopes of SRTE with different objectives as follows. First, we performed a preliminary study of SRTE with standard operators' objective function (i.e. minimize the maximum link utilization). Due to the large problem size, we then proposed the stretch bounding approach to limit the number of candidate intermediate nodes, leading to a smaller problem size, compared with generic SRTE. The intuition behind our stretch bounding approach is that we should not include candidate intermediate nodes that are too far away from source-destination pairs, as they are rarely helpful to minimize the maximum link utilization. Our approach allows us to construct different sets of candidate intermediate nodes for each source-destination pairs, which is the main contributor to our performance gains. The idea of stretch bounding has been studied before in the literature in wireless sensor networks (WSNs). The purpose of stretch bounding is to explicitly control the path length for transmission (i.e., stretch), thereby reducing transmission latency [43, 30]. However, it remains unknown from these works how to leverage stretch bounding for link utilization in SRTE. We then compared our approach against the optimal routing of SRTE.

Second, in order to make our approach robust against traffic changes, we must recompute our optimization program in a tiny interval. We are then motivated to investigate the trade-off relationship between the link utilization and computation time. When network operators want to reduce the maximum link utilization, the network operator needs to sacrifice the computation time, as all candidate intermediate nodes must be taken into account for routing. Conversely, when the network operators want to reduce the computation time, the network operator can select a small number of candidate intermediate nodes, then the computation time can be reduced at the price of increasing link utilization. Indeed, one can see that this problem is non-trivial due to the conflicting objective functions. Furthermore, when we need to select the candidate intermediate nodes to reduce the problem size as the selection is combinatorial in nature. This also makes the problem non-trivial, as integer variables are introduced to the problem. Another challenge is that the problem becomes nonlinear after the problem formulation because of involved decision variables. In brief, we formulated bi-objective mixed integer nonlinear programming (BOMINLP) to characterize the trade-off characteristics of the link utilization and computation time. Due to the hardness in solving the problem directly, we then motivated to resort to a problem decomposition approach. Then, we proposed a randomized sampling approach based on stretch bounding. We evaluated our results with both real and synthesis datasets against various approaches.

1.4 Dissertation Organization

The rest of this dissertation is structured as follows. We discuss the background and related work in Chapter 2. In Chapter 3, we discuss the preliminary study of the stretch bounding approach for SRTE. Then, we present system models, problem formulation, and algorithm design for the SRTE problem. We evaluated the proposed approach with both real and synthetic network topologies. In Chapter 4, we study the trade-off relationship between computation time and link utilization in SRTE. We formulated the problem as a bi-objective mixed-integer nonlinear program, which is NP-hard in general. To this end, we proposed a problem decomposition approach that we decompose the main problem into two sequential sub-problems of node selection and flow assignment. Then, we proposed to solve the node selection problem with a randomized sampling based stretch bounding approach, and we solve the flow assignment sub-problem with linear programming (LP) solver. Finally, We conclude this dissertation, discuss limitations, and provide directions for future works in Chapter 5.

Chapter 2

Background

Communication networks have been evolved rapidly from distributed network architecture to logically centralized network architecture in recent years. The key reason behind this massive change is that managing existing communication networks with ossified distributed protocols are sub-optimal. Similarly, traffic engineering (TE) makes substantial improvement from the tangled link weight adjustment approach to the multi-protocol label switching traffic engineering (MPLS-TE) approach. However, the traditional MPLS-TE incurs high overhead due to its resource reservation mechanism. In the light of simplicity and efficiency, segment routed traffic engineering (SRTE) has become a promising approach to replace complex conventional TE approaches. Moreover, network operators prefer simplicity over complexity, and also openness over proprietary. For these reasons, SRTE has become an indispensable part of many autonomous systems. This chapter provides background and relevant knowledge of the underlining architecture of SRTE [29].

2.1 Software-defined network architecture

Traditional distributed network architecture focused on providing robustness in case of failures (e.g. link failures and switch failures). At the time that the internet was invented, a centralized network architecture was thought to be too vulnerable to failures (e.g. link failures, device failures, and etc.). However, today, communication networks are too complex to manage with distributed network architecture, and also distributed network management may result in sub-optimal resource utilization.

Moreover, due to the accomplishment of internet services, communication networks are rapidly growing in size, which results in large scale complex systems that are hard to manage. The concept of a software define network (SDN) was introduced to increase simplicity and make the system easier to evolve with its openness in the control plane. As a result, SDN has transformed many aspects of computer networking. The concept of SDN is to separate the control from the data plane. In other words, the control plane of network switches can be moved into a logically centralized controller, or simply called a controller. And the role of a controller is to manage switches according to network events. As the network has the centralized controller to make decisions in responding to events, the role of switches is simply packet forwarding according to the commands of the controller

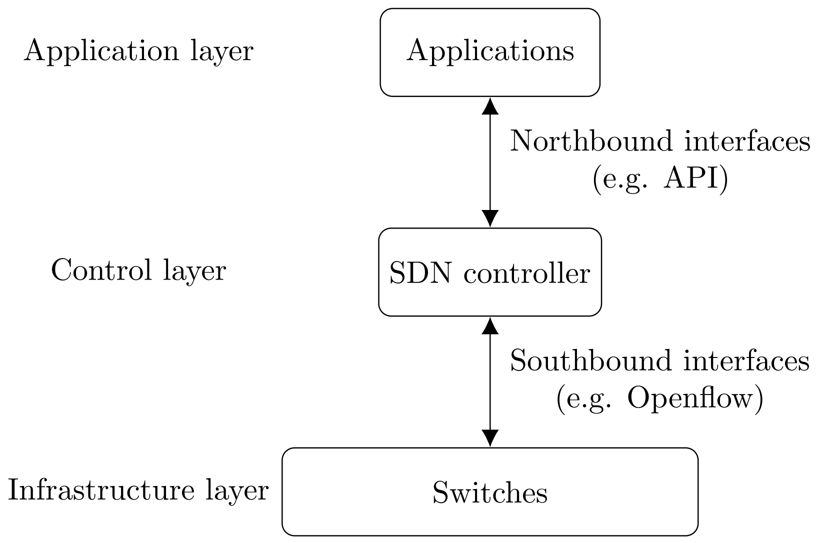


Figure 2.1: SDN architecture

[20, 21]. The SDN architecture is illustrated in Figure 2.1. Network operators can put their business logic or application program (e.g. TE, network monitoring) onto applications layer, which can be deployed on the SDN controller. The application communicates with SDN controller through the northbound interfaces (i.e. Application programming interface (API)). After receiving application inputs, the controller can communicate through the southbound interface (e.g. Openflow [55]) to regulate switches' behavior for flow assignment, network monitoring, and so on. Recently, SRTE has been proposed to utilize the SDN controller for TE purpose [26, 29].

2.2 Traffic Engineering

The unprecedented growth of network traffic has brought great challenges to network operators as traditional internet protocol (IP) based routing cannot provide the quality of services when traffic is substantially growing in size [72]. Therefore, network operators need a method to steer traffic away from congested paths to less congested paths in order to keep service quality unchanged or even provide better service quality.

The diversification of traffic types and the explosive growth of traffic demands have prompted great research attention to traffic engineering. By means of TE [13, 75, 33, 63], network flows can be dynamically embedded into physical substrate networks corresponding to the information about traffic from source to destination, thereby avoiding network congestion and optimizing routing performance for network operators. Recently, the benefit of the centralized controller for TE has been advocated in [41, 3, 46]. Network operators typically aim to minimize the utilization of the most utilized link (bottleneck link), as the higher the link utilization implies, the higher the congestion.

In order to guarantee a quality of service, a traffic engineering program must be fast to execute while efficient in terms of performance. Network operators typically recompute their TE program in short interval (e.g. 5-10 minutes) [41] in order to adapt to traffic changes. Then, newly obtained information can be used for a new traffic matrix forecast [7, 58, 56, 27].

2.2.1 Multi-protocol label switching traffic engineering

In practice, a commonly adopted mechanism for TE is MPLS-TE [11, 12, 67, 49, 74, 25], 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 [14], which may result in poor network scalability [5].

2.2.2 IP based traffic engineering

As alternative to MPLS-TE, IP based TE approaches [34, 33, 32, 8, 31, 31] require hurdle operations from network operators to carefully set link weights according to traffic changes. Furthermore, by tweaking link weights, all traffic flows may be affected, which is not the case for MPLS-TE since a tunnel can be created per flow.

2.2.3 Segment Routed Traffic Engineering

Recently, SRTE has been proposed to help network operators to manage their resources efficiently. Indeed, various pitfalls of the previous approaches have been addressed by SRTE. First, SRTE has overcome the scalability issues of MPLS-TE since SRTE is not relied on a resource reservation protocol traffic engineering (RSVP-TE) and label distribution protocol (LDP) [6] to request, maintain and distribute network resource states along a routing path as in MPLS-TE [11]. Thanks to the advancement of SDN, SDN architecture [2] helps SRTE to simplify TE mechanism by leveraging a centralized controller to maintain network states [29, 4].

Second, similar to label Switching Path (LSP) of multi-protocol label switching (MPLS), Segment Routing utilizes source routing paradigm to distribute routing information. Segment routing paths can be constructed by combining two or more segments into an end-to-end path, and each segment is the shortest path between two nodes. The end-to-end path can be seen as a logical tunnel from ingress to egress with the segment label specified in the packet header. However, not every segment has to be specified in the packet header with a segment label since we can leverage the shortest path to specify each part of the end-to-end path that traverses to some intermediate nodes between source and destination. Thus, we can use these intermediate nodes or a destination node as segment labels, as shown in Figure 2.2.

This mechanism relaxes the restrict condition of the shortest path routing because a considerable amount of paths can be constructed with the shortest path information given by interior Gateway Protocol (IGP) [29]. Moreover, this mechanism helps limit the overhead processing of packet header greatly since only a few segment labels are still needed for constructing an end-to-end tunnel. Therefore, 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.

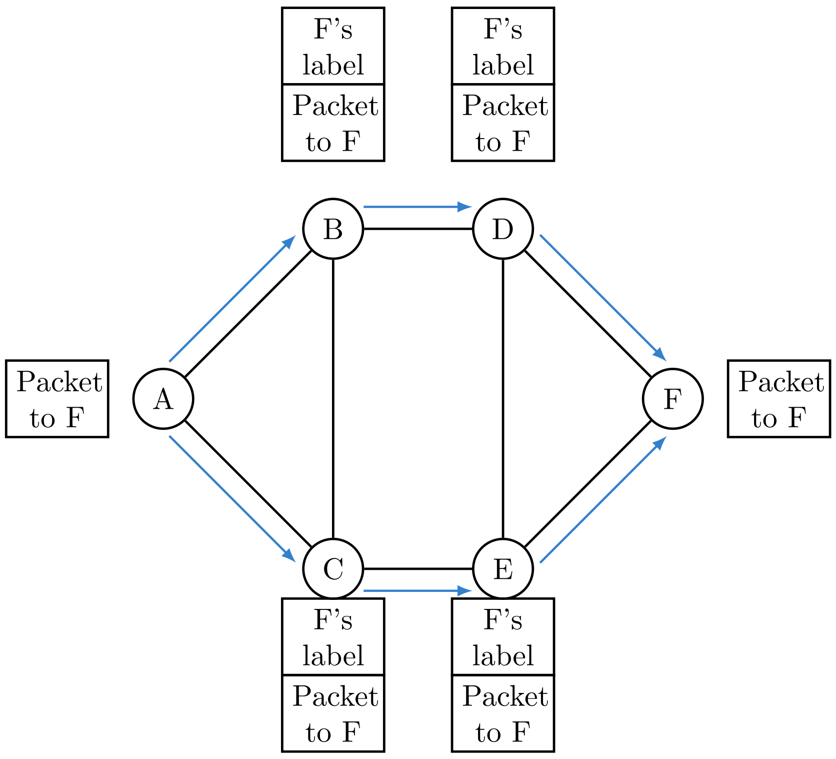


Figure 2.2: Segment routing with equal cost shortest paths: the segment label of node is denoted as "F's label".

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.

There are two types of routing data plane that can be used for segment routing:

1. MPLS [65],

2. Segment Routing over IPv6 dataplane (SRv6) [28].

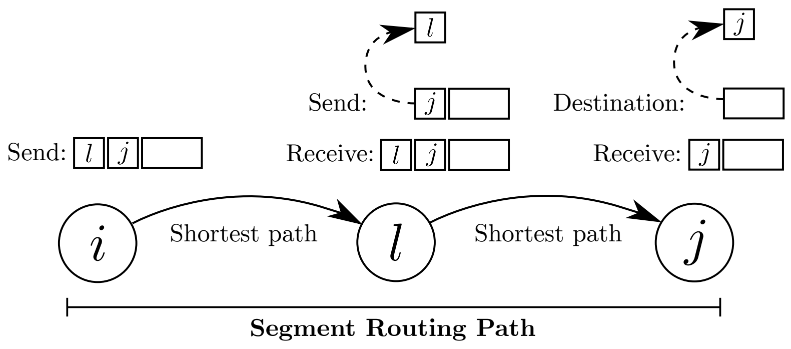


Figure 2.3: Segment routing examples: the operation of segment.

We illustrate how segment routing works in practice in Figure 2.3 and Figure 2.4. 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) [42]. A *segment routing path* is an established end-to-end path that is constituted by a sequence of segments.

In Figure 2.3, 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.

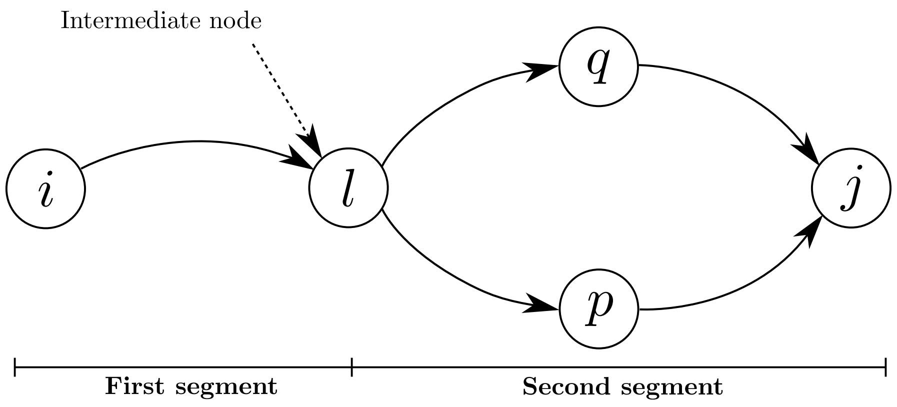


Figure 2.4: Segment routing examples: first and second segments illustration.

In Figure 2.4, 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 maximum link utilization performance of 2-segment is identical to ∞-segment. Moreover, the effectiveness of the 2-segment setting has been asserted by [61, 18, 62]; therefore, we restrict our focus to the 2-segment setting in the following chapters. However, solving SRTE problem is non-trivial for real-world networks in general due to the large problem size in SRTE [64, 69]. This motivates us to develop efficient approaches for SRTE.

Chapter 3

Segment Routing Traffic Engineering with Stretch Bounding

The main idea behind this dissertation is, one of the essential issues in segment routed traffic engineering (SRTE), to reduce the computation time for traffic engineering (TE) program execution. As the network is growing in size, the computation time requires to compute the TE program is becoming more and more demanding. It is highly desirable for network operators to have an efficient approach for SRTE.

3.1 Preliminary

TE is an essential application for network operators to manage network resources. Recently, SRTE has been adopted widely among network operators as an essential tool for TE. By using SRTE, the control plane of TE application can be simplified without the need to use resource reservation protocol traffic engineering (RSVP-TE) or manually configure link weights. Generally, a TE program must be fast and efficient in order to facilitate the network operator's demands. These two characteristics are vital since network operators usually deploy a TE program to optimize network resources, and repeatedly execute the program in short intervals (e.g., 5 minutes).

SRTE is emerging as an important application for network operators to manage resource utilization by using segment routing paths as candidates for route selection. In order to facilitate the network operator demands, a TE program should be fast and efficient. These two characteristics are essential since the TE program must be invoked periodically in short intervals. The segment routing paths can be constructed by concatenating the shortest paths between two nodes such that there is a path from source to destination. We are interested in the problem of finding intermediate nodes to construct segment routing paths minimizing the maximum link utilization. However, the existing approaches have the shortcomings that either they require a substantial amount of time to find a solution, or they must sacrifice a considerable amount of link utilization.

In this chapter, we propose the segment routing traffic engineering with Bounded Stretch (SRBS) to cope with the computation complexity of the SRTE problem. In contrast to the works in [18] and [69], our approach aims to reduce the computation time of the problem by limiting the number of candidates for intermediate nodes with a stretch bounding constraint relative to the shortest path of each source-destination pair. Notice that these intermediate node candidates can be obtained in advance with only the shortest paths information.

3.2 System Model

In this section, we introduce the theoretical model of SRTE. We first describe the model assumptions, and we then describe the SRTE problem in details.

3.2.1 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 directed edges (i.e. links), respectively, and each edge $e \in \mathcal{E}$ is associated with the weight $w(e) \in \mathbb{R} > 0$ and the capacity $c(e) \in \mathbb{R} > 0$. We assume that each router is controlled by a logically centralized controller, which is responsible for traffic flow management and monitoring.

In addition, each source-destination pair (i, j) can have a traffic demand t_{ij} for all $i, j \in \mathcal{V}$. This traffic demand has to be accommodated in the network.

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 $u, v \in \mathcal{V}$. Note that the shortest paths can be constructed by the interior Gateway Protocol (IGP) extension protocol for segment routing, which has been under standardization[59].

3.2.2 Flow Splitting

Let $\mathcal{L}$ be the set of all intermediate nodes and $\mathcal{L} \subseteq \mathcal{V}$. For each traffic demand, the traffic from a source node i to a destination node j is routed through an intermediate node or multiple intermediate nodes $l \in \mathcal{L}$. Segment Routing path from i to j can be constructed by concatenating two shortest paths: the shortest path from the source node i to an intermediate node l and the shortest path from an intermediate node l to the destination node j. Each shortest path on the segment routing path is called a segment of the routing path. For each segment of a segment routing path, the amount of traffic demand is divided into equal proportions if multiple equal-cost shortest paths are used. Let $|\mathcal{L}|$ be the number of intermediate nodes used in total. Each possible path from i to j can traverse up to M intermediate nodes or $M + 1 < |\mathcal{L}|$ segments.

Consider a traffic demand from a source node i to a destination node j, let l be an intermediate node selected by the centralized controller for this traffic flow. Then, a source node i can send a traffic flow to an intermediate node l with segment labels of l and j. When the intermediate node l receives the first packet

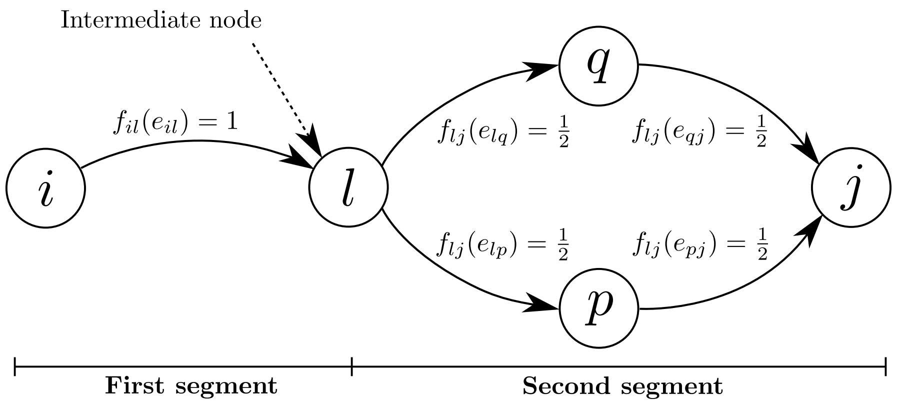

Intermediate node
$f_{il}(e_{il}) = 1$
$f_{lj}(e_{lq}) = \frac{1}{2}$
$f_{lj}(e_{qj}) = \frac{1}{2}$
$f_{lj}(e_{lp}) = \frac{1}{2}$
$f_{lj}(e_{pj}) = \frac{1}{2}$
i
l
q
p
j
First segment
Second segment

Figure 3.1: Segment routing examples: 1-segment splitting ratio of the first and second segments.

of the flow, the intermediate node l can be popped out its segment label. Then, the intermediate node l can reroute the traffic flow to the destination node j with the segment label of the destination node j. The segment labels that need to be specified in the packet header are the only segment labels of rerouting nodes and a destination.

Segment routing leverages equal-cost multi-path routing (ECMP) to distribute sub-flows across multiple paths. Note that we assume traffic can be split equally with ECMP. In order to prevent packet reordering effect in TCP, we note that traffic splitting is performed at IP flow level. 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. We give an example in Figure 3.1.

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

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

$$f_{uv}(e) = \begin{cases} f'_{uv}(e), & \text{if } e \in \mathcal{E}_{uv}, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall e \in \mathcal{E}, u, v \in \mathcal{V}, \tag{3.1}$$

where $f'_{uv}(e)$ is a ratio of sub-flow of 1-segment splitting ratios from u to v on link e.

Without loss of generality, each edge e can be rewritten as a tuple of vertices (q, r) that associated with edge e such that we can rewrite $f'_{uv}(e)$ as $f'_{uv}(q, r)$. Note that we will use the two terms interchangeably throughout this dissertation. With the above information, we can obtain an assignment of $f'_{uv}(q, r)$ which satisfies the following equations:

$$\sum_{q\in \mathrm{in}_{uv}(r)} f'_{uv}(r,q) = \sum_{q\in \mathrm{out}_{uv}(r)} f'_{uv}(q,r),$$
$$\forall u,v \in \mathcal{V}, q \in \mathcal{V}_{uv} \setminus \{u,v\}, \forall (r,q),(q,r) \in \mathcal{E} \tag{3.2}$$

$$\sum_{q\in \mathrm{out}_{uv}(u)} f'_{uv}(u,q) = 1, \quad \forall u,v \in \mathcal{V}, \forall (u,q) \in \mathcal{E}, \tag{3.3}$$

$$\sum_{q\in \mathrm{in}_{uv}(v)} f'_{uv}(q,v) = 1, \quad \forall u,v \in \mathcal{V}, \forall (q,v) \in \mathcal{E}, \tag{3.4}$$

$$f'_{uv}(q_1,r) = f'_{uv}(q_2,r),$$
$$\forall q_1,q_2 \in \mathrm{out}(r), \forall u,v \in \mathcal{V}, \forall (q_1,r),(q_2,r) \in \mathcal{E}, \tag{3.5}$$

where (3.2)-(3.4) refer to flow conservation at an intermediate node, the source node, and the destination node, respectively. (3.5) 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. For example, if there is only one shortest path from i to l, then $f_{il}(e) = 1$ for all links e on the path from i to l. However, If a link e is not on the shortest path from i to l, then $f_{il}(e) = 0$. If there are multiple equal cost shortest paths from i to l, the amount of flow is divided equally. Thus, $f_{il}(e)$ can be fractional for each link e on the shortest path from i to l. We omit an example for $f_{lj}(e)$ as it is similar to $f_{il}(e)$.

Subsequently, we define the 2-segment splitting ratio (aggregate the splitting ratios of the two segments) $g_{ilj}(e)$ as

$$g_{ilj}(e) = f_{il}(e) + f_{lj}(e), \quad \forall e \in \mathcal{E}, i,l,j \in \mathcal{V}. \tag{3.6}$$

3.2.3 Stretch Bounding

Two of the most important characteristics for TE program are the efficiency of a solution and the computation time to obtain a solution. The reason is that an optimal solution with the best quality may not be a good solution if the computation time to obtain the solution is too long. We argue that a good solution should be a solution which can be obtained in a short time while retaining a near-optimal performance.

In this section, we propose a fast and efficient approach called SRBS. The SRBS introduce a stretch bounding constraint to the linear programming (LP) formulation. The role of the stretch bounding constraint is to limit the number of candidates for intermediate nodes.

In wireless sensor network (WSN), the purpose of stretch bounding is to explicitly control the path length for transmission (i.e., stretch), thereby reducing transmission latency [43, 30]. However, it remains unknown from these works how to leverage stretch bounding for link utilization in SRTE.

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

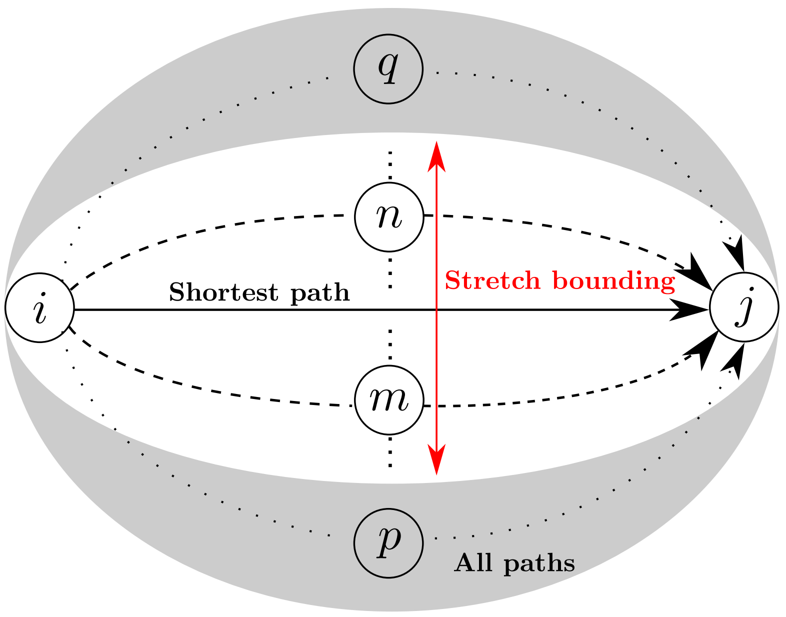


Figure 3.2: The stretch bounding concept.

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

$$\mathrm{str}(i, l, j) = \frac{\sum_{e \in \bar{\mathcal{E}}_{s_1}} w(e) + \sum_{e \in \bar{\mathcal{E}}_{s_2}} w(e)}{\sum_{e \in \bar{\mathcal{E}}_{s_3}} w(e)}, \quad \forall i, l, j \in \mathcal{V}. \tag{3.7}$$

By leveraging the stretch concept, the number of candidate intermediate nodes can be greatly reduced through stretch bounding. In Figure 3.2, node n and node m are in a stretch bounding; as a result, these nodes should be included as candidate intermediate nodes for routing. While node q and node p are too far away for the source-destination pair, hence these two nodes should not be included for routing.

To indicate restrict whether node l serves as a candidate intermediate node after applying stretch bounding, we define leverage a stretch bounding coefficient as

$$\text{str}(i, l, j) \leq \alpha, \quad \forall i, l, j \in \mathcal{V}, \tag{3.8}$$

which is used to ensure that $\text{str}(i, l, j)$ is not greater than a stretch bounding coefficient $\alpha \geq 1$. In other words, the weight of a segment routing path from i to j through l must be less than or equal to the weight of the stretched path that is scaled by α times the length of the shortest path from i to j where $\alpha >= 1$. Note that $\alpha = 1$ states that only the shortest paths are used, and $\alpha = \infty$ represents no stretch bounding takes effect. We denote S_{ij} as a set of candidates intermediate nodes that satisfy a stretch bounding constraint relative to the shortest path from i to j, and $S_{ij} \subset \mathcal{V}$. The size of S_{ij} is $|\mathcal{V}|^2$. Then, we leverage stretch bounding to select candidate intermediate nodes into S_{ij}.

3.3 Problem Formulation

We formulate SRTE as an optimization problem with an objective to minimize the maximum utilization. This variant of TE is a standard objective function for TE in carrier networks where traffic matrix can be approximated in advanced, and the network operator aims to minimize the traffic congestion in the network.

We can now introduce the SRTE problem. We denote a decision variable x_{ilj} as the amount of flow, which is routed from i to j through an intermediate node l. Given traffic flow demand t_{ij}, the goal is to find x_{ilj} the amount of traffic flow from i to j passing through an intermediate node l with the objective to minimize the maximum link utilization θ. The linear programming is as follow:

$$
\begin{array}{lll}
\mathbb{P}_1 & \textbf{(Minimize maximum link utilization)}: & \\
\min\limits_{x} & \theta, & \longleftarrow \text{maximum link utilization} \\
\text{s.t.} & \textbf{C1.1: } \sum\limits_{l \in S_{ij}} xilj = t_{ij}, & \forall i, j \in \mathcal{V}, \\
 & \textbf{C1.2: } \sum\limits_{i \in \mathcal{V}} \sum\limits_{j \in \mathcal{V}} \sum\limits_{l \in S_{ij}} g_{ilj}(e) xilj \leq \theta \cdot c(e), & \forall e \in \mathcal{E}, \\
 & \textbf{C1.3: } x_{ilj} \geq 0, & \forall i, j \in \mathcal{V}, \forall l \in S_{ij},
\end{array}
$$

The objective is to minimize the maximum utilization θ of all links. Constraint **C1.1** ensures that each traffic is routed through some intermediate node. Constraint **C1.2** is a capacity constraint which ensures that sum of all the traffic routed through a link does not exceed the link capacity. Constraint **C1.3** assures that the amount of traffic is non-negative.

3.4 Algorithm Design

3.4.1 Offline Computation

We observed that the stretch bounding (3.8) could be computed offline since the stretch bounding requires only the shortest paths information from the IGP. Note that we can compute the S_{ij} for all source-destination pairs in advanced.

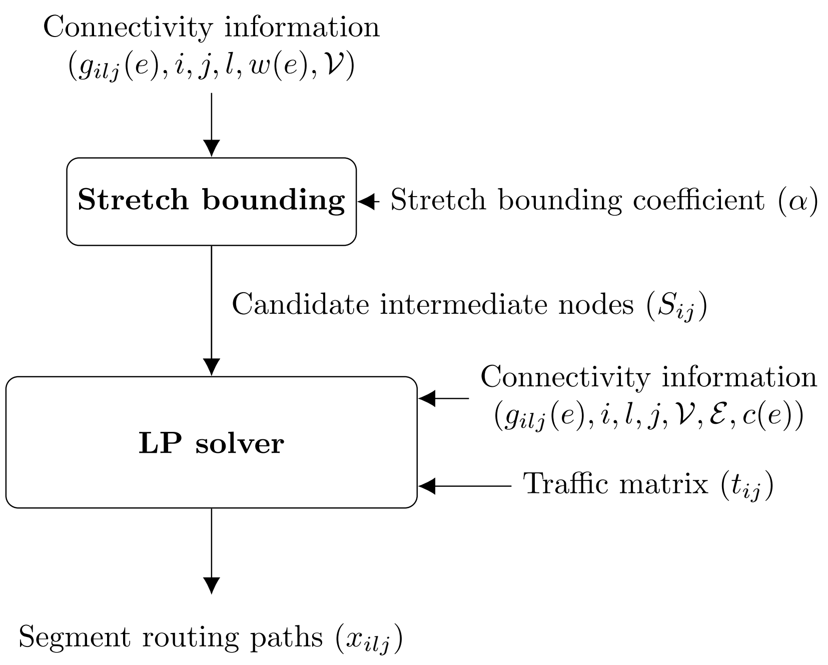

Connectivity information
$(g_{ilj}(e), i, j, l, w(e), \mathcal{V})$
Stretch bounding
Stretch bounding coefficient (α)
Candidate intermediate nodes (S_{ij})
Connectivity information
$(g_{ilj}(e), i, l, j, \mathcal{V}, \mathcal{E}, c(e))$
LP solver
Traffic matrix (t_{ij})
Segment routing paths (x_{ilj})

Figure 3.3: The algorithmic overview of SRBS.

As a consequence, we can replace an intermediate node l with a candidate for intermediate node $s \in S_{ij}$ which satisfied the stretch bounding (3.8) in the constraints (**C1.1**)-(**C1.3**).

Algorithm 1 presents the procedures to obtain S_{ij} and solve the LP program sequentially. For each source-destination pair (i, j), we select $l \in \mathcal{L}$ into S_{ij} satisfying the stretch bounding (3.8). After obtaining S_{ij}, we solve $\mathbb{P}_1$ based on obtaining S_{ij} as intermediate nodes set for each source-destination pair.

Algorithm 1 A SRBS algorithm

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

Output: $\boldsymbol{x}$.

```
1: $S_{ij} \leftarrow \emptyset$
2: for $(i, j) \in \mathcal{V}$ do
3:     for $l \in \mathcal{V}$ do
4:         if $str(i, l, j) \leq \alpha$ then
5:             $S_{ij} \leftarrow l \cup S_{ij}$
6: Solve $\mathbb{P}_1$ based on $S_{ij}$;
7: Output $\boldsymbol{x} \leftarrow \{x_{ilj}, \forall l \in S_{ij}, i, j \in \mathcal{V}\}$.
```

3.5 Performance Evaluation

In this section, we evaluate the performance of SRBS in comparison to SRTE with two metrics: (1) the computation time, and (2) the maximum link utilization.

We conduct our experiments to evaluate the validity of our proposed approaches on two network topologies from [40]: 100 nodes 572 links topology with 9,817 traffic demands, and 79 nodes 294 links topology with 6,161 traffic demands. We show visualization of 100 nodes and 79 nodes topologies as network graph in Figure 3.4 and Figure 3.5. The 100 nodes topology is a synthetic network topology, and the 79 nodes topology is a real topology from the Rocketfuel project [66]. On each topology, we use link weights, link capacities and traffic demands provided with the topologies. The traffic demands matrices are generated with

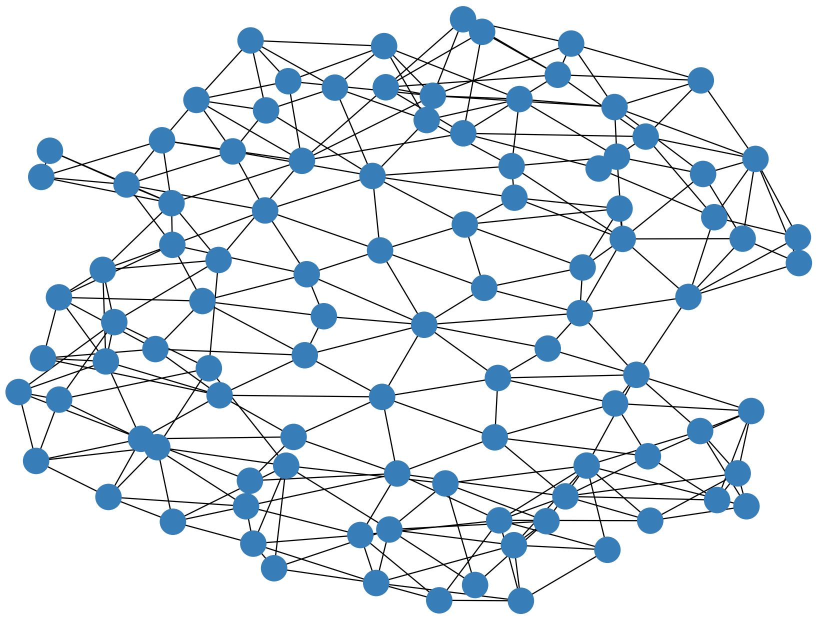


Figure 3.4: Visualization of 100 nodes network topology.

the gravity model fed by exponentially distributed random variables. However, we scale the traffic demand volume by 10 in order to saturate the network links. We conduct our simulations on 2.60 GHz 16-core Xeon processor and 64 GB of memory running Ubuntu Linux Server version 16.04 with Linux kernel 4.4.0. As for linear programming solver, we use IBM ILOG CPLEX version 12.8 for all of our simulations. We ran the experiment 10 times for each data point. In addition to that, we have error bars with 95 percent confident intervals on all data points. The results show that the fluctuation from the bars is minimal.

A common question that may arise is "how large the stretch bounding coefficient should be?". If we choose the stretch too large, then the computation time

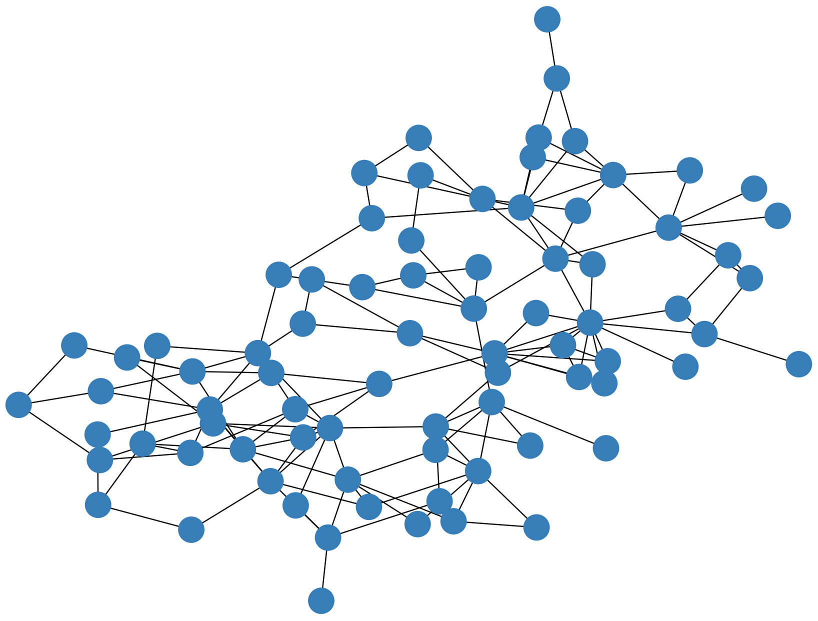
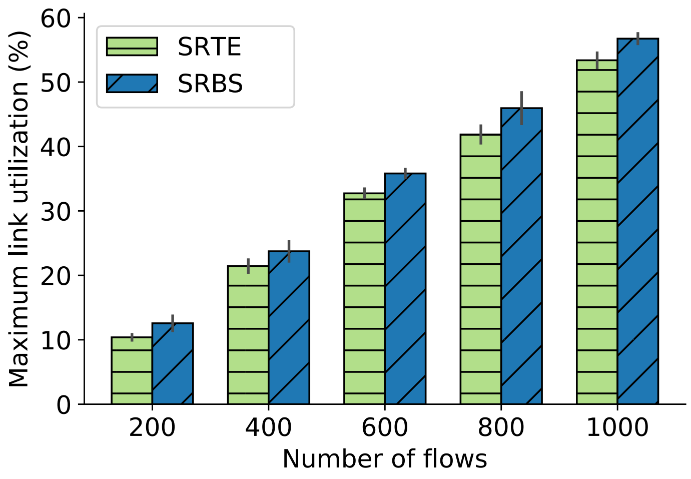


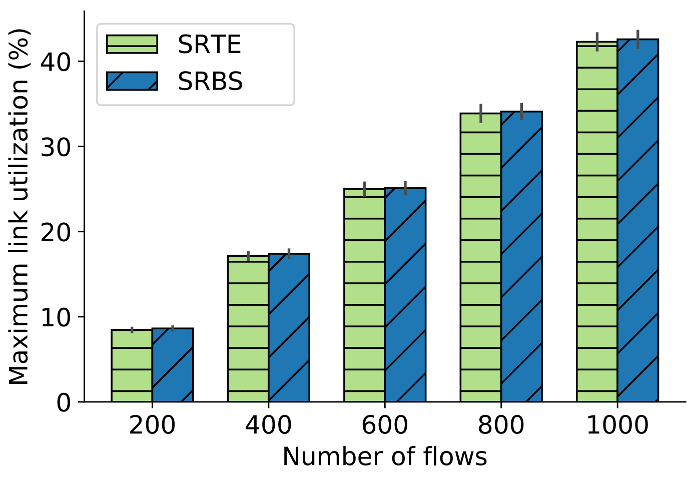
Figure 3.5: Visualization of 79 nodes network topology (Exodus) from the Rocketfuel project [66].

to solve the linear programming will be high, or the maximum utilization may increase if the stretch is too short. Hence, we can set the stretch bounding coefficient to be a small value such as 1.3, and we use this value throughout the chapter. The complete evaluation of the stretch bounding coefficient is not presented here due to limited space.

To show the performance of the proposed SRBS, we compare the SRBS to the optimal solution SRTE [18]. Figure 3.6a(a) and Figure 3.7a(b) depict the maximum link utilization and the computation time of 100 nodes topology. The results show that SRBS reduces the computation time by at most 99.6%, and the maximum link utilization is increased by only 5% at most from the optimal link



(a) The maximum link utilization ratio in 100 nodes topology.

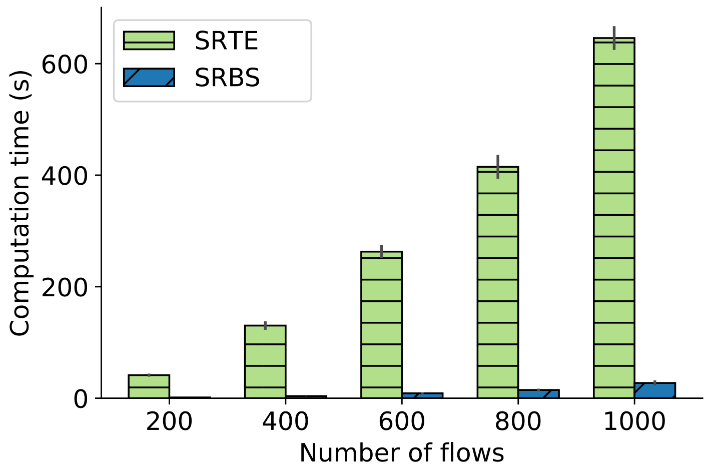


(b) The maximum link utilization ratio in 79 nodes topology.

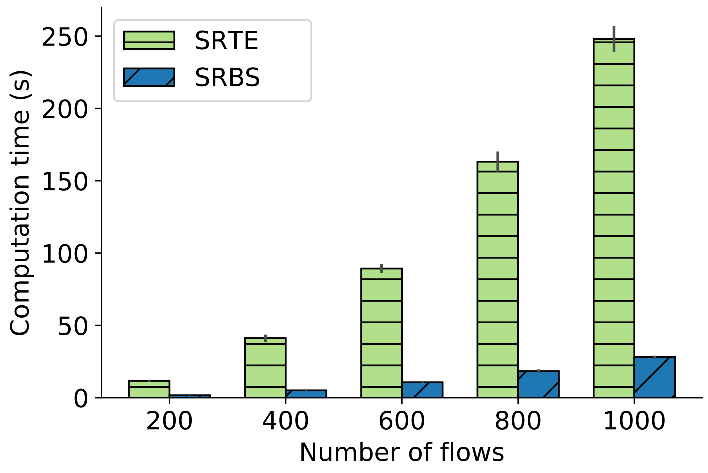
Figure 3.6: The maximum link utilization.

utilization in SRTE.

Figure 3.6b(c) and Figure 3.7b(d) show the maximum link utilization and the computation time of 79 nodes topology. The results show that our approach achieves almost the same value of the optimal maximum link utilization in SRTE,



(a) The computation time in 100 nodes topology.



(b) The computation time in 79 nodes topology.

Figure 3.7: The computation time.

while the computation time is reduced by at most 82% in this topology.

3.6 Summary

In this chapter, we proposed segment routing traffic engineering with stretch bounding. The main idea of this approach is that we leverage the stretch bounding approach, which is the path stretch that relative to the shortest path of the source-destination pair to limit the number of intermediate node candidates. Then, we compared our approach against generic SRTE approach in [18]. The simulation results show that our approach achieves a near-optimal solution while the computation time is reduced by 99.6% at most. This shows the effectiveness of stretch bounding to reduce the problem size of SRTE by excluding candidate intermediate nodes that are too far away from source-destination pairs. We believe that our approach is a promising solution for SRTE.

Chapter 4

A Computation-Efficient Approach for Segment Routing Traffic Engineering

In the previous chapter, we introduced an approach to improve the performance of segment routed traffic engineering (SRTE). In this chapter, we focus on the trade-off relationship between link utilization and computation time. In this chapter, we address SRTE with two different objectives: 1. minimize link utilization 2. minimize computation time. We then formulate a bi-objective mixed integer nonlinear programming (BOMINLP) with respect to the two objective functions. Finally, we introduce randomized sampling with stretch bounding approach to tackle this problem.

4.1 Preliminary

In this chapter, our goal is to investigate the trade-off relationship between link utilization and computation time in SRTE. To this end, we formulate a 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 as:

1. node selection,
2. 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 chapter are as follows.

- We investigate the trade-off relationship 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 linear programming (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.

4.2 System Model

4.2.1 Node Selection

By leveraging the stretch concept, the number of candidate intermediate nodes can be greatly reduced through stretch bounding. The purpose of stretch bounding is to specifically manipulate the path length for transmission (i.e., stretch), thereby reducing transmission latency [43, 30]. However, it remains unknown from these works how to use stretch bounding for link utilization in SRTE.

To indicate whether node l serves as a candidate intermediate node after applying stretch bounding, we define the indicator as

$$v_{ilj}(\alpha) = \begin{cases} 1, & \text{str}(i,l,j) \leq \alpha, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall i,l,j \in \mathcal{V}, \tag{4.1}$$

which is used to ensure that $\text{str}(i,l,j)$ is not greater than a stretch bounding coefficient $\alpha \geq 1$. Note that $\alpha = 1$ states that only the shortest paths are used, and $\alpha = \infty$ represents no stretch bounding takes effect. For example, see Figure 4.1, $v_{inj} = 1$ and $v_{inj} = 1$ indicate that node n and node m are inside stretch bounding. While $v_{iqj} = 0$ and $v_{ipj} = 0$ indicate that node p and node q are too far away from

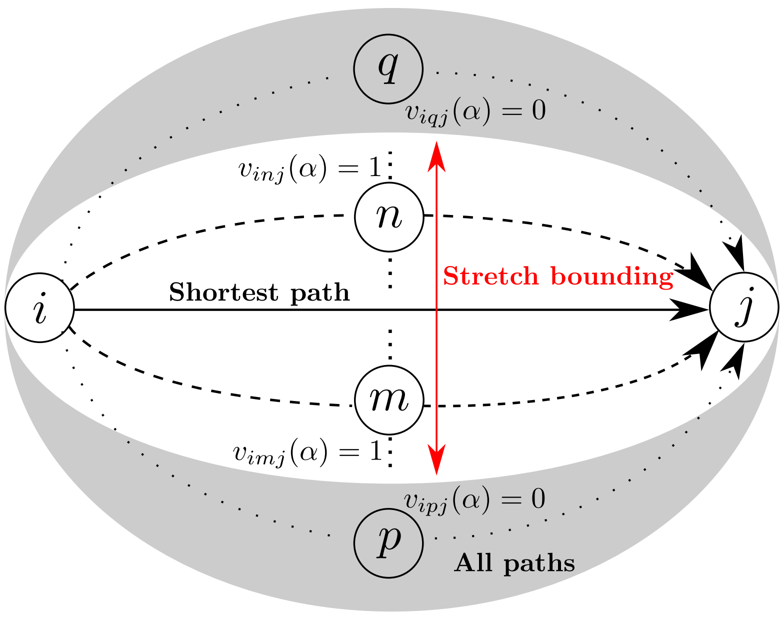


Figure 4.1: Stretch bounding approach.

the source node i and destination node j.

4.3 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 BOMINLP to jointly minimize maximum link utilization θ and computation time ζ via

$$\boldsymbol{x} = \{x_{ilj}, \forall i, l, j \in \mathcal{V}\}, \qquad \text{(flow assignment)} \tag{4.2}$$

$$\boldsymbol{y} = \{y_{ilj}, \forall i, l, j \in \mathcal{V}\}, \qquad \text{(node selection)} \tag{4.3}$$

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

$$
\begin{aligned}
&\mathbb{P}_2 && (\textbf{Joint Node Selection and Flow Assignment}):\\
&\min_{\boldsymbol{y}} && \zeta \triangleq h(\boldsymbol{y}), \qquad \longleftarrow \text{computation time}\\
&\min_{\boldsymbol{x}} && \theta, \qquad \longleftarrow \text{maximum link utilization}\\
&\text{s.t.} && \textbf{C2.1: } \sum_{l\in\mathcal{V}} x_{ilj} y_{ilj} = t_{ij}, \quad \forall i, j \in \mathcal{V},\\
& && \textbf{C2.2: } \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}\sum_{l\in\mathcal{V}} g_{ilj}(e) x_{ilj} y_{ilj} \le \theta \cdot c(e), \ \forall e \in \mathcal{E},\\
& && \textbf{C2.3: } \sum_{l\in\mathcal{V}} y_{ilj} = \min(\sum_{l\in\mathcal{V}} v_{ilj}(\alpha), \lceil \beta|\mathcal{V}|\rceil), \ \forall i, j \in \mathcal{V},\\
& && \textbf{C2.4: } y_{ilj} \in \{0, 1\}, \qquad \forall i, l, j \in \mathcal{V},\\
& && \textbf{C2.5: } x_{ilj} \ge 0, \qquad \forall i, l, j \in \mathcal{V},
\end{aligned}
$$

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. This monotonically increasing property can be observed in general SRTE (e.g. [69, 64]). **C2.1** represents that the flow assignment should satisfy all traffic demands. **C2.2** ensures that the flows routed through a link do not exceed the link capacity. **C2.3** 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. However, 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. Furthermore, as a complement, β can be used to limit the number of candidate intermediate nodes (e.g. $\beta = 2.5$-7% [69]). **C2.4** and **C2.5** refer to the auxiliary constraints for node selection and flow assignment, respectively. Note that constraint **C2.1** and **C2.2** are different from the **C1.1** and **C1.1** in a way that **C2.1** and **C2.2** have $\boldsymbol{y}$ as a multiplier for node selection.

In essence, $\mathbb{P}_2$ is formulated to characterize the trade-off between maximum link utilization and computation time. However, there is a lack of solution approaches for tackling $\mathbb{P}_2$ 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 $\mathbb{P}_2$ optimally is in essence NP-hard (as a general integer linear program (ILP) [36]).

- *Nonlinear constraints.* **C2.1** and **C2.2** 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 $\mathbb{P}_2$ optimally. Therefore, we are motivated to consider the following two sequential sub-problems:

$$\begin{aligned} &\mathbb{P}_3 \;\; (\textbf{Node Selection}): \\ &\min_{\boldsymbol{y}} \;\; \zeta, \\ &s.t. \textbf{C2.3}, \textbf{C2.4}, \end{aligned}$$

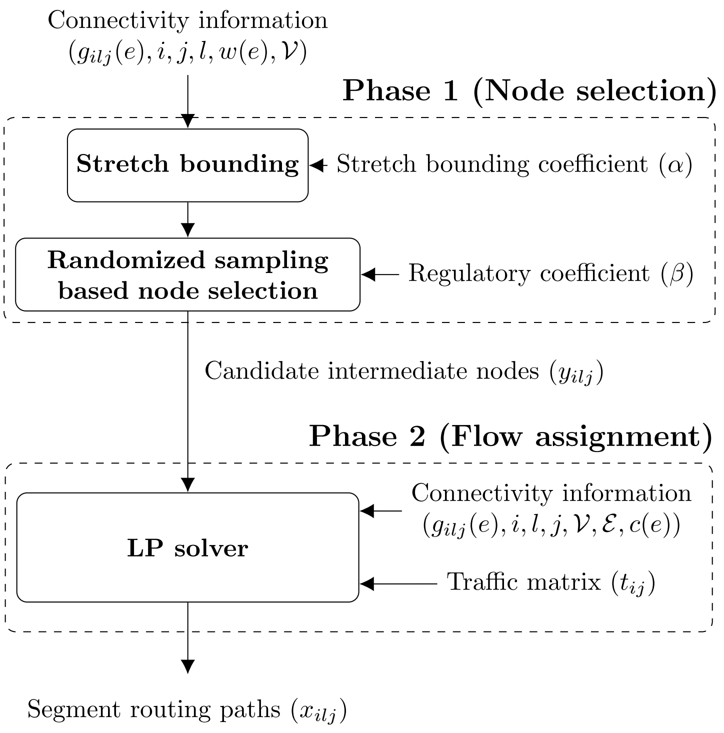
and

$$\begin{aligned} &\mathbb{P}_4 \;\; (\textbf{Flow Assignment}): \\ &\min_{\boldsymbol{x}|\boldsymbol{y}} \;\; \theta, \\ &s.t. \textbf{C2.1}, \textbf{C2.2}, \textbf{C2.5}. \end{aligned}$$

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 $\mathbb{P}_3$ through $\boldsymbol{y}$, based on which we find out a flow assignment in $\mathbb{P}_4$ through $\boldsymbol{x}$. Since $\mathbb{P}_3$ and $\mathbb{P}_4$ are an ILP and an LP, respectively, we can then design a computation-efficient algorithm for solving them in practice.

Remark 1 (Decomposition loss)**.** *In fact,* $\mathbb{P}_3$ *and* $\mathbb{P}_4$ *retain all of the constraints in* $\mathbb{P}_2$*; therefore the feasible solution space of* $\mathbb{P}_2$ *can be kept intact after the problem decomposition.*

4.4 Algorithm Design



Connectivity information
$(g_{ilj}(e), i, j, l, w(e), \mathcal{V})$
Phase 1 (Node selection)
Stretch bounding
Stretch bounding coefficient (α)
Randomized sampling based node selection
Regulatory coefficient (β)
Candidate intermediate nodes (y_{ilj})
Phase 2 (Flow assignment)
LP solver
Connectivity information
$(g_{ilj}(e), i, l, j, \mathcal{V}, \mathcal{E}, c(e))$
Traffic matrix (t_{ij})
Segment routing paths (x_{ilj})

Figure 4.2: The algorithmic overview of SRTE$^+$.

In this section, we propose a two-phase algorithm (denoted by SRTE$^+$) to address the decomposed sub-problems $\mathbb{P}_3$ and $\mathbb{P}_4$. 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 in which each intermediate node has equal probability to be chosen as candidate intermediate nodes during the sampling process. The purpose of the randomized sampling is to distribute the traffic load among a limited number

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

```
Input: $\alpha$, $\beta$, $\mathcal{V}$, $\mathcal{E}$, $\{c(e), \forall e \in \mathcal{E}\}$, $\{t_{ij}, \forall i, j \in \mathcal{V}\}$, $\{g_{ilj}(e), \forall i, l, j \in \mathcal{V}, e \in \mathcal{E}\}$.
Output: $\boldsymbol{x}$.
    (Phase 1) Randomized Sampling based Node Selection
 1: Initialize $\bar{\mathcal{V}}_{ij} \leftarrow \{l | v_{ilj}(\alpha) = 1 \text{ in } (4.1), \forall i, l, j \in \mathcal{V}\}$;
 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 $|\bar{\mathcal{V}}_{ij}| \le \lceil \beta |\mathcal{V}| \rceil$ 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 $\mathbb{P}_4$ based on $\boldsymbol{y}$;
14: Output $\boldsymbol{x} \leftarrow \{x_{ilj}, \forall l \in \bar{\mathcal{V}}_{ij}, i, j \in \mathcal{V}\}$.
```

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 $\mathbb{P}_4$. Note that Algorithm 2 leverages α and β as partial information to reduce the computation time, since the exact form of h may not be available.

Algorithm 2 presents a computation-efficient 2-phase algorithm for SRTE$^+$ (see Figure 4.2 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 $\mathbb{P}_4$ with the reduced number of candidate intermediate nodes in $\boldsymbol{y}$ (the number of $\boldsymbol{y}$ being 1 is reduced), which implies the reduced problem size of $\mathbb{P}_4$.

Since LP can be solved in polynomial time using Karmarkar's interior-point algorithm, which grows in cubic order of the number of variables [47]. 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. 4.5, 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 $\mathbb{P}_4$ is in essence an LP, which can be solved by various software tools (e.g. CPLEX[45], GUROBI [39] or GLPK [53]).

Remark 2. *To implement SRTE$^+$ on real networks, the most common way is to leverage a software define 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 the flow assignment.*

4.5 Performance Evaluation

In this section, we first choose balance stretch bounding coefficients for our proposed solution in different network topologies. Then, we compare our proposed solution with various traffic engineering (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.

4.5.1 Simulation settings

4.5.1.1 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 [45].

4.5.1.2 Datasets

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

- The REPETITA dataset [38] 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. We show the selected topologies on world map in Figure 4.3 - Figure 4.7.

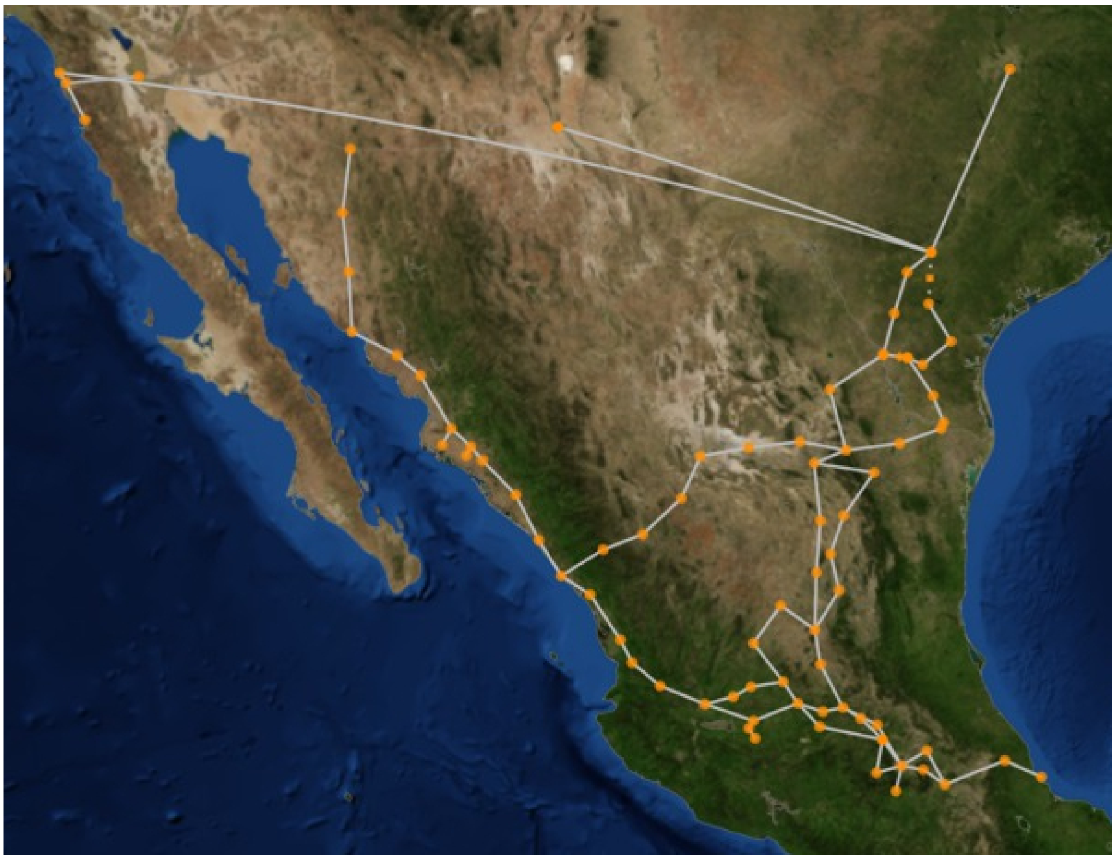


Figure 4.3: RedBestel network topology [48].

Figure 4.4: VtlWavenet network topology [48].

- The GEANT dataset [70] contains one network topology and 10,772 real-world traffic matrices (96 of which will be chosen for our performance evaluation). We show the topology in Figure 4.8.

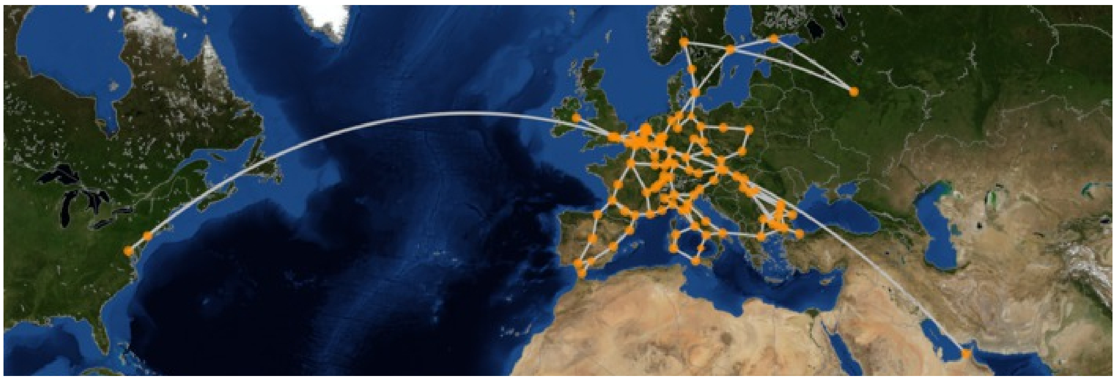


Figure 4.5: Interoute network topology [48].

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 4.2.

4.5.1.3 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.

4.5.1.4 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

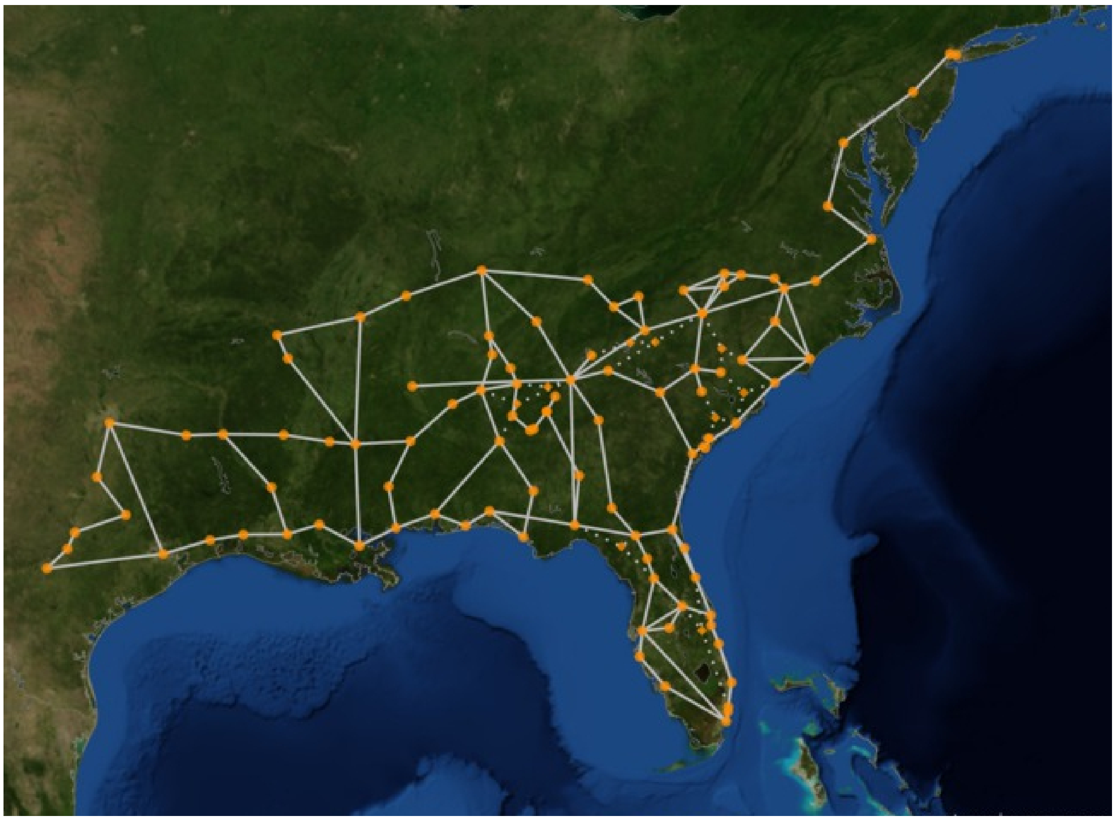


Figure 4.6: Deltacom network topology [48].

by a shortest path algorithm (e.g. Dijkstra algorithm [24] and Bellman-Ford algorithm[16]).

Note that 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.

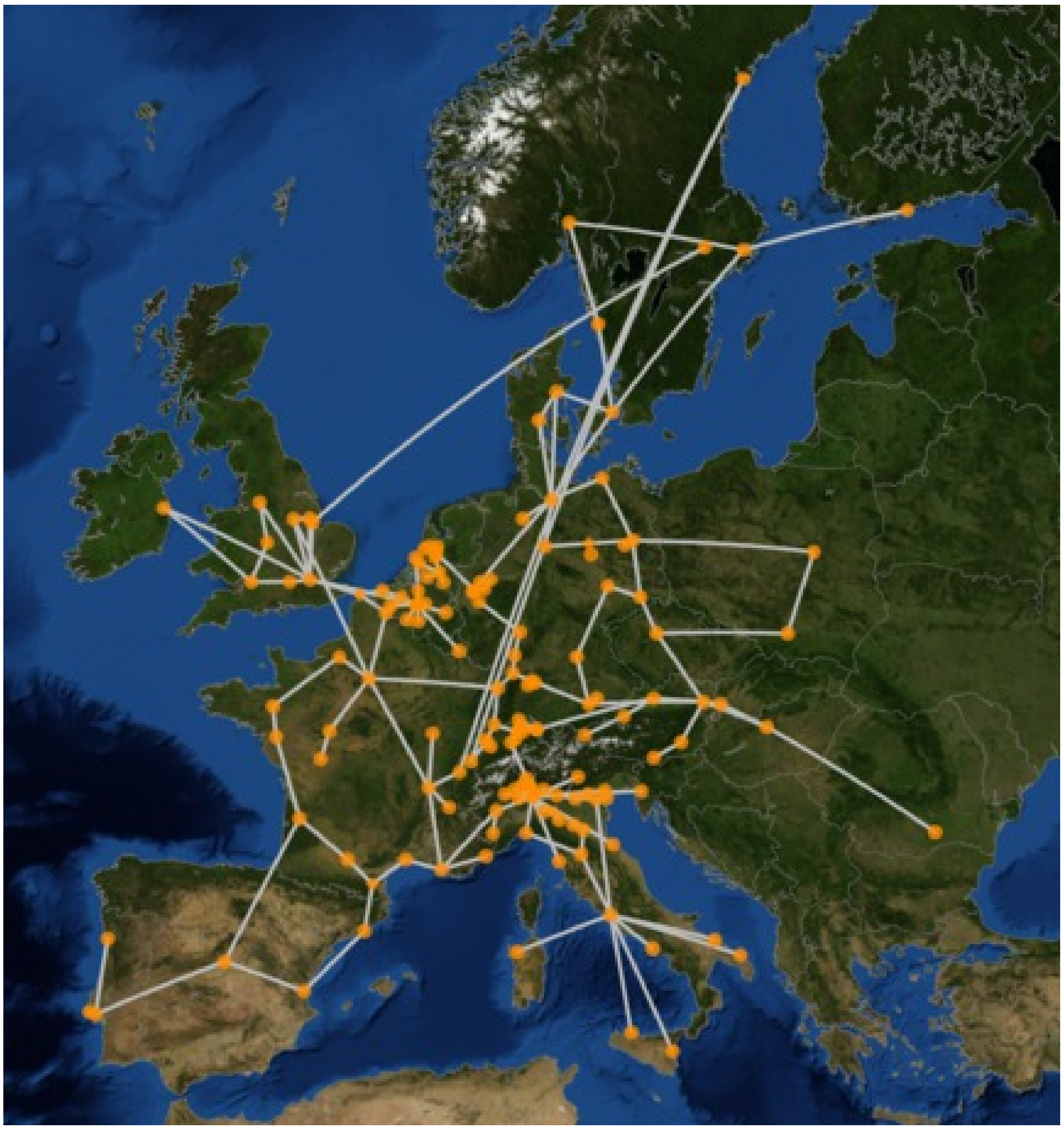


Figure 4.7: Colt network topology [48].

4.5.1.5 Comparison Schemes

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

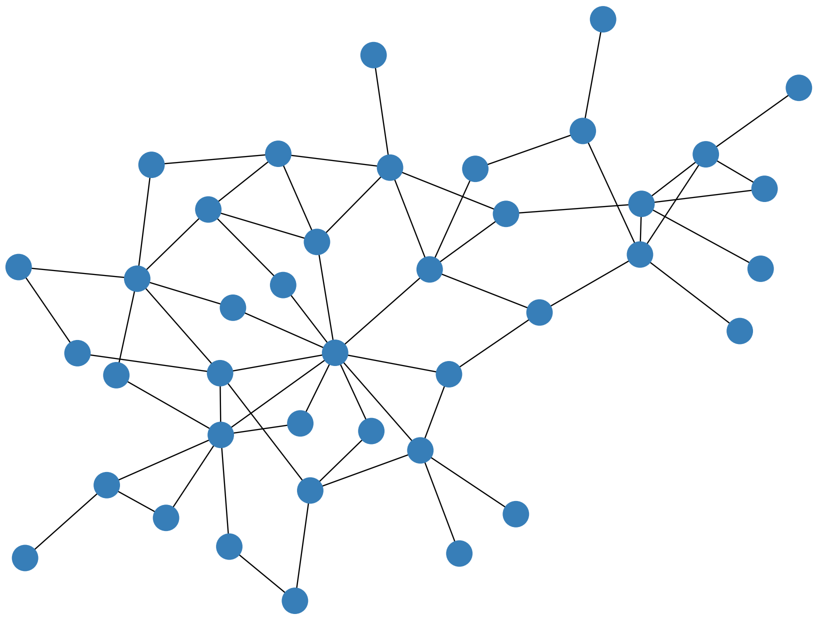


Figure 4.8: Visualization of GEANT network topology.

- SRTE[18]: all of the nodes in $\mathcal{V}$ are chosen as candidate intermediate nodes.
- SRBS[64]: candidate intermediate nodes are chosen similarly as SRTE$^+$, except that SRBS neither regulates the size of candidate intermediate nodes nor performs randomized sampling.
- DEG[69]: 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[69]: 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[69]: $\lceil\beta|\mathcal{V}|\rceil$ of nodes are chosen uniformly as candidate intermediate

Table 4.1: Datasets, network topologies and traffic matrices.

Dataset name	Topology name	Traffic matrix
REPETITA	RedBestel (see Figure 4.3)	RedBestel.0000
REPETITA	VtlWavenet (see Figure 4.4)	VtlWavenet2011.0003
REPETITA	Interoute (see Figure 4.5)	Interoute.0001
REPETITA	Deltacom (see Figure 4.6)	Deltacom.0001
REPETITA	Colt (see Figure 4.7)	Colt.0001
GEANT	GEANT (see Figure 4.8)	IntraTM-2005-04-29-09-15

nodes.

For simplicity, we focus on the unary routing metric (i.e., all links have equal weights) in Sec. 4.5.2.2 and 4.5.2.3. The impact of various routing metrics will be left to Sec. 4.5.2.4.

4.5.2 Simulation results

4.5.2.1 Comparison between SRTE (2-seg) and SRTE (∞-seg)

Figure 4.9 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 results show that 2-seg has similar performance compared with ∞-seg. Thus,

Table 4.2: Datasets, network topologies and traffic matrices.

Topology name	**# of nodes**	**# of edges**	**# of traffic matrices**	**# of flows**
RedBestel	84	93	5	5000
VtlWavenet	92	96	5	5000
Interoute	110	148	5	5000
Deltacom	113	161	5	5000
Colt	153	177	5	5000
GEANT	27	38	96	729

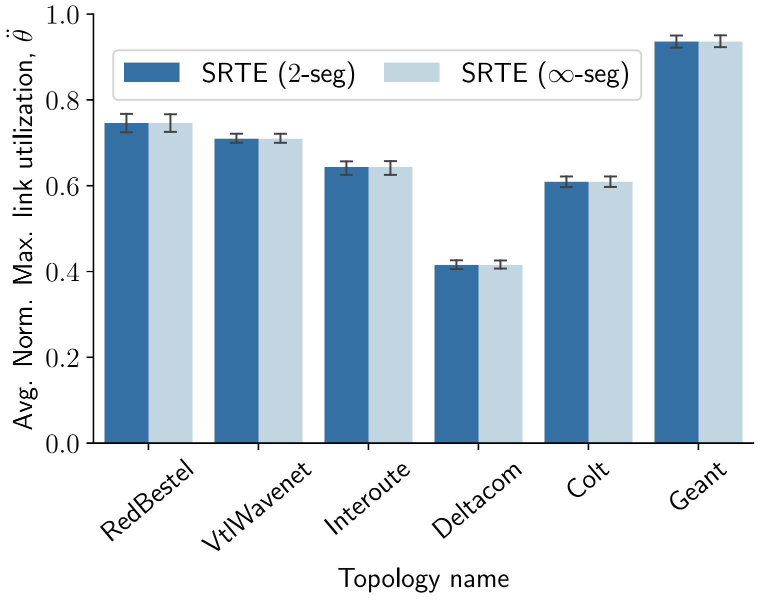


Figure 4.9: A comparison between SRTE (2-seg) and SRTE (∞-seg).

based on these results, we focus on 2-seg in this dissertation.

Stretch bounding coefficient	Topology name					
	RedBestel	VtlWavenet	Interoute	Deltacom	Colt	GEANT
1.0–1.1	−0.193	−0.445	−1.851	−0.300	−0.104	0
1.1–1.2	−0.193	−0.843	−0.853	−2.784	−1.740	0
1.2–1.3	−0.569	−0.673	−0.026	0	−1.037	−0.003
1.3–1.4	0	−0.005	0	0	−0.257	−3.583
1.4–1.5	0	0	0	0	0	−1.279
1.5–1.6	0	0	0	0	0	0
1.6–1.7	0	0	0	0	0	0
1.7–1.8	0	0	0	0	0	0
1.8–1.9	0	0	0	0	0	0
1.9–2.0	0	0	0	0	0	0
2.0–∞	0	0	0	0	0	0

(↑ a zero/negative value indicates a flat/decreasing line segment)

Table 4.3: The line slopes of θ corresponding to Figure 4.10.

4.5.2.2 The choices of stretch bounding coefficients

Choosing stretch bounding coefficients differently has direct impacts on both of the performance metrics as depicted in Figure 4.10 and Figure 4.11.

Large stretch bounding coefficients improve link utilization. In Figure 4.10, as the stretch bounding coefficient increases, the maximum link utilization tends to decrease for all network topologies. This is because larger stretch bounding coefficients can offer more candidate intermediate nodes and more path

Stretch bounding coefficient	**Topology name**					
	RedBestel	VtlWavenet	Interoute	Deltacom	Colt	GEANT
1.0–1.1	32.092	61.36	48.816	49.118	10.192	0.033
1.1–1.2	150.125	131.084	134.086	179.566	326.281	1.914
1.2–1.3	105.08	129.373	138.285	166.828	238.813	2.669
1.3–1.4	141.799	124.471	143.782	169.909	293.664	7.567
1.4–1.5	116.581	141.88	171.006	192.088	245.424	1.104
1.5–1.6	91.373	91.906	116.472	111.613	198.656	4.967
1.6–1.7	99.266	125.363	126.973	164.539	211.177	2.937
1.7–1.8	115.424	132.483	167.05	242.768	444.01	2.917
1.8–1.9	61.117	104.514	124.094	57.422	54.932	1.559
1.9–2.0	111.783	149.623	209.362	483.603	243.254	4.438
2.0–∞	1211.555	1826.717	1417.152	1257.509	2473.967	28.273

($\uparrow$ a positive value indicates a monotonically increasing line segment)

Table 4.4: The line slopes of ζ corresponding to Figure 4.11.

choices. In addition, we observe that the maximum link utilization remains unchanged when the stretch bounding coefficient becomes large. The reason is that large stretch bounding coefficients 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 4.10 can vary greatly with network topologies.

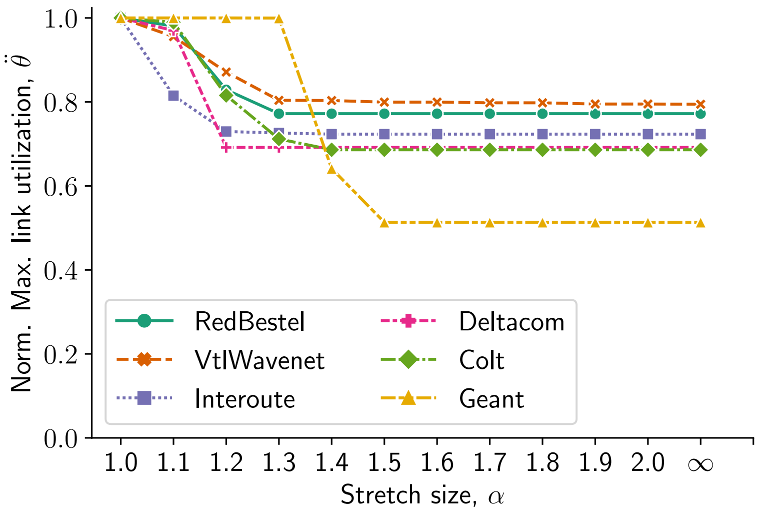


Figure 4.10: The impact of stretch bounding coefficient (unary): The maximum link utilization.

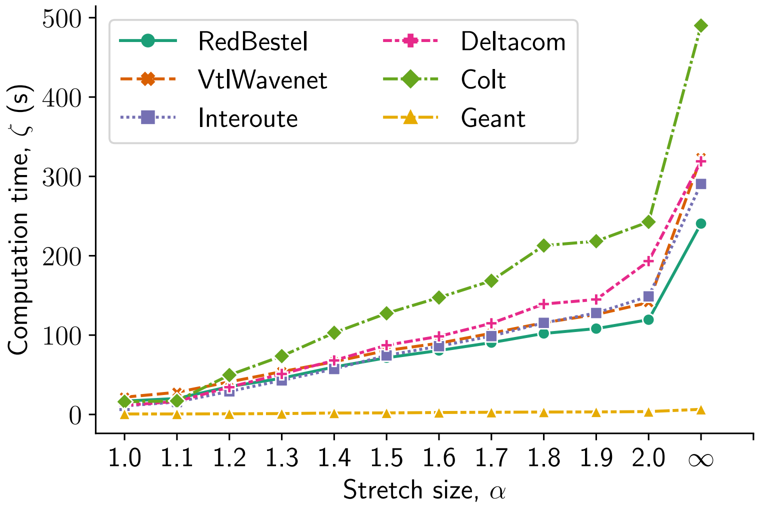


Figure 4.11: The impact of stretch bounding coefficient (unary): the computation time.

Large stretch bounding coefficients require heavy computation. In Figure 4.11, we see that the computation time is monotonically increasing with the stretch bounding coefficient.

The reason is that large stretch bounding coefficients 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.

Stretch bounding coefficients balance the performance metrics. As the maximum link utilization remains unchanged and the computation time increases monotonically, we can certainly choose the smallest stretch bounding coefficient (the saturated point) that corresponds to the lowest maximum link utilization. In other words, increasing the stretch bounding coefficient beyond the saturated point does not help decrease the link utilization, but will incur higher computation time. By looking at Figure 4.10 and Figure 4.11, we see that there exist stretch bounding coefficients that balance the maximum link utilization and computation time on each network topology. Accordingly, we have the balanced stretch bounding coefficients for each network topology, as shown in Table 4.5, which will be used for the following subsections.

4.5.2.3 The comparison among TE approaches

Figure 4.12 and Figure 4.13 illustrate how SRTE$^+$ outperforms other TE approaches in terms of the two performance metrics.

SRTE$^+$ improves link utilization effectively. In Figure 4.12, SRTE$^+$ outperforms DEG, BETW, and RAND in terms of the maximum link utiliza-

Table 4.5: The list of balanced stretch bounding coefficients (the grey-shaded column and row refer to the settings used in Figure 4.12-4.15.

Dataset name	**Topology name**	**Unary**	**Delay**	**Inverse**
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
GEANT	GEANT	1.5	1.4	1.2

Topology	**SRTE$^+$ (θ)**	**SRTE (θ)**	**SRTE$^+$ (ζ)**	**SRTE (ζ)**
RedBestel	0.775	0.746	14.823	362.901
VtlWavenet	0.724	0.710	7.521	412.117
Interoute	0.655	0.643	14.629	382.064
Deltacom	0.436	0.415	28.503	409.602
Colt	0.653	0.609	43.417	489.671
GEANT	0.936	0.936	0.106	4.481

Table 4.6: A comparison between SRTE$^+$ and SRTE (corresponding to Figure 4.12 and Figure 4.13).

tion. The reason is that stretch bounding allows SRTE$^+$ to use diverse candidate intermediate nodes, whereas the centrality approaches use static candidate inter-

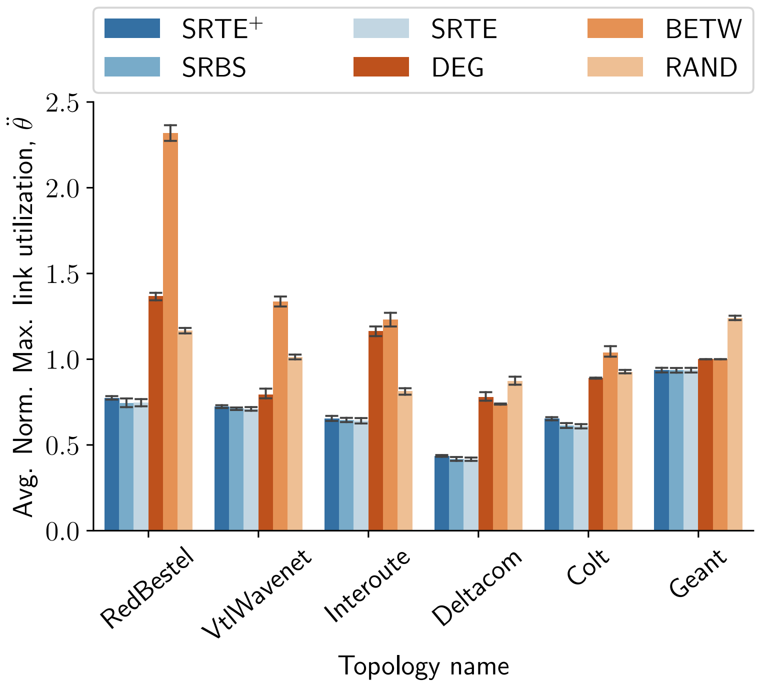


Figure 4.12: Performance comparison among various approaches ($\beta = 0.04$, unary): the maximum link utilization.

Dataset name	Topology name	# of nodes	# of edges	Computation time (s)
REPETITA	RedBestel	84	93	14.823
REPETITA	VtlWavenet	92	96	7.251
REPETITA	Interoute	110	148	14.629
REPETITA	Deltacom	113	161	28.503
REPETITA	Colt	153	177	43.416
GEANT	GEANT	27	38	0.106

Table 4.7: The computation time of SRTE$^+$ (corresponding to Figure 4.13).

mediate 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

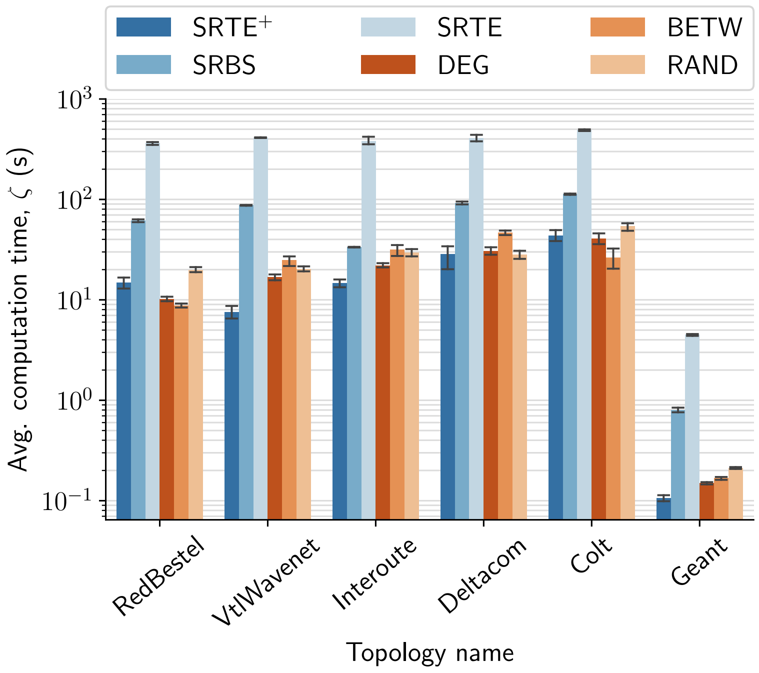


Figure 4.13: Performance comparison among various approaches ($\beta = 0.04$, unary): The computation time (in log scale).

SRBS, the maximum link utilization remains low in SRTE$^+$.

SRTE$^+$ reduces the computation time enormously. In Figure 4.13 (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

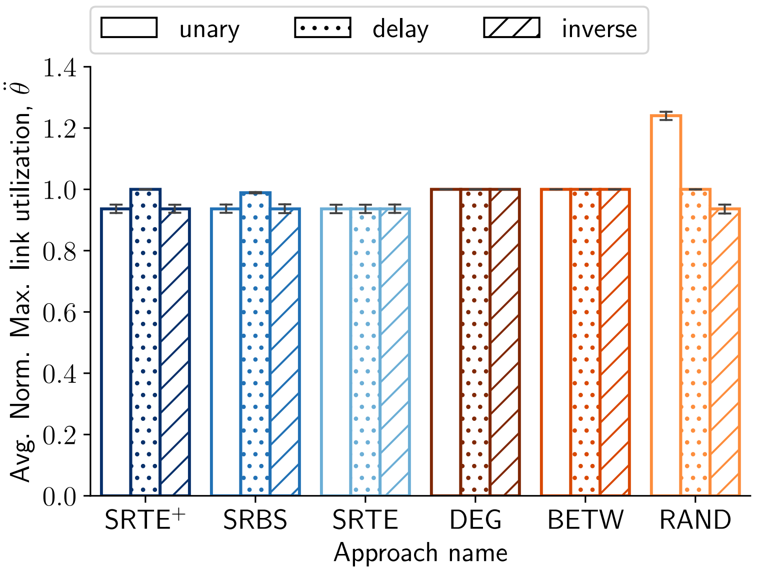


Figure 4.14: The impact of routing metrics ($\beta = 0.04$, Geant): the maximum link utilization.

intervals, e.g. 5-10 minutes[41].

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.

4.5.2.4 The impact of routing metrics

Figure 4.14 and Figure 4.15 show how SRTE$^+$ varies with other routing metrics and how it outperforms other TE approaches.

SRTE$^+$ thrives on various routing metrics. In Figure 4.14, SRTE$^+$

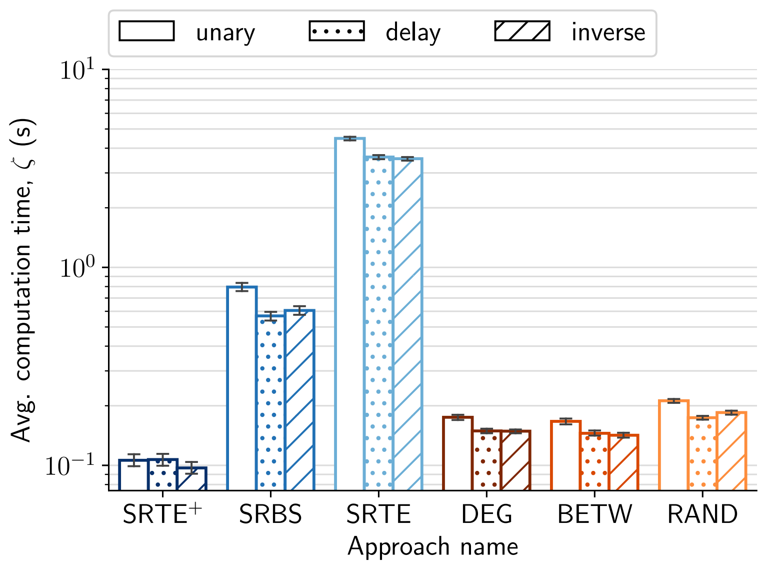


Figure 4.15: The impact of routing metrics ($\beta = 0.04$, Geant): the computation time (in log scale).

has comparable maximum link utilization as compared to SRTE and SRBS even SRTE$^+$ has a much fewer 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 4.15, SRTE$^+$ outperforms all comparison TE approaches across all routing metric in terms of computation time since it has small LP problem size.

4.6 Discussion

4.6.1 Implementation on real networks

Solving general linear or nonlinear programs could be time-consuming and challenging in dynamic systems. In [41], Hong *et al.* state that traffic engineering programs must be invoked periodically by network operators in short intervals, e.g. 5-10 minutes. To meet this time requirement, we are thus motivated to propose the computation-efficient two-phase algorithm (i.e. SRTE$^+$) to facilitate segment routing traffic engineering in practice. Our simulation results demonstrate that, for all network topologies that we have tested, the computation time of SRTE$^+$ can be kept within one minute (see the last column of Table 4.7). Since our performance evaluation is conducted entirely based on publicly available datasets, it reveals that SRTE$^+$ indeed provides a computation-efficient solution in real networks.

On the other hand, to implement SRTE$^+$ on real networks, the most common way is to leverage a SDN architecture. By using the SDN architecture, we can directly place our optimization program (namely SRTE$^+$) on an SDN controller. After running the optimization program, the SDN controller will configure routers accordingly. More details regarding SDN and segment routing architectures can be found in [26] and [29], respectively.

4.6.2 Decomposition loss

The problem decomposition (from P1 to P2 and P3) actually does not result in a loss, since P2 and P3 retain all constraints of P1, thereby keeping the feasible solution space of P1 intact after the problem decomposition. However, SRTE$^+$ may result in losses as it selects part of the network nodes as candidate intermediate nodes.

Nevertheless, we can show that SRTE$^+$ is actually computation-efficient through our simulations. To show this, we compare SRTE$^+$ with SRTE. Since SRTE leverages all network nodes (while SRTE$^+$ only uses part of the network nodes) to reduce network congestion, it serves as a lower bound of SRTE$^+$ with respect to θ. Table 4.6 compares the performances of θ and ζ between SRTE$^+$ and SRTE, from which we see that

Although SRTE$^+$ sacrifices a small amount of θ (by at most 7%), the computation time ζ can be reduced enormously (by at least 91%), which explains that SRTE$^+$ is a computation-efficient solution approach.

4.6.3 The relationship between the line curves and a network topology

The relationship between the line curves (in Figure 4.10 and Figure 4.11) and a network topology can be described as follows. Figure 4.10, there is no explicit relationship between the line curves and the network topologies, since it depends on link weights and network connectivity (i.e. how vertices are connected), which varies greatly with network topologies.

However, we see that the link utilization θ decreases with stretch bounding coefficients α and then becomes saturated for all network topologies. Figure 4.11, the larger the network topology, the higher the computation time. In addition, the computation time tends to grow faster for larger network topologies, and it increases monotonically with stretch bounding coefficients for all network topologies.

Note that in order to determine the line curves, we have to specify a range of α first, and then calculate the line curves based on the simulation results of θ and ζ. Particularly, in this paper, we concentrate on the range of $[1.0, 2.0]$ since balanced stretch bounding coefficients for the network topologies that we have tested all lie within the range of $[1.2, 1.5]$. Then, we will have the simulation results of θ and ζ with respect to α, based on which we can get the line curves (provided in Tables 2 and 3).

From these two tables, we arrive at the following observations.

- We can see that the link utilization generally decreases with stretch bounding coefficients (i.e. negative slopes) and will be saturated (i.e. zero slopes) as the stretch bounding coefficients are sufficiently large.
- We can observe that the computation time is monotonically increasing (i.e. positive slopes) with stretch bounding coefficients.

4.7 Summary

In this chapter, 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.

Chapter 5

Conclusion

How do we improve the performance of SRTE such that SRTE can be used in practice? This dissertation attempted to answer this question: by studying the network operators' desired properties for TE in general and also considering different objectives to reduce network congestion. First, we introduced a stretch bounding approach to reduce the problem size of SRTE. Our results showed that we arrive at a performance gain for SRTE that network operators can be used under their tight computation intervals.

Apart from the performance gain in SRTE, this dissertation also studies the trade-off characteristics of link utilization and computation time. We formulate the problem as a bi-objective mixed-integer nonlinear program (BOIMNLP). Due to the hardness in solving the problem directly, we decompose the problem into two sequential sub-problems of node selection and flow assignment. Consequently, we proposed a randomized sampling approach based on stretch bounding to solve the node selection problem. Then, we leverage linear programming (LP) solvers for the flow assignment problem. The results showed that the proposed approach could effectively balance the link utilization and computation time.

In summary, we believe that the approaches developed in this dissertation can help improve the performance of SRTE such that it can be used in real networks. In the rest of this chapter, we conclude some of the lessons learned and also highlight possibilities for future work in Sec. 5.1. Finally, we summarize this dissertation in Sec. 5.2.

5.1 Discussion and Future Work

Certainly, our approaches and results are bound to limitations in several ways. In this section, we discussed how to extend the main idea of this dissertation so that the assumption is less restrictive or more practical in the real systems. We also discussed the possibility of other considerations regarding SRTE in various aspects.

5.1.1 Traffic matrix

Throughout this dissertation, we have made an assumption that network operators have obtained traffic matrices using a combination of methods (e.g. traffic analysis and traffic measurement) [15, 19, 71, 73]. While this is possible in practice, there are some cases where the traffic matrix may not be representative of the current network traffic. First, the limitation of traffic measurement may limit the quality of traffic matrix derivation. Second, realistic traffic matrix datasets are rarely available since this information is sensitive. Besides, large-scale traffic measurement is not easy to perform [50].

5.1.2 Online segment routing traffic engineering

Throughout this dissertation, we have an assumption that a traffic matrix at a given time is known. While the traffic matrix can be obtained through traffic matrix monitoring and approximation methods. Often, the obtained traffic matrix is not accurate, or the computation interval is too large to adapt to traffic changes. As a consequence, one might consider an online approach for SRTE problem without knowledge of future arrival of traffic nor traffic matrix. Note that the online method has its limitation as well; as the future arrival of traffic is unknown, then the performance may be somewhat far from the optimal [18].

5.1.3 Maximize total network throughput

Although we only consider minimizing congestion with the minimize link utilization objective in our optimization programs, other objectives can also be considered, such as maximize total network throughput. Apart from minimizing congestion variant of objective functions, this variant of objective function is also being used inside autonomous systems as well.

5.1.4 Machine learning approach

In this dissertation, we leverage the stretch bounding approach and also randomized sampling for candidate intermediate nodes selection, respectively. However, one can consider an alternative approach for candidate intermediate nodes selection based on machine learning algorithms (e.g. neural networks) or even reinforcement learning to learn intermediate nodes selection based on stretch bounding or even solely use machine learning for routing.

5.1.5 Stretch bounding improves delay and fairness

Stretch bounding approach can be considered as an approach that helps improve both delay and fairness. Stretch bounding limits the path length for each source-destination pairs such that network delay is naturally limited. Also, fairness regarding each source-destination pairs is also improved as stretch bounding balance the traffic to route to different paths. Note that in the network designers' perspective, there is a trade-off between link utilization and delay when designing a fully utilized network. However, it is beyond the scope of this work.

5.2 Final Remarks

We have presented two approaches to address SRTE problem. The first one is segment routing traffic engineering with bounded stretch. We showed that the proposed approach has near-optimal performance in terms of link utilization, while computation time can be reduced enormously. The second one is a computation-efficient approach for segment routing traffic engineering. Even though we cannot claim the optimality of the proposed approach, we have shown that our proposed approaches can balance the link utilization and computation time effectively. The approaches described in this dissertation aim to tackle SRTE problem where the centralized computing element (e.g. path computation element, software-defined network (SDN) controller) is assumed to be present in the network. To the best of our knowledge, this dissertation is the first to study stretch bounding approach in SRTE. In addition, we are first to study the trade-off between two essential metrics in SRTE (i.e. link utilization and computation time). We believe that this dissertation will benefit network operators who utilize the simplicity and efficiency of segment routing in general.

Bibliography

[1] A. Abdelsalam et al. “Implementation of virtual network function chaining through segment routing in a linux-based NFV infrastructure”. In: *IEEE NetSoft*. July 2017, pp. 1–5.

[2] S. Agarwal, M. Kodialam, and T. V. Lakshman. “Traffic engineering in software defined networks”. In: *IEEE INFOCOM*. Ap. 2013, pp. 2211–2219.

[3] Sugam Agarwal, Murali Kodialam, and TV Lakshman. “Traffic engineering in software defined networks”. In: *Proc. IEEE INFOCOM*. Apr. 2013, pp. 2211–2219.

[4] I. F. Akyildiz et al. “Research challenges for traffic engineering in software defined networks”. In: *IEEE Network* 30.3 (May 2016), pp. 52–58.

[5] Ian F Akyildiz et al. “A new traffic engineering manager for DiffServ/MPLS networks: design and implementation on an IP QoS Testbed”. In: *Elsevier Comput. Commun.* 26.4 (2003), pp. 388–403.

[6] L. Andersson et al. “LDP Specification”. In: *RFC 5036* (Oct. 2015).

[7] A. Asgari et al. “Scalable monitoring support for resource management and service assurance”. In: *IEEE Network* 18.6 (Nov. 2004), pp. 6–18.

[8] Ashwin Sridharan, R. Guerin, and C. Diot. “Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks”. In: *IEEE INFOCOM*. Vol. 2. Mar. 2003, 1167–1177 vol.2.

[9] F. Aubry et al. “SCMon: Leveraging segment routing to improve network monitoring”. In: *IEEE INFOCOM*. Apr. 2016, pp. 1–9.

[10] D. O. Awduche. “MPLS and traffic engineering in IP networks”. In: *IEEE Commun. Mag.* 37.12 (Dec. 1999), pp. 42–47.

[11] D. Awduche et al. “Requirements for Traffic Engineering Over MPLS”. In: *RFC 2702* (Sept. 1999).

[12] Daniel O Awduche. “MPLS and traffic engineering in IP networks”. In: *IEEE Commun. Mag.* 37.12 (1999), pp. 42–47.

[13] Daniel O. Awduche et al. *A Framework for Internet Traffic Engineering*. Tech. rep. IETF, Nov. 2001. URL: `https://tools.ietf.org/html/draft-ietf-tewg-framework-05`.

[14] Daniel Awduche et al. “Overview and principles of Internet traffic engineering”. In: *RFC 3272* (May 2002).

[15] A. Azzouni and G. Pujolle. “NeuTM: A neural network-based framework for traffic matrix prediction in SDN”. In: *IEEE/IFIP NOMS*. Apr. 2018, pp. 1–5.

[16] Richard Bellman. “On a routing problem”. In: *Quart. Appl. Math.* 16.1 (1958), pp. 87–90.

[17] Dimitri Bertsekas and Robert Gallager. *Data Networks (2Nd Ed.)* Prentice-Hall, Inc., 1992.

[18] R Bhatia et al. “Optimized network traffic engineering using segment routing”. In: *Proc. IEEE INFOCOM*. Apr. 2015.

[19] X. Cao et al. “Interactive Temporal Recurrent Convolution Network for Traffic Prediction in Data Centers”. In: *IEEE Access* 6 (Dec. 2018), pp. 5276–5289.

[20] Martin Casado, Nate Foster, and Arjun Guha. “Abstractions for Software-defined Networks”. In: *Commun. ACM* 57.10 (Sept. 2014), pp. 86–95.

[21] Martin Casado et al. “Ethane: Taking Control of the Enterprise”. In: *SIGCOMM Comput. Commun. Rev.* 37 (2007), pp. 1–12.

[22] M. Chiesa, G. Kindler, and M. Schapira. “Traffic Engineering With Equal-Cost-MultiPath: An Algorithmic Perspective”. In: *IEEE/ACM Trans. Netw.* 25.2 (Apr. 2017), pp. 779–792.

[23] A Cianfrani, M Listanti, and M Polverini. "Incremental Deployment of Segment Routing Into an ISP Network: a Traffic Engineering Perspective". In: *IEEE/ACM Trans. Netw.* 25.5 (Oct. 2017), pp. 3146–3160.

[24] E W Dijkstra. "A note on two problems in connexion with graphs". In: *Numer. Math.* 1.1 (1959), pp. 269–271.

[25] A. Elwalid et al. "MATE: MPLS adaptive traffic engineering". In: *Proc. IEEE INFOCOM.* Vol. 3. Apr. 2001, pp. 1300–1309.

[26] Adrian Farrel et al. "An Architecture for Use of PCE and the PCE Communication Protocol (PCEP) in a Network with Central Control". In: *RFC 8283* (Dec. 2017).

[27] A. Feldmann et al. "Deriving traffic demands for operational IP networks: methodology and experience". In: *IEEE/ACM Trans. Netw.* 9.3 (June 2001), pp. 265–279.

[28] Clarence Filsfils et al. *IPv6 Segment Routing Header (SRH).* Tech. rep. IETF, Oct. 2019. URL: `https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-25`.

[29] Clarence Filsfils et al. "Segment Routing Architecture". In: *RFC 8402* (July 2018).

[30] R Flury, S V Pemmaraju, and R Wattenhofer. "Greedy Routing with Bounded Stretch". In: *Proc. IEEE INFOCOM.* Apr. 2009.

[31] B. Fortz, J. Rexford, and M. Thorup. “Traffic engineering with traditional IP routing protocols”. In: *IEEE Commun. Mag.* 40.10 (Oct. 2002), pp. 118–124.

[32] B. Fortz and M. Thorup. “Optimizing OSPF/IS-IS weights in a changing world”. In: *IEEE J. Sel. Areas. Commun.* 20.4 (May 2002), pp. 756–767.

[33] B Fortz and M Thorup. “Internet traffic engineering by optimizing OSPF weights”. In: *Proc. IEEE INFOCOM.* Mar. 2000.

[34] Bernard Fortz, Mikkel Thorup, and Mikkel Thorup. “Increasing Internet Capacity Using Local Search”. In: *Comput. Optim. Appl.* 29.1 ().

[35] Yirou Gang et al. “Throughput Maximization Routing in the Hybrid Segment Routing Network”. In: *Proc. Telecomun. and Commun. Eng.* Nov. 2018.

[36] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1990.

[37] S Gay, R Hartert, and S Vissicchio. “Expect the unexpected: Sub-second optimization for segment routing”. In: *Proc. IEEE INFOCOM.* May 2017.

[38] Steven Gay, Pierre Schaus, and Stefano Vissicchio. “REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms”. In: *CoRR* (Oct. 2017). arXiv: `1710.08665 [cs.NI]`. URL: `http://arxiv.org/abs/1710.08665`.

[39] "Gurobi". In: (2019). URL: `https://www.gurobi.com/`.

[40] Renaud Hartert et al. "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks". In: *ACM SIGCOMM Comput. Commun. Rev.* 45.5 (Oct. 2015), pp. 15–28.

[41] Chi-Yao Hong et al. "Achieving High Utilization with Software-driven WAN". In: *ACM SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 15–26.

[42] Christian E. Hopps. "Analysis of an Equal-Cost Multi-Path Algorithm". In: *RFC 2992* (Nov. 2000).

[43] K. Huang et al. "Bounded stretch geographic homotopic routing in sensor networks". In: *Proc. IEEE INFOCOM.* Apr. 2014.

[44] L. Huang et al. "Optimizing Segment Routing With the Maximum SLD Constraint Using Openflow". In: *IEEE Access* 6 (Apr. 2018), pp. 30874–30891.

[45] "IBM ILOG CPLEX Optimization Studio". In: (2014). URL: `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer`.

[46] Sushant Jain et al. "B4: Experience with a Globally-deployed Software Defined WAN". In: *ACM SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 3–14.

[47] N. Karmarkar. "A New Polynomial-time Algorithm for Linear Programming". In: *Proc. ACM STOC.* Dec. 1984.

[48] S. Knight et al. “The Internet Topology Zoo”. In: *IEEE J. Sel. Areas Commun.* 29.9 (Oct. 2011), pp. 1765–1775.

[49] M. Kodialam and T. V. Lakshman. “Minimum interference routing with applications to MPLS traffic engineering”. In: *Proc. IEEE INFOCOM*. Vol. 2. Mar. 2000, pp. 884–893.

[50] V. A. Le, P. Le Nguyen, and Y. Ji. “Deep Convolutional LSTM Network-based Traffic Matrix Prediction with Partial Information”. In: *IFIP/IEEE IM*. Apr. 2019, pp. 261–269.

[51] X. Li and K. L.Yeung. “ILP Formulation for Monitoring-Cycle Construction Using Segment Routing”. In: *IEEE LCN*. Oct. 2018, pp. 485–492.

[52] X. Li and K. L. Yeung. “Traffic Engineering in Segment Routing using MILP”. In: *Proc. IEEE ICC*. May 2019.

[53] A Makhorin. “GLPK (GNU Linear Programming Kit)”. In: (Oct. 2008). URL: `http://www.gnu.org/software/glpk/glpk.html`.

[54] A. Mayer et al. “An Efficient Linux Kernel Implementation of Service Function Chaining for Legacy VNFs Based on IPv6 Segment Routing”. In: *IEEE NetSoft*. June 2019, pp. 333–341.

[55] Nick McKeown et al. “OpenFlow: Enabling Innovation in Campus Networks”. In: *SIGCOMM Comput. Commun. Rev.* 38.2 (Mar. 2008), pp. 69–74.

[56] A. Medina et al. “Traffic Matrix Estimation: Existing Techniques and New Directions”. In: *ACM SIGCOMM Comput. Commun. Rev.* 32.4 (Aug. 2002), pp. 161–174.

[57] F. Paolucci. “Network service chaining using segment routing in multi-layer networks”. In: *IEEE/OSA J. Opt. Commun. Netw.* 10.6 (June 2018), pp. 582–592.

[58] K. Papagiannaki et al. “Long-term forecasting of Internet backbone traffic: observations and initial models”. In: *IEEE INFOCOM.* Vol. 2. Mar. 2003, pp. 1178–1188.

[59] Peter Psenak and Stefano Previdi. *OSPFv3 Extensions for Segment Routing.* Tech. rep. IETF, Jan. 2019. URL: `https://datatracker.ietf.org/doc/html/draft-ietf-ospf-ospfv3-segment-routing-extensions-23`.

[60] T. Schüller et al. “Traffic Engineering Using Segment Routing and Considering Requirements of a Carrier IP Network”. In: *IEEE/ACM Trans. Netw.* 26.4 (Aug. 2018), pp. 1851–1864.

[61] Timmy Schüller et al. “Traffic engineering using segment routing and considering requirements of a carrier IP network”. In: *Proc. IFIP Netw.* June 2017.

[62] T Schüller et al. “Predictive Traffic Engineering with 2-Segment Routing Considering Requirements of a Carrier IP Network”. In: *Proc. IEEE LCN.* Oct. 2017.

[63] S. Secci et al. “Resilient Traffic Engineering in a Transit-Edge Separated Internet Routing”. In: *2011 IEEE International Conference on Communications (ICC).* June 2011, pp. 1–6.

[64] T Settawatcharawanit et al. “Segment Routed Traffic Engineering with Bounded Stretch in Software-Defined Networks”. In: *Proc. IEEE LCN.* Oct. 2018.

[65] Rob Shakir et al. *Segment Routing with MPLS data plane.* Tech. rep. IETF, May 2019. URL: `https://tools.ietf.org/html/draft-ietf-spring-segment-routing-mpls-22`.

[66] Neil Spring, Ratul Mahajan, and David Wetherall. “Measuring ISP Topologies with Rocketfuel”. In: *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.* SIGCOMM ’02. New York, NY, USA: ACM, 2002, pp. 133–145.

[67] G. Swallow. “MPLS advantages for traffic engineering”. In: *IEEE Communications Magazine* 37.12 (Dec. 1999), pp. 54–57.

[68] G. Trimponias et al. “Node-Constrained Traffic Engineering: Theory and Applications”. In: *IEEE/ACM Trans. Netw.* 27.4 (Aug. 2019), pp. 1344–1358.

[69] George Trimponias et al. *On traffic engineering with segment routing in SDN based WANs.* Tech. rep. Mar. 2017. URL: `https://arxiv.org/abs/1703.05907`.

[70] Steve Uhlig et al. "Providing Public Intradomain Traffic Matrices to the Research Community". In: *ACM SIGCOMM Comput. Commun. Rev.* 36.1 (Jan. 2006), pp. 83–86.

[71] J. Wang et al. "Spatio temporal modeling and prediction in cellular networks: A big data enabled deep learning approach". In: *IEEE INFOCOM.* May 2017, pp. 1–9.

[72] N. Wang et al. "An overview of routing optimization for internet traffic engineering". In: *IEEE Commun. Surveys Tuts.* 10.1 (Apr. 2008), pp. 36–56.

[73] K. Xie et al. "Accurate recovery of Internet traffic data: A tensor completion approach". In: *IEEE INFOCOM.* Apr. 2016, pp. 1–9.

[74] Xipeng Xiao et al. "Traffic engineering with MPLS in the Internet". In: *IEEE Network* 14.2 (Mar. 2000), pp. 28–33.

[75] Junjie Zhang et al. "Dynamic hybrid routing: Achieve load balancing for changing traffic demands". In: *Proc. IEEE IWQoS.* May 2014.

[76] P. Zhang et al. "Bandwidth Allocation With Utility Maximization in the Hybrid Segment Routing Network". In: *IEEE Access* 7 (June 2019), pp. 85253–85261.

[77] G. Zhong, J. Yan, and L. Kuang. "Improving Integrated Terrestrial-Satellite Network Utilization using Near-Optimal Segment Routing". In: *2018 IEEE/CIC ICCC Wkshps.* Aug. 2018.

List of Publications

Journal Paper

T. Settawatcharawanit, Y. Chiang, V. Suppakitpaisarn and Y. Ji, "A Computation-Efficient Approach for Segment Routing Traffic Engineering," in IEEE Access, vol. 7, pp. 160408-160417, 2019.

International Conference

T. Settawatcharawanit, V. Suppakitpaisarn, S. Yamada and Y. Ji, "Segment Routed Traffic Engineering with Bounded Stretch in Software-Defined Networks," 2018 IEEE 43rd Conference on Local Computer Networks (LCN), Chicago, IL, USA, 2018, pp. 477-480.

Domestic conference

T. Settawatcharawanit and Y. Ji, "Segment Routed Traffic Engineering using Randomized Sampling with Bounded Stretch," BS-4-25, 2019 IEICE General Conference. [English Session Award]

T. Settawatcharawanit and Y. Ji, "Segment Routed Traffic Engineering using Randomized Sampling with Bounded Stretch," NS2019-118, IEICE Technical Report on Networks Systems.