COMPACT AND NON-COMPACT FORMULATIONS FOR SEGMENT ROUTING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

By
Osman Kağan Yayla
August 2025

Compact and Non-Compact Formulations for Segment Routing
By Osman Kağan Yayla
August 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Oya Karaşan(Advisor)

Ezhan Karaşan

Barış Yıldız

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

COMPACT AND NON-COMPACT FORMULATIONS FOR SEGMENT ROUTING

Osman Kağan Yayla
M.S. in Industrial Engineering
Advisor: Oya Karaşan
August 2025

Segment routing allows flexible control of packet routing using a segment list. A segment list consists of node-segments that utilize equal-cost multiple paths and adjacency-segments that directly use available links between source and destination nodes. This characteristic of segment routing makes it a good candidate for simplified and scalable traffic engineering. This thesis aims to minimize maximum link utilization (congestion) and processing energy costs simultaneously. In the thesis, a segment-based compact model and a path-based column generation model are developed. Initially, the models are utilized with maximum link utilization as the sole objective; both the compact model and the column generation model outperform existing methods in the literature, with the column generation approach excelling in its ability to solve large-scale and complex instances. Then, a multi-objective framework is tested for various parameter configurations that influence the energy consumption and congestion. The experiments over realistic network instances show potential for exploiting the energy-congestion trade-off for energy savings with minimal impact on congestion.

Keywords: Segment routing, Traffic engineering, Energy saving, Routing, Optimization. .

ÖZET

SEGMENT YÖNLENDIRME ILE ENERJI FARKINDALIKLI TRAFIK MÜHENDISLIĞI

Osman Kağan Yayla
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Oya Karaşan
Ağustos 2025

Segment yönlendirme, paket yönlendirmesini bir segment listesi aracılığıyla esnek bir şekilde kontrol etme olanağı sağlar. Segment listesi, eşit maliyetli çoklu yolları (ECMP) kullanan düğüm segmentleri ile kaynak ve hedef düğümler arasındaki mevcut bağlantıları doğrudan kullanan komşuluk segmentlerinden oluşur. Segment yönlendirmenin bu özelliği, onu basitleştirilmiş ve ölçeklenebilir trafik mühendisliği için uygun bir aday haline getirir. Bu tezde, maksimum bağlantı kullanımını (tıkanıklık) ve işlemeye bağlı enerji maliyetlerini eşzamanlı olarak en aza indirmek amaçlanmaktadır. Tez kapsamında, segment tabanlı bir kompakt model ve yol tabanlı bir kolon üretimi modeli geliştirilmiştir. Başlangıçta, modeller yalnızca maksimum bağlantı kullanımı amacıyla değerlendirilmiş; hem kompakt model hem de kolon üretimi modeli, literatürdeki mevcut yöntemlerden daha iyi performans göstermiştir. Özellikle kolon üretimi yaklaşımı, büyük ölçekli ve karmaşık örnekleri çözmedeki başarısıyla öne çıkmaktadır. Ardından, enerji tüketimi ve tıkanıklığı etkileyen çeşitli parametre yapılandırmaları altında çok amaçlı bir çerçeve test edilmiştir. Gerçekçi ağ örnekleri üzerinde yapılan deneyler, enerji-tıkanıklık dengesinden yararlanarak minimum tıkanıklık etkisiyle enerji tasarrufu sağlanabileceğini ortaya koymaktadır.

Anahtar sözcükler: Segment yönlendirme, Trafik mühendisliği, Enerji tasarrufu, Yönlendirme, Optimizasyon.

Acknowledgement

First and foremost, I would like to thank my advisor, Prof. Oya Karaşan. Without her unwavering support and insightful feedback, this thesis would not have materialized, and I would not have been able to develop the skills and foundation I possess today. I will be forever grateful to her.

I would also like to express my sincere thanks and gratitude to my jury members, Prof. Ezhan Karaşan and Assoc. Prof. Barış Yıldız. Their comments and suggestions were instrumental in shaping this thesis and deepening my understanding of my research.

I would like to acknowledge my dear friends who turned the third floor into a second home for me: Yunus Emre, Zeynep Göze, Mustafa, Doğuş Berk, and İrem. I am also grateful to Bora, Kaan, Sena, Efecan, and Muhittin for being a reliable and supportive cohort. Special thanks to Bilge and İbrahim for their invaluable companionship.

A special paragraph is reserved for Deniz Akkaya—my office mate, friend, and neighbor. You have made the last two and a half years both bearable and enjoyable.

I would like to thank my dear family. To my father, Hüseyin Yayla, for always pushing me forward; to my sister, Işıl Yayla, for her constant support; and to my French Bulldog, Angel, for being the perfect companion anyone could ask for. A special acknowledgement is owed to my mother, Hacer Yayla. Thank you for always believing in me, even when I couldn't.

This thesis is dedicated to my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rapid adaptation and expansion of internet services are resulting in unmatched demand, thus internet networks and network operators are expected to face unprecedented challenges. To alleviate the toll on the networks, incoming and outgoing traffic has to be managed effectively. As a result, traffic engineering, henceforth TE, arises as an answer to manage complex networks while managing multiple key performance metrics. In the Internet Engineering Task Force (IETF) Request For Comments (RFC) 9522, TE is defined as a subfield of network engineering concerned with evaluating and optimizing the performance of operational IP networks [1]. Under this definition, it is clear that TE needs to adapt and overcome increasing demand in challenging networks. Traditionally Multi-Protocol Label Switching (MPLS) and Software Defined Networks (SDN) have been used in traffic engineering.

As classified in [2], the literature focuses on three primary TE objectives: (i) the minimization of network energy consumption, (ii) the minimization of congestion, and (iii) the minimization of the number of rejected requests. This thesis focuses on the first two objectives: minimizing energy consumption and minimizing maximum link utilization (MLU). While the routing flexibility offered by SR enables congestion control through load balancing, it can simultaneously suffer from high energy consumption by spreading data over many links. Handling

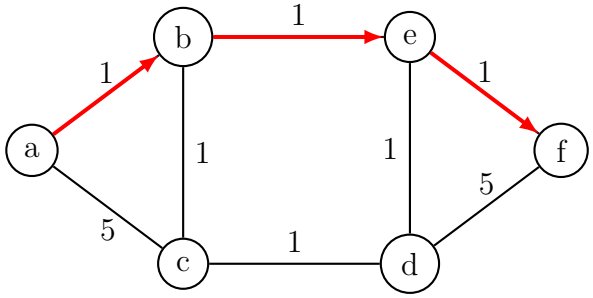


Figure 1.1: An SR-path with segment list $\{f\}$.

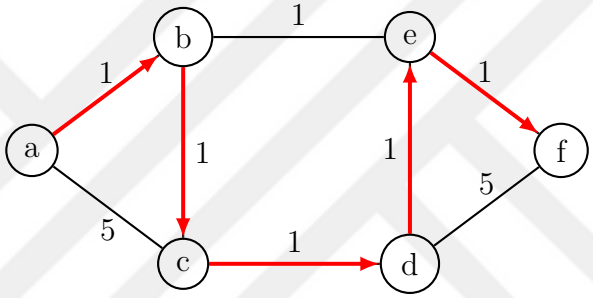


Figure 1.2: An SR-path with segment list $\{b, c, f\}$.

these two conflicting objectives is paramount in effective TE, and we take on this challenge by handling them within a bi-criteria optimization framework.

To satisfy the ever-increasing requirements of TE, Segment Routing, henceforth SR, is a relatively new network routing paradigm that allows the routing path to be decomposed into an ordered list of *segments* [3], specified by Segment-IDs (henceforth SID). In SR, a *node-segment*, Node SID, is the shortest path between the two end nodes given in the identifier. Refer to Figure 1.1 for an example; the SR path originating from node a is given the Node SID f. The shortest path from node a to f is a-b-f, that path is followed since there are no further instructions. This example corresponds to the shortest path routing.

However, now assume that the segment list $\{b, c, f\}$ is given; we obtain the path shown in Figure 1.2. First, the package is routed to b, then c, and finally f using the shortest paths.

If there exists more than one shortest path between the current node on the path and the destination node specified by the Node SID, then Equal-Cost Multi

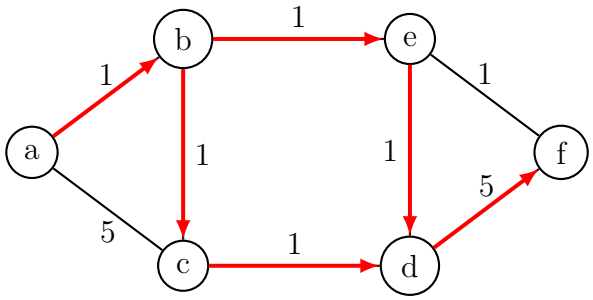


Figure 1.3: An SR-path with segment list $\{d, < d, f >\}$.

Paths (abbreviated as ECMPs) are utilized. Additionally, an *adjacency-segment* or a *link-segment* corresponds to routing through the supplied link. Now consider Figure 1.3 with the segment list $\{d, < d, f >\}$. The package follows the shortest path to d and utilizes the two equal cost shortest paths from b to d. At node b, package is split equally into paths: b-e-d and b-c-d. Then *adjacency-segment* $< d, f >$ is utilized instead of following the shortest path d-e-f.

Segment routing presents opportunities and challenges for modern traffic engineering, offering greater control over routing decisions while requiring scalable optimization frameworks to fully harness its potential. In this thesis, we address these challenges by proposing compact and column generation-based models that jointly optimize congestion and energy consumption.

1.1 Problem Statement

The models we will develop in this thesis aim to minimize two objectives considered in TE through a bi-objective framework. The first objective is minimizing the maximum link utilization (MLU, hereafter), directly impacting network performance. Consequently, the second objective is minimizing the energy consumption of the network nodes and links; we will abbreviate it as EC moving forward. While the energy consumed by internet networks carries monetary and environmental importance, prior work has not fully addressed the trade-offs between the objectives mentioned earlier. Our problem is finding a routing plan

for each demand in a given network and a set of parameters that can exploit the energy-congestion trade-off through a multi-objective framework. We will define this problem as the **Energy Aware Segment Routing Problem**, ESRP in Chapter 3. Moreover, time to achieve solutions is of the essence when it comes to traffic engineering; in practice, solutions under 5 or 10 minutes are desired [4]. Thus, while formulating our models, we prioritized reducing the number of constraints and variables. Our compact formulation is an enhancement to existing literature in this regard. Also, we present valid inequalities for special problem scenarios that reduce the solution space and running times significantly. On the other hand, our column generation model is very fast since it does not require enumerating all the paths and can generate paths on a need basis.

1.2 Contribution

The main contributions of this thesis to the literature on TE with SR are as follows:

We propose a compact, flow-based model that can solve modest-sized problem instances to optimality. Moreover, for hop limits $K = 1$ and $K = 2$, we exploit the SR architecture to introduce cuts that eliminate unusable segments and significantly improve solution times. Specifically, the $K = 2$ cuts are consequential for the literature since it is possible to reach near-optimality with two hops for minimizing link congestion [5], [6].

We introduce a path-based model that is solved through column generation. The novel formulation we develop enables us to handle the pricing problem step of the CG algorithm in polynomial time and solve realistic problem instances to optimality. The method in [7] also utilizes column generation; however, their method first solves the maximum throughput problem and then uses a heuristic to find solutions for minimizing MLU.

We expand our models with a multi-objective framework and investigate the

relationship between energy consumption and traffic congestion in SR networks. To our knowledge, ours is the only work that addresses both objectives in the same framework [2], [8], [9]. We discover solutions that obtain energy savings with marginal increases in network congestion.

1.3 Organization

This thesis is organized in the following manner:

Chapter 2 presents the literature review of traffic engineering techniques used to minimize maximum link utilization and energy consumption. More attention is placed on methods utilizing Mixed Integer Linear Programming (MILP), as the thesis builds upon and enhances present formulations. Chapter 3 is dedicated to establishing a flow/segment-based model using a bi-objective framework to minimize energy consumption and link congestion. Chapter 4 develops a path-based model that leverages column generation. Chapter 5 outlines the experimental setup for testing the performance of models presented in Chapters 3 and 4 in a setting where only the link utilization objective is considered. Chapter 6 extends the experimental setup to a bi-objective setting. In this chapter, the performance of the proposed models is evaluated under varying network parameters, including processing costs, capacity multipliers, and hop limits. Moreover, this chapter presents a detailed analysis of the energy-congestion trade-off. Chapter 7 offers a conclusion on the thesis.

Chapter 2

Related Work and Problem Setting

This chapter is divided into sections that represent our two objectives. Section 2.1 defines and discusses minimizing maximum link utilization, MLU, with SR using Linear Programs (LP) and Mixed Integer Linear Programs (MILP). Section 2.2 gives an overview of the work done on minimizing energy consumption with SR. Finally, in Section 2.3, we briefly review other methods such as heuristics and machine learning for solving TE challenges with SR.

2.1 Minimizing Maximum Link Utilization

One of SR's prominent application areas is Traffic Engineering (TE), which optimizes network resource utilization, performance, and stability through smart routing [1]. As classified in [2], the literature focuses on three primary TE objectives: (ii) the minimization of maximum link utilization [4–7, 10–17], (i) the minimization of network energy consumption [18–23], and (iii) the minimization of the number of rejected requests [7, 24–27].

With traffic splitting allowed, the flow is split into several routing paths, each composed of at most a maximum allowed number, say K, of node- and/or link-segments. Some studies in the literature allow the demand to be split into multiple paths [5, 10, 12, 17], while some others do not allow traffic splitting [7, 15, 16, 26]. Additionally, some works in the literature have fixed K, usually $K = 2$ [4–6, 11, 13]. Our formulation allows for any number of segments on a path and allows for the demand to be split into multiple paths. We assume that an ECMP load balancing algorithm, e.g., round-robin, is used to distribute the intra-segment traffic equally. The two most related studies to our approach are [7] and [10] since, unlike the previously mentioned studies, they allow non-shortest path link-segments to be used to utilize all of the flexibility provided by SR. The study in [10] further reveals that the routing might be sub-optimal without non-shortest path link-segments. We will quantify this effect with our experimentation. We shall focus on the first two objectives jointly. While the routing flexibility offered by SR enables congestion control through load balancing, it can simultaneously suffer from high energy consumption due to spreading data over many links. Handling these two conflicting objectives is paramount in green TE, and we take on this challenge by handling them within a bi-criteria optimization framework.

The utilization of a link is defined as the ratio of the flow on the link to the link's capacity. Our first objective is to minimize the maximum link utilization (MLU), i.e., the highest link utilization, in a given network topology and traffic demand, thus achieving lower congestion levels. For this problem, the literature offers both path-based and segment-based solutions. For a given segment limit K, path-based solutions explicitly enumerate and optimize over complete K-segment paths from source to destination. These models guarantee optimality only if the path enumeration is sufficiently large to include all paths required by the optimal solution. In contrast, segment-based solutions optimize flow fractions over individual segments without explicitly constructing full paths during optimization. While segment-based models are more scalable and compact, they typically yield approximate solutions, as some valid combinations may be overlooked.

The model in [5] is an influential paper on utilizing LPs to solve traffic engineering problems with SR. Their formulation, 2-LP, provides path-based solutions

for $K = 2$ but does not support link-segments. This model was later extended to support link-segments and $K = 3$ by [10], building on points raised in [11] and [6]. Another 3-segment routing extension is given in [12]. In [5], a pre-determined parameter elegantly captures how traffic is distributed over links within a 2-segment path under ECMP routing. We extend this idea to any node- or link-segment of arbitrary length, allowing us to efficiently compute the proportion of flow each link carries within a segment. They also develop and test additional models that can handle online demand arrivals and absence of a known traffic matrix [5], however, in this paper our attention is centered on offline segment routing where a traffic matrix is supplied. In [13], a similar formulation is presented with an extension to reduce solution times that uses a bounded stretch constraint to limit the length of the paths. A path is discarded if a 2-segment path using an intermediate node is longer than the shortest path. Moreover, in [11], the model from [5] is tested on real network topologies and demands. Extensions for minimizing the number of used segments and only allowing one path per demand are provided for easier network management [11], in [4], the extension for limiting demand splitting is used alongside heuristics for finding a set of segments/tunnels that will perform well against unknown traffic matrices. In [14], an MILP and a reinforcement learning framework is developed for a scenario where only some of the nodes in the network are SR capable.

Some notation is necessary before we discuss a select set of models in detail. Note that our notation deviates from the original notation established for the models in their respective papers. We aim to present an overview of the models with consistent notation and use the notation here for our models that we will develop in Chapters 3 and 4.

We denote the optical network with an undirected graph $G = (N, E)$. Let $A = \{(i, j) \cup (j, i) : \{i, j\} \in E\}$ be the set of directed arcs/links representing the directional flow of data in this optical network. The data transfer capacity of an edge $e \in E$ is shown by κ_e. A *node-segment* σ in G contains a set of edges $E_\sigma \subseteq E$, which consists of all edges lying on the union of the shortest paths from the origin o_σ to the destination d_σ. Let $R = \{(i, j) : i \in N \; j \in N \setminus \{i\}\}$ be the set of all node-segments.

We shall represent a *link-segment* by an ordered pair of its origin and destination, i.e., $\sigma = (o_\sigma, d_\sigma)$. For every arc $a \in A$, we define a *link-segment* δ_a. If a link-segment δ_a for $a = (i, j)$ is used to transfer data from i to j, the flow does not follow the shortest path(s) from i to j but instead is directly transferred from i to j on the link $\{i, j\}$ without splitting. We let $R^A = R \cap A$ denote the set of all link-segments.

A data transfer demand q is a triplet $\langle o_q, d_q, f_q \rangle$, where o_q denotes the origin, d_q denotes the destination, and f_q indicates the volume of the demand. We define Q as the set of demands considered in the problem. A path π, for a demand $q \in Q$, is a sequence of node-segments and link-segments that connects the origin o_q to the destination d_q. The total number of segments contained in a path π is denoted by K_π. A path π is called *feasible*, if $K_\pi \leq K$ for a given stopover limit $K \geq 1$. The set of all feasible paths for a demand q is denoted by P_q, and the set of feasible paths that use a link $e \in E$ is denoted by P_e.

For a link $e \in E$, and a path $\pi \in P_e$, the percentage of the flow on the path π that is carried over the link e is shown by g_π^e. We will discuss how the g_π^e values are obtained in more detail in Chapter 3; however, two examples are necessary here as [5] uses these pre-determined parameters. In Figure 3.1 we illustrate the formation of a path π from node a to node f. In the graph, all the edges have a unit length, except for the edges $\{a, c\}$ and $\{d, f\}$ with a uniform length of five units. The path π, which obeys the segment list $\{d, (d, f)\}$, is composed of a node-segment σ_{ad} that starts at node a and ends at d, and a link-segment δ_{df} defined for the arc (d, f). Note that, in the given graph there are two shortest paths from node a to d, namely the paths $a - b - c - d$ and $a - b - e - d$, hence the node-segment σ_{ad} contains the edges $\{a, b\}$,$\{b, c\}$,$\{c, d\}$,$\{b, e\}$ and $\{e, d\}$ with the respective percentage of flows we show in the figure. The link-segment δ_{df} does not correspond to a shortest path between nodes d and f.

Refer to Figure 2.2 for an example with more than two bifurcations. Figure 2.2 aims to illustrate the formation of a path with the node-segment σ_{19}. In the graph, all the edges have a unit length except for the edge $\{5, 8\}$ that has a length of 2 units. Values given with the red arcs are the g_σ^e values of the edges in this

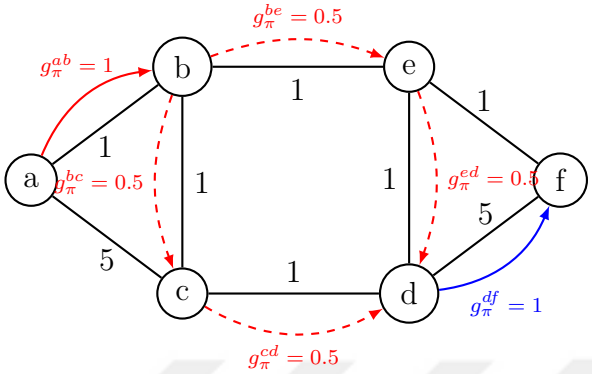


Figure 2.1: g_π^e values for path π with segment list $\{d, < d, f >\}$.

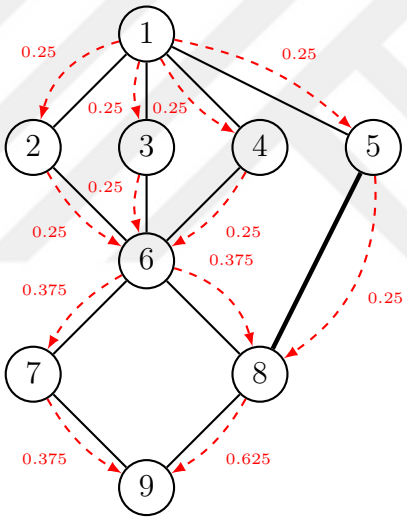


Figure 2.2: g_σ^e values displayed on the node segment σ_{19}.

segment.

In the following four subsections, we present and discuss the Multi-Commodity Flow (henceforth **MCF**) model, the **2-LP** model from [5], **K-MILP** and **K-sMILP** models from [10], and the **CG4SR** model that utilizes column generation from [7]. Note that the models in this section do not consider the energy objective. We give an overview of every index, set, and parameter defined so far in Table 2.1.

Table 2.1: Indices, sets, and parameters defined in Section 2.1.

Indices:	
e:	link
q:	demand
σ:	segment
π:	path
Sets:	
N:	set of nodes
E:	set of links
E_σ:	set of links contained in a segment σ
E_π:	set of links contained in a path π
A:	set of arcs, i.e., $A = \{(i,j) \cup (j,i) : \{i,j\} \in E\}$
Q:	set of demands
R:	set of segments (node- and link-segments)
R^A:	set of link-segments, i.e., $R \cap A$
P:	set of feasible paths
P_q:	set of the feasible paths of a demand $q \in Q$
P_e:	set of the feasible paths that contain the link $e \in E$
Parameters:	
κ_e:	capacity of an edge $e \in E$
o_q:	the origin of a demand $q \in Q$
d_q:	the destination of a demand $q \in Q$
o_σ:	the origin node of node-segment $\sigma \in R$
d_σ:	the destination node of node-segment $\sigma \in R$
f_q:	the volume of a demand $q \in Q$
k:	stopover limit
k_π:	the number of segments on path $\pi \in P$
α:	relative weight of the congestion objective

2.1.1 Multi-Commodity Flow

Multi-Commodity Flow, **MCF**, is a widely studied network flow problem with many applications in distribution, computer, communication and transportation networks [28]. Here we restrict our attention to its use in communication/internet networks as **MCF** allows traffic splitting and paths with no hop limit. These attributes of **MCF** guarantee the minimum MLU as the limiting factor for any SR model aiming to minimize MLU are the hop/segment limit K and restrictions on arbitrary traffic splitting. As a result, **MCF** serves as a good candidate to be a baseline model for minimizing MLU, [10], [5], [7] all use **MCF** as a benchmark model.

MCF

$$\min \ \Theta \tag{2.1}$$

$$\text{subject to} \tag{2.2}$$

$$\sum_{e_o=n} m_e^q - \sum_{e_d=n} m_e^q = \begin{cases} 1, \text{ if } o_q = n \\ -1, \text{ if } d_q = n \\ 0, \text{ otherwise} \end{cases}, \forall q \in Q \forall n \in N \tag{2.3}$$

$$\sum_q m_e^q f_q \leq \Theta \kappa_e, \forall e \in E \tag{2.4}$$

$$0 \leq m_e^q \leq 1, \forall q \in Q, \forall e \in E \tag{2.5}$$

Where m_e^q is a continuous variable representing the flow carried on arc e to satisfy demand from o_q to d_q. The objective, (2.1), is minimizing MLU. Constraints (2.3) assure that the flow entering and leaving that node is balanced. Constraints (2.4) enforce a bound on the flow carried on each edge e. This bound is determined by the capacity limit/bandwidth, κ_e, inherent to that edge. Then, Θ is multiplied by κ_e, which allows us to determine the MLU for edge e. For example, assume that there is no flow on edge e, then Θ is free to take any non-negative

value combined with the objective (2.1). On the other hand, if the flow on edge e is some fraction of κ_e, let us assume the flow on edge e is $\kappa_e \times 0.5$. Then a lower bound for Θ is established on edge e, as it can only be minimized/lowered to 0.5. Moreover, we can interpret this as half of the available link capacity is utilized. The constraints of type (2.4) are used in our models and in this chapter.

While not directly implementable in SR due to their fractional and unconstrained nature, **MCF** solutions represent the ideal performance limits regarding congestion since flow is split arbitrarily and there are no segment limits.

As such, they are instrumental in quantifying the performance gap introduced by SR-specific limitations, offering a reference point against which the efficiency and practicality of SR models can be assessed.

2.1.2 2-LP

2-LP is a model developed in [5], an influential paper on utilizing LPs to solve traffic engineering problems with SR. The original formulation can only handle 2-segment paths. However, the model has been enhanced to handle 3 and 4-segment paths [10].

2-LP

$$\min \ \Theta \tag{2.6}$$

subject to

$$\sum_{t \in N} x_t^q = f_q, \forall q \in Q \tag{2.7}$$

$$\sum_{q \in Q} \sum_{t \in N} g_t^q(e) x_t^q \leq \theta \kappa_e, \forall e \in E \tag{2.8}$$

$$x_t^q \geq 0, \forall q \in Q, \forall k \in N \tag{2.9}$$

The objective (2.6), Θ, is to minimize MLU. The variable x_t^q corresponds to

Table 2.2: Variables and parameters introduced for **MCF** and **2-LP**.

MCF	
m_e^q:	continous variable for flow carried on edge e for demand q
2-LP	
$g_t^q(e)$:	parameter for percentage of flow on edge e in a 2-segment path for demand q that uses the intermediary node t
x_t^q:	continous variable for flow carried on path $o_q - t - d_q$

the volume of flow carried on the path from t to d_q. Moreover, $g_t^q(e)$ denotes the fraction of flow carried on the edge e if one unit of flow originating from o_q visits the middle node t to reach the destination d_q. Refer to Figures 3.1 and 2.2 for examples on fractions of flow, however notice that their usage of this parameter differs from ours. In our notation, g_σ^e corresponds to the fraction of flow on edge e within the segment σ. This parameter is handled differently in **2-LP** as the model is only limited to 2 segments. Since the model is limited to 2 segments, t (as the middle node) combined with o_q and d_q identifies a 2-segment path. In our notation $g_t^q(e)$ is equivalent to g_π^e where the path $\pi = \{t, d_q\}$, meaning that π is a 2-segment path with the t and destination d_q. Satisfaction of all demands is guaranteed by Constraint (2.7). Constraint (2.8) allows us to find the MLU for each edge, and Constraint (2.9) prevents negative flows.

We will discuss how $g_t^q(e)$ variables are calculated and used in further detail in Chapter 3.

2.1.3 K-MILP and K-sMILP

In [10], two models are developed: **K-MILP** and **K-sMILP**. K-MILP is a path-based formulation that directly enumerates SR paths to satisfy traffic demands. However, the enumeration of all feasible paths incurs significant computational overhead. To mitigate this, [10] proposed K-sMILP, a segment-based alternative that avoids path enumeration by selecting segment sequences to construct SR paths.

2.1.3.1 K-MILP

To further discuss models from [10], we need some notation; specifically, we need to make more specific distinctions between the segments. We have previously established R and R^A, which correspond to all segments and link-segments. In [10], R is divided into 4 subsets:

1. the set of all unique shortest paths between every node pair, R^U;
2. the set of all ECMP routes between every node pair, R^{ECMP};
3. the set of all non-shortest-path links, R^{NE};
4. the set of all ECMP links, R^{EE}.

Segments in R^U can be explicitly identified through a single Node SID [10]. However, segments in R^{ECMP} cannot be identified by a single Node SID but require multiple Node SIDs. There could be many paths in an ECMP cluster; however, choosing a single path requires a longer segment list [10]. Segments in R^{NE} are links between an origin and destination; however, they do not constitute a shortest path. Their usage is still important, however, as they help in alleviating link congestion [10]. On the other hand, segments in R^{EE} refer to links that share the exact cost, origin, and destination with other paths in some ECMP.

In this setting our previously established sets correspond to $R = R^U \cup R^{ECMP} \cup R^{NE} \cup R^{EE}$ and $R^A = R^{NE} \cup R^{EE}$. However, [10] partitions them differently by letting $R^1 = R^{ECMP}$ and $R^2 = R^U \cup R^{NE} \cup R^{EE}$ This partition is required by their model since unlike [5] (which manage traffic splitting through the usage of pre-determined g_t^q variables) they manage traffic splitting within the MILP.

K-MILP

$$\min \ \Theta \tag{2.10}$$

subject to

$$\sum_{\sigma \in R | o_\sigma = n} x^q_{\sigma|_m} - \sum_{\sigma \in R | d_\sigma = n} x^q_{\sigma|_m}$$

$$= \begin{cases} 1 & o_q = n \\ -1 & d_q = n \\ 0 & \text{otherwise} \end{cases}, \quad \forall m \in M, \forall q \in Q, \forall n \in N \tag{2.11}$$

$$\sum_{m \in M} \left(\sum_{\sigma \in R | o_\sigma = n} z^q_{\sigma|_m} - \sum_{\sigma \in R | d_\sigma = n} z^q_{\sigma|_m} \right)$$

$$= \begin{cases} 1 & o_q = n \\ -1 & d_q = n \\ 0 & \text{otherwise} \end{cases}, \quad \forall q \in Q, \forall n \in N \tag{2.12}$$

$$z^q_{\sigma_1|_m} - z^q_{\sigma_2|_m} \leq \gamma \left(2 - x^q_{\sigma_1|_m} - x^q_{\sigma_2|_m} \right), \quad \forall q \in Q, \forall \sigma_1, \sigma_2 \in R, \forall m \in M \tag{2.13}$$

$$z^q_{\sigma_1|_m} - z^q_{\sigma_2|_m} \geq \gamma \left(x^q_{\sigma_2|_m} + x^q_{\sigma_1|_m} - 2 \right), \quad \forall q \in Q, \forall \sigma_1, \sigma_2 \in R, \forall m \in M \tag{2.14}$$

$$x^q_{\sigma|_m} \geq z^q_{\sigma|_m}, \quad \forall q \in Q, \forall \sigma \in R, \forall m \in M \tag{2.15}$$

$$z^q_{\sigma|_m} \geq x^q_{\sigma|_m} - 1, \quad \forall q \in Q, \forall \sigma \in R, \forall m \in M \tag{2.16}$$

$$\sum_{m \in M} \sum_{q \in Q} \sum_{\sigma \in P} e_\sigma z^q_{\sigma|_m} f_q \leq \Theta \kappa_e, \quad \forall e \in E \tag{2.17}$$

$$\sum_{m \in M} \sum_{\sigma \in R^1 | l_1 \in \sigma} z^q_{\sigma|_m} = \sum_{m \in M} \sum_{\sigma \in R^1 | l_2 \in \sigma} z^q_{\sigma|_m}, \quad \forall q \in Q \tag{2.18}$$

$$\sum_{\sigma \in R} x^q_{\sigma|_m} \leq K, \quad \forall m \in M, \forall q \in Q \tag{2.19}$$

$$0 \leq z^q_{\sigma|_m} \leq 1, \quad \forall q \in Q, \forall m \in M \tag{2.20}$$

$$x^q_{\sigma|_m} \text{ is a binary variable}, \quad \forall q \in Q, \forall \sigma \in R, \forall m \in M \tag{2.21}$$

In K-MILP, the objective,(2.10), is minimizing MLU. The set of constraints given in (2.11) ensures flow balance and conservation with the use of binary variable $x^q_{\sigma|_m}$. $x^q_{\sigma|_m}$ is a binary variable and it becomes 1 when segment $\sigma \in R$ is used by the m'th ($m \in M$) K-segment path for the demand $q \in Q$. Constraints (2.12) manage flow splitting and ensure that it is split accurately through the use

Table 2.3: Sets, parameters and variables introduced in **K-MILP** and **K-sMILP**.

Sets in [10]	
R^U	: set of all unique shortest paths between every node pair
$R^{ECMP} = R^1$	: set of all ECMP routes between every node pair
R^{NE}	: set of all non-shortest-path links
R^{EE}	: set of all ECMP links
R^2	: $R^U \cup R^{NE} \cup R^{EE}$
Parameters	
γ	: large enough integer constant
K-MILP Variables	
$x^q_{\sigma\|m}$	: binary variable tracking segment usage for demand q and path m
$z^q_{\sigma\|m}$	: fraction of flow on segment σ for demand q and path m
K-sMILP Variables	
y^q_σ	: fraction of f_q placed on σ
b^q_σ	: binary variable tracking if σ carries flow for demand q
v^q_σ	: integer variable tracking the voltage assigned to segment σ carries flow for demand q

of continuous $z^q_{\sigma|_m}$ variables, which denote the fraction of flow f_q on segment σ of the $m-$th $K-$segment path. Constraints (2.13) and (2.14) ensure that same flow is carried from segment σ_1 to segment σ_2. Constraints (2.15) and (2.16) ensure that if some flow is placed on a segment, that segment must be used, and if a segment is used, some flow must traverse it. Here γ is a large enough integer constant, e.g., $\gamma \geq 1000$. Constraint (2.17) ensures that the flow on link e is not exceeded, where e_σ is a binary variable indicating whether edge e is used in σ. Constraint (2.18) forces outgoing links from the same node to have the same amount of flow. Constraint (2.19) enforces the segment/hop limit of K onto each path. Constraints (2.20) and (2.21) are domain constraints. Table 2.1 presents the notation introduced in [10].

K-MILP is a path-based model, meaning that it iterates over SR paths to find solutions, denoted by the set M in the model. However, the enumeration process requires much time. Consequently, segment-based **K-sMILP** is developed in [10].

2.1.3.2 K-sMILP

K-sMILP is a segment-based model, meaning that the solutions it produces yield individual segments rather than complete paths [10]. Then, an algorithm is used to turn selected segments into feasible SR paths. They introduce a set of "voltage" constraints to ensure the constructed paths are feasible.

We can define **K-sMILP** with the notation and sets from **K-MILP** by only introducing new variables, given in Table 2.1 as follows:

K-sMILP

$$\min \ \Theta \tag{2.22}$$

subject to

$$\sum_{\sigma \in R | o_\sigma = n} y_\sigma^q - \sum_{\sigma \in R | d_\sigma = n} y_\sigma^q = \begin{cases} 1 & o_q = n \\ -1 & d_q = n \\ 0 & \text{otherwise} \end{cases}, \quad \forall q \in Q, \forall n \in N \tag{2.23}$$

$$\sum_{q \in Q} \sum_{\sigma \in R^A} y_\sigma^q f_q \leq \Theta \kappa_e, \quad \forall e \in E \tag{2.24}$$

$$\sum_{\sigma \in R^1 | l_1 \in \sigma} y_\sigma^q = \sum_{\sigma \in R^1 | l_2 \in \sigma} y_\sigma^q, \quad \forall q \in Q \tag{2.25}$$

$$b_\sigma^q \geq y_\sigma^q, \quad \sigma \in R, \forall q \in Q \tag{2.26}$$

$$v^q_{\sigma|_{o_\sigma = o_q}} = b^q_{\sigma|_{o_\sigma = o_q}}, \quad \forall \sigma \in R, \forall q \in Q \tag{2.27}$$

$$v^q_{\sigma_1|_{o_{\sigma_1} = n}} - v^q_{\sigma_2|_{d_{\sigma_2} = n}} \geq 1 - \gamma \left(2 - b^q_{\sigma_1} - b^{\sigma_2}_q\right), \quad \forall \sigma_1, \sigma_2 \in R \tag{2.28}$$

$$v^q_{\sigma|_{d_\sigma = d_q}} \leq K + \gamma \left(1 - b^q_{\sigma|_{d_\sigma = d_q}}\right), \quad \forall \sigma \in R, \forall q \in Q \tag{2.29}$$

$$0 \leq y_\sigma^q \leq 1, \quad \forall q \in Q, \forall \sigma \in R \tag{2.30}$$

$$b_\sigma^q \text{ is a binary variable}, \quad \forall q \in Q, \forall \sigma \in R \tag{2.31}$$

$$v_\sigma^q \geq 0 \text{ and integer}, \quad \forall q \in Q, \forall \sigma \in R \tag{2.32}$$

The objective of **K-sMILP** is to minimize MLU, given by (2.22). Constraint (2.23) ensures flow balance and conservation through the continuous variable y_σ^q, which denotes the fraction of f_q placed on σ. Constraint (2.24) ensures the link capacity κ_e is not exceeded. Notice that, compared to Constraint (2.17) from **K-MILP**, **K-sMILP** does not enumerate over all possible paths in the model. Constraint (2.25) guarantees equal traffic splitting over ECMP. Constraint (2.26) establishes the binary variable b_σ^q, which indicates if there is flow carried on segment σ for satisfying demand q. Constraint (2.26) ensures that this variable works accurately. Constraints (2.30) and (2.32) are domain constraints. Additionally, Constraint (2.30) denotes that the flow can be arbitrarily split among feasible paths.

Up to this point, the two models are similar; however, the selection of segments such that they will form feasible paths requires a new set of variables and constraints. The variable v_σ^q denotes the *voltage* assigned to segment σ used in satisfying demand q. In short, *voltage* variables help with where the segment is located in the segment list. For example, if segment σ is the first segment utilized then $v_\sigma^q = 1$, this is enforced in Constraint (2.27). Then, each segment in a potential path is counted in this manner and its voltage value is incremented based on the voltage value of the previous segment, which is enforced in Constraint (2.28). Finally, infeasible segments (in terms of segment limit K) are prevented by Constraint (2.29).

Figure 2.3 illustrates how voltage constraints operate. Each segment is colored differently, and the number to the right of the segment is the corresponding v_σ^q value. Note that the Constraint (2.29) is relaxed for illustration purposes; however, we assume $K = 3$. Consider Figure 2.3a, the model selects a total of five segments, and none violate the segment limit. Using Algorithm 1, two paths can be constructed using this solution: *a-b-e-f* and *a-c-d-f*. Algorithm 1 is designed to create paths from the selected segments; it ensures valid paths are supplied to the router and allows for splitting the demand into multiple paths.

Similarly, if we now consider Figure 2.3b, we observe the same thing, but now we can construct three paths: *a-b-e-g*, *a-e-g*, and *a-d-e-g*. In contrast, in

Figure 2.3c σ_7 has a voltage value of 4, which violates the segment limit; thus, this solution is invalid, and every selected segment is discarded. While we have prevented creating an infeasible path, several feasible paths have also been discarded during the process: *a-b-d-g*, *a-d-g*, *a-c-d-g*. As a result, **K-sMILP** cannot guarantee optimality as it discards potential paths. However, this deficiency has been shown to have no significant impact on solution quality [10]. Moreover, our **FLOW** based model suffers from the same issue as we leverage a similar mechanism to the voltage constraints to limit the number of segments on a path.

Algorithm 1 Path Construction using segments found by **K-sMILP**

Input: y_σ^q, G, q
$P_s \leftarrow$ set of segments with non-zero values of y_σ^q $(\sigma \in R)$
`List` $\leftarrow \{\}$; /* set of segment lists for carrying flow q */
`Ratio` $\leftarrow \{\}$; /* fraction of traffic carried by each segment list */
while P_s is not empty **do**
    $path_m \leftarrow$ a K-segment path from o_q to d_q, by connecting the segments from P_s with the largest value of y_σ^q at each node.
    $list_m \leftarrow$ segment list for $path_m$, one SID for one segment in $path_m$.
    $ratio_m \leftarrow$ the smallest y_σ^q among all segments σ in $path_m$.
    subtract $ratio_m$ from y_q^σ for every segment in $path_m$ and remove the segments with $y_\sigma^q = 0$ from P_s.
    `List` $\leftarrow$ `List` $\cup\, list_m$; `Ratio` $\leftarrow$ `Ratio` $\cup\, ratio_m$
end while
Return `List` and `Ratio` for flow q

In essence, while **K-MILP** provides a path-based formulation for SR, the exponential growth in the number of paths limits its scalability. On the other hand, **K-sMILP** addresses this challenge by a segment-based formulation, enabling more efficient solution times at the cost of reduced flexibility in path selection.

2.1.4 CG4SR

Most existing studies in the literature that solve traffic engineering optimization problems exactly utilize mathematical formulations involving decision variables with high orders of indices that inevitably fail to solve realistically sized problems, due to memory requirements. Although our flow-based model outperforms

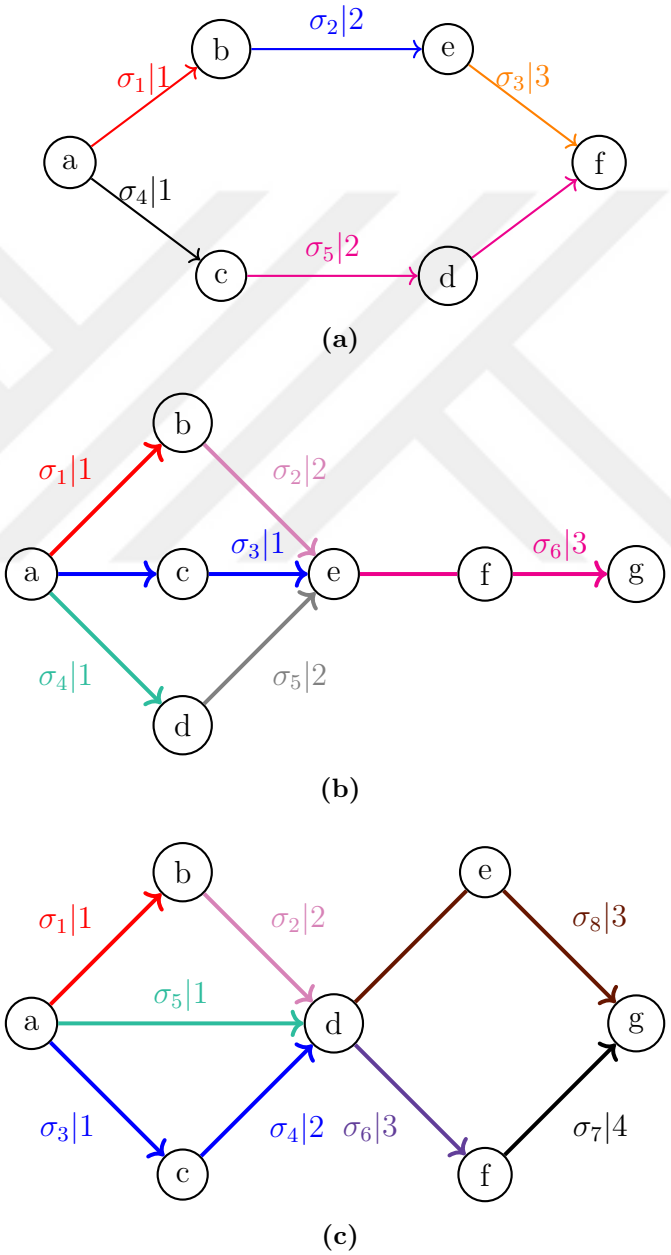


Figure 2.3: Example voltage assignments for 3 possible solutions.

existing compact models in the literature—as attested to by our computational results in Chapter 5.1—it also encounters scalability challenges on larger network topologies. However, with CG, variables likely to be present in desirable solutions are added as needed during the pricing stage; thus, solving large linear programs is possible with much less memory requirements than compact formulations. The first study to develop a CG-based approach to apply SR to TE problems is [7]. Similar to our work, they allow paths with up to K-segments and permit link segments. However, instead of directly minimizing MLU with a CG model, their framework [7] first maximizes the amount of demand routed without violating link capacities by a given threshold. Then the solution is post-processed by an algorithm to obtain a set of K-segment paths. However, they cannot guarantee optimality since their solution, solving a variant of the problem, might not include the necessary set of paths to achieve the minimum MLU.

The goal in [7] is to minimize the maximum link utilization; however, as mentioned, they start with a variant of the problem. This variant assings demands to SR paths, aiming to allocate as much demand as possible without exceeding a given threshold [7], this problem is called **SRTED** (where **SRTE** stands for Segment Routing Traffic Engineering problem). The Integer Linear Programming (ILP) formulation for **SRTED** is as follows:

SRTED-ILP

$$\max \sum_{\pi \in P} \sum_{q \in Q} f_q x_\pi^q \tag{2.33}$$

subject to

$$\sum_{\pi \in P} x_\pi^q \leq 1 \quad , \forall q \in Q \tag{2.34}$$

$$\sum_{\pi \in P} \sum_{q \in Q} g_\pi^q f_q x_\pi^q \leq \Theta \kappa_e \quad , \forall q \in Q \tag{2.35}$$

$$x_\pi^q \text{ are binary variables, } \forall \pi \in P, \forall q \in Q \tag{2.36}$$

The objective, (2.37), is to maximize the amount of demand routed. Here, x_π^q is a binary variable indicating whether π is used for satisfying demand q, also

given in domain Constraint (2.48). Constraint (2.46) ensures that only one path is selected for each demand. Constraint (2.47) assures that a threshold, Θ, binds the flow on each link. Here we introduce g_π^q, which corresponds to the ratio of flow on edge e if segment path π is used; we discuss this parameter in more detail in the following chapters. LP relaxation is applied to **SRTED-ILP** to obtain **SRTED-LP**, which we can take the dual of it and then use it in CG.

SRTED-LP

$$\max \sum_{\pi \in P} \sum_{q \in Q} f_q x_\pi^q \tag{2.37}$$

subject to

$$\sum_{\pi \in P} x_\pi^q \leq 1 \quad , \forall q \in Q \tag{2.38}$$

$$\sum_{\pi \in P} \sum_{q \in Q} g_\pi^q f_q x_\pi^q \leq \Theta \kappa_e, \forall e \in E \tag{2.39}$$

$$x_\pi^q \in [0,1], \forall \pi \in P, \forall q \in Q \tag{2.40}$$

Then the dual is taken as follows:

SRTE-DUAL

$$\min \ \lambda \sum_{e \in E} \kappa_e y_e + \sum_{q \in Q} z^q \tag{2.41}$$

subject to

$$z^q + \sum_{\pi \in P} \sum_{q \in Q} g_\pi^q f_q y_e \geq f_q, \forall \pi \in P, \forall q \in Q \tag{2.42}$$

$$y_e, z^q \geq 0, \forall \pi \in P, \forall q \in Q \tag{2.43}$$

Where y_e is the dual variable corresponding to Constraint (2.47) and z^q is the dual variable corresponding to Constraint (2.46). Now that we have the dual, through duality theory it can be shown that finding a path, π for demand q, with

the following condition means that **SRTED-LP** is not optimal yet:

$$z^q + \sum_{e \in E^\pi} g_\sigma^e f_q y_e < f_q \tag{2.44}$$

Thus, the pricing problem in [7], is to find path and demand pairs that (2.44) holds. They achieve this with a dynamic programming approach that finds the minimum cost SR paths and uses the dual variables y_e and z^q. We will use a similar approach in our column generation model by reducing the pricing problem to finding minimum cost SR paths. Notice that the above formulation finds a solution for **SRTED** problem and not minimizing MLU. Moreover, a binary search is established to find the threshold Θ such that each demand is routed. Finally, a set of paths, P^*, that maximize the amount of flow routed with one path per demand is found.

To find minimum MLU, **SRTED-ILP** is reconstructed with minimizing Θ as the objective function.

SRTED-UTIL-ILP

$$\min \ \Theta \tag{2.45}$$

subject to

$$\sum_{\pi \in P} x_\pi^q = 1 \quad , \forall q \in Q \tag{2.46}$$

$$\sum_{\pi \in P} \sum_{q \in Q} g_\pi^q f_q x_\pi^q \leq \Theta \kappa_e \quad , \forall q \in Q \tag{2.47}$$

$$x_\pi^q \text{ are binary variables, } \forall \pi \in P, \forall q \in Q \tag{2.48}$$

SRTED-UTIL-ILP cannot guarantee optimal solutions as the generated paths are optimal for a different problem [7]. Our formulation utilizes a similar framework and directly minimizes MLU.

Working on the problem instances from RocketFuel topologies, the authors show that their algorithm can provide performance guarantees for the MLU and achieve solutions within small optimality gaps (i.e., less than 6 percent). Unlike our study, [7] does not consider minimizing the network's energy usage along with

congestion. Solving the pricing problem efficiently is the key factor in the success of a CG approach. Providing a significant computational advantage in the CG phase, the formulation we develop in our study enables us to tackle the pricing problem in polynomial time and solve realistic problem instances to optimality.

2.2 Minimizing Network Energy Consumption

Our second objective is to minimize the energy costs resulting from transmission and processing.

We adopt the power model of [29], which quantifies the energy efficiency of the Internet infrastructure at the granularity of packet processing and byte storage operations. The energy consumption of routers and switches is considered the primary source of energy consumption, with two components: fixed (corresponding to idle or baseline power) and variable (corresponding to per-packet energy processing/transmission and per-byte storage energy).

Switching off routers and links is a fundamental way of conducting energy-aware TE as introduced in [30] and [31]. With SR, packets can be forced into specific nodes and links, thus making it a good choice for energy-aware traffic engineering. TE studies that utilize SR, such as [18], [20], [19], consider switching off routers and links in order to save energy. In our work, we are not interested in the fixed cost component of energy consumption since we do not propose switching the routers or links on and off. Instead, we focus on the transmission energy cost over the link, which depends on the link length and the processing cost incurred at the router. In [18], Carpa et al. develop the Segment Routing-based Energy Efficient Traffic Engineering framework, which puts links to sleep and reroutes the traffic on available links using SR. In [19], a similar framework is established with a different approach to determine which links to turn off. Otten et al. introduce the Green Segment Routing Problem in [20] for which they modify the 2-LP formulation from [5] to minimize the energy consumption of the network when the MLU objective is given as a predetermined parameter as a performance

threshold.

Another paper that utilizes SR for energy savings is [21]. They focus on incrementally upgrading a regular network with SR capabilities to control the sleep state of links and switches. In [23], SR is the routing protocol for a multi-objective virtual network embedding framework that aims to minimize congestion, energy usage, and cost saving. Tunneling-based algorithms to reduce the energy consumption of the SR network are introduced in [22]. This reduction is accomplished by overwriting the ECMP paths using pre-established tunnels. While this approach reduces energy consumption as only one of the ECMPs is used, in the meantime, this also reduces the fine control over the routing that SR introduces.

While studies considering TE with SR either solely optimize the congestion or the energy consumption, we exploit the energy-congestion trade-off through the multi-criteria optimization framework introduced in this paper.

2.3 Preprocessing and Other Non-LP Techniques

While LP-based methods have been popular for traffic engineering, several non-LP methodologies have been used to minimize MLU, mainly utilizing machine learning methods and heuristics. Moreover, several studies discuss the preprocessing of the network topology to improve running time and solution quality.

When the segment limit is 2, i.e., $K = 2$, SR paths only contain a single middle point connecting the origin and destination of each demand. Consider the SR path described in Figure 3.1, nodes a and d are connected via the midpoint d. This example has four potential middlepoints, but as topology sizes increase, potential candidates present a computational challenge. Several papers in the literature use these middle points to improve their models. The study in [6] proposes a middlepoint-based approach to improving link utilization and maximum

throughput solutions. Their framework evaluates nodes based on their centrality, which is measured using different metrics. It selects a limited number of them for each flow, thus reducing the number of potential SR paths and significantly reducing solution times. The authors in [24] develop a machine learning based recommendation framework, which utilizes historical solutions to reduce the number of candidates for intermediate nodes and groups particular demands before the optimization step, inspired by [5]. They turn off arbitrary traffic splitting by forcing the flow variables to be binary for most of their experiments. They run many instances from the SNDLib and DEFO libraries on synthetic and real traffic matrices to obtain a dataset of past solutions, from which they train a session-based recommendation framework. From the historical solutions they have obtained, they find that flows can be clustered and middlepoints (they assume $K = 2$) can be chosen only from a smaller subset of nodes while keeping solutions close to optimal in Section 3 we also use these characteristic of 2 segment paths to introduce valid inequalities that greatly help solution times, refer to Section 5.1.3 for our results.

The study presented in [32] offers an extensive survey of the preprocessing approaches to SR, and they propose a method that aggregates these approaches. They include node selection, demand pinning, and centrality/middlepoint-based methods. After their preprocessing is applied to the instances, they result in 10x speedups in the solution times using a model based on [5].

The Q-SR framework is presented in [25], a flexible and extensible approach for utilizing SR for the maximum throughput problem. They develop offline and online formulations using a fully polynomial-time approximation scheme. Their framework is compared with the CG4SR framework from [7] as both approaches are scalable and can give maximum throughput guarantees [25]. However, unlike [7], they do not give an extension for covering the minimum link utilization. In [26], SR is used for network throughput with an algorithm based on link criticality, similar to our CG model. The Bellman-Ford algorithm is used for its hop constraint to find SR paths. Study in [27] develops a primal-dual approach to handle link failures and consequent traffic restoration for the maximum throughput problem using SR.

In [33], the authors propose a hybrid constraint programming framework with a novel SR-path variable that can represent an SR-path in a single variable. The heuristic in [15] focuses on sudden demand changes through a Local Search algorithm. Another study that utilizes Local Search is [16], where they develop 3 ILP models with different SR capabilities and compare them with their heuristic. Authors in [17] focus on a scenario where only some of the nodes are SR-capable, which they solve with an MILP initially, and a discussion on a heuristic, which can find the best node to upgrade to be SR-capable, is presented.

Chapter 3

Compact Model Formulation

3.1 Model Setting

In this chapter we will formulate a compact model for solving the minimizing MLU and minimizing energy consumption simultaneously. Most of the necessary notation has been established in Table 2.1

As discussed in Section 2.1, g_σ^e corresponds to the ratio of flow carried on edge e of segment σ. This value can be calculated through different algorithms and methods however we have elected in this thesis to develop Algorithm 2. For

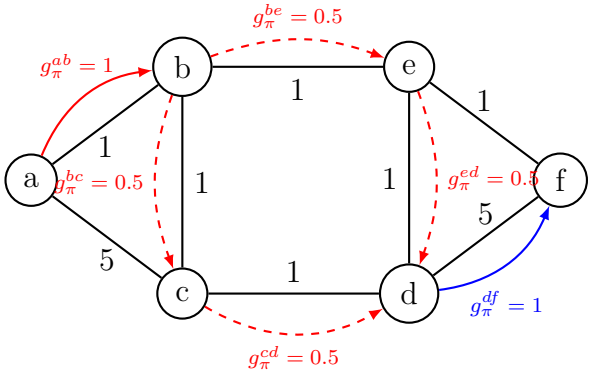


Figure 3.1: g_π^e values for path π with segment list $\{d, < d, f >\}$.

Table 3.1: Sets, Indices and Parameters for **FLOW**

Indices:	
e:	link
q:	demand
σ:	segment
π:	path
Sets:	
N:	set of nodes
E:	set of links
E_σ:	set of links contained in a segment σ
E_π:	set of links contained in a path π
A:	set of arcs, i.e., $A = \{(i,j) \cup (j,i) : \{i,j\} \in E\}$
Q:	set of demands
R:	set of segments (node- and link-segments)
R^A:	set of link-segments, i.e., $R \cap A$
P:	set of feasible paths
P_q:	set of the feasible paths of a demand $q \in Q$
P_e:	set of the feasible paths that contain the link $e \in E$
Parameters:	
κ_e:	capacity of an edge $e \in E$
o_q:	the origin of a demand $q \in Q$
d_q:	the destination of a demand $q \in Q$
o_σ:	the origin node of node-segment $\sigma \in R$
d_σ:	the destination node of node-segment $\sigma \in R$
f_q:	the volume of a demand $q \in Q$
k:	stopover limit
k_π:	the number of segments on path $\pi \in P$
α:	relative weight of the congestion objective

link-segments δ_e, g_σ^e is always 1.

Algorithm 2 Computation of g_σ^e

procedure COMPUTE$g(N, E)$ ▷ (N, E): directed graph of the network topology
  for each segment $\sigma \in R$ **do**
    Construct $G_\sigma = (N_\sigma, E_\sigma)$, the subgraph induced by all shortest paths of equal weights from o_σ to d_σ
    Compute a topological ordering T_σ of G_σ
    Initialize node inflows: $f(n) \leftarrow 0$ for all $n \in N_\sigma$; set $f(o) \leftarrow 1$
    Initialize $g_\sigma^e \leftarrow 0$ for all $e \in E_\sigma$
    for each node $n \in T_\sigma \setminus \{d_\sigma\}$ **do**
      Let $\mathrm{sn}(n)$ be the set of successor nodes of n in G_σ
      Let $\mathrm{oe}(n)$ be the set of outgoing edges out of n in G_σ
      Let $\mathrm{ex}(n) \leftarrow |\mathrm{oe}(n)|$ ▷ Number of outgoing edges
      for each node $u \in \mathrm{sn}(n)$ **do**
        $f(u) \leftarrow f(u) + \frac{f(n)}{\mathrm{ex}(n)}$ ▷ Calculate flow entering u
      end for
      for each edge $e = \mathrm{oe}(n)$ **do**
        $g_\sigma^e \leftarrow \frac{f(n)}{\mathrm{ex}(n)}$ ▷ Assign flow to outgoing edges of u
      end for
    end for
  end for
  return g_σ^e for each segment $\sigma \in R$
end procedure

A similar variable is employed in [5] only for 2-segment paths, however we utilize parameter g_σ^e for each $\sigma \in R$. Incorporating this parameter into our novel formulations enabled us to escape from enumerating ECMP paths in node-segments, considerably reducing the model sizes. In [10], instead of these pre-determined ratios, intra-segment ratios are calculated within the problem.

Moreover, we have to define g_π^e, which corresponds to the ratio of flow on edge e utilized in path π. g_π^e is nothing but the union g_σ^e values corresponding to each segment σ contained in path π. We will use g_σ^e in our non-compact formulation **PATH**. A similar parameter is also used in [7].

3.2 Objectives

The link utilization rate (congestion) θ_e on a link $e \in E$ is defined as the ratio between the amount of data transferred over the edge and the capacity κ_e. The congestion in the network Θ is defined as the maximum of the link congestion on edges, i.e., $\Theta = \max\{\theta_e : e \in E\}$.

Now we can introduce some notation for the energy usage of the network. The amount of energy needed to process a unit demand in a router is denoted by λ and the amount of energy used to transfer a unit demand over an edge $e \in E$ is denoted by λ_e. We define λ_π as the amount of energy required to send a unit demand with path π, i.e.,

$$\lambda_\pi = \sum_{e \in E_\pi} (g_\pi^e \lambda_e) + \lambda(K_\pi - 1), \tag{3.1}$$

where E_π is the set of links in path π.

The Pareto frontier is the set of solutions where you can't improve one objective without making another one worse [34]. In our multi-objective framework we are interested in finding out this set of solutions for energy-congestion trade-off. For a given network instance, i.e. G and Q, we let Θ^N and Λ^N be the worst possible values for the congestion and energy objectives, respectively. We will present a discussion on how these points are calculated in Chapter 6.

A routing plan Π indicates the fraction of a demand $q \in Q$ to be sent over each feasible path $\pi \in P_q$. The energy consumption of a routing plan is denoted by Λ_Π, and the network congestion incurred is shown as Θ_Π.

We handle Θ and Λ objectives through the weighted sum scalarization method [34] with the scalarization multiplier α. Varying the α parameter allows us to discover the Pareto frontier. We define $\alpha \in [0, 1]$ as the relative weight of the

congestion objective and let

$$Z_{\Pi}^{\alpha} = \alpha \frac{\Theta_{\Pi}}{\Theta^{N}} + (1-\alpha)\frac{\Lambda_{\Pi}}{\Lambda^{N}} \tag{3.2}$$

be the efficiency score of a routing plan Π. Through the normalization above we scale both components of the efficiency score to the same range of $[0, 1]$. Thus, $Z_{\Pi}^{\alpha} \in [0, 1]$ for any $\alpha \in [0, 1]$ and routing plan Π.

For an easy reference, the notation we use is outlined in Table 3.2. We formally define the ESRP as follows:

Definition 1 *For a given network G, demand set Q, hop limit K and the congestion weight α, the Energy Aware Segment Routing Problem (ESRP) is to find a routing plan Π^* with the best (smallest) efficiency score Z_{Π^*}.*

3.3 The FLOW Model

Now we can define our model **FLOW**:

For each demand $q \in Q$, we will send 1 unit of flow from the source node o_q to the destination node d_q over feasible paths that are sequences of segments. The y variables keep track of the flow over node-segments and the z variables keep track of the flow over link-segments. Both y and z variables are non-negative continuous variables. To ensure that the flows are transferred along feasible paths obeying the hop limit, we use binary u variables to indicate whether a specific segment (node or link) is used or not and h variables to count the number of hops from source nodes to intermediate nodes along feasible paths. Non-negative continuous variables Θ and Λ keep track of the maximum link usage and the total energy consumption, respectively.

FLOW

Table 3.2: Parameter and Variable Notation

Parameters:	
κ_e:	capacity of an edge $e \in E$
λ_e:	energy needed to transmit a unit data on an edge $e \in E$
λ:	energy needed to process a unit data on a router
g_σ^e:	percentage of flow on segment σ that uses the link $e \in E$
g_π^e:	the percentage of flow of a path π that uses the link $e \in E$
o_q:	the origin of a demand $q \in Q$
d_q:	the destination of a demand $q \in Q$
o_σ:	the origin node of node-segment $\sigma \in R$
d_σ:	the destination node of node-segment $\sigma \in R$
f_q:	the volume of a demand $q \in Q$
k:	stopover limit
k_π:	the number of segments on path $\pi \in P$
Θ_K:	the minimum possible network congestion with stopover limit k
Λ_K:	the minimum possible energy consumption with stopover limit k
α:	relative weight of the congestion objective
Variables:	
Θ:	the maximum link utilization
Λ:	the total energy consumption
Θ^I:	minimum Θ for a given parameter configuration
Λ^I:	minimum Λ for a given parameter configuration
Θ^N:	maximum Θ for a given network instance
Λ^N:	maximum Λ for a given network instance
Λ:	the total energy consumption
y_σ^q:	fraction of demand q over node-segment σ
z_σ^q:	link segment σ for demand q over
u_σ^q:	binary variable indicating the usage of segment σ for demand q
h_n^q:	number of hops on the path from o_q to node n for demand q
x_π^q:	the fraction of a demand q that is sent over a path $\pi \in P_q$

$$\min \quad \alpha\frac{\Theta}{\Theta^N} + (1-\alpha)\frac{\Lambda}{\Lambda^N} \tag{3.3}$$

$$\text{s.t.} \quad \sum_{\sigma\in R:\sigma_o=n} y_\sigma^q - \sum_{\sigma\in R:\sigma_d=n} y_\sigma^q + \sum_{\sigma\in R^A:\sigma_o=n} z_\sigma^q -$$

$$\sum_{\sigma\in R^A:\sigma_d=n} z_\sigma^q = \begin{cases} 1 & \text{if } n = o_q \\ -1 & \text{if } n = d_q \\ 0 & \text{otherwise} \end{cases} \quad q\in Q, n\in N \tag{3.4}$$

$$y_\sigma^q \le u_\sigma^q \qquad \sigma\in R, q\in Q \tag{3.5}$$

$$y_\sigma^q + z_\sigma^q \le u_\sigma^q \qquad \sigma\in R^A, q\in Q \tag{3.6}$$

$$\sum_{q\in Q} f_q \left(\sum_{\sigma\in R}(g_\sigma^e y_\sigma^q) + z_e^q\right) \le \Theta\kappa_e \qquad e\in E \tag{3.7}$$

$$h_{o_\sigma}^q + 1 \le h_{d_\sigma}^q + (K+1)\left(1-u_\sigma^q\right) \; q\in Q, \sigma\in R \tag{3.8}$$

$$h_n^q \le K \qquad q\in Q, n\in N \tag{3.9}$$

$$\sum_{q\in Q} f_q \sum_{\sigma\in R}\sum_{e\in E_\sigma} \lambda_e \left(g_\sigma^e y_\sigma^q + z_\sigma^q\right) = en_t \tag{3.10}$$

$$\sum_{q\in Q} f_q \sum_{\sigma\in R, d_q\neq d_\sigma} \lambda \left(y_\sigma^q + z_\sigma^q\right) = en_p \tag{3.11}$$

$$en_t + en_p \le \Lambda \tag{3.12}$$

$$u_\sigma^q \in \{0,1\} \qquad \sigma\in R, q\in Q \tag{3.13}$$

$$y_\sigma^q \ge 0 \qquad \sigma\in R, q\in Q \tag{3.14}$$

$$z_\sigma^q \ge 0 \qquad \sigma\in R^A, q\in Q \tag{3.15}$$

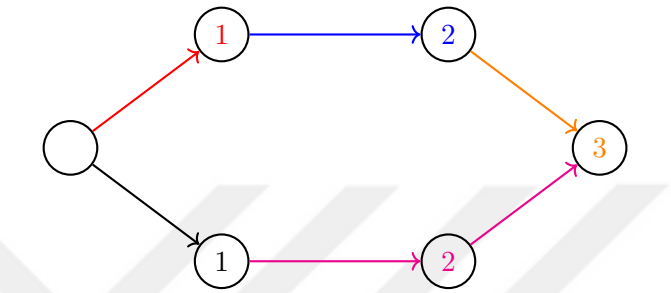
$$h_n^q \ge 0 \text{ and integer} \qquad n\in N, q\in Q \tag{3.16}$$

In the mathematical formulation, our objective (3.3) minimizes the efficiency score corresponding to the routing plan generated by the model. Constraints (3.4) are the flow balance constraints. For each demand one unit of flow is going to initiate at its source and terminate at its destination node. This unit of flow will disperse and traverse through node-segments and link-segments while ensuring flow balance. Constraints (3.5) ensure that if any node-segment is used (even partially), its corresponding indicator variable will take value one. Similarly, constraints (3.6) do the same for link-segments. Note that there is no difference between a link-segment and a node-segment if the only shortest path between

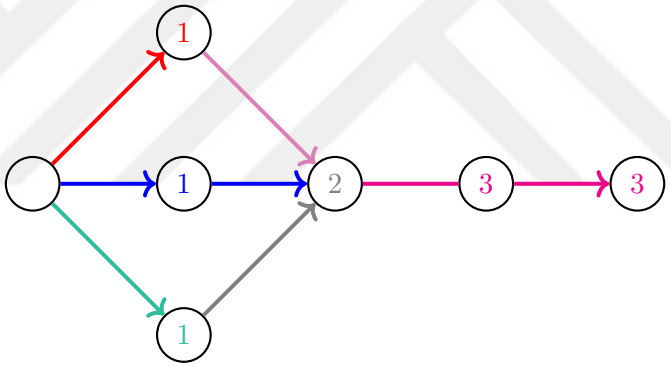
σ_o and σ_d is the link (σ_o, σ_d). Moreover, the endpoints of an arc in the network may be identified by node-segments, link-segments, or both. Constraints (3.7) are used to determine the network congestion Θ. For each edge $e = \{i, j\}$, and for each demand q, the values in parentheses count the fraction of the unit flow over all node- and link-segments using this link, and the sum over all demands find the total data transferred over this edge. Since the constraint is written for each edge, Θ will correspond to the congestion value.

If a particular segment (o_σ, d_σ) is used for the connection of a demand, constraints (3.8) force the number of hops from the demand source node to node o_σ be augmented by one for node d_σ. Otherwise, these constraints are simply redundant. Constraints (3.9) ensure that the number of hops to any node on a feasible path for a demand obeys the hop limit. Constraints (3.8) and (3.9) are similar to voltage constraints introduced in [10]. Our advantage is that we attach the hop count to nodes rather than their segments (ECMP paths). Refer Figure 3.2 for an illustration how they work. The number inside of each node corresponds to the h_n^q value. In Figure 3.2a and 3.2b, all segments are valid and we can form paths assuming a hop limit of $K = 3$. On the other hand, since in Figure 3.2c, the final node has a value of 4 all possible solutions are discarded. A detailed discussion on this was given in Section 2.1.3.2.

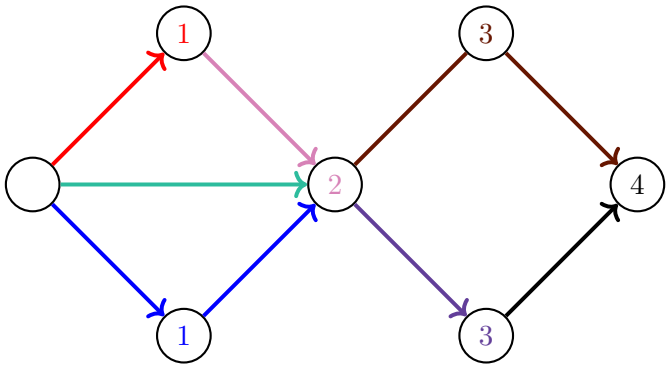
The energy consumption of the routing plan of this model is calculated via constraints (3.10)-(3.12), where the total energy needed for transmission and processing are calculated in (3.10) and (3.11), respectively. The amount of flow transferred on any link for any demand is weighted by the link's unit energy requirement and summed over all demands; auxiliary variable en_t corresponds to this transmission cost. Similarly, for each hop utilized in the collection of feasible paths, the energy required to process a unit demand in a router is weighted by the demand volume; auxiliary variable en_p corresponds to this processing cost. With constraints (3.12), these two energy requirements are summed to bound Λ, the energy requirement of the model's resulting routing plan. Finally, constraints (3.13)-(3.16) are domain restrictions.



(a)



(b)



(c)

Figure 3.2: Example hop limit assignments for 3 possible solutions.

3.3.1 Valid Inequalities for $K = 1$ and $K = 2$

In the special cases when the hop limit is at most 2, we can strengthen our formulation by introducing valid inequalities.

Proposition 1 *The following are valid inequalities for the feasible region given by* (3.4)-(3.16).

For $K = 1$*:*

$$u_\sigma^q \leq 0 \qquad \forall \sigma \in R \text{ and } q \in Q : \sigma_o \neq o_q \tag{3.17}$$

$$u_\sigma^q \leq 0 \qquad \forall \sigma \in R \text{ and } q \in Q : \sigma_d \neq d_q \tag{3.18}$$

For $K = 2$*:*

$$u_\sigma^q \leq 0 \quad \forall \sigma \in R \text{ and } q \in Q : \sigma_d \neq d_q \text{ and } \sigma_o \neq o_q \tag{3.19}$$

Proof 1 *When* $K = 1$*, only segments from the origin to the destination of each demand are permitted, and no intermediate segments can be used. Constraints* (3.17) *and* (3.18) *enforce this restriction for any demand by eliminating any segment that does not start at the demand origin or end at the demand destination, respectively. Similarly, when* $K = 2$*, segments can only be used if they share either the origin or the destination of the demand. Constraint* (3.19) *enforces this by eliminating segments whose endpoints are both intermediate nodes. This also means that Constraint* (3.19) *does not prevent the formation of single segment paths.*

We shall show in Section 5.1 that these cuts have been very effective.

In conclusion, our formulation introduces three key enhancements:

1. Instead of enforcing valid traffic splitting through additional constraints, we pre-process the network to compute flow distribution ratios g_σ^e, for each edge

e used by segment σ, thereby reducing the total number of variables and constraints required. This approach has been applied in [5] to 2-segment paths and generalized to K-segment paths; however, since all intermediate potential nodes are explicitly enumerated, it is not viable for large K. In contrast, our formulation handles this implicitly through a novel modeling approach.

2. To prevent the inclusion of segments that violate the hop limit K, we introduce the variable h_n^q, which explicitly restricts segment selection by counting the number of segments used to reach node n from the origin of demand q. A similar set of constraints was used in the segment-based MILP of [10], albeit with a larger number of variables and constraints. While our formulation scales better, it suffers from the same issue of potentially discarding feasible paths and being sub-optimal. However, as confirmed through our experiments in Section 5.1, we reach the same conclusion as [10] that these lost paths do not appear to have a significant impact on solution quality in practice.

3. For scenarios with $K \leq 2$, we introduce valid inequalities that add negligible overhead to the model build time but significantly reduce solution times, as demonstrated in Section 3.19. Moreover, these constraints are applicable to any model that uses binary variables to indicate whether a segment carries flow.

Chapter 4

Path based formulation

4.1 Path-Based Formulation

When constructing paths from segments, hop counts often overestimate and exclude feasible combinations, a fundamental limitation of all segment-based approaches. Path-based models remedy this problem, however, enumerating all potential SR paths within a compact model, as proposed by [10] incurs significant computational overhead. In contrast, our proposed path-based method, **PATH**, is highly scalable, guarantees optimality for ESRP, and outperforms non-optimal segment-based methods in both solution quality and efficiency since paths are only generated on an as-needed basis within a CG approach.

PATH

$$\min \quad \alpha\frac{\Theta}{\Theta^N} + (1-\alpha)\frac{\Lambda}{\Lambda^N} \tag{4.1}$$

$$s.t. \sum_{\pi \in P_q} x_\pi^q \geq 1 \qquad \forall q \in Q \tag{4.2}$$

$$\Theta \geq \sum_{q \in Q} \sum_{\pi \in P_e} \frac{f^q g_\pi^e x_\pi^q}{\kappa_e} \qquad \forall e \in E \tag{4.3}$$

$$\Lambda \geq \sum_{q \in Q} \sum_{\pi \in P_q} f^q \lambda_\pi x_\pi^q \tag{4.4}$$

$$x_\pi^q \geq 0 \qquad \forall q \in Q, \pi \in P_q \tag{4.5}$$

The objective is again to obtain the minimum efficiency score. Constraints (4.2) ensure that all data transfer demands are satisfied as at least a path is enforced for each demand. Note that, due to the structure of the objective function, writing (4.2) as an equality would not change the optimal solution. We preferred the current form to simplify the dual in the pricing problem, using nonnegative rather than free variables. Constraints (4.3) set the network congestion to the highest link utilization in the network. g_π^e values are calculated in the same manner with g_σ^e values, however in this case we calculate them over the entire path(which are sequences of segments) instead of just segments. In constraint (4.4) the total energy use of the network, represented by the auxiliary variable Λ, is calculated by summing up the energy consumed to transfer each demand (for the same reasons mentioned for (4.2), we prefer to write (4.4) as an inequality). Finally, variable domains are indicated by (4.5), note that we omit the upper bound of 1 on x_π^q's as our goal is to minimize the objective, this is not necessary.

As a LP model, the main difficulty to solve **PATH** for realistic size problem instances is to consider all feasible paths in the network, which grows exponentially with the number of edges and the hop limit K. In the next subsection, we introduce our CG algorithm (CG) to address this computational difficulty and to be able to solve large real-world ESRP instances.

4.2 Column Generation Algorithm (CG)

Column generation is very useful transforming/decomposing edge flow problems to path based problems [35]. One disadvantage of this approach in usual application areas is that the flow/commodity carried by the paths are integer which necessitates more complicated solution approaches. However, as we allow traffic splitting, our formulation is free of this burden however an extension for disabling

this feature for easier network operation and service management can be easily designed. Note that, despite the fact that the number of feasible paths in an ESRP instance can be quite large, only a very small fraction of them would be utilized in the optimal solution. Aiming to take advantage of this structure, we propose a column generation approach to solve **PATH** by initially considering a restricted set of feasible paths and iteratively adding new ones on a need basis.

At each CG iteration ξ, we consider a restricted problem $\mathbf{PATH}_\xi$ that contains a subset of the feasible paths, say P_ξ. We solve the *pricing problem* to detect feasible paths that are not included in P_ξ but have negative reduced costs, or conclude no such paths exist and terminate the procedure. We propose to solve the pricing problem by solving a series of hop-constrained shortest path problems, one for each demand $q \in Q$ over a pricing graph G^q.

Let ρ, ϱ and υ be the dual variables associated with the constraints (4.2), (4.3) and (4.4), for the restricted problem PATH_ξ.

We can write the dual problem $\mathbf{PATH\text{-}DUAL}_\xi$ as follows.

$\mathbf{PATH\text{-}DUAL}_\xi$

$$\max \quad \sum_{q \in Q} \rho_q \tag{4.6}$$

$$s.t. \rho_q - \sum_{e \in E_\pi} \frac{\varrho_e f^q g^e_\pi}{\kappa_e} - \upsilon f^q \lambda_\pi \leq 0 \qquad \forall q \in Q, \pi \in P_q \cap P_\xi \tag{4.7}$$

$$\sum_{e \in E} \varrho_e = \frac{\alpha}{\Theta^N} \tag{4.8}$$

$$\upsilon = \frac{1-\alpha}{\Lambda^N} \tag{4.9}$$

$$\rho_q \geq 0 \qquad \forall q \in Q \tag{4.10}$$

$$\varrho_e \geq 0 \qquad \forall e \in E \tag{4.11}$$

$$\upsilon \geq 0 \tag{4.12}$$

Let $(\rho^*, \varrho^*, \upsilon^*)$ be the optimal solution of $\mathbf{PATH\text{-}DUAL}_\xi$. Then, using (3.1) the reduced cost c^q_π of a path variable $\pi \in P_q$ for some $q \in Q$ can be calculated as follows.

$$c_\pi^q = \upsilon^* f^q \lambda (K_\pi - 1) + \sum_{e \in E_\pi} f^q g_\pi^e \left(\frac{\varrho_e^*}{\kappa_e} + \lambda_e \upsilon^* \right) - \rho_q^* \tag{4.13}$$

For a demand $q \in Q$, we define the *pricing graph* $G^q = (N, \bar{A})$ as a complete graph in which each arc represents either a node-segment or a link-segment in the original optical network. Let $c_\sigma = \sum_{e \in E_\sigma} g_\sigma^e (\varrho_e / \kappa_e + \lambda_e \upsilon)$ be the reduced cost of a segment σ and $c_{\delta_{ij}} = \varrho_{ij} / \kappa_{ij} + \lambda_{ij} \upsilon$ be the reduced cost of a direct connection $\delta_{ij}, (i,j) \in A$. For each arc $(i,j) \in \bar{A}$, we denote the segment from the node i to the node j with σ_{ij} and define the arc weights $c_{ij}^q, (i,j) \in \bar{A}$ in the pricing graph G^q as follows:

$$c_{ij}^q = \begin{cases} c_{\sigma_{ij}} + \upsilon\lambda, & \text{if } j \neq d_q \text{ and } (i,j) \notin A, \\ \min\{c_{\sigma_{ij}}, c_{\delta_{ij}}\} + \upsilon\lambda, & \text{if } j \neq d_q \text{ and } (i,j) \in A, \\ c_{\sigma_{ij}}, & \text{if } j = d_q \text{ and } (i,j) \notin A, \\ \min\{c_{\sigma_{ij}}, c_{\delta_{ij}}\}, & \text{otherwise.} \end{cases} \tag{4.14}$$

Note that, for a demand, $q \in Q$, the arc weights in the pricing graph G^q are composed of the shadow costs for the capacity and energy use for the respective node-segment or link-segment. Observe that those arcs that end in the destination (d_q) do not include the shadow cost for energy usage in the routers $(\upsilon\lambda)$. Also, for the direct connection links $(i,j) \in A$, the respective arc costs are determined by taking the minimum of the reduced costs for the link-segment and path-segment alternatives to sending data from the node i to node j for demand q. With these arc costs one can see that the pricing problem can be solved by solving hop-constrained shortest paths in the pricing graphs $G^q, q \in Q$. We formally state this observation with the following proposition.

Proposition 2 *Considering the solution of the restricted problem in a CG iteration ξ, there exists a negative reduced cost path variable that is not included in the model if and only if there exists a path with at most K number of arcs from o_q to d_q in the pricing graph G^q, for some $q \in Q$, with a weight less than ρ_q / f_q.*

Proof 2 *The result simply follows from the fact that for a demand $q \in Q$, the length of a path $\pi \in P_q$ in the pricing graph G^q is equal to $\upsilon\lambda(k_\pi - 1) + \sum_{e \in E_\pi} g_\pi^e \left(\frac{\varrho_e}{\kappa_e} + \lambda_e \upsilon \right)$.*

The hop-constrained shortest path problem is a special case of the resource constrained shortest path problem. Providing a significant computation advantage for the CG algorithm we propose, one can solve hop-constrained shortest path problems in polynomial time by taking advantage of the well-known shortest path finding algorithms such as the Bellman Ford algorithm [28] that can work with graphs having negative cost arcs and takes the hop constraints into account. Taking advantage of this property, we solve the pricing problem in each CG iteration by solving the K-hop shortest path problem for each demand $q \in Q$, on the pricing graph G^q and add all the negative reduced cost paths we detect or terminate the CG procedure if no such paths are detected.

In the following two chapters we are going to provide a sequence of experiments to highlight our contribution. In Chapter 5 we experiment solely with the Θ objective and in Chapter 6 we are interested in the energy-congestion trade-off.

Chapter 5

Minimizing Maximum Link Utilization

In this section, we shall strict our attention to the single objective of minimizing maximum link utilization and compare our solution approaches (our compact model FLOW as well as the CG algorithm based on the non-compact formulation PATH) against the state of the art compact formulation of [10]. Additionally, we test our valid inequalities given in (3.17)-(3.18) and (3.19). Moreover, we quantify the importance of modeling non-shortest path segments. Finally, to highlight the abilities of our PATH model we also solve instances from the Rocketfuel topologies obtained from [36] repository.

Our CG algorithm is implemented in Java. The compact model and the instance generation are implemented in Python. Both models use the Gurobi 12 solver. All experiments are conducted on a Linux machine with an AMD Ryzen 5800H CPU at 4.4 GHz and with 32 GBs of RAM. All reported times in tables and figures refer to clock time, encompassing both model construction and solution phases unless otherwise specified.

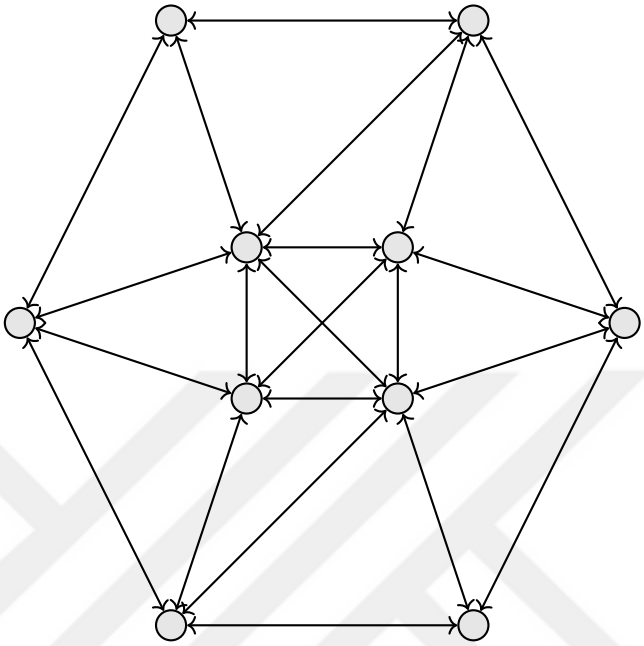


Figure 5.1: Network Visualization for the smallnet topology.

5.1 Comparison of Models

In order to make the comparison with the literature fair, as solution times are crucial for the SR problem, in our models we take $\alpha = 1$, $\Theta_N = 1$ and exclude the energy related constraints and variables. In summary, we only want to find out minimum Θ disregarding Λ.

5.1.1 Instance Generation & Experimental Design

We use five network topologies for our single experiment objectives, : four from SNDLib [37]—*Abilene*, *Nobel-Us*, *Nobel-Germany*, and *Geant*—and one from [38], *Smallnet*. The topologies from [37] are real network topologies and *Smallnet* is an artificial topology, we give the number of nodes and links (directed). Illustrations of the network topology are given in Figures 5.1, 5.2, 5.3, 5.4 and 5.5.

We set random link lengths and demands independently sampled from a uniform distribution on $[1, 10]$, we create 10 random instances in this manner. For

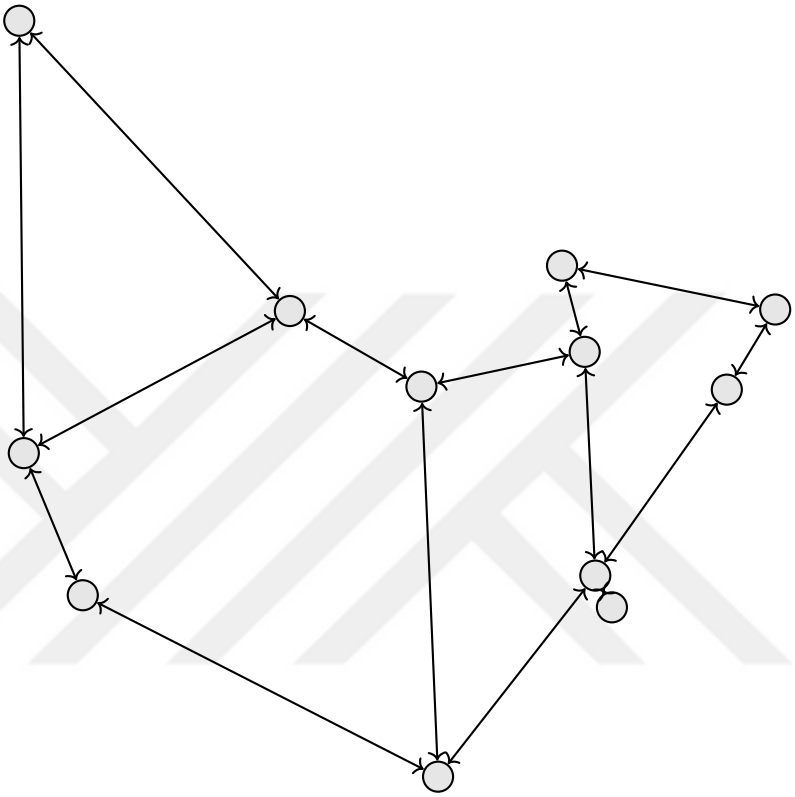


Figure 5.2: Network Visualization for the abilene topology.

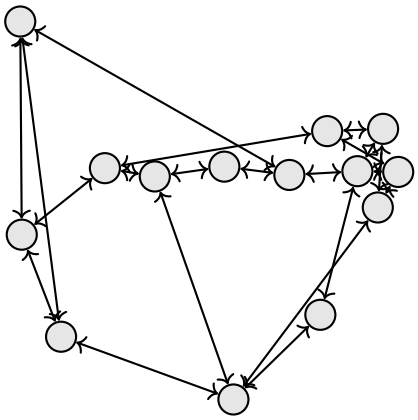


Figure 5.3: Network Visualization for the nobel-us topology.

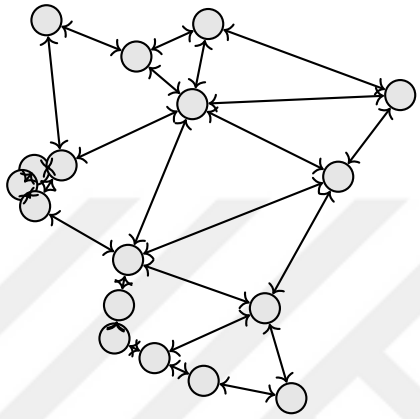


Figure 5.4: Network Visualization for the nobel-germany topology.

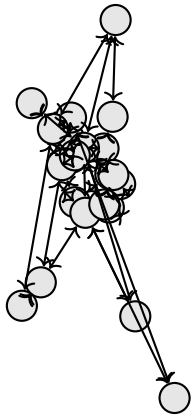


Figure 5.5: Network Visualization for the geant topology (*ny1.ny* node not shown).

each instance, we calculate the capacity for each edge as described in [10]. In particular, we route all demand through shortest paths and obtain the baseline capacity from the link with the highest flow, T_e. Then, we randomly pick a capacity for all edges from the range $[T_e, (1+\ell)T_e)]$ for $\ell \geq 0$.

In our single-objective experiments we set $\lambda = 1$, and $\ell = 1$. To understand and test model performance on different segment limits, we vary $K \in \{1, 2, 3, 4, 5\}$ and we test different K values on the same network topology - same link lengths and capacities- and demands. In order to measure the performance of our model, we compare our solutions with the Multi-Commodity Flow model, MCF, as it allows arbitrary traffic splitting and hops. As done in [10], we define

$$\eta = \frac{\Theta_{MODEL}}{\Theta_{MCF}}, \tag{5.1}$$

where Θ_{MODEL} and Θ_{MCF} are the congestion levels of MODEL (FLOW OR PATH) and MCF, respectively.

Then for each instance and for each $K \in \{1, 2, 3, 4, 5\}$, we solve FLOW, with and without cuts, PATH and K-sMILP.

5.1.2 Results of the Experiments

In Table 5.1 we present a comparative evaluation of solution times, and performance ratio values with $K = 2$ for sMILP from [10] and our models FLOW, with and without the cuts, and PATH, across five network topologies. For each model, average solution times including model build times (in seconds) and performance ratios (η values as given by (5.1)) are reported.

From Table 5.1, we observe that our valid inequalities yield significant computational savings, and our segment-based FLOW model is orders of magnitude faster than the sMILP in the literature, even without cuts. Moreover, the PATH model consistently outperforms all others, offering substantial time savings. In

terms of performance ratio, both our segment-based model and those in the literature achieve same values, as expected. Remarkably, the PATH model captures the congestion level of the MCF even with $K = 2$.

Table 5.1: Solution times and performance ratios for various models evaluated on benchmark topologies for various K.

Topology	K	Time				η			
		sMILP	FLOW w/o	FLOW w	PATH	sMILP	FLOW w/o	FLOW w	PATH
Smallnet	1.00	7.18	1.45	1.42	0.015	3.54	3.54	3.54	3.28
$\|N\| = 10, \|A\| = 42$	2.00	17.02	3.15	1.44	0.032	1.14	1.14	1.14	1.0006
	3.00	7.75	2.39	-	0.032	1.14	1.14	-	1
	4.00	6.88	2.42	-	0.030	1.14	1.14	-	1
	5.00	6.05	2.46	-	0.030	1.14	1.14	-	1
Abilene	1.00	13.02	1.71	1.90	0.014	1.56	1.56	1.56	1.56
$\|N\| = 12, \|A\| = 30$	2.00	27.79	4.83	1.89	0.023	1.04	1.04	1.04	1.04
	3.00	17.01	2.75	-	0.020	1.00	1.00	-	1.00
	4.00	13.89	2.59	-	0.023	1.00	1.00	-	1.00
	5.00	12.69	2.61	-	0.020	1.00	1.00	-	1.00
Nobel-US	1.00	38.74	3.64	4.06	0.042	1.99	1.99	1.99	1.99
$\|N\| = 14, \|A\| = 42$	2.00	59.01	11.74	3.97	0.070	1.00	1.00	1.00	1.00
	3.00	37.63	9.82	-	0.068	1.00	1.00	-	1.00
	4.00	27.96	8.94	-	0.071	1.00	1.00	-	1.00
	5.00	26.01	9.79	-	0.068	1.00	1.00	-	1.00
Nobel-Germany	1.00	147.24	10.89	10.84	0.053	2.50	2.50	2.50	2.50
$\|N\| = 17, \|A\| = 52$	2.00	1456.17	184.64	10.63	0.109	1.08	1.08	1.08	1.07
	3.00	211.77	27.09	-	0.115	1.00	1.00	-	1.00
	4.00	158.07	21.52	-	0.136	1.00	1.00	-	1.00
	5.00	137.51	19.56	-	0.117	1.00	1.00	-	1.00
Geant	1.00	856.92	43.02	40.02	0.11	2.94	2.94	2.94	2.94
$\|N\| = 22, \|A\| = 72$	2.00	1642.62	512.47	38.18	0.21	1	1	1	1
	3.00	739.97	167.15	-	0.22	1	1	1	1
	4.00	716.73	189.09	-	0.21	1	1	1	1
	5.00	569.17	176.69	-	0.16	1	1	1	1

For the Smallnet topology, in Table 5.1 observe that **PATH** achieves $\eta = 1.00$ for $K \geq 3$ with runtimes between 0.030–0.032 s, offering the fastest and most consistent performance. **FLOW w/o** performs well, achieving $\eta = 1.14$ for $K \geq 2$ with runtimes ranging from 2.39–3.15 s. **sMILP** shows similar η values but requires significantly more time, peaking at 17.02 s for $K = 2$. **FLOW w** matches the η of other models and is slightly faster than **FLOW w/o** for $K = 1$ and $K = 2$, but results are unavailable for higher K. At $K = 1$, all models have higher performance ratios, with $\eta = 3.54$ for **sMILP**, **FLOW w/o**, and **FLOW w**, and $\eta = 3.28$ for **PATH** this indicates that due to complex topology of Smallnet the segment limit constraints are discarding valid solutions.

For the Abilene topology, in Table 5.1 we observe that **PATH** reaches $\eta = 1.00$ for $K \geq 3$ with runtimes between 0.020–0.023 s, offering the fastest and most scalable performance. **FLOW w/o** achieves the same η with higher runtimes of 2.59–2.75 s, while **sMILP** is the slowest, reaching up to 27.79 s at $K = 2$. **FLOW w** offers comparable runtimes to **FLOW w/o** and **sMILP** for $K = 1$ and $K = 2$. At $K = 1$, all models exhibit elevated performance ratios with $\eta = 1.56$, suggesting insufficient routing flexibility. Increasing K to 2 sharply reduces η to 1.04, and from $K = 3$ onward, all models stabilize at $\eta = 1.00$, indicating that further segment limit increases do not yield additional gains.

For the Nobel-US topology, in Table 5.1 we observe that **PATH** consistently achieves $\eta = 1.00$ with runtimes between 0.042–0.071 s, making it the most efficient. **FLOW w/o** reaches the same η with runtimes of 8.94–11.74 s, while **sMILP** is the slowest, ranging from 26.01–59.01 s. **FLOW w** achieves optimal η with runtimes of 4.06 s and 3.97 s for $K = 1$ and $K = 2$, slightly outperforming **FLOW w/o** in time. At $K = 1$, all models have $\eta \approx 1.99$, but increasing to $K = 2$ reduces η to 1.00, with no further improvement for higher K.

For the Nobel-Germany topology, in Table 5.1 we observe that **PATH** achieves $\eta = 1.00$ for $K \geq 3$ with runtimes ranging from 0.109–0.136 s, offering fast and consistent performance. **FLOW w/o** also reaches $\eta = 1.00$ in this range but with much higher runtimes of 19.56–27.09 s. **sMILP** is significantly slower, peaking at 1456.17 s for $K = 2$, despite offering no improvement in η. **FLOW w** provides

the lowest runtime (10.63 s) among the non-PATH models at $K = 2$. At $K = 1$, all models show elevated performance ratios of $\eta = 2.50$, which drop to $\eta = 1.08$ at $K = 2$ and stabilize at $\eta = 1.00$ for $K \geq 3$, once again indicating that additional segments beyond this point yield no further improvement.

For the Geant topology, in Table 5.1 we observe that **PATH** achieves $\eta = 1.00$ for $K \geq 2$ with runtimes between 0.16–0.22 s, making it the fastest and most efficient model. **FLOW w/o** and **sMILP** also reach $\eta = 1.00$ for $K \geq 2$, but with significantly higher runtimes, reaching up to 512.47 s and 1642.62 s respectively at $K = 2$. **FLOW w** delivers fast results at $K = 1$ and $K = 2$ (40.02 s and 38.18 s), outperforming **FLOW w/o**, but is not applicable for larger K due to how we construct the cuts. At $K = 1$, all models have $\eta = 2.94$, showing high congestion when only one path is allowed. Increasing K to 2 drastically reduces η to 1.00 across all models, with no further gains for higher K.

Across all five topologies, **PATH** consistently delivers the fastest solution times while achieving optimal or near-optimal performance ratios (η), especially for $K \geq 2$. It maintains runtimes below 0.25 s even in large networks like Geant and Nobel-Germany, where **sMILP**, **FLOW** and **FLOW w/o** often exceed several hundred seconds. **sMILP**, while always matchin the η of **FLOW**, exhibits significantly higher computational costs, with peak runtimes up to 1642.62 s on the Geant topology. **FLOW w/o** provides a good trade-off between runtime and solution quality, achieving $\eta = 1.00$ in most cases, though it is still one to two orders of magnitude slower than **PATH**. **FLOW w** performs extremely well when we can apply it, often outperforming **FLOW w/o** in runtime while achieving identical η. For all topologies, performance ratios are high at $K = 1$, typically between 1.5 and 3.0, but consistently drop to $\eta = 1.00$ at $K = 2$ or $K = 3$, indicating that at least 2 segments are sufficient to achieve near-optimal routing performance.

To conclude this portion of the single objective experiments, our models consistently outperform the models from [10], including the valid inequalities to **FLOW** formulation improves solution times immensely, and finally our column generation approach, **PATH**, offers extremely fast solutions without sacrificing

any solutions to segment limit constraints.

5.1.3 Impact of Valid Inequalities

The improvement in overall solution times of including our valid inequalities can be seen in Table 5.1, under the **FLOW w** column. To elaborate further on the impact of our valid inequalities for the $K = 1$ and $K = 2$ case, we conduct an empirical comparison using the *Geant* topology from [37]. As it is a larger topology, the differences in build time and solution time are clearer. The cuts are designed to exclude segments that would violate the hop limit, thus reducing the solution space and tightening the formulation. However, additional constraints must be added to the model. On the *Geant* topology, when $K = 2$, we add 43% more constraints to the problem however this only results in 4% longer model building times. When $K = 1$, the cuts cause 47% more constraints to be added to the model, however reduce the solution times tremendously and result in 70% reduction overall solution times. We conclude that our valid inequalities are very simple to introduce to any MILP that assigns variables to individual segments, does not incur any substantial cost to model building times and results in much faster solution times especially for $K = 2$.

5.2 Inclusion of non-shortest path segments

Additionally, we have tested the importance of non-shortest path link segments for the FLOW model across $K \in \{1, 2, 3\}$. Excluding these segments are suboptimal as incorporating all $K-$segment paths are required for optimal solutions [10]. In Table 5.2 we provide the η for each configuration averaged over 10 equal network topologies and demands. As observed, inclusion of the non-shortest path segments is very crucial for the synthetic *smallnet* topology. Disabling these segments almost doubles the η when $K \in \{2, 3\}$. However, for the Nobel-US topology the results are much less drastic, where we see an 8% increase only. One explanation for this difference is that *smallnet* contains many links that connect

Table 5.2: η on the *nobel-us* and *smallnet* topologies for varying K (1–3) with and without non-shortest path adjacency segments.

	Nobel-Us			*Smallnet*		
K	1	2	3	1	2	3
Without Adj. Seg.	2.37	1.08	1.08	4.61	1.97	1.97
With	2.33	1.00	1.00	3.71	1.05	1.05
Ratio	**1.02**	**1.07**	**1.08**	**1.25**	**1.88**	**1.88**

each demand point on the other hand nodes in Nobel-US are only connected to nodes that are close to them geographically. We can quantify this by measuring the degree centrality of the nodes in the network: in the Nobel-US topology a node is connected to 23% of the nodes on average, on the other hand this metric is 48% for the *smallnet* topology. Moreover, we observe that disabling non-shortest path link-segments are less of an impediment when $K = 1$. This can be attributed to the simple fact that when $K = 1$ we are only allowed to pick one segment, when more segments are allowed disabling certain segments create more congestion. In conclusion, modeling the non-shortest path link segments is crucial for obtaining SR solutions however its effect is dependent on the network topology.

5.3 Capabilities of CG

We evaluate the performance and scalability of our CG model through single-objective experiments on large-scale network topologies drawn from the Rocketfuel dataset, as made available in the REPETITA repository [36]. These experiments highlight the suitability of the CG framework for solving complex instances where compact formulations become computationally intractable due to memory limitations. Table 5.3 presents the average solution times (in seconds, including model build time, over 5 traffic matrices given in repository) and number of columns generated to reach optimality across various hop limits ($K = 1, 2, 3$). As the table illustrates, the CG approach remains tractable for sizable networks such as *rf3257*, though the computational burden and number of generated columns increase with larger topologies and higher K values. Notably, only the largest

instance, *rf1239*, exceeds available memory for $K = 3$.

Table 5.3: Average solution times (including model build time) for five traffic matrices from the REPETITIA repository [36]. Each topology is evaluated for hop limits $K \in \{1, 2, 3, 4, 5\}$. Empty entries represent instances where the the system ran out of memory.

				$K = 1$		$K = 2$		$K = 3$	
Network	$\|N\|$	$\|A\|$	$\|D\|$	Time(s)	# Columns	Time(s)	# Columns	Time(s)	# Columns
rf3967	79	294	6162	4	6162	16	14600	16	18408
rf1755	87	322	7482	5	7482	38	23021	45	27018
rf1221	104	302	10712	6	10712	17	17247	19	18906
rf6461	138	744	18906	25	18906	210	54664	1059	58816
rf3257	160	655	25759	39	25759	230	59089	533	74453
rf1239	315	1944	98910	313	98910	5334	237677	-	-

Chapter 6

Energy vs. Congestion Trade-Off

This section is dedicated to exploring results of our multi-objective framework with our compact model and with the column generation algorithm. Heavier attention is placed on analysis of the results that we have obtained through the CG model using larger network topologies that are not possible to solve with our compact model.

Our goal in this section is to simulate a more realistic setting. We use the *Geant* topology from [37] for the FLOW model and *Germany50* for the PATH model. The topologies are chosen for their realistic sizes and easy access to aggregated traffic matrices. The link lengths are assigned using the Haversine distance (great-circle distance) between node pairs [39], while demand matrices are sourced directly from [37] corresponding to 10 5-minute intervals, yielding 10 distinct demand scenarios per topology. Then, we assign capacity for all edges uniformly with the value $(1 + \ell)T_e$ for $\ell \geq 0$. We vary key parameters across experiment groups to evaluate model behavior under different system settings. Unless otherwise noted, the default values are $\ell = 1, \lambda = 1$, and $K = 3$. Link energy costs are set uniformly as $\lambda_e = 1$ for all $e \in E$. The experiments are grouped as follows:

- **E1 – Processing Cost Sensitivity:** We vary the router processing energy

Algorithm 3 Obtaining Solutions with ϵ-Constraints

procedure ϵ-FIND
  Compute $\Theta_I \leftarrow \min \Theta$
  Compute $\Lambda_N \leftarrow \min \Lambda$ **subject to** $\Theta \leq \Theta_{min}$
  Compute $\Lambda_I \leftarrow \min \Lambda$
  Compute $\Theta_N \leftarrow \min \Theta$ **subject to** $\Lambda \leq \Lambda_{min}$
  Compute step size: $\epsilon \leftarrow (\Theta_N - \Theta_I)/10$
  for $n = 0$ **to** 10 **do**
    Compute $\Lambda^{(n)} \leftarrow \min \Lambda$ **subject to** $\Theta \leq \Theta_I + n \cdot \epsilon$
    Store solution: $(\Theta^{(n)}, \Lambda^{(n)}) \leftarrow (\Theta, \Lambda)$
  end for
  return $\{(\Theta^{(i)}, \Lambda^{(i)}) \mid i = 0, 1, \ldots, 11\}$
end procedure

cost parameter $\lambda \in \{0.25, 0.5, 0.75, 1.0, 1.25, 2.0, 4.0\}$ to evaluate its effect on energy-congestion trade-offs.

- **E2 – Capacity Variation:** We analyze the impact of varying link capacity multipliers $\ell \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$ on model performance.
- **E3 – Hop Limit Restriction:** We explore how increasing the allowed hop limit, $K \in \{1, 2, 3, 4, 5\}$, influences solution quality and variety.

6.1 FLOW Model with ϵ-constraint Extension

In this section we extend the **FLOW** model with the $\epsilon-$ constraint method. ϵ-constraint method allows us to generate a diverse set of non-dominated solutions [34]. In order to generate solutions with ϵ-constraint method we use the method described in Algorithm 3.

For each experiment group, we generate 10 instances by using the traffic matrices from [37] while keeping the network topology fixed. This ensures that the comparison is fair across different parameter values within the experiment. We employ two bi-objective solution strategies: (i) the ϵ-constraint method, where ten

constraint levels are applied per instance, and (ii) scalarization using a weighted-sum approach with $\alpha \in 0, 0.1, \ldots, 1.0$. This comprehensive setup results in a total of 4,180 solved instances across all configurations.

While the Pareto frontier of a bi-objective linear program is theoretically convex, generating all efficient solutions using weighted sum scalarization is not always straightforward. Specific values of the scalarization parameter α may yield duplicate solutions. In contrast, the ϵ-constraint method systematically enforces diversity in solutions by restricting one objective and optimizing the other. This difference is empirically illustrated in Figures 6.3, 6.4 and 6.5, where each method traces a distinct subset of the efficient frontier.

Moreover, we observe that solution times between two methods are similar in Figures 6.1 and 6.2. where we report the solution times across varying scalarization weights (α) and the ϵ-constraint multiplier n. While the solution times are similar for both methodologies, we observe that as the congestion objective have more weight($\alpha = 1$) or has more restriction($n = 0$) the solution times are higher regardless of methodology and experiment. Additionally, we observe that solution times for the weighted sum scalarization method has more variance compared to the ϵ-constraint method.

From Figure 6.1 we observe that Experiment Types 1 and 2 maintain relatively stable solution times across all values of α, with medians clustering between 112 and 117. In contrast, Experiment Type 3 exhibits consistently higher variance, particularly pronounced for $\alpha \geq 0.9$, where both the median and values outside the quartiles increase sharply. All experiment types show a notable increase in solution time at $\alpha = 1.0$.

From Figure 6.1 we observe that across all values of n, Experiment Type 1 and Type 2 consistently yield lower solution times than Type 3, with median times stabilizing around 107.5–110.0 units from $n = 3$ onward. Type 3 generally exhibits higher variability, particularly at $n = 0$ and $n = 1$, where it shows both the highest median and the widest interquartile range. As n increases, solution times decrease for all experiment types and converge, but Type 3 remains the

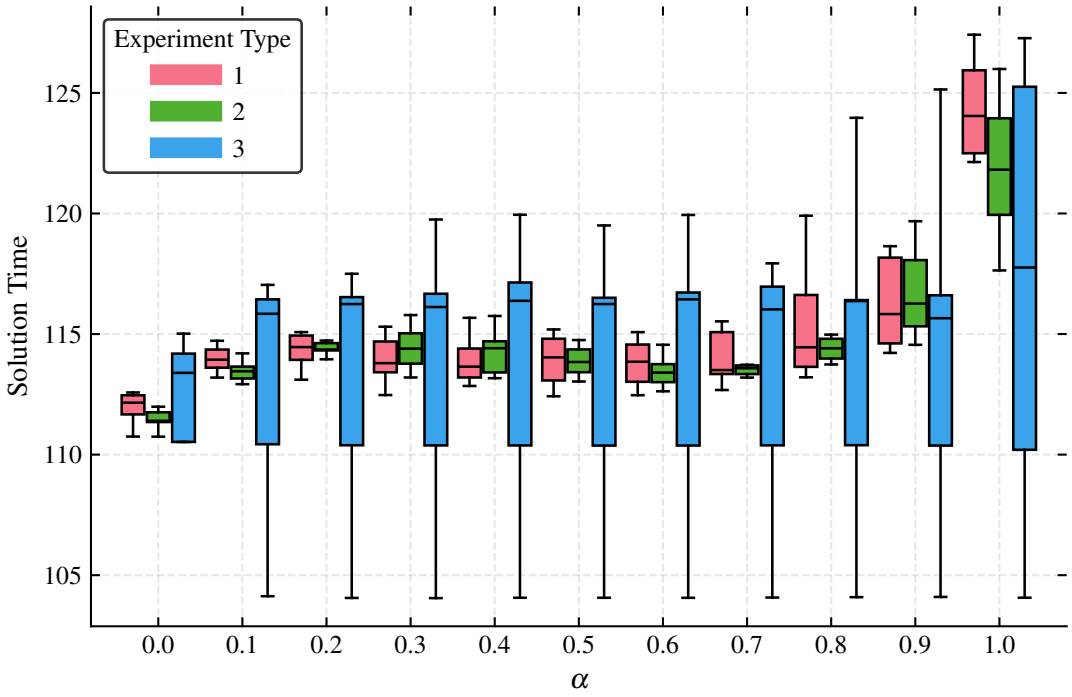


Figure 6.1: Solution times for each α across all experiments.

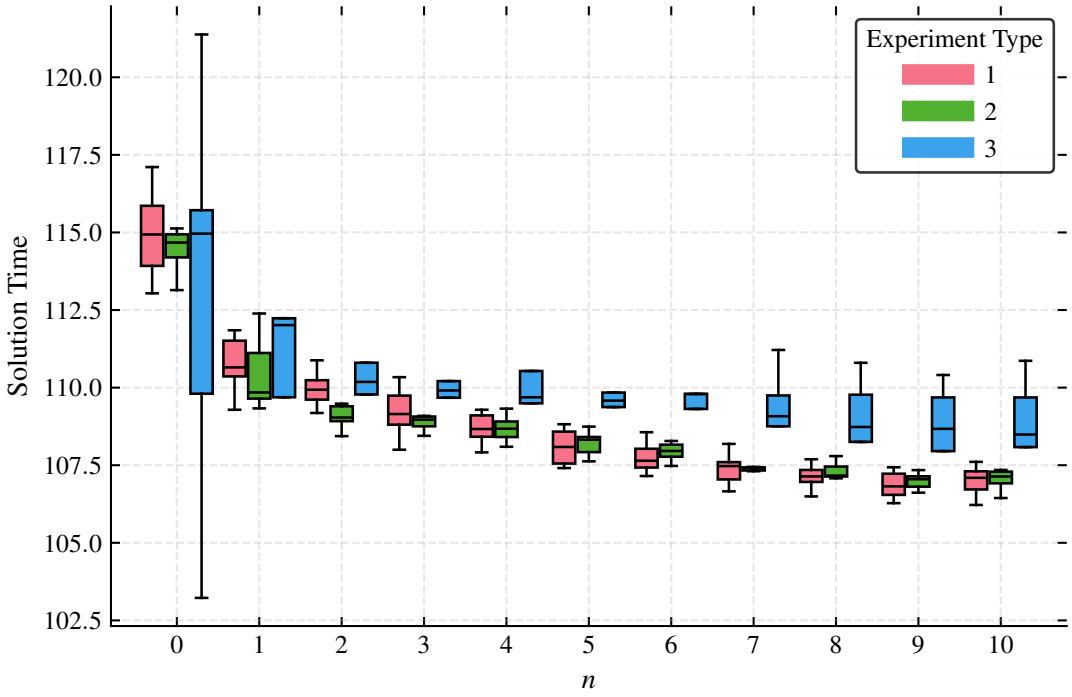


Figure 6.2: Solution times for each n across all experiments.

slowest on average. The gap between the experiment types narrows for $n \geq 4$, indicating that solely optimizing for congestion can lead to higher variance in solutions.

6.1.1 Results for Experiment 1

In this experiment, we investigate how the variation of the router processing energy cost parameter λ affects the trade-off between energy consumption and congestion. By testing a range of values for λ, we aim to understand how sensitive the models are to energy-related cost factors and how this impacts routing decisions.

As shown in Figure 6.3, solution values are normalized to $[0, 1]$, each point corresponds to the average of 10 instances, increasing the processing energy cost parameter λ shifts the trade-off curve upward along the energy axis, indicating higher total energy consumption. Importantly, the maximum link utilization (MLU) remains nearly unchanged across λ values, implying that the model prioritizes fewer hops under higher costs without compromising traffic balancing. These results suggest that investing in energy-efficient routers can significantly reduce energy consumption without negatively impacting congestion. As an interesting observation, in Figure 6.3 we observe that the α-method found solutions that dominated the solutions from the ϵ-constraint method. This is possible due to ϵ-constraint method can only guarantee weakly efficient solutions [34]

6.1.2 Results for Experiment 2

This experiment evaluates the robustness of the models under varying levels of network capacity. By scaling link capacities through the multiplier ℓ, we observe how congestion and feasibility are influenced under both constrained and relaxed settings.

Figure 6.4, solution values are normalized to $[0, 1]$, each point corresponds to

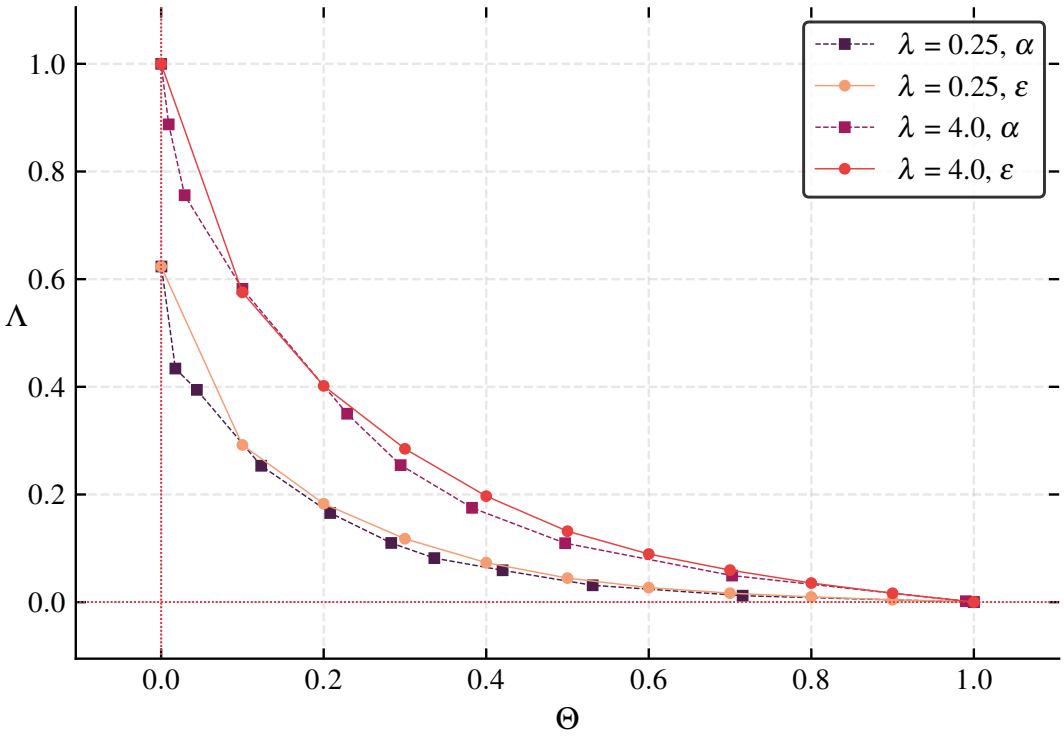


Figure 6.3: Energy–congestion trade-offs for processing cost settings in Experiment **E1**, conducted on the *Geant* topology. The results compare performance under two processing cost parameters ($\lambda = 0.25$ and $\lambda = 4.0$).

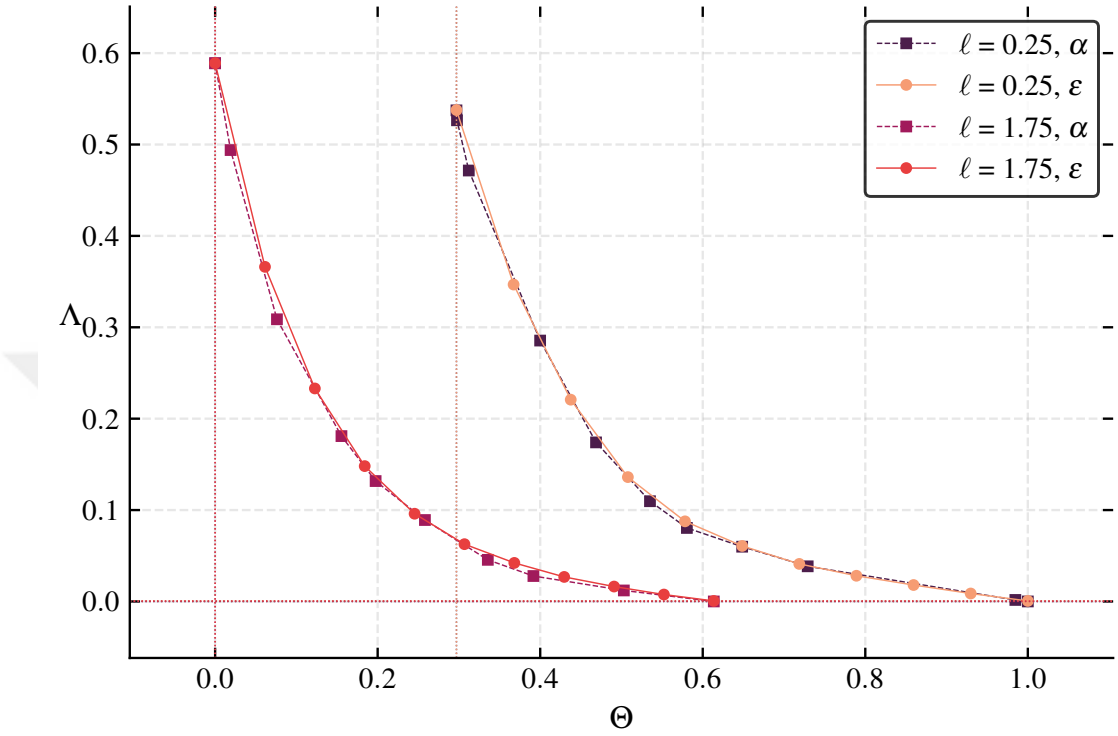


Figure 6.4: Energy vs. congestion trade-offs for two capacity multiplier settings ($\ell = 0.25$ and $\ell = 1.75$), using both the weighted sum scalarization (α-method) and ϵ-constraint approach.

the average of 10 instances, compares the trade-offs under varying link capacity multipliers. As expected, higher values of ℓ result in both lower energy usage and congestion. Higher link bandwidth - if it comes at no additional energy cost - seems to favor both of our objectives since it allows us to pick shorter links with less hops and edges.

6.1.3 Results for Experiment 3

Here, we examine the impact of limiting the number of allowed hops, K, on the solution space. This allows us to study the trade-offs between routing flexibility and performance, particularly how smaller K values may restrict the model and affect solution quality.

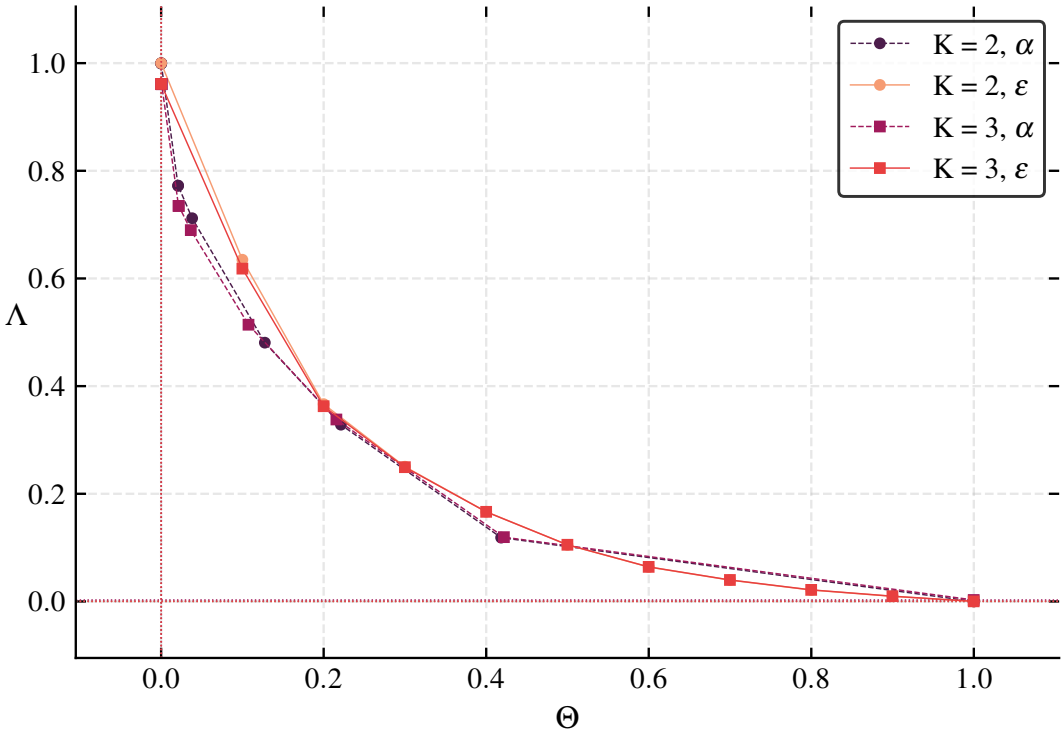


Figure 6.5: Energy vs. congestion trade-offs under different hop limits ($K = 2$ and $K = 3$), using both scalarization (α) and ϵ-constraint methods.

In Figure 6.5, we observe that allowing more hops (i.e., increasing K) enhances the model's flexibility to optimize routing paths. The ϵ-constraint method captures more non-dominated solutions than scalarization, especially as K increases. Increasing the allowed hop count from $K = 2$ to $K = 3$ improves the model's routing flexibility, enabling marginal gains in both objectives. Solution values are normalized to $[0, 1]$ in Figure 6.5, each point corresponds to the average of 10 instances. Notably, after $K = 3$, the marginal improvement in MLU and energy is negligible, refer to Figure 6.6, aligning with prior literature suggesting that limited detouring is sufficient for optimal trade-offs [10]. Increasing the allowed hop count from $K = 3$ to 4 or 5 produces no improvement.

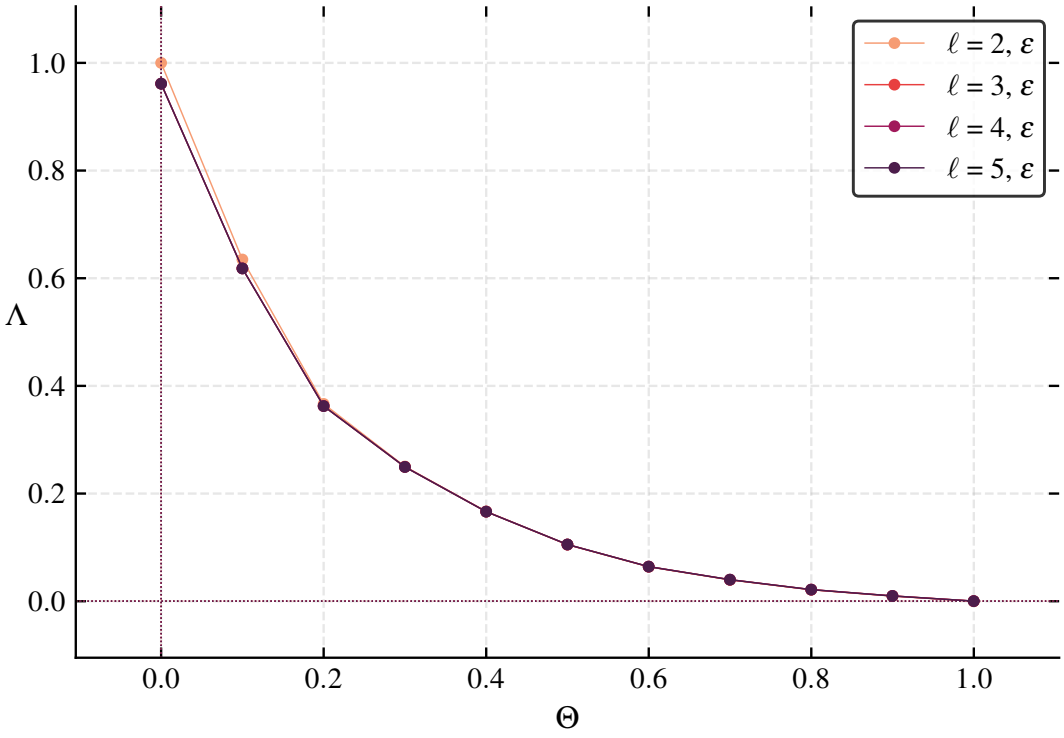


Figure 6.6: Energy vs. congestion trade-offs under different hop limits ($K \in \{2, 3, 4, 5\}$), using only ϵ-constraint methods for clarity.

6.2 PATH Model with Germany50 Topology

In this section, we assess the scalability and capabilities of our column generation, PATH approach by applying it to larger, real-world networks by multi-objective experiments.

We use the *Germany50* topology with 50 nodes and 176 arcs from [37] to conduct a multi-objective analysis. This topology provides access to real traffic matrices and enables us to investigate energy–congestion trade-offs. We use the weighted sum scalarization method with the objective (3.3) where we take the scalarization multiplier α from $\{0.0, 0.25, 0.5, 0.75, 1.0\}$. The experimental framework mirrors that of earlier sections: we vary the router processing energy cost (λ), link capacity multiplier (ℓ), and hop constraint (K), corresponding to experiment groups **E1**, **E2**, and **E3**, we generate 20 instances using demand matrices from [37]. Note that in contrast to the single-objective experiments, we do not assign a random link capacity from a range in this set of experiments. As there are overlaps between the ranges generated by varying ℓ, it defeats the purpose of testing for a set of link capacities. The results demonstrate both the practicality of the CG approach for large-scale network optimization and its robustness across different parameter settings.

6.2.1 Results for Experiment 1

In Experiment **E1**, we analyze the energy–congestion trade-offs for varying processing cost parameters $\lambda \in \{0.25, 0.5, 0.75, 1.0, 1.25, 2.0, 4.0\}$. In Table 6.1, detailed solution results are given. We observe that regardless of parameter setting when we disregard the congestion objective, i.e. $\alpha = 0$, generating one path per demand is enough. If more weight is given to the congestion objective, generation of more paths seem to be necessary as the model searches for paths that could improve the congestion level. Moreover, the trends in the efficiency score Z is stable across different λ, the highest values are observed when $\alpha = 0.5$. Which implies that the farthest solutions to the ideal point, i.e. Θ_I, Λ_I, are found when both

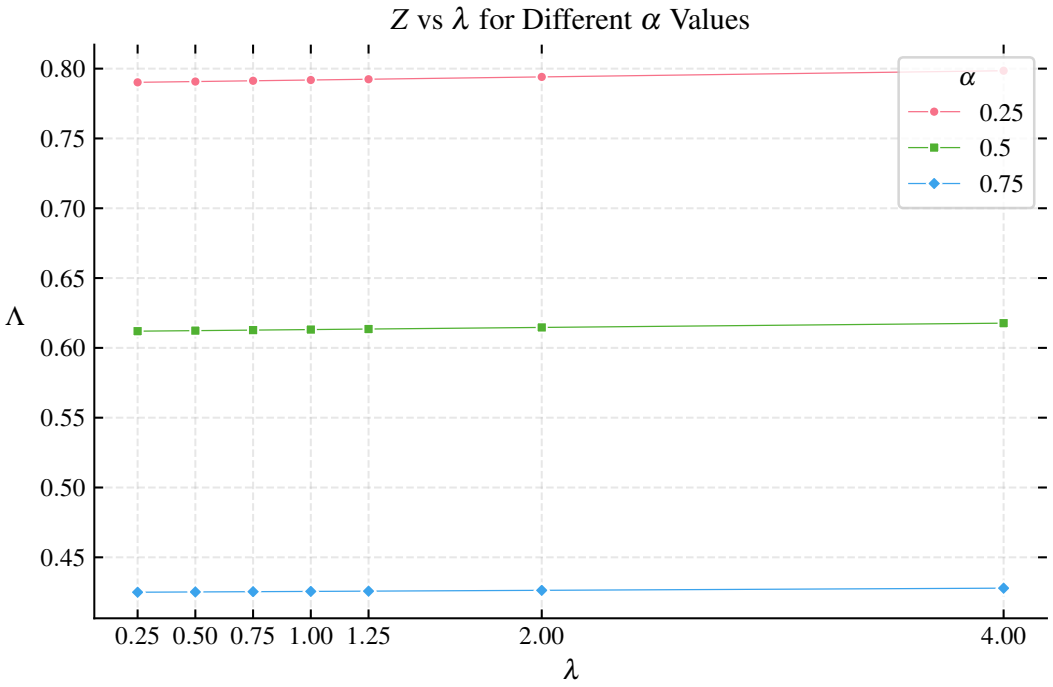


Figure 6.7: Z vs λ

objectives are considered equally. Solution times across both λ values are similar and higher for higher α values as more paths are needed. One other observation for $\lambda \geq 2$, $\alpha = 0$ and $\alpha = 0.25$ seems to find almost the same set of solutions with a small deviation on $\alpha = 0.25$ for $\lambda = 2$. This is partly a consequence of using the weighted sum scalarization method, as it cannot guarantee unique solutions as the weight parameter changes. However, this also implies that higher energy costs might necessitate more attention on the congestion objective to start finding solutions with better congestion trade-offs.

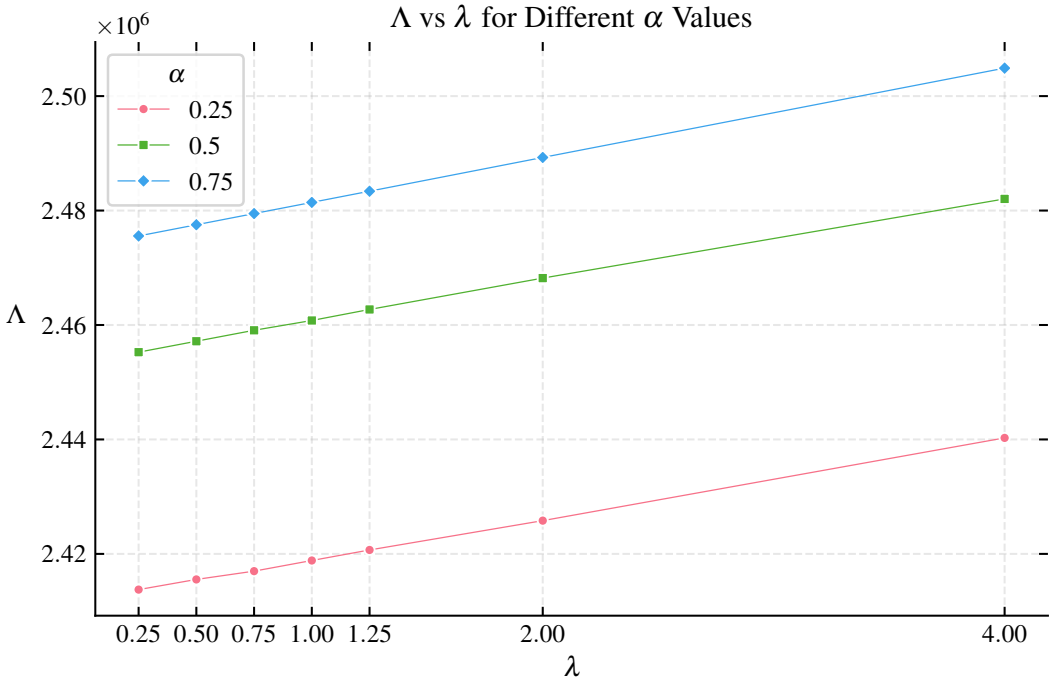


Figure 6.8: Λ vs λ

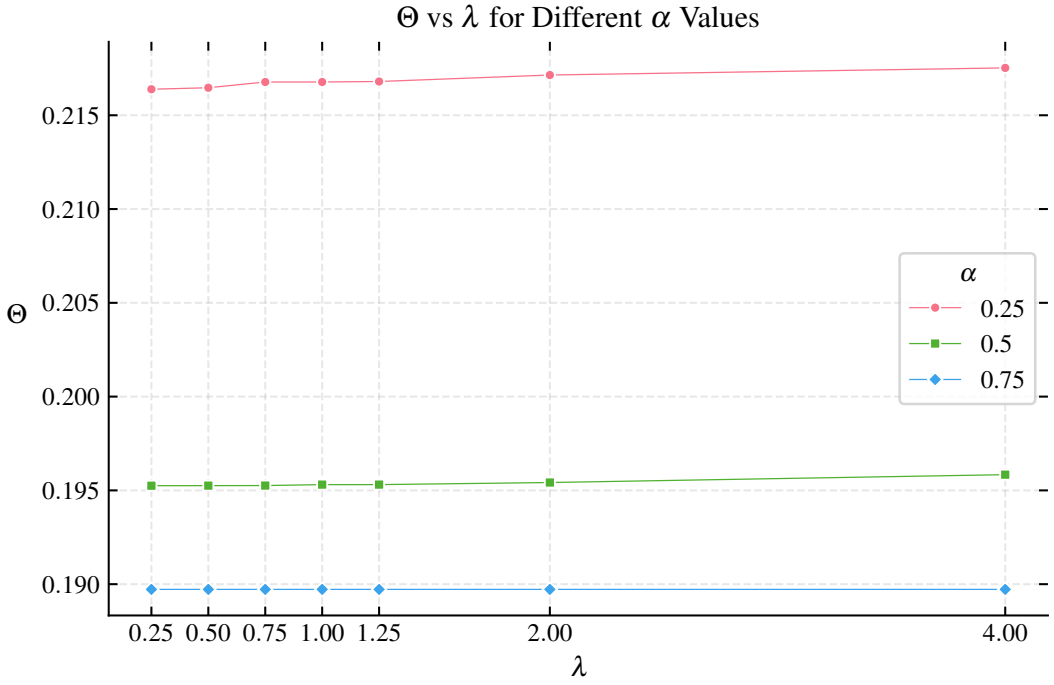


Figure 6.9: Θ vs λ

Table 6.1: Results of Experiment **E1** for the *Germany50* topology. We omit values that do not change with changes in λ. $\Theta_{\text{MCF}} = 0.1897$, $\Theta^N = 0.7997$, $\Lambda^N = 2504877.1$.

λ	α	Θ	Λ	Z	No. Col.	Time
0.25	0.00	0.5000	2366816.46	0.945513	2450.00	1.09
	0.25	0.2164	2413760.10	0.790192	2969.60	1.95
	0.50	0.1952	2455251.99	0.611966	3218.40	2.07
	0.75	0.1897	2475559.47	0.425003	3354.10	2.16
	1.00	0.1897	2792515.56	0.237215	3342.10	2.39
0.50	0.00	0.5000	2368346.53	0.946124	2450.00	1.09
	0.25	0.2165	2415537.23	0.790746	2967.90	1.92
	0.50	0.1953	2457164.13	0.612348	3216.95	2.05
	0.75	0.1897	2477515.83	0.425197	3353.75	2.14
	1.00	0.1897	2794568.19	0.237215	3342.10	2.36
0.75	0.00	0.5000	2369876.89	0.946734	2450.00	1.07
	0.25	0.2168	2416985.44	0.791299	2966.55	1.94
	0.50	0.1953	2459077.76	0.612731	3215.30	2.05
	0.75	0.1897	2479470.94	0.425391	3350.60	2.14
	1.00	0.1897	2796620.77	0.237215	3342.10	2.37
1.00	0.00	0.5000	2371406.75	0.947345	2450.00	1.07
	0.25	0.2168	2418846.30	0.791852	2965.40	1.90
	0.50	0.1953	2460788.40	0.613113	3210.30	1.99
	0.75	0.1897	2481425.75	0.425585	3351.60	2.11
	1.00	0.1897	2798673.15	0.237215	3342.10	2.35
1.25	0.00	0.5000	2372937.36	0.947956	2450.00	1.09
	0.25	0.2168	2420687.06	0.792406	2963.75	1.91
	0.50	0.1953	2462715.79	0.613495	3209.20	2.01
	0.75	0.1897	2483382.26	0.425779	3349.85	2.13
	1.00	0.1897	2800726.13	0.237215	3342.10	2.36
2.00	0.00	0.5000	2377527.65	0.949788	2450.00	1.11
	0.25	0.2171	2425811.15	0.794068	2958.05	1.94
	0.50	0.1954	2468194.95	0.614645	3210.50	2.04
	0.75	0.1897	2489278.84	0.426363	3347.05	2.15
	1.00	0.1897	2806883.83	0.237215	3342.10	2.35
4.00	0.00	0.5000	2389769.45	0.954675	2450.00	1.08
	0.25	0.2175	2440270.65	0.798492	2949.75	1.89
	0.50	0.1958	2482033.45	0.617697	3201.55	2.01
	0.75	0.1897	2504889.86	0.427912	3339.75	2.12
	1.00	0.1897	2823304.83	0.237215	3342.10	2.36

6.2.2 Results for Experiment 2

In Experiment **E2**, we analyze the energy–congestion trade-offs for varying link capacity multipliers ($\ell \in 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75$). In Table 6.2, detailed solution results are given. We observe that the model is robust against changes in the capacity multiplier, we find similar or the same paths while generating the same number of columns across ℓ values. The construction of G^q is dependent on edge capacities, as G^q affects the column generation process this stability is important.

Figures 6.10, 6.11 and 6.12,show the energy and congestion trade-offs in changes over ℓ and α respectively. In Fig. 6.10 we observe that the efficiency score decreases as capacity is increased. From Fig. 6.12 we observe that as ℓ decreases(lower link capacity in the network), the congestion levels increase as expected. On the other hand, energy consumption at the same congestion level becomes higher as ℓ decrease. In Fig. 6.11, we observe a similar trend with **E1**, importance of considering the energy objective is clear (e.g. $\alpha = 0.50$ compared with $\alpha = 0.75$). The general trend observed in **E1** holds: higher capacities enable better trade-offs between energy and congestion. As ℓ increases, thus link capacity, the congestion levels drop off, however the differences in energy consumption across ℓ values is marginal. Notably, across a broad range of ℓ values, congestion can be reduced substantially with minimal increases in energy consumption. When capacity becomes more constrained (lower ℓ), the model requires more energy to reroute traffic efficiently as it uses more nodes and links to reduce congestion.

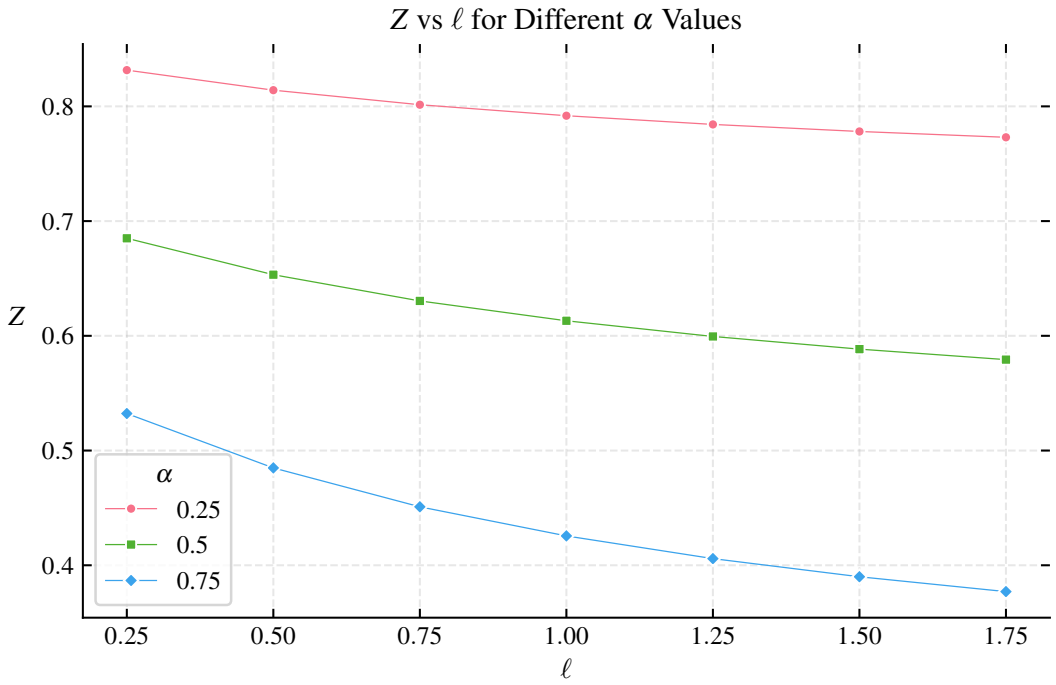


Figure 6.10: Z vs λ for Experiment 2

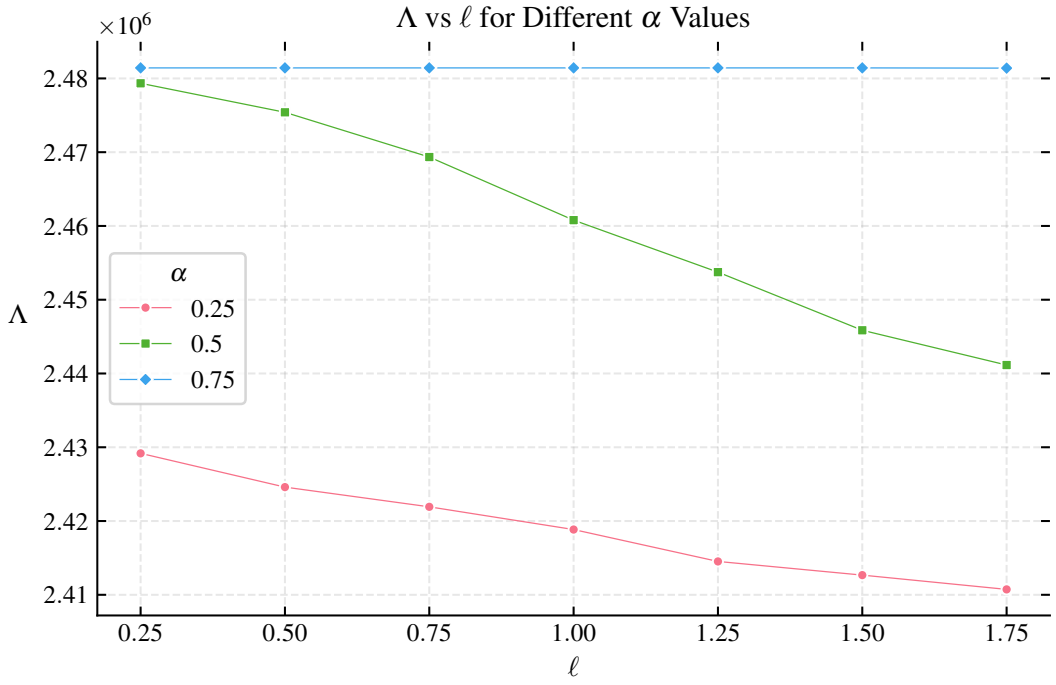


Figure 6.11: Λ vs λ for Experiment 2

Table 6.2: Results of Experiment **E2** for the *Germany50* topology. We omit values that do not change with changes in λ. $\Theta_{\text{MCF}} = 0.1897$, $\Theta^N = 0.7997$, $\Lambda^N = 2504877.1$.

ℓ	α	Θ	Λ	Z	No. Col.	Time
0.25	0.00	0.7998	2371406.75	0.947345	2450.00	1.08
	0.25	0.3333	2429176.00	0.831597	3052.95	1.93
	0.50	0.3042	2479339.75	0.685019	3327.85	2.12
	0.75	0.3035	2481425.75	0.532256	3353.80	2.12
	1.00	0.3035	2792889.18	0.379444	3341.50	2.35
0.50	0.00	0.6666	2371406.75	0.947345	2450.00	1.06
	0.25	0.2824	2424598.83	0.814114	3013.80	1.87
	0.50	0.2549	2475407.45	0.653268	3285.20	2.08
	0.75	0.2529	2481425.75	0.484857	3353.25	2.12
	1.00	0.2529	2775169.23	0.316245	3342.00	2.34
0.75	0.00	0.5713	2371406.75	0.947345	2450.00	1.06
	0.25	0.2448	2421926.60	0.801434	2990.65	1.92
	0.50	0.2204	2469342.35	0.630409	3242.80	2.06
	0.75	0.2168	2481425.75	0.450955	3352.70	2.08
	1.00	0.2168	2795042.83	0.271042	3341.95	2.33
1.00	0.00	0.5000	2371406.75	0.947345	2450.00	1.07
	0.25	0.2168	2418846.30	0.791852	2965.40	1.90
	0.50	0.1953	2460788.40	0.613113	3210.30	1.99
	0.75	0.1897	2481425.75	0.425585	3351.60	2.11
	1.00	0.1897	2798673.15	0.237215	3342.10	2.35
1.25	0.00	0.4444	2371406.75	0.947345	2450.00	1.07
	0.25	0.1960	2414521.68	0.784261	2937.65	1.88
	0.50	0.1756	2453739.93	0.599439	3181.90	2.00
	0.75	0.1686	2481425.75	0.405794	3350.70	2.11
	1.00	0.1686	2820049.79	0.210827	3342.10	2.33
1.50	0.00	0.4000	2371406.75	0.947345	2450.00	1.07
	0.25	0.1779	2412668.83	0.778120	2909.45	1.88
	0.50	0.1601	2445860.85	0.588401	3147.00	1.98
	0.75	0.1518	2481425.75	0.389991	3349.90	2.10
	1.00	0.1518	2796695.45	0.189757	3342.05	2.33
1.75	0.00	0.3636	2371406.75	0.947345	2450.00	1.07
	0.25	0.1633	2410733.85	0.773042	2887.75	1.86
	0.50	0.1471	2441146.55	0.579248	3124.80	1.98
	0.75	0.1380	2481396.40	0.377045	3346.75	2.14
	1.00	0.1380	2799264.54	0.172495	3342.25	2.34

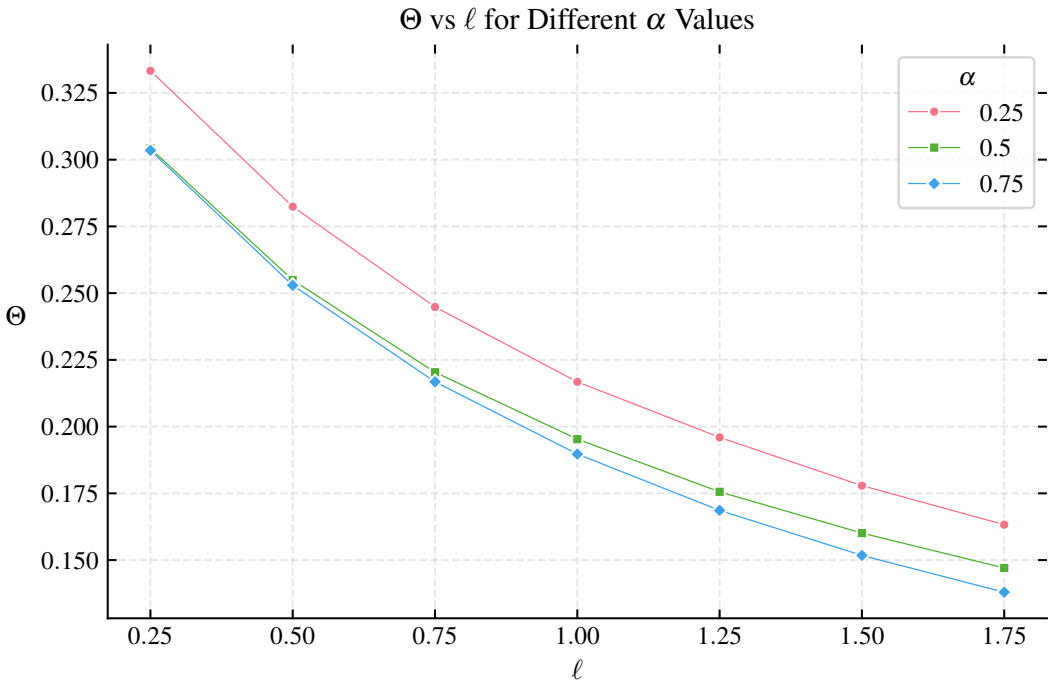


Figure 6.12: Θ vs λ for Experiment 2

6.2.3 Results for Experiment 3

In Experiment **E3**, we analyze the energy–congestion trade-offs for varying the hop limit $K \in \{1, 2, 3, 4, 5\}$. In Table 6.3 we give detailed solution results. Figs 6.13, 6.14 and 6.13 show the energy and congestion trade-offs with changes over K and α respectively. Increasing K improves the model's routing flexibility, resulting in lower congestion without significant energy penalties. We find that allowing just one intermediate node (i.e., $K = 2$) is sufficient to capture most of the gains from segment routing. This suggests that limited path detouring offers a practical and effective mechanism for reducing congestion. Also note that while increasing the segment limit increases flexibility, it also introduces more processing costs as a result, taking shorter paths (in terms of link length) might cost more energy as it requires more segments. Compared to smaller networks, the difference between $K = 2$ and $K \geq 3$ is even smaller. Moreover, we observe that going from $K = 2$ to $K = 3$ increases the energy consumption when $\alpha = 0.5$.

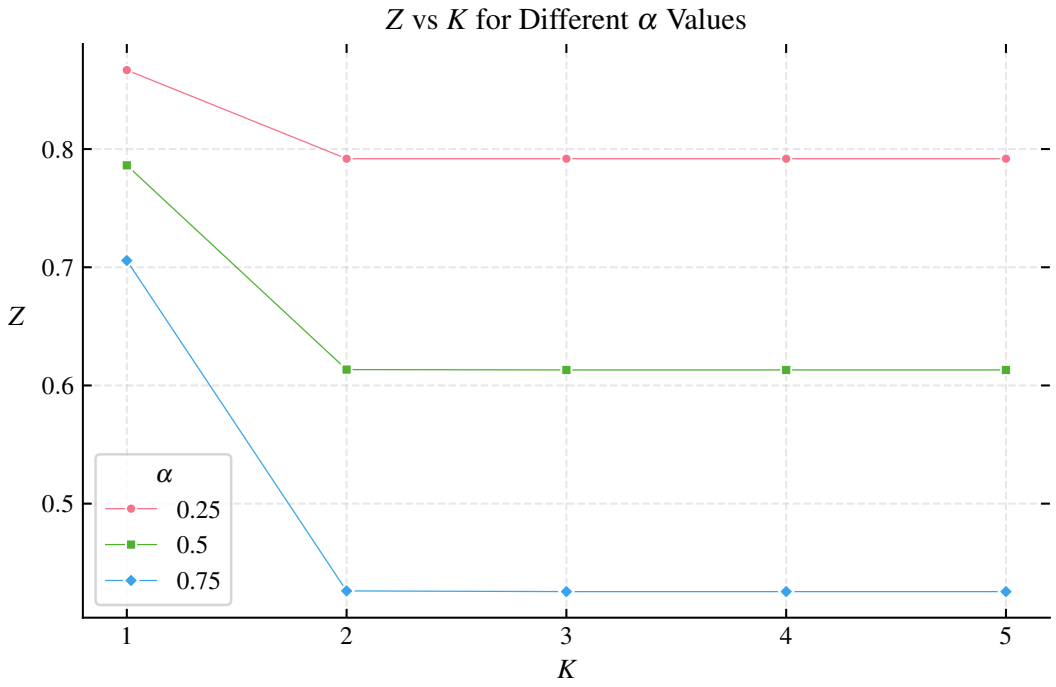


Figure 6.13: Z vs λ for Experiment 3

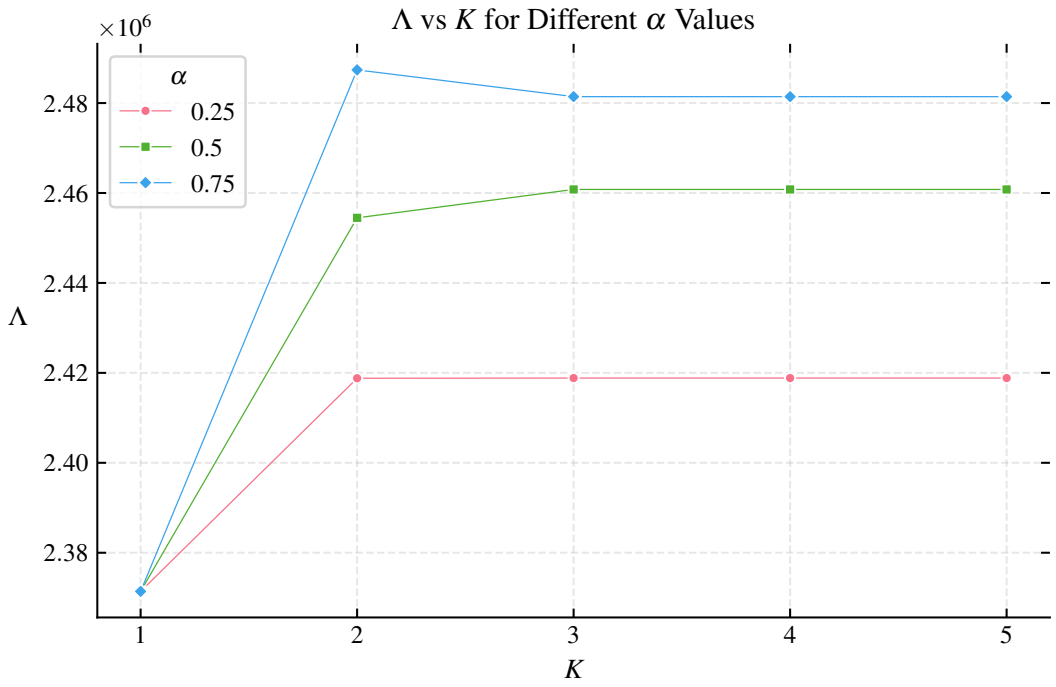


Figure 6.14: Λ vs λ for Experiment 3

Table 6.3: Results of Experiment **E2** for the *Germany50* topology. We omit values that do not change with changes in λ. $\Theta_{\text{MCF}} = 0.1897$, $\Theta^N = 0.7997$, $\Lambda^N = 2504877.1$.

K	α	Θ	Λ	Z	No. Col.	Time
1.00	0.00	0.5000	2371406.75	0.9473	2450.00	0.98
	0.25	0.5000	2371406.75	0.8668	2450.00	0.96
	0.50	0.5000	2371406.75	0.7863	2450.00	0.96
	0.75	0.5000	2371406.75	0.7057	2450.00	0.96
	1.00	0.5000	2371406.75	0.6252	2450.00	0.95
2.00	0.00	0.5000	2371406.75	0.9473	2450.00	1.08
	0.25	0.2168	2418801.85	0.7919	2961.75	1.83
	0.50	0.1977	2454475.10	0.6134	3184.55	1.91
	0.75	0.1897	2487372.71	0.4262	3324.80	2.01
	1.00	0.1897	2709291.00	0.2372	3326.20	2.17
3.00	0.00	0.5000	2371406.75	0.9473	2450.00	1.07
	0.25	0.2168	2418846.30	0.7919	2965.40	1.90
	0.50	0.1953	2460788.40	0.6131	3210.30	1.99
	0.75	0.1897	2481425.75	0.4256	3351.60	2.11
	1.00	0.1897	2798673.15	0.2372	3342.10	2.35
4.00	0.00	0.5000	2371406.75	0.9473	2450.00	1.07
	0.25	0.2168	2418846.30	0.7919	2965.40	1.91
	0.50	0.1953	2460788.45	0.6131	3210.10	2.01
	0.75	0.1897	2481425.31	0.4256	3351.20	2.10
	1.00	0.1897	2796601.12	0.2372	3361.10	2.31
5.00	0.00	0.5000	2371406.75	0.9473	2450.00	1.09
	0.25	0.2168	2418846.30	0.7919	2965.40	1.91
	0.50	0.1953	2460788.45	0.6131	3210.10	2.00
	0.75	0.1897	2481425.31	0.4256	3351.20	2.07
	1.00	0.1897	2796601.12	0.2372	3361.10	2.32

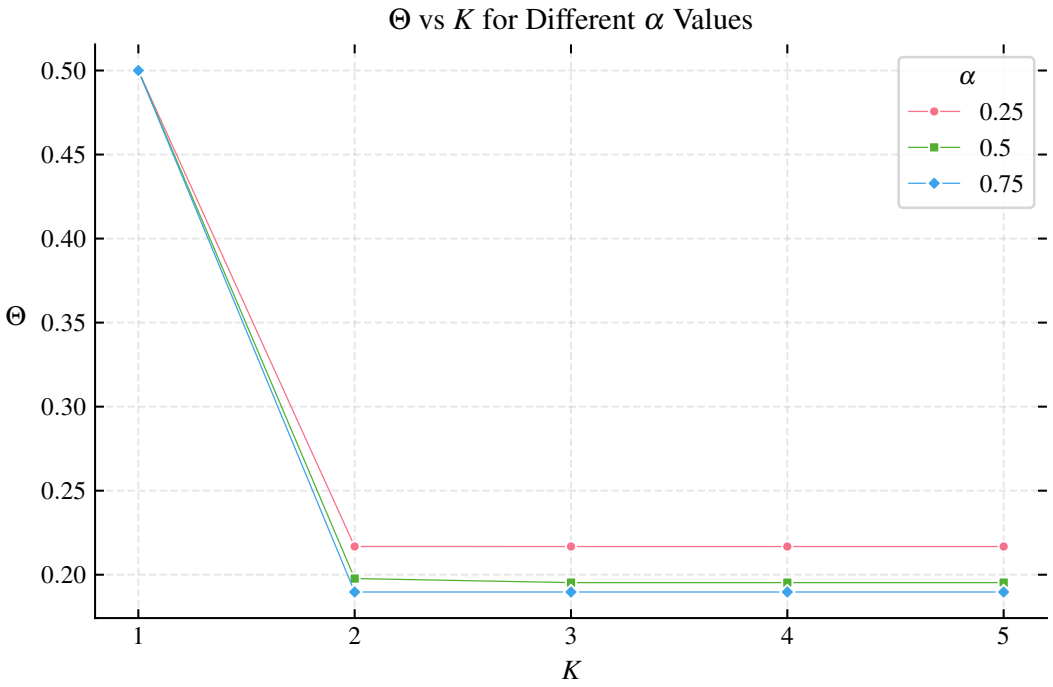


Figure 6.15: Θ vs λ for Experiment 3

Chapter 7

Conclusion

In this study, we set out to minimize the internet networks' maximum link utilization and energy consumption via segment routing. For this purpose, we developed two models, a compact flow-based model and a path-based model utilizing column generation. While our compact model is competitive with the literature, our column generation model is faster and can solve extensive network instances to optimality. Moreover, we confirm through our single objective experiments on MLU that setting the hop limit to 3 is enough to leverage the capabilities of segment routing fully.

In our multi-objective experiments, we observed that the energy processing costs do not significantly impact the network congestion. While the network is incentivized to use fewer hops with higher energy processing costs, the consequences of using fewer hops increase energy usage marginally. On the other hand, changing the link capacities has an immense effect on both objectives. With higher link capacities, the model can choose shorter links and fewer hops, significantly reducing the energy consumption. However, with reduced link capacities, the path options are limited; thus, the flow is split onto more links and nodes, as a result, energy consumption is higher compared to the same congestion levels on higher capacity instances. Finally, the hop limit does not impact both objectives after $K = 3$ on almost all instances. Comparing $K = 2$ against $K = 3$,

the differences are minimal. At the same time, when $K = 1$, the solution space is significantly reduced, and we cannot generate more than one solution as the model uses the ECMP route for each demand.

Our findings show that it is possible to reduce the energy consumption of a network without drastically increasing the congestion levels. Moreover, in our experiments, we noticed that the solvers stop at dominated solutions if the energy variables are not in the objective. This issue can be prevented by introducing bounds or through lexicographic optimization. However, it is possible to find better solutions in terms of energy with very minimal weights ($0.25 \geq \alpha > 0$) assigned to energy consumption in the objective.

There are many avenues for future work to discover. Smart node selection and middle-point-based approaches could enhance our column generation. While current node selection methods are developed on the singular objective of minimizing MLU, the impact of adding the energy objective should be studied. Moreover, online algorithms allow the demands to be routed as they arrive in the network; our CG algorithm can be enhanced to become online. Finally, the cuts we have developed for our flow-based model can be added to many other flow-based models. However, it should be possible to add them to path-based models as a preprocessing step

Bibliography

[1] A. Farrel, "Overview and Principles of Internet Traffic Engineering." RFC 9522, Jan. 2024.

[2] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 182–221, 2021.

[3] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2015.

[4] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Predictive traffic engineering with 2-segment routing considering requirements of a carrier ip network," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 667–675, IEEE, 2017.

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

[6] G. Trimponias, Y. Xiao, H. Xu, X. Wu, and Y. Geng, "Centrality-based middlepoint selection for traffic engineering with segment routing," 2019.

[7] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "Cg4sr: Near optimal traffic engineering for segment routing with column generation," in

IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 1333–1341, 2019.

[8] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 464–486, 2019.

[9] D. Wu and L. Cui, "A comprehensive survey on segment routing traffic engineering," *Digital Communications and Networks*, vol. 9, no. 4, pp. 990–1008, 2023.

[10] X. Li and K. L. Yeung, "Traffic engineering in segment routing networks using milp," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1941–1953, 2020.

[11] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic engineering using segment routing and considering requirements of a carrier ip network," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1851–1864, 2018.

[12] H. Roomi and S. Khorsandi, "Semi-oblivious segment routing with bounded traffic fluctuations," in *Electrical Engineering (ICEE), Iranian Conference on*, pp. 1670–1675, 2018.

[13] T. Settawatcharawanit, V. Suppakitpaisarn, S. Yamada, and Y. Ji, "Segment routed traffic engineering with bounded stretch in software-defined networks," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 477–480, IEEE, 2018.

[14] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, and J. Yang, "Traffic engineering in partially deployed segment routing over ipv6 network with deep reinforcement learning," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1573–1586, 2020.

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

[16] E. Moreno, A. Beghelli, and F. Cugini, "Traffic engineering in segment routing networks," *Computer Networks*, vol. 114, pp. 23–31, 2017.

[17] A. Cianfrani, M. Listanti, and M. Polverini, "Incremental deployment of segment routing into an isp network: A traffic engineering perspective," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3146–3160, 2017.

[18] R. Carpa, O. Glück, and L. Lefevre, "Segment routing based traffic engineering for energy efficient backbone networks," in *2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, pp. 1–6, 2014.

[19] K. S. Ghuman and A. Nayak, "Per-packet based energy aware segment routing approach for data center networks with sdn," in *2017 24th International Conference on Telecommunications (ICT)*, pp. 1–6, 2017.

[20] D. Otten, A. Brundiers, T. Schüller, and N. Aschenbruck, "Green segment routing for improved sustainability of backbone networks," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, pp. 1–9, 2023.

[21] X. Jia, Y. Jiang, Z. Guo, G. Shen, and L. Wang, "Intelligent path control for energy-saving in hybrid sdn networks," *Computer Networks*, vol. 131, pp. 65–76, 2018.

[22] C. Thaenchaikun, G. Jakllari, B. Paillassa, and W. Panichpattanakul, "Mitigate the load sharing of segment routing for sdn green traffic engineering," in *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 1–6, 2016.

[23] M. Pham, D. B. Hoang, and Z. Chaczko, "Congestion-aware and energy-aware virtual network embedding," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 210–223, 2020.

[24] L. Wang, M. Wang, C. Lin, and Y. Zhang, "Accelerating traffic engineering optimization for segment routing: A recommendation perspective," *Computer Networks*, vol. 264, p. 111224, 2025.

[25] J. Zhang and C. Zhao, "Q-sr: An extensible optimization framework for segment routing," *Computer Networks*, vol. 200, p. 108517, 2021.

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

[27] F. Hao, M. Kodialam, and T. V. Lakshman, "Optimizing restoration with segment routing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, 2016.

[28] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Englewood Cliffs, N.J: Prentice Hall, 1993.

[29] A. Vishwanath, K. Hinton, R. W. A. Ayre, and R. S. Tucker, "Modeling energy consumption in high-capacity routers and switches," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 8, pp. 1524–1532, 2014.

[30] M. Gupta and S. Singh, "Greening of the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 19–26, 2003.

[31] N. Vasić and D. Kostić, "Energy-aware traffic engineering," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pp. 169–178, 2010.

[32] A. Brundiers, T. Schüller, and N. Aschenbruck, "Mind the paths you choose: Speeding up segment routing-based traffic engineering with path preprocessing," *Computer Communications*, vol. 238, p. 108156, 2025.

[33] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving segment routing problems with hybrid constraint programming techniques," in *Principles and Practice of Constraint Programming* (G. Pesant, ed.), (Cham), pp. 592–608, Springer International Publishing, 2015.

[34] M. Ehrgott, *Multicriteria optimization*. Springer, 2005.

[35] J. Desrosiers and M. E. Lübbecke, *A Primer in Column Generation*, pp. 1–32. Boston, MA: Springer US, 2005.

[36] S. Gay, P. Schaus, and S. Vissicchio, "Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017.

[37] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[38] H. Zeng, C. Huang, and A. Vukovic, "A novel fault detection and localization scheme for mesh all-optical networks based on monitoring-cycles," *Photonic Network Communications*, vol. 11, p. 277–286, May 2006.

[39] C. C. Robusto, "The cosine-haversine formula," *The American Mathematical Monthly*, vol. 64, no. 1, pp. 38–40, 1957.