Preprocessing for segment routing optimization

Hugo Callebaut, Jérôme de Boeck, Bernard Fortz

Preprocessing for segment routing optimisation

Hugo Callebaut[2,3] | **Jérôme De Boeck**[1,4] | **Bernard Fortz**[1,2,3]

[1]HEC - Management School of the University of Liège, Liège, Belgium

[2]Départment d'Informatique, Université libre de Bruxelles, Brussels, 1050, Belgium

[3]INOCS, INRIA, 59650 Villeneuve d'Ascq, France

[4]Decision Support & Operations Research, University of Fribourg, Fribourg, 1700, Switzerland

Correspondence
Bernard Fortz, HEC - Management School of the University of Liège, Rue Louvrex, 14, 4000 Liège, Belgium
Email: bernard.fortz@uliege.be

Funding information

In this article we introduce a preprocessing technique to solve the Segment Routing Traffic Engineering Problem optimally using significantly fewer computational resources than previously introduced methods. Segment routing is a recently developed interior gateway routing protocol to be used on top of existing protocols that introduces more flexibility in traffic engineering. In practice, segment routing allows to deviate traffic from its original path by specifying a list of intermediate nodes or links, called segments, to visit before going to its destination. The issue we tackle in this article is that the number of segment paths scales exponentially with the maximum number of segments allowed leading to scalability issues in mathematical formulations. This article introduces the notion of dominated segment paths, these are paths that can be eliminated from the solution space when searching for an optimal solution. We propose a dynamic programming algorithm eliminating dominated paths for any number of segments. Numerical results show that respectively 50%, 90% and 97% of paths are dominated when considering up to 2, 3 and 4 segments on benchmark network topologies.

KEYWORDS
segment routing, mixed integer linear programming, network optimisation, network flows, routing algorithms, graph theory, telecommunication networks, traffic engineering

1 | INTRODUCTION

Segment Routing (SR) is a routing protocol developed by the IETF [4] to address known limitations of traditional routing protocols in IP networks. Protocols such as Open Shortest Path First (OSPF) [14] and Intermediate System-Intermediate System (IS-IS) [11] use link weights as the metric for Shortest Path Routing (SPR). These protocols have a single predefined routing between each pair of source-destination nodes which is technically hard to change and thus offers little flexibility [18]. Segment routing offers the possibility to deviate from the shortest path by using detours in the form of nodes or links respectively called node segments and adjacency segments. Segment routing also benefits from the advantages of the underlying routing protocols such as scalability and even load balancing between paths of equal cost called Equal-Cost Multi-Path (ECMP) [10] as it is built upon the already existing protocols.

Traffic Engineering (TE) is the field that aims at improving the quality of service in an existing network. In traditional shortest path routing, this consists mainly in adjusting link weights based on traffic patterns to avoid congestion. This objective is usually achieved by minimising the Maximum Link Utilisation (MLU), where the utilisation of a link is defined as the flow traversing it divided by its capacity. We refer to [1] for a survey on TE with SPR protocols. The problem considered in this article is the Segment Routing Traffic Engineering Problem (SRTEP). In this problem we are given a network and demands represented by a traffic matrix. Our goal is to route all demands by choosing a unique segment path for each demand while minimising the MLU.

While both TE problems require finding a unique path between each pair of source-destination nodes, SR still offers much more flexibility than regular SPR and has better optimal MLU, given the OSPF weights are optimal. Indeed, using the optimal weights for the shortest path TE problem, SR should at least perform as well as SPR since using no detour for each source-destination pair would result in an acceptable solution for the SRTEP having the same routing and therefore the same utilisation on each link (and therefore MLU) as the SPR. On the other hand, SR can also perform worse than an optimal SPR with a bad weight initialisation. Finding an optimal set of weights for SPR and finding optimal segment paths given a set of weights are both NP-hard problems [5, 9]. In this article only the second problem is considered.

Formulations for the SRTEP suffer from scalability issues as any set of routers or links in the network can be considered as a segment path. Some of these paths contain loops or are equivalent to other paths. These paths can then be removed from the mathematical formulation of the problem when searching for an optimal routing strategy. This article introduces the notion of dominated segment paths, these are paths for which there exists a different segment path with a load smaller or equal on each link. Dominated paths can also be eliminated from the solution space of any SR formulation in a preprocessing step to address scalability issues. We propose a dynamic programming algorithm to generate non-dominated segment paths for any number of segments. Numerical results show that respectively 50%, 90%, and 97% of paths are dominated when considering 2, 3, and 4 segments on benchmark network topologies, significantly decreasing the number of paths to consider. A simple path-based formulation is then used to solve instances by considering only non-dominated segment paths. Optimal solutions are found in a significantly faster time than state-of-the-art methods.

This paper is organised as follows. Section 2 provides an introduction to the segment routing protocol and Section 3 discusses some works related to its optimisation. In Section 4 we define the SRTEP problem formally and introduce mathematical models used to solve the SRTEP optimally. In Section 5 we present the theory related to dominated paths, we describe the dynamic programming algorithm used to generate all non-dominated paths and introduce an efficient data structure for this purpose. In Section 6 we provide numerical results obtained using our method and compare them to other heuristics and optimal methods. Finally, Section 7 concludes this article and provides insights in further improvements for segment routing.

2 | SEGMENT ROUTING PROTOCOL

In the last decade, the dependence of many businesses on network performance and availability became increasingly relevant. Strict Service Level Agreements (SLA) in terms of packet loss, delay, jitter, and available bandwidth became key business differentiators. In a context where networks evolve towards application-centric platforms, operators now require more flexible, yet scalable, and simple to operate network architectures. The IETF (Internet Engineering Task Force) started works on standardising an architecture aimed at fulfilling these requirements, called Segment Routing (SR) [4].

SR is a source routing technique that allows a host or an edge router to steer a packet through the network by using a list of instructions called segments. SR requires extensions and modifications to existing protocols. In this article, we use SR on top of OSPF, meaning that we are able to deviate from the shortest path routing through the use of segments.

From an abstract point of view the source node adds a SR header to each packet sent. This header contains a list of segments and the top segment is the active one. Once the active segment has been executed, it is popped and the next segment becomes active [2, 4].

Two types of segments are commonly used:

- **Node segments** : These segments correspond to a node, the action associated is to forward the packet on the shortest path to the corresponding node.
- **Adjacency segments** : These segments correspond to a link, the action associated is to forward the packet on the corresponding link.

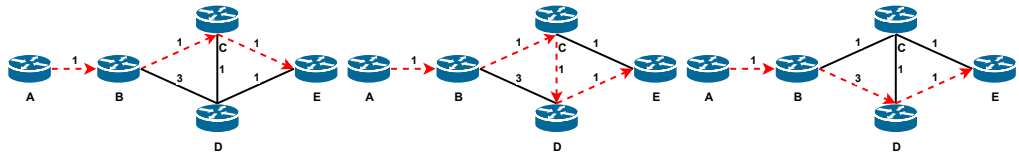
A technical difference between node and adjacency segments is that node segments are significant for the whole domain while adjacency segments are only locally significant to a node. This means that to forward a packet using an adjacency segment, the packet should already be at the starting node of the corresponding link. This can be the case either because that node is the source node or because the previous segment routed to that node.

We illustrate how segment routing works and can bypass the shortest path routing in Figure 1. We assume traffic goes from source node A to destination node E. The network is symmetric and the number on each link corresponds to the link weight. The red-dotted links are forwarding traffic and the black ones are idle. We also use the following notation A-AB-...-C-D to denote a segment path with source node A and segments AB, ..., C, D where AB is an adjacency segment corresponding to link AB and the other segment are node segments. Note that the first element (A in this case) is always a node and is technically not part of the segment path but is shown for clarification. We use standard shortest path routing in Figure 1a. At the source node, only one node segment is added corresponding to the destination node E.

In Figure 1b, we want to bypass the standard shortest path and push nodes E and D to the list. At the start, the active node segment is D and the packet is routed to D using SPR. At D the segment is popped, E becomes the next active segment and the packet is then routed to E using SPR.

In Figure 1c, we want to specifically use the link BD of cost 3. We cannot do this solely using node segments since the shortest path between B and D never uses that link. We then need three segments on the stack. The first one routing to B using SPR, the second segment corresponding to link BD, and finally, the segment corresponding to the egress node E.

In practice, all these routings could have been induced by changing the weights on the arcs. One might then wonder in which aspects SR differs from SPR. A first difference is that changing the SR paths dynamically does not



(a) SPR (Segment path A-E) (b) Segment path A-D-E (c) Segment path A-B-BD-E

FIGURE 1 Usage of segment routing to route traffic from A to E

introduce perturbations in the network unlike changing OSPF weights, but this is not our focus point as we focus on static optimisation of networks. The main gain from SR is that changing the routing of one demand does not influence the routing of another demand. Using SPR, when changing the weights such that we obtain the routing shown in Figure 1c starting with the routing in Figure 1a, the shortest path between B and E also changes, meaning that the routing from B to E changes too. This is not the case when using SR, which is why SR offers more flexibility than SPR.

With enough segments, any path could possibly be chosen to route traffic. In practice the maximum number of segments is limited. IPv4 can use a maximum number of three segments while the new IPv6 protocol offers the possibility to use up to ten segments [3]. We denote by k-SR the case where k is the maximum number of segments. 1-SR without adjacency segments is then equivalent to SPR and 2-SR allows for up to one detour. As explained earlier, adjacency segments are only locally relevant and there usually needs to be a node segment going to the right node immediately before the adjacency segment in the segment list. For this reason some studies consider adjacency segments as counting for two segments and then do not need to account for the locality of adjacency segments.

We use the full potential of k-SR, such that if a link is incident to the source node, no node segment is needed prior to it and if the last adjacency segment ends in the destination node, the destination node does not need to be added to the segment list. This is why 1-SR is different from SPR. Indeed using only node segments, only the shortest path can be chosen with 1-SR, but in the network shown in Figure 1, an adjacency segment could be used to route demands differently from SPR between node pairs BD and DB as the shortest path between these nodes does not go over the link directly connecting them.

Finally, note that in the SPR model we consider, we use Equal-Cost Multi-Path (ECMP) [10], a commonly used technique to split the traffic when multiple shortest paths exist. Some network operators choose their weights in such a way that shortest paths are unique but this constraint complicates the choice of weights and is less robust than using ECMP. In ECMP, the traffic is evenly split across all shortest paths using only local information. We illustrate this in Figure 2. Unit weights are used to compute the shortest paths. The numbers on the arcs are the resulting ratios of the flow on the arcs if data is transmitted from A to B. There are three different shortest paths going through nodes ACDB, ACFB and AEFB in order. Despite there being three shortest paths, the traffic is split in two even flows at node A because it only sees two different links on the shortest path. This illustrates the locality of ECMP. At node C, the traffic is then again evenly split on the different links and it recombines at node F resulting in 3/4 of the total load on arc FB. The fact that we use ECMP creates some special cases in the path domination criterion we use later in this paper.

3 | LITERATURE REVIEW

SR optimization faces challenging scalability and measurement issues. To the best of our knowledge, studies have yet to be able to address both of these issues. Most approaches for optimizing SR assume the traffic is known in

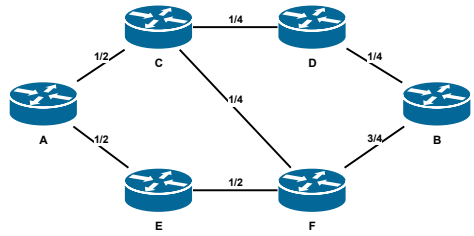


FIGURE 2 Distribution of load going from A to B using ECMP and hop count as weight metric

advance and consider one or multiple traffic matrices (TMs) over which the optimization is performed. The number of segments is generally also limited to two or three to limit scalability issues. Bhatia et al. [2] propose an exact optimisation method using a path formulation and linear programming for SR with two segments. They also propose an oblivious formulation to account for demand uncertainty and introduce an online routing method that determines the segments to use for incoming requests based on the real-time traffic over the network. A heuristic method is presented by Pereira et al. [16] for SR with three segments together with a method to deal with link failures. This paper is a rare case where adjacency segments are used, although not freely since the authors force to use one adjacency segment in each segment path. The only model proposed for an unlimited number of segments was proposed by Jadin et al. [12]. They tackle the problem with a column generation approach followed by a branch-and-bound procedure as a heuristic solution for SR. Hartert et al. [9] try to quantify the gain obtained by using SR compared to common routing protocols. They show that SR can decrease the congestion from 30% to 50%, assuming that the traffic is known and provide a Mixed Integer Linear Programming (MILP) based heuristic to compute results faster than using the model proposed in [2]. Gay et al. [6] proposed SRLS, a local search heuristic that tackles the problem of reacting quickly to sudden network changes. The main motivation for this heuristic is that previous work like [2] and [9] are based on MILP and suffer from high computation times while traffic spikes result in very temporary congestion and need very fast re-optimisation. Moreno et al. [13] question the efficiency of the routing protocol used between two different segments, namely OSPF combined with ECMP. They provide examples where ECMP can decrease the quality of segment routing by creating congestion when combined with SR. In [17], Schüller et al. propose an exact failure resilient MILP model that minimises MLU based on the model from [2], but this model has very high computational costs. The only study combining jointly the optimisation of segment paths and SPR weights is the work of Parham et al. [15]. They try to quantify to which extent combining jointly the optimisation of both metrics can improve the congestion rate of a network and provide a heuristic to do so. They considered multiple demand matrices with only one source and one destination node. For complete and recent surveys on segment routing we refer to Ventre et al. [19] and Wu and Cui [20].

Work has also been done to compare the quality of different algorithms proposed in the literature on benchmark instances. The testing framework REPETITA (Gay et al. [7]) contains over a hundred network topologies and allows users to integrate their routing algorithms for efficient comparison.

4 | PROBLEM DEFINITION AND MATHEMATICAL MODELS

In this section, we define the Segment Routing Traffic Engineering Problem (SRTEP) and present three different formulations proposed in the literature to solve the problem. Our main contribution, presented in Section 5, is a preprocessing algorithm used to solve the path model presented in Section 4.3 efficiently. The two other models are used

for benchmarking purposes.

4.1 | Problem definition

We are given a network represented as a capacitated graph $G(N, A)$ with node set N and directed arc set A, and a Traffic Matrix (TM) $\mathbf{D}$. The demands $\mathbf{D}(s, t)$ of the traffic matrix indicate the traffic that needs to be routed from source node s to destination node t for all $s, t \in N, s \neq t$. Moreover each arc a has a capacity c_a and a weight w_a. Since the weights w_a are already given and we chose OSPF with ECMP as underlying protocol, the shortest paths between each pair of nodes and the resulting load ratios when routing a demand from s to t can be computed. The goal of the SRTEP is to find a unique segment path between each pair of source-destination nodes that minimises the maximum link utilisation when routing all demands of the given traffic matrix on the network [8].

4.2 | Compact 2-SR formulation

A first compact formulation for 2-SR was proposed in Bhatia et al. [2]. We present here a slightly adapted version of this formulation that uses binary variables, as the initial formulation only solved the relaxation of the SRTEP. We use the notation $f_a^{(s,t)}$ to denote the load ratio on arc a when routing a demand from s to t. The following model can then be used to solve the SRTEP problem optimally using 2-SR with only node segments:

$$\text{min} \quad \alpha \tag{1}$$

$$\text{s.t.} \quad g_a^{(s,t)}(v) = f_a^{(s,v)} + f_a^{(v,t)} \qquad \forall s, t, v \in N;\ s \neq v;\ s \neq t;\ \forall a \in A \tag{2}$$

$$\sum_{v \in N} x^{(s,t)}(v) = 1 \qquad \forall s, t \in N \tag{3}$$

$$l_a = \sum_{(s,t) \in N \times N} \mathbf{D}(s, t) \sum_{v \in N} g_a^{(s,t)}(v) x^{(s,t)}(v) \qquad \forall a \in A \tag{4}$$

$$l_a / c_a \leq \alpha \qquad \forall a \in A \tag{5}$$

$$x^{(s,t)}(v) \in \{0, 1\} \qquad \forall s, t, v \in N;\ s \neq v;\ s \neq t \tag{6}$$

We minimise the MLU represented by variable α that is greater or equal to each link utilisation by (5). The utilisation of a link a is defined as its load l_a divided by its capacity c_a.

Variable $g_a^{(s,t)}(v)$ can be interpreted as an extension of $f_a^{(s,t)}$ to represent forwarding graphs for segment paths. It is defined as the load ratio on link a when routing traffic from s to t using intermediate node v. This variable is defined in equation (2). Note that when no intermediate segment is used, we consider an "artificial" segment with $v = t$.

$x^{(s,t)}(v)$ is a variable indicating if node v is used as intermediate node for traffic going from s to t. Equations (3) and (6) then force a unique path between each pair of source-destination nodes.

Finally in equation (4), we define the load on each arc using the values of the demand matrix.

4.3 | Path model

To extend SRTEP to use an arbitrary limit on the number of segments and both node and adjacency segments, a path formulation was proposed in Jadin et al. [12]. As this formulation does not scale well, it was only used so far

in a heuristic method based on column generation. In this paper, we propose a preprocessing technique to solve it exactly.

First, we introduce some definitions.

Definition Given a network G, we define $\mathcal{P}^k_{(s,t)}$ as the set of all possible segment paths of size at most k between s and t.

Definition Given a set of segment paths $\mathcal{P}^k_{(s,t)}$, we define a path $p \in \mathcal{P}^k_{(s,t)}$ as a sequence of segments (node or adjacency) $p_0 - p_1 - p_2 - \ldots - p_n$ where $n \leq k$, $p_0 = s$ and $p_n = t$ or p_n is an adjacency segment that ends in t.

Definition The length of a path $p \in \mathcal{P}^k_{(s,t)}$ noted $|p|$ is defined as the number of segments minus one in the sequence of segments that constitutes p (the minus one comes from the fact that the source node is technically not part of the segment list). This is also equal to n if we write the sequence p as $p_0 - p_1 - p_2 - \ldots - p_n$.

Definition We define the ratio of demand (s,t) on arc a for a path $p \in \mathcal{P}^k_{(s,t)}$ as $f^p_a = \sum_{i=1}^{|p|} f_a^{(p_{i-1},p_i)}$, which corresponds to adding all the ratios of subsequent segments.

The SRTEP can then be solved using the following model provided that we can enumerate all possible SR paths and enough computational resources are assigned to the task.

$$\min \quad \alpha \tag{7}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}^k_{(s,t)}} x_p = 1 \qquad \forall s,t \in N \tag{8}$$

$$\sum_{(s,t) \in N \times N} \mathbf{D}(s,t) \sum_{p \in \mathcal{P}^k_{(s,t)}} f^p_a x_p \leq c_a \alpha \qquad \forall a \in A \tag{9}$$

$$x_p \in \{0,1\} \qquad \forall p \in \mathcal{P}^k_{(s,t)};\ \forall s,t \in N \tag{10}$$

This formulation is similar to the previous one where $x_p, p \in \mathcal{P}^k_{(s,t)}$ indicates whether SR path p is used to route demands from s to t. We once again minimise the MLU in (7), we force a unique segment path between each pair of nodes in (8) and (10), and in (9) we define α to be greater or equal to the utilisation of each link.

4.4 | Segment model

A compact model for k-SR, based on a flow formulation, is the segment model proposed by Hartert et al. [8]. Contrary to the path model, this model allows to handle increasing SR path length limit k without increasing the spacial complexity of the model, but does not allow the use of adjacency segments. It also suffers from a very weak linear programming relaxation. In practice, despite its exponential behaviour, the path model remains smaller for 2-SR and both models are similar for 3-SR, independently of the network size, as shown in [8]. However, when considering $k \geq 4$, the segment model gets smaller than the path model. We present the segment model below.

$$\min \quad \alpha \tag{11}$$

$$\text{s.t.} \quad \sum_{u \in N} x_{s,u}^{(s,t)} = \sum_{u \in N} x_{u,t}^{(s,t)} = 1 \qquad \forall s,t \in N \tag{12}$$

$$\sum_{v \in N \setminus \{u\}} x_{v,u}^{(s,t)} = \sum_{v \in N \setminus \{u\}} x_{u,v}^{(s,t)} \qquad \forall s,t \in N;\ \forall u \in N \setminus \{s,t\} \tag{13}$$

$$\sum_{(s,t) \in N \times N} \mathbf{D}(s,t) \sum_{u,v \in N; u \neq v} f_a^{(u,v)} x_{u,v}^{(s,t)} \leq \alpha c_a \qquad \forall a \in A; \tag{14}$$

$$\sum_{u,v \in N; u \neq v} x_{u,v}^{(s,t)} \leq k \qquad \forall s,t \in N \tag{15}$$

$$x_{u,v}^{(s,t)} \in \{0,1\} \qquad \forall s,t,u,v \in N \tag{16}$$

$x_{u,v}^{(s,t)}$ is a binary variable that indicates whether demand (s,t) traverses the forwarding graph induced when routing traffic on the shortest path from u to v. This concretely means that segment path u-v is a subpath of the segment path used to route demand (s,t).

Constraints (12) and (13) are the flow conservation constraints, constraint (12) ensures that for each demand the flow outgoing (respectively incoming) from the source (respectively destination) node is equal to one. Constraint (13) ensures that at each intermediate node, the incoming flow is equal to the outgoing flow. Constraint (14) is similar to (9) in the path model and ensures that α is equal to the maximum utilisation. Constraint (15) sets the limit k on the number of node segments a SR path can use. Finally the domain constraint (16) ensures that a unique SR path is used.

5 | PREPROCESSING

Using only node segments, there would be a total of $|N|^{k+1}$ different segment paths for k-SR. When adding adjacency segments, this number would grow even more, which makes the path model quickly impractical. A possible approach to tackle this issue and solve the path model exactly could be to develop a full branch-and-price scheme extending the column generation approach of Jadin et al. [12]. But we observed that in practice, many segments paths are never present in an optimal solution and they can be detected easily. Instead of developing an ad-hoc algorithm, we developed a preprocessing algorithm that eliminates all these dominated paths and makes the formulation tractable with an off-the-shelf solver.

5.1 | Equivalent paths

First we need to distinguish between segment paths also called segment lists, and forwarding graphs. A segment path is a list of segments that induces a forwarding graph. A segment path states which segments are visited and in which order regardless of the underlying OSPF weights. A forwarding graph corresponds to the links used and their ratio of utilisation when using a certain routing. In our case, this routing is SR. Unlike a segment path, a forwarding graph, depends on the underlying protocol setting, in our case, the OSPF weight setting, as the links used for a particular routing might change with a different weight setting.

When considering the set $\mathcal{P}_{(s,t)}^k$, multiple different segment paths can result in the same forwarding graph. We illustrate this in Figure 3. In this figure, for traffic with source-destination nodes (S,T), all three segment paths S-C-T, S-D-T and S-C-D-T result in the same forwarding graph, and there are in fact even more than these three equivalent

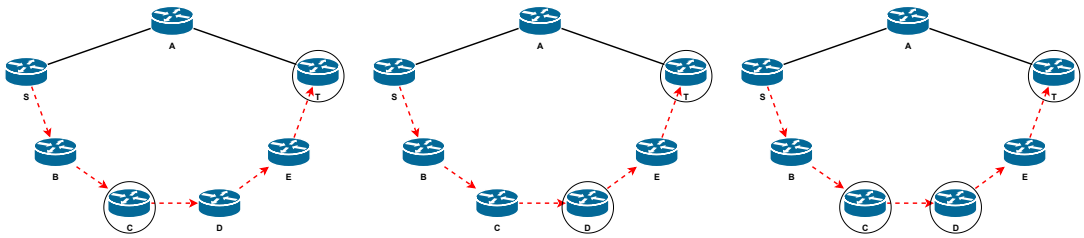


FIGURE 3 Different SR paths giving the same routing

segment paths.

When using the formulation from Section 4.3, we can see that no information about the segments present in the segment path is used, only information about the flow ratios induced by the forwarding graphs is used. In any solution to that problem, we could then replace a path by another equivalent path without changing the objective value and therefore the optimality of that solution. It is therefore not useful to consider all these equivalent paths in the model and is in fact even counterproductive as it increases the size of the model and solvers need more resources to find the optimal solution to the problem. When equivalent segment paths are found, only the first one is added to our model. Using our method to generate these paths ensures that the shortest one (in terms of number of segments) is kept.

5.2 | Dominated paths

It makes sense that lists in which a same node occurs multiple times should also not be used in the formulation of the problem as there would be a loop and removing this loop would result in a strictly better path. While it is easy to make a rule such as not to have the same segment twice in a list, there are other ways to create loops in the forwarding graphs and some of these paths should also be removed. For this we introduce the notion of dominated paths.

Definition Let $p, q \in \mathcal{P}^k_{(s,t)}$. p **dominates** q if

$$\forall a \in A : \sum_{i=1}^{|p|} f_a^{(p_{i-1},p_i)} \leq \sum_{i=1}^{|q|} f_a^{(q_{i-1},q_i)} \tag{17}$$

and

$$\exists a \in A : \sum_{i=1}^{|p|} f_a^{(p_{i-1},p_i)} < \sum_{i=1}^{|q|} f_a^{(q_{i-1},q_i)} \tag{18}$$

This definition means that for a path p to dominate another path q, they must have the same source and destination nodes. Secondly the load ratio of each link in the network when routing with path p must be smaller or equal to the load ratio of the same link when routing with path q and finally there must be at least one link where that load is strictly smaller. If only that last part was not the case, then both segment paths would be equivalent.

Using this definition of path domination, we can remove all dominated paths from the problem formulation since a dominated path present in the optimal solution could be replaced by the path dominating it while preserving feasibility and keeping the same optimal objective value. In general, replacing a dominated path by the dominating one can only decrease the maximum utilisation since there will be at least one edge with lower utilisation and the utilisation of

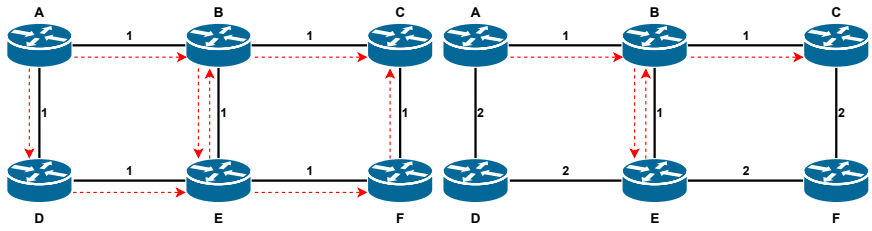


FIGURE 4 Segment path A-E-C is not dominated on the left but is dominated on the right despite both having a loop on arc BE/EB

the other edges will not change. Note that this notion of path domination is independent from the presence of loops in the general case. On the other hand, because of flow conservation, a dominated path must contain a loop. The reverse is not true as some paths containing (partial) loops because of ECMP can still be non-dominated and be part of an optimal solution. We show an example of this in Figure 4 where the numbers on the arcs indicate the weights used for the shortest paths.

In the network on the left in Figure 4, because of ECMP, each red dotted arrow carries a load ratio of 1/2 when using segment path A-E-C. We can see that on links BE and EB that a part of the traffic makes somewhat a loop and by removing all traffic on these links, the forwarding graph would be strictly better. Nevertheless, this segment path is not dominated and could be useful to find an optimal solution to the SRTEP. Indeed, this better forwarding graph cannot be induced using SR only and weight changes would be needed to obtain such a path. In the right network of Figure 4 on the other hand, segment path A-E-C is indeed dominated by segment path A-C and should be eliminated from the set of paths considered to find an optimal solution.

In practice, despite being strictly worse than A-C, segment path A-E-C could still occur in an optimal solution of the SRTEP if preprocessing was not used. This is due to the MLU objective function, because as we only care about the maximally utilised link, the link utilisation of the other links could become larger without changing the objective function. Such a case of a loop was in practice found in the optimal solution of the SRTEP on a benchmark topology. If preprocessing was used to remove these paths, this would not be the case without changing the objective function.

5.3 | Parallel links domination

Finally we show how some non-dominated segment paths can still be removed from the problem formulation while preserving an optimal solution. This particular case happens when there are multiple parallel links each with the same capacity and weight which are the only links on the shortest path between the two nodes connected by these links. The node segment from the first to the second node is a non-dominated path and the flow is split equally on each link. All adjacency segments corresponding to the links are also non-dominated paths, forcing the flow to follow this link only. In practice the adjacency segments could all be removed as using only the node segments and ECMP results in a perfectly even split which would be optimal in this particular setting. We prove this in the following.

Lemma 1 *Given two nodes A and B with multiple parallel links connecting them, if all the links have the same capacity and if the shortest path between A and B distributes the flow equally on all the parallel links and only them, using adjacency segments A-AB_i cannot improve the optimal MLU obtained using only node segments A-B, where each AB_i corresponds to a link between A and B.*

Proof First note that using a node segment instead of an adjacency segment never results in more segments used,

meaning that anything we route using an adjacency segment can also be routed using a node segment without needing to change the maximum number of segments. We remind that we are in a setting where there are multiple parallel links with the same weights and capacity and that no other path is on the shortest path between the two nodes connecting the links. Let us consider an optimal solution to the SRTEP in which one or more segment paths use one of these adjacency segments. Let n be the number of parallel links and x_i the load of each link for $i \in \{1, ..., n\}$. Suppose we replace all the adjacency segments by the corresponding node segments. In that case we obtain a maximum link utilisation for these parallel links of $\sum_{i=1}^{n} x_n/n$, which is smaller or equal to the previous MLU of $\max_{i \in \{1,...,n\}} x_i$. Note also that the rest of the network is unchanged, so if the most utilised link was not one of these parallel links, the MLU would not change, showing that adding paths using adjacency segments to go over these parallel links is not useful.

5.4 | Path generation

Generating all non-dominated paths can be performed using a dynamic programming algorithm. We start with a set of segment paths composed of all 1-SR paths. Next we add each possible segment at the end of each path generated in the previous iteration creating each time a new path. This new path is then compared to all the paths already generated to test for domination. If the path is not dominated, it is added to the set of non-dominated paths. Some paths can also dominate paths created earlier, in that case the earlier path should be removed from the set. This simple algorithm described in Algorithm 1 could also remove equivalent paths by removing constraint (18) in the definition of domination and the special case that can sometimes happen when using parallel links could also be handled when adding all 1-SR paths (in practice we did both of these optimisations).

Algorithm 1 Dynamic programming algorithm to generate non-dominated paths

```
L ← {}
add all 1-SR paths to L
for (i ← 2; i++; i ≤ k) do
    for each (i − 1)−SR path p do
        for each path p′ obtained by adding one segment at the end of p do
            dominated ← 0
            for each path p* with same origin-destination as p′ do
                if p* dominates p′ then
                    dominated ← 1
                    break
                else if p′ dominates p* then
                    remove p*
                end if
            end for
            if dominated = 0 then
                Add p′ to L
            end if
        end for
    end for
end for
```

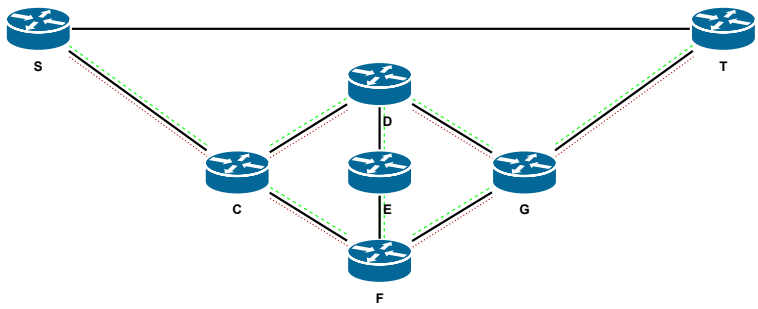


FIGURE 5 Network in which segment path S-E-T is dominated S-C-G-T which contains more segments

This algorithm can be improved by taking into account some properties of segment paths that also allow to develop an efficient data structure (described in the next section) for storing them. A first property is that paths can only be dominated by other paths having the same source-destination nodes. It is therefore of interest to group the paths by source-destination nodes.

Another improvement to the simple dynamic algorithm is to prune paths by using information about paths already generated. More precisely, it is not necessary to consider the addition of every possible segment at the end of a path already created: when trying to add a new segment D to segment path A-B-...-C, if segment path B-...-C-D was not generated in previous iterations, then B-...-C-D is dominated and adding segment D to A-B-...-C would also lead to a dominated path. Hence it can be skipped from the enumeration.

5.5 | Longer path domination

One should observe that 1-SR paths can never be dominated, and it is quite easy to see that in the case of unique shortest paths (without using ECMP), a path can never be dominated by a longer path. Unfortunately, this is not the case when ECMP is used. We show such a case in Figure 5. In this figure we consider unit weights and 2-SR path S-E-T. The flow resulting from using this path is indicated in dashed green lines. This 2-SR path is not dominated by any other 1-SR or 2-SR path but is dominated by the 3-SR path S-C-G-T for which the flow is indicated in red dotted lines.

Because of the data structure described in Section 5.6, this poses a problem as we need to keep these dominated paths in memory. Fortunately, this special case occurs very rarely in practice and the number of dominated paths kept by the algorithm is very small. Out of 257 topologies, only 8 of them had 2-SR paths dominated by a longer 3-SR path. The number of dominated paths kept in the end because of this special case ranged from 0.001% to 0.026%. Since testing for domination between two paths can easily be done in both ways at the same time (i.e. testing if p dominates p' and if p' dominates p), these special dominated paths are computed during the preprocessing step. They are then kept in the data structure, but a boolean value is associated to these paths to indicate that they should not be used in the linear program.

5.6 | Data structure representation

Thanks to the properties described in the previous section, we can generate all non-dominated paths more efficiently by using a tailored data structure for managing paths. We use a prefix tree in which the root node corresponds to the

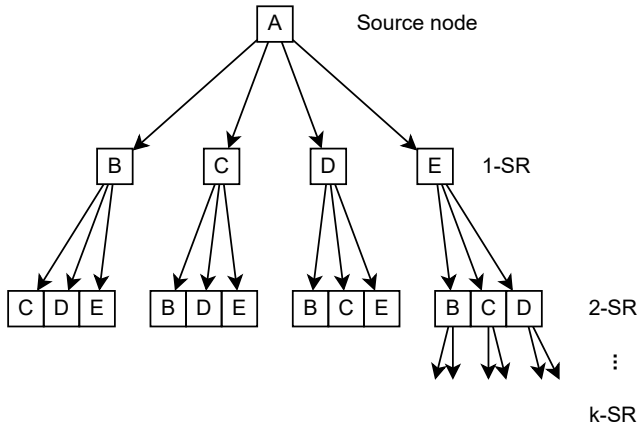


FIGURE 6 Tree representation of paths with source node A

source node of a segment path and each subsequent node in the tree corresponds to a segment. Each path in the tree from the source node to any other node also corresponds to a segment path consisting of all segments encountered on the path. We show the representation of such a tree in Figure 6.

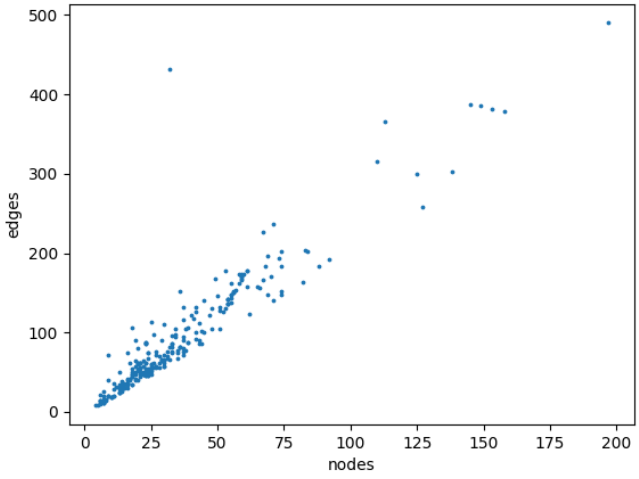
This representation makes searching for an existing path straightforward and the pruning step can therefore be done efficiently. In practice instead of trying to add each segment to a segment path p = A-B-...-C, we look at the children of the path B-...-C, only add them and test the newly created paths for domination. On the other hand, it has the disadvantage that a path dominated by another longer path cannot be removed from the data structure as the shorter dominated path could still be a subpath of another non-dominated path and therefore needs to be kept for the pruning step. This case can happen because of the limit set on the number of segments. If a segment path p is dominated by a longer path p', if p' is of size k (which is the maximum number of segments), path p with a new segment at the end of it could still be non-dominated because the path dominating it would be of size $k+1$. Fortunately, as previously said, this special case occurs very rarely in practice.

Some additional data is also stored at each node such as the flows on each arc for the segment path the node corresponds to. These flows are stored as sparse lists as storing the flow on each link can create memory problems on topologies with many edges. Storing a sparse list for each path on the other hand is not an issue since the information stored is needed in the mathematical formulation anyway which is the most demanding part in memory and computational time.

6 | NUMERICAL RESULTS

The method described in this article was implemented using Java 11.0.2 and Gurobi 9.1.2. All computational experiments were run using 62 GB of RAM and up to 8 cores in parallel running at 2.60 GHz. The Gurobi parameters were kept to their default values, meaning the tolerance is 10^{-4} or 0.01%.

Finally a time limit of 30 minutes was also set for every experiment. Since the preprocessing step of generating all non-dominated paths can be done only once per topology independently of the demand file, a timeout of 30 minutes was set on the preprocessing step and also on the model solving. This means that, in theory, an instance could use up to 60 minutes of total time. In practice, on the most challenging instances that were solved optimally, only around



average nodes	36.38
average edges	94.42
maximum nodes	197
maximum edges	490

FIGURE 7 All 260 topologies of REPETITA used represented as dots on the graph with the number of nodes on the x-axis and number of arcs on the y-axis.

10% of the time was used on the preprocessing part. For fairness in the comparisons, we consider that an instance using more than 30 minutes in total (i.e. preprocessing + solving time) failed due to a timeout even if the optimal solution was found later.

6.1 | Data

The benchmark instances used all come from REPETITA [7]. We used 260 different topologies all with initial weights set to the inverse capacity of the links. Each topology also comes with 5 different demand files which are created in such a way that an optimal routing would have around 90% maximum link utilisation. Figure 7 shows the topologies' sizes. Each point on the graph corresponds to a topology, on the x-axis, we denote the number of vertices, and on the y-axis, we denote the number of directed arcs.

6.2 | Results

In this section we conduct experiments to assess the performance of the method we introduced. We use the following notations for the different models used for benchmarking.

- **SRPP** is the method we introduced throughout this article and stands for Segment Routing with Pre-Processing.
- **Full** is the method in which we solve the path model for SRTEP without preprocessing the dominated paths.
- **CG4SR** is the column generation heuristic introduced in [12], we did not use adjacency segments for this method.
- **MIP** corresponds to solving the SRTEP problem using the segment model 4.4.

We did not reimplement the CG4SR and MIP methods but rather used the implementations from REPETITA which also use Java and Gurobi.

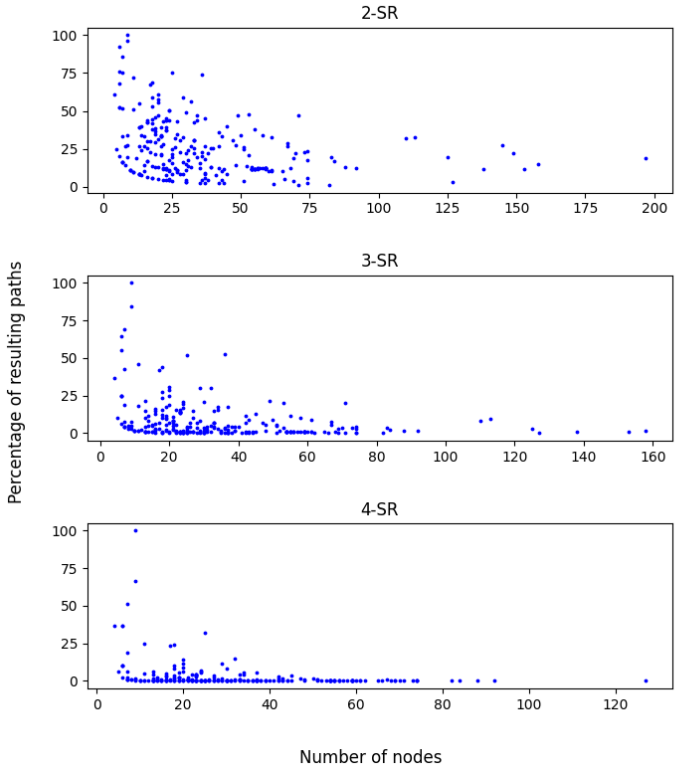


FIGURE 8 Percentage of resulting paths using only node segments after preprocessing w.r.t. the size of the network

6.3 | Path domination

In this section we analyse the number of paths our preprocessing step manages to remove from the formulation. If this number was small, our preprocessing step would likely not be useful and it would be better to solve the model with all paths without going through the preprocessing step. Figure 8 shows the percentage of paths left after preprocessing for each topology on the y-axis for 2-SR, 3-SR and 4-SR using only node segments. On the x-axis we show the number of nodes for each topology, one should see that the bounds (on the x-axis) are not the same on the three graphs; this is because the most difficult topologies could sometimes not even be preprocessed within the 30 minutes allowed.

The results show that the proportion of paths removed grows with the number of segments allowed. This could be expected since removing a path also removes all subsequent paths in which it is a subpath. Also note that the topology for which there is each time 100% of the resulting paths is a complete graph on 9 vertices with all links having the same weight and capacity. In this graph it is then impossible to remove any segment path when using only node segments.

When using adjacency segments, even more segment paths can be removed since, most of the time, these adjacency segments are equivalent to the corresponding node segments. In practice, only for 57 out of the 260 topologies does adding adjacency segments add new (potentially) useful paths. For the 203 other topologies, all paths using adjacency segments were removed during the preprocessing step because all paths using adjacency segments were

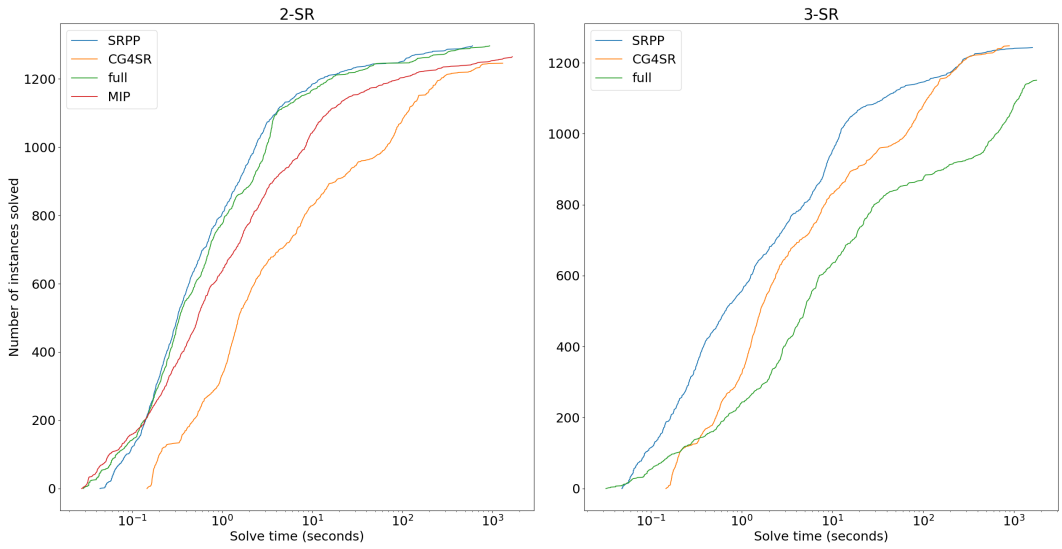


FIGURE 9 Cumulative distribution plot of the number of instances solved within a certain time

dominated by another path using only node segments. This could be expected since usually the shortest path between two adjacent nodes is the one using the link connecting them, making the use of adjacency segments useless. This shows that considering adjacency segments does not contribute much to the difficulty of the problem when preprocessing is used to remove segment paths that can not improve the optimal solution.

6.4 | Computation time

In this section we compare the computation times of the different techniques. We compare the method with and without preprocessing, CG4SR and MIP on 2-SR and 3-SR. No method uses adjacency segments here, but using them gives practically the same results in the case of preprocessing. The implementation of MIP in REPETITA did not allow to change the number of segments parametrically. Therefore only 2-SR results are obtained with this method.

We show the results obtained in Figure 9. This figure shows the cumulative distribution plot of the number of topologies solved with respect to the computation time. On the x-axis we show the computation time (in seconds on a logarithmic scale) and on the y-axis we show how many instances were solved optimally within the given time.

We can see that using preprocessing generally gives the best results although when using only 2-SR, the difference with or without the use of preprocessing is quite small. When using 3 segments on the other hand, the difference in computation time is much more significant. The MIP model from REPETITA performs quite poorly because, for 2-SR, this model is larger than the path model and it also has a much worse linear relaxation than the path model. CG4SR which is a heuristic, obtains poor results when only 2-SR is used, but this is somewhat compensated by the fact that the method scales very well.

Using 2-SR, two instances were not solved optimally within the 30 minutes allowed for the methods with and without preprocessing. The two instances were from the topology Nsfnet which is surprisingly very small (13 nodes, 30 edges). Moreover for the three other instances of the same topology, an optimal solution was found within one second. It is then likely that the solution returned is within the 10^{-4} optimality range but that Gurobi needs to evaluate

Method	Average MLU
Pre optimisation	1.286924312613915
TabuIGPWO	0.9545468916114692
CG4SR (2 segments no adjacency)	0.9512004395974866
CG4SR (3 segments no adjacency)	0.9481075131403686
2-SR (no adjacency)	0.9368902632031229
2-SR (unary weights)	0.9350733300498427
3-SR (no adjacency)	0.9344596237112538
2-SR	0.9276537128719935
3-SR	0.9252219043817592
3-SR (unary weights)	0.9214282906624273
TabuIGPWO + 2-SR	0.9083883109525657
TabuIGPWO + 3-SR	0.9071847608563889

TABLE 1 Comparison in maximum link utilisation between different methods for minimising the maximum MLU on a network (1222 instances).

more solutions to assess the (near) optimality of the solution currently found. For the segment model, 34 instances were not solved optimally (including the two problematic previous instances). Finally, CG4SR did not manage to solve 53 instances for 2-SR but succeeded in solving the two problematic instances.

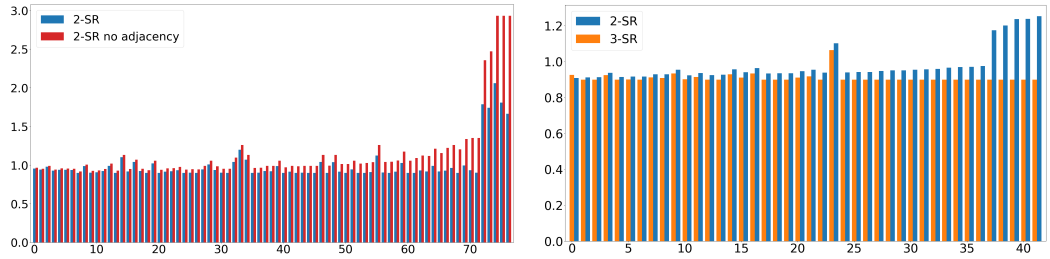
For 3-SR, CG4SR did not find a solution for 53 instances. The model with preprocessing did not find an optimal solution for 55 instances and, without preprocessing, it did not find an optimal solution for 148 instances. This shows that both CG4SR and the model with preprocessing scale much better than regular methods. We expect CG4SR to outperform by far SRPP when used for larger k, but we do remind that CG4SR is a heuristic and does not necessarily provide optimal solutions. Moreover, Hartert [8] showed that most of the benefits of using SR can be obtained with small values of k.

6.5 | Optimality

In this section we analyse the results obtained using the different techniques with respect to the optimal MLU obtained for each instance. This analysis shows how useful adjacency segments are, how much improvement we can expect from using 3 segments instead of 2 and finally how well or bad the CG4SR heuristic is compared to an optimal method.

We first start by giving a summary of the results obtained using each method in Table 1. Unless otherwise specified, the initial weight setting used was the inverse capacity weight setting and adjacency segments were used too. All results for SR were obtained using the preprocessing technique and the average is computed using only 1222 out of the 1300 instances. As for the 78 other instances, at least one method did not get optimal results within the given time limit.

We can first observe that all methods using SR yield better results on average than simply optimising the weights using the TabuIGPWO method and not considering SR. This shows that SR should effectively be able to reduce con-



(a) Comparison between 2-SR with and without adjacency segments

(b) Comparison between 2-SR and 3-SR

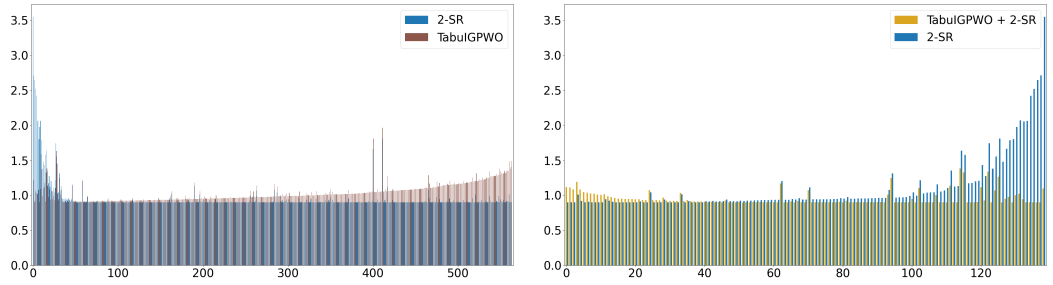
FIGURE 10 Comparison in optimal MLU when using adjacency segments or more segments

gestion on networks. Secondly CG4SR which is a heuristic performs as expected on average worse than any other method even when we compare this heuristic using 3 segments to an optimal solution using only 2 segments. Other results that were expected are that using adjacency segments improves the optimal MLU and the same can be said about using more segments. Finally the last two methods correspond to first optimising the IGP weights using the TabuIGPWO method and then performing SR on that new network. These methods prove to be very successful as they are on average within 1% of the optimal MLU that could be obtained using general routing (the optimal MLU for the general routing problem should be of 0.9 for each instance). We do mention that the computation time allowed for these last two methods was longer. This is because the TabuIGPWO method always used the maximum time allowed (i.e. 30 minutes) to obtain a result. After that another 30 minutes of computation time was allowed to find the optimal SR solution to the new network meaning the total computation time could be over 1 hour. Only instances where the second part (the SR optimisation) was over 30 minutes were discarded as the solution could not be optimal. On the other hand, giving more computation time to the SR and CG4SR methods would not give better results as the MLU returned is already optimal for the problem considered.

A drawback of the matrix representation of the average MLU in Table 1 is that only the average over 1222 instances can be shown. In the following we compare two methods side by side to see in more detail how these differences in MLU occur. Since plotting about 1300 instances in a graph becomes quite difficult to read, only instances where the difference in MLU is higher than 1% are shown in the following plots. Once again all instances where one of the two compared methods use over 30 minutes are discarded. In Figure 10 we compare 2-SR with 3-SR and 2-SR using adjacency segments with 2-SR not using adjacency segments.

We now explain how to interpret these plots. Each bar corresponds to the MLU obtained for an instance. The pairs of adjacent bars each correspond to the same instance using the method with the corresponding color, they are ordered by increasing difference in MLU. A method that has a bar which is higher than the corresponding bar for the other method is then worse for that instance since it means that the MLU is higher. The more significant the difference in height between the bars, the worse one method is compared to the other one. The number of instances reported in the plot (on the x-axis) also has an impact on the interpretation of the results: Since all instances for which the difference in MLU is less than 1% are removed, plots with few instances imply that both methods generally give similar results. In contrast a large number of instances means that the solution obtained is often very different for the two methods.

When we compare Figures 10a and 10b we see that, as expected, using adjacency segments always gives better results than not using them and using 3-SR is also always better than using 2-SR. In fact there is one instance where



(a) Comparison between 2-SR and TabuIGPWO

(b) Comparison between 2-SR and TabuIGPWO followed by 2-SR

FIGURE 11 Comparison in optimal MLU between using adjacency segments or more segments

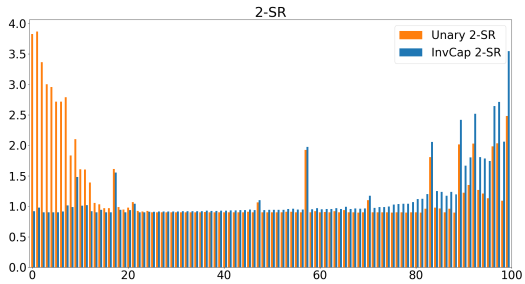
3-SR performs more than 1% worse than 2-SR as shown in the plot, this is because Gurobi has multiple optimality criteria and that they can sometimes interfere with the optimality tolerance which should be 0.01% in our case. We can then see that using 3-SR instead of 2-SR is less often useful than using adjacency segments as it improves the MLU of about 40 instances whereas using adjacency segments improves the MLU for over 70 instances. The improvement in MLU is often also more significant when using adjacency segments than when using one more segment. This is consistent with the results from Table 1 where we can see that 2-SR with adjacency segments performs on average better than 3-SR without adjacency segments.

Next we compare the TabuIGPWO method and the combination of TabuIGPWO and 2-SR against 2-SR in Figure 11. In Figure 11a we can see that 2-SR often largely outperforms TabuIGPWO but that there are some instances where it performs much worse likely because of a poor initial weight setting. This is the reason why we tested the iterative method consisting in optimising the IGP weights first and then applying SR on these new weights. The results for this method are shown in Figure 11b and are compared with 2-SR. As expected, the instances where a poor weight setting resulted in a poor MLU are not problematic anymore but for about 20 instances the optimal MLU increased. This shows that to obtain optimal results, the joint optimisation of IGP weights and SR paths should be performed. Although from Table 1, we know that on average we should expect less than a 1% decrease in optimal MLU for an optimal routing compared to this sequential method.

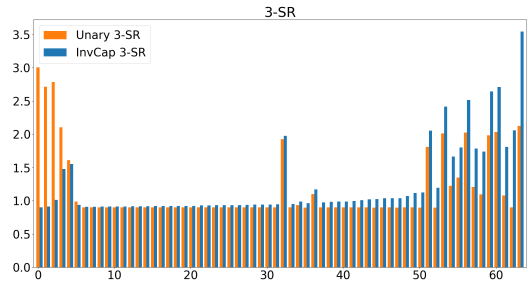
Finally, in Figure 12, we compare the results obtained with unary and inverse capacity weight settings. Table 1 showed that using 2-SR, inverse capacity weights worked better, but using 3-SR, unary weights obtained better results. This is indeed what we can observe on these two graphs, using only 2-SR, the unary weight setting has a few instances where the optimal MLU is far above the one obtained with invCap weight setting. When adding one more segment, the number of these instances gets much smaller. This can also be observed on the right side of the graphs where invCap performs worse, but the decrease is here less noticeable. This result was in fact already obtained in [8], they noticed that unary weights scaled better than invCap weights with the number of segments, although this observation was only done on the relaxation of the SRTEP (using continuous variables instead of binary for the paths).

6.6 | Optimal path length

While the path domination criterion ensures that an optimal solution is found, provided that the resulting MILP is solved optimally, this criterion still leaves many paths in the formulation. Generally speaking one can expect that an



(a) Comparison between unary and invCap weight setting for 2-SR



(b) Comparison between unary and invCap weight setting for 3-SR

FIGURE 12 Comparison in optimal MLU between unary and inverse capacity weight setting for SR

optimal SR-path should not be much longer than the shortest path connecting both nodes.

After having computed all optimal paths for each topology, we compared the length of the optimal SR-paths to the length of the shortest path between the origin and destination nodes. Figure 13 represents this in the form of a heat map for 2-SR and Figure 14 as boxplots for 2- and 3-SR. In Figure 13, the number on the y-axis corresponds to the length of the shortest path and the x-axis corresponds to the length of an optimal SR-path. The way this heat map is to be interpreted is the following: for each row the color corresponding to a square (i, j) corresponds to the proportion node pairs with a non-zero demand that have a shortest path length equal to i and for which an optimal SR-path had length equal to j (the cyan numbers in each square correspond to the same data in absolute values). For example, square (1, 43) tells us that out of 116284 node pairs between which the shortest path distance was equal to one and that had a non-zero demand, in an optimal solution, there were 34 SR-paths which had a distance equal to 43 (i.e. 0.037%, notice that the color follows a logarithmic scale). The second image shows the same data in the form of a boxplot for 2- and 3-SR in blue and orange respectively. Please note that for Figure 14, the y-axis and x-axis are inverted compared to Figure 13 for improved readability.

From these images we can see that very long paths are quite often used. After some tests on a few topologies which used very long paths we noticed that we were able to obtain each time the same optimal solution (in terms of MLU) while heavily limiting the length of the available SR-paths.

Therefore, we added a new criterion to our domination criterion to remove these very long paths. Assuming that x is the distance of the shortest path between the origin and destination nodes, we consider that an SR-path is dominated if its length is longer than $ax + b$ where a and b are parameters. If these parameters are chosen carefully we should be able to obtain (near) optimal solutions while generating less paths. This should make the preprocessing step and solving the MILP faster, resulting in faster computations.

First, we mention that these tests were conducted on the unary topologies only as the value of b is then well-defined. Since all edge weights are equal to 1, each unit of b corresponds to one more edge that can be taken in a path, on the other hand for inverse capacity weights, the value of b would need to be dependent on the topology since the edge weights can heavily vary between different topologies and even within one topology. Our initial test used as values $a = 2$ and $b = 1$ meaning that an SR-path that is longer than 2 times the shortest path +1 would be removed. While the results were satisfying, we can easily see in Figure 14 that a factor 2 is already overkill for shortest paths of length greater than 24 since no pair of nodes for which the shortest path between them is greater than 24 uses SR-paths of length over 48. On the other hand, some topologies did not achieve optimal results, so we wanted to

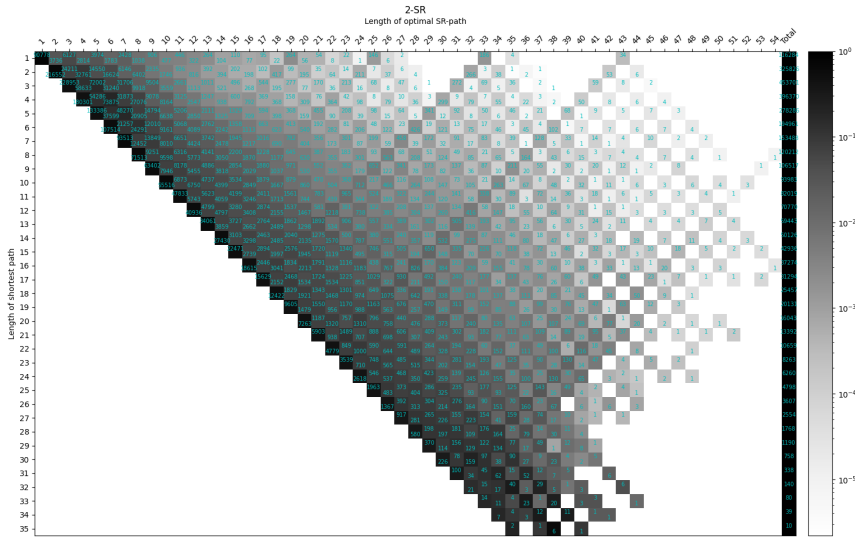


FIGURE 13 Heatmap of lengths of optimal SR-paths compared to the length of the shortest path

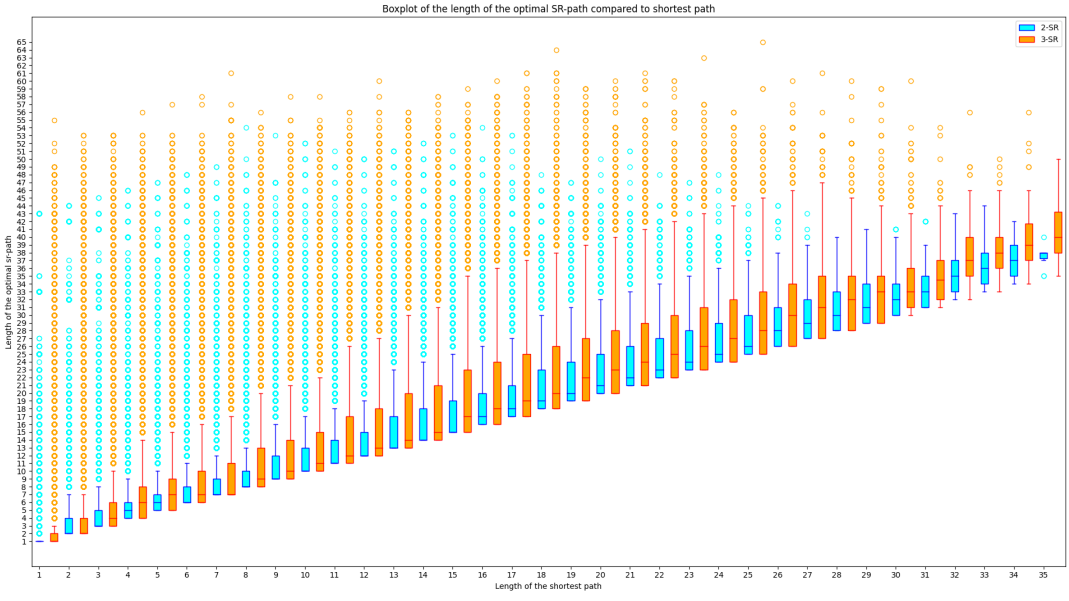
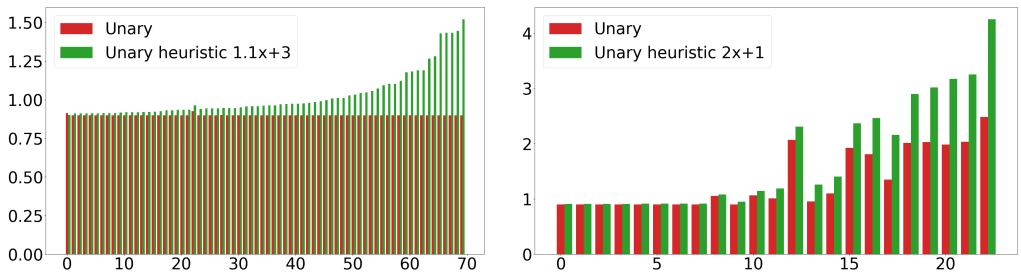


FIGURE 14 Length of optimal SR-paths compared to the length of the shortest path

2-SR Method	Average MLU (1300 instances)	3-SR Method	Average MLU (1233 instances)
$2x + 1$	0.9405958841678318	$2x + 1$	0.9294954247428531
$1.1x + 3$	0.9401210890246896	$1.1x + 3$	0.9265071360502246
2-SR	0.9333870935280182	3-SR	0.9216284140454601

TABLE 2 Comparison in maximum link utilisation between different methods for minimising the maximum MLU on a network for 2- and 3-SR.



(a) Comparison between optimal unary and $1.1x + 3$ heuristic for 2-SR

(b) Comparison between optimal unary and $2x + 1$ heuristic for 2-SR

FIGURE 15 Comparison in optimal MLU between optimal 2-SR and $ax + b$ heuristics

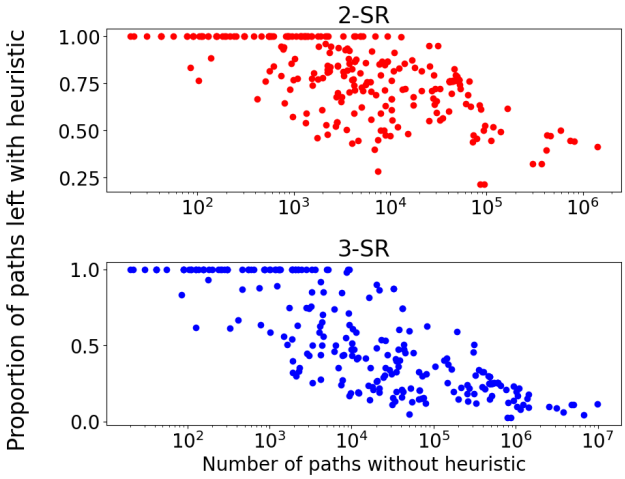
include new paths.

We then tested with the values $a = 1.1$ and $b = 3$ (decimal values are always rounded down). Notice that this only adds new paths (compared to $2x + 1$) when the shortest path length between two nodes is equal to 1. Both methods are equivalent when the shortest path length is equal to 2 and for shortest path lengths greater than or equal to 3, this new method removes even more paths.

We give in Table 2 the resulting average MLU of each heuristic and the optimal solution for each of the instances solved optimally by each method using unary weight setting (the results for the optimal 2-SR and 3-SR methods are slightly different from Table 1 since more instances were solved here; 1300 for 2-SR and 1233 for 3-SR compared to 1222 for all methods). We immediately see that the results obtained by both heuristics are quite good since they are on average within 1% of the optimal solution. Both heuristics also seem similar since the average optimal MLU of both methods are extremely close to each other.

In fact when analysing both solutions we can see that despite the average being close, the individual topologies can be quite different. We show this in Figure 15. We can see here that the $1.1x + 3$ heuristic performs more than 1% worse for 69 out of 1300 instances while the $2x + 1$ heuristic performs worse for only 23 instances. Still the difference in MLU is much higher (notice the different scales of the y-axis) which explains why this heuristic is, on average, slightly worse. It is although interesting to see that the MLU can sometimes be highly improved by simply allowing adjacent nodes to use SR-paths of maximum length 4 instead of 3 which are the only paths that the $1.1x + 3$ heuristic allows that the $2x + 1$ heuristic does not.

In Figure 16 we show how many more paths we can eliminate using these heuristics. The image on the left plots for each topology, the number of paths left using the $1.1x + 3$ heuristic proportional to the paths left after a



2-SR	
$1.1x + 3$	0.5253641965974845
$2x + 1$	0.8064406220851759

3-SR	
$1.1x + 3$	0.1572948641365103
$2x + 1$	0.43277042223078693

FIGURE 16 Proportion of paths left (compared to using no heuristic) using a heuristic with 2- or 3-SR

preprocessing without using any heuristic for 2- and 3-SR on the y-axis. The x-axis corresponds to the size of the topology (in terms of non-dominated paths), each dot corresponds to a preprocessed topology. In the table on the right we show the proportion of paths left in total over all topologies using both heuristics for 2- and 3-SR. From these results, we observe that the $1.1x + 3$ heuristic removes a significantly larger number of paths than the $2x + 1$ heuristic. Moreover, we find that larger topologies and more segments result in the removal of more paths.

In Figure 17 the computation time of the different methods is compared. As expected the $1.1x + 3$ heuristic is the fastest because it removes the most paths. Interestingly this heuristic also managed to solve the most challenging instances which an optimal algorithm could not solve even using preprocessing. Since from Figure 16 it seemed that for the smallest topologies the heuristics did not manage to remove more paths, we would have expected the regular method to follow closely the two heuristics at the start of this graph. This is not the case, the reason behind this is because the preprocessing is slightly faster using any of the heuristic methods. The reason behind this is that testing for the heuristic criterion is much faster than testing for regular path domination criterion and even though this criterion does not remove any more paths, it still manages to remove some dominated paths before testing for the domination criterion.

7 | CONCLUSIONS AND FUTURE WORK

In this article we introduce an efficient way to generate all paths that could potentially be useful in a mathematical formulation to solve the Segment Routing Traffic Engineering Problem. We demonstrate the effectiveness of our algorithm for reducing the size of a path model for the Segment Routing Traffic Engineering Problem, resulting in significantly faster computational times for solving the problem optimally.

We show that using 2-SR, the results obtained with preprocessing are slightly better on big topologies but that the gain is more important when using even more segments. Unfortunately, using only 3 segments, some instances could not be solved even using preprocessing. Using even more segments is only feasible on relatively small topologies. We also compare the results obtained with the CG4SR heuristic. Results show that this heuristic often finds a

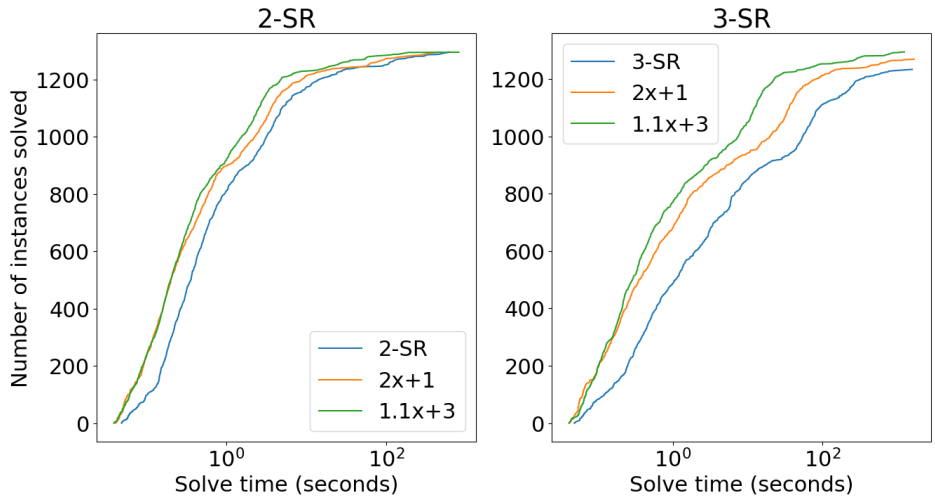


FIGURE 17 Cumulative distribution plot of the number of instances solved within a certain time

solution close to the optimal value, but that there are some instances for which it performs quite poorly. Moreover this particular heuristic is especially useful when many segments are used as it scales extremely well. Another important observation is that the gain in optimal value by using 3-SR instead of 2-SR is limited and that there is often no gain at all. On the other hand, our findings suggest that using adjacency segments can lead to a greater improvement in the optimal MLU than using 3-SR instead of 2-SR. We then show that combining the optimisation of the OSPF weights and then applying OSPF on it is on average within 1% of the optimal general routing solution.

It also turns out that eliminating dominated paths removes the possibility of having suboptimal loops in the resulting solution. Indeed, without preprocessing, it happens sometimes that because the MLU is used as objective function, some dominated paths appear in optimal solutions.

Despite this, there often appear extremely long paths in an optimal solution. We show that it is generally not useful to consider very long paths (respective to the shortest path length between the origin and destination nodes). We then use a heuristic to remove these long paths and manage to remove up to 47.5% and 84.3% of the non-dominated paths for respectively 2- and 3-SR while still keeping the average solution within 1% of the optimal solution. This last result highlights one of the shortcomings of the MLU objective function which is that this objective function only minimises the most used edge and disregards the edge load on all other edges in the network.

ACKNOWLEDGMENTS

Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

DATA AVAILABILITY STATEMENT

The topologies, demand files and some of the methods used to obtain our results are available at the REPETITA GitHub repository `https://github.com/svissicchio/Repetita`. Our implementation of the the SRTEP problem with preprocessing and all the results obtained in CSV format are available at the following GitHub repository `https://github.com/hcalleba/SRPreProcessing`

References

[1] A. Altın, B. Fortz, M. Thorup, and H. Ümit, Intra-domain traffic engineering with shortest path routing protocols, Ann. Oper. Res. **204** (April 2013), 65–95.

[2] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, Optimized network traffic engineering using segment routing, IEEE Conference Comput. Commun. (INFOCOM) (2015), 657–665.

[3] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, Ipv6 segment routing header, Tech. report, RFC Editor, 2020.

[4] C. Filsfils, N. Kumar Nainar, C. Pignataro, J. Camilo Cardona, and P. François, The segment routing architecture, IEEE Global Commun. Conference (GLOBECOM) (2015), 1–6.

[5] B. Fortz and M. Thorup, Optimizing ospf/is-is weights in a changing world, IEEE J. Selected Areas Commun. **20** (2002), 756–767.

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

[7] S. Gay, P. Schaus, and S. Vissicchio, Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms, arXiv:1710.08665v1 (2017).

[8] R. Hartert, Fast and scalables optimization for segment routing, Ph.D. thesis, Université catholique de Louvain, 2018.

[9] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. François, A declarative and expressive approach to control forwarding paths in carrier-grade networks, SIGCOMM (2015).

[10] C. Hopps, Rfc2992: Analysis of an equal-cost multi-path algorithm, Tech. report, USA, 2000.

[11] Information technology — Telecommunications and information exchange between systems — Intermediate System to Intermediate
Standard, International Organization for Standardization, Nov. 2002.

[12] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, Cg4sr: Near optimal traffic engineering for segment routing with column generation, IEEE INFOCOM 2019 - IEEE Conference Comput. Commun. (2019), 1333–1341.

[13] E. Moreno, A. Beghelli, and F. Cugini, Traffic engineering in segment routing networks, Comput. Networks **114** (2017), 23–31.

[14] J. Moy, Rfc2328: Ospf version 2, Tech. report, USA, 1998.

[15] M. Parham, T. Fenz, N. Süss, K.T. Foerster, and S. Schmid, Traffic engineering with joint link weight and segment optimization, Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies, Association for Computing Machinery, New York, NY, USA, 2021, CoNEXT '21 pp. 313–327.

[16] V. Pereira, M. Rocha, and P. Sousa, Traffic engineering with three-segments routing, IEEE Trans. Network Service Manage. **17** (2020), 1896–1909.

[17] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, Failure resiliency with only a few tunnels – enabling segment routing for traffic engineering, IEEE/ACM Trans. Networking **29** (2021), 262–274.

[18] D. Sidhu, T. Fu, S. Abdallah, R. Nair, and R. Coltun, Open shortest path first (ospf) routing protocol simulation, ACM SIGCOMM Comput. Commun. Review **23** (1993), 53–62.

[19] P. 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 Commun. Surveys Tutorials **PP** (11 2020), 1–1.

[20] D. Wu and L. Cui, A comprehensive survey on segment routing traffic engineering, Digital Commun. Networks (2022).