Midpoint Optimization for Segment Routing

Alexander Brundiers°, Timmy Schüller•°, Nils Aschenbruck°

°Osnabrück University, Institute of Computer Science
Osnabrück, Germany
Email: {brundiers, schueller, aschenbruck}@uos.de

•Deutsche Telekom Technik GmbH
Münster, Germany
Email: timmy.schueller@telekom.de

***Abstract*—In this paper, we discuss the concept of Midpoint Optimization (MO) for Segment Routing (SR). It is based on the idea of integrating SR policies into the Interior Gateway Protocol (IGP) to allow various demands to be steered into them. We discuss the benefits of this approach when compared to end-to-end SR and potential challenges that might arise in deployment. We further develop an LP-based optimization algorithm to assess the Traffic Engineering capabilities of MO for SR. Based on traffic and topology data from a Tier-1 Internet Service Provider as well as other, publicly available data, we show that this algorithm is able to achieve virtually optimal results with regards to the maximum link utilization, that are on par with state-of-the-art end-to-end SR approaches. However, our MO approach requires substantially less policies to do so. For some instances the achieved reduction ranges up to more than 99%.**

I. Introduction

To cope with the continuous growth of Internet traffic, many Internet Service Providers (ISPs) deploy some form of Traffic Engineering (TE) to utilize existing infrastructure more efficiently. A recent approach to TE that received a lot of attention is based on Segment Routing (SR). SR allows for precise steering of a packets path through a network by applying waypoints, so called segments, to a packet, that have to be visited in a specific order before heading for the original destination. Segments are applied to a packet via so called *SR policies* that are configured on a per-node basis. They can be interpreted as some form of rules that specify which segments have to be added to a packet that is steered into them [14].

Various research (e.g., [5], [21], [30]) has shown that SR is able to achieve (near) optimal results with regards to several TE objectives. However, to the best of our knowledge, all these publications focus solely on what we refer to as "*end-to-end*" SR. This means that each SR policy is dedicated to only route the traffic between just one pair of nodes, namely its respective start- and endpoint. Other demands, that do not originate/end at these nodes, but just visit them in transit will not be steered into the policy. This allows for precise traffic control on an individual, per-demand basis but also has its downsides.

In this paper, we show that using end-to-end SR can result in a high number of policies, especially in larger networks like ISP backbones. Even though SR introduces much lower overhead than, for example, Multiprotocol Label Switching (MPLS) tunnels, network operators still prefer solutions with low policy numbers for reasons of clarity, manageability, and robustness. We also demonstrate that current SR TE algorithms can heavily underestimate the number of policies if solutions are computed based on preprocessed topology information.

To tackle these problems, we pursue the idea of steering multiple different demands into a single policy by integrating them into the Interior Gateway Protocol (IGP). Contrary to current end-to-end SR approaches, this allows for a single policy to route multiple demands at once. Therefore, it has the potential to substantially reduce the number of policies that need to be configured in a network. We refer to this idea as Midpoint Optimization (MO) because traffic is detoured (or "*optimized*") at arbitrary *midpoints* along its path through the network, instead of its ingress node. Furthermore, we propose a Linear Program (LP)-based optimization algorithm that utilizes the MO concept. Based on the example of the backbone network of a globally operating Tier-1 ISP as well as other, publicly available network data, we show that it is able to achieve virtually optimal results with regards to the Maximum Link Utilization (MLU), that are on par with state-of-the-art end-to-end SR algorithms. However, our approach requires substantially (sometimes up to 99%) less SR policies, which is a major improvement over the current state-of-the-art.

We further believe that MO is of great interest for SR research and TE in general. This is backed up by the fact that several large routing vendors are working on proprietary approaches that could be classified as some form of MO.

II. Background

Before further discussing the concept of MO, we first need to provide some more information on three relevant topics: The integration of MPLS TE tunnels into the IGP, the SR architecture, and its applications for TE. In the following, the term *demand* refers to the amount of traffic that is exchanged between two nodes in the context of an Ingress-Egress (IE) traffic matrix as described in [33].

A. Integration of MPLS Tunnels into IGP Routing

Before SR was developed, MPLS Label Switched Paths (LSPs) were often used for TE purposes. Besides their use as simple end-to-end tunnels, there are some approaches to incorporate them into the IGP routing (cf. [4], [25], [28]) and, thus, make them usable by more than one demand. A rather simplistic approach, often referred to as *Basic IGP Shortcut*, is to steer a packet into a tunnel starting at the current node if the tunnel endpoint corresponds to the packets destination. This way, a tunnel between nodes X and Y can route every packet that is addressed to Y and passes over X. A more sophisticated version of this strategy is *IGP Shortcut*. Here, packets are steered into a tunnel if the respective destination

DOI: https://doi.org/10.1109/INFOCOM48880.2022.9796747

is a downstream router of the tunnel endpoint [31]. In general, the term downstream denotes a router lying "behind" the tunnel endpoint with respect to IGP Shortest Path Routing (SPR), but exact definitions can vary [4]. The definition that we follow in this paper is this one (cf. [4]): A packet is only steered into a tunnel if the tunnel endpoint lies on the shortest path from the tunnel startpoint to the packets destination. The rationale behind choosing his exact definition is further explained in Section V-A.

Basic IGP Shortcut and IGP Shortcut are both only locally significant. The existence of a tunnel is only known at its respective startpoint. As a result, it can only influence the routing decisions at this node. However, there also exist globally significant approaches in which tunnels are advertised to the IGP just like normal links [4]. This enables the IGP and, hence, all nodes to consider these tunnels in their shortest path computations. While this can be beneficial to some extent, it also introduces problems that are similar to the limitations of metric optimization TE approaches (cf. e.g., [16], [35]).

B. Segment Routing

Segment Routing (SR) [13] is based on the source routing paradigm and commonly implemented using either MPLS or a dedicated IPv6 extension. In general, different types of segments can be used depending on the intended action (e.g., routing to a specific node, using an adjacency, or applying a service). However, in this paper, we only consider node segments. Each segment is identified by a specific Segment Identifier (SID). Combining multiple SIDs into a so called *segment list* that has to be processed in the given order allows for a precise control of a packet's path through the network. The respective sub-paths between individual segments are determined by the IGP. Segment lists are defined within *SR policies* that can be configured on each SR-capable node. Such a policy can basically be interpreted as a rule specifying which segments to apply to a packet that is steered into it [14].

Another network tunneling technique that provides even better traffic steering capabilities is Resource Reservation Protocol (RSVP)-TE [3] in combination with MPLS. However, this comes at the cost of significant signaling overhead because an RSVP-TE tunnel has to be set-up and maintained on every associated node. This has negative impact on the scalability of this approach. In contrast, the information required for SR is encoded in the packet itself. Therefore, an SR policy just needs to be configured on the respective ingress node but not along the actual path, resulting in a significant reduction of the introduced control-plane overhead. However, there is some (data-plane) overhead resulting from the segment list that is appended to a packet (see e.g., [1] or [34]).

C. Segment Routing-based Traffic Engineering

SR-based TE is a topic that recently received a lot of attention. There are several publications that deal with its applications for various use-cases and objectives (cf. [34]) but a large portion focuses on the minimization of the MLU. This objective will also be the main focus of this paper.

$$\begin{aligned}
&\min \theta && && (1)\\
&\text{s.t.} \quad \sum_k x_{ij}^k &= 1 && \forall ij && (2)\\
&\sum_{ij} t_{ij} \sum_k g_{ij}^k(e) x_{ij}^k &\le \theta\, c(e) && \forall e && (3)\\
&x_{ij}^k &\ge 0 && \forall ijk && (4)
\end{aligned}$$

Problem 1: 2SR formulation (inspired by [5]).

One of the first publications regarding SR TE is [5]. It proposes an LP-based optimization algorithm called 2SR and demonstrates that two segments are often sufficient to achieve near-optimal results. The respective LP formulation is depicted in Problem 1. The objective is to minimize the MLU denoted by θ. The variables x_{ij}^k indicate the percentage share of the demand t_{ij} between nodes i and j, that is routed over the intermediate segment k. Equation (2) ensures that each demand is satisfied. Equation (3), together with the objective function, minimizes the MLU. For every edge e, $g_{ij}^k(e)$ indicates the load that is put on e if a uniform demand is routed from i to j over the intermediate segment k. These values are constants and can be efficiently precomputed. All in all, the left side of the constraint denotes the traffic that is put on e by the SR configuration represented by the x_{ij}^k. This is then limited to the edges capacity $c(e)$ scaled by θ. By minimizing this scaling factor, a SR configuration with minimal MLU is computed.

The 2SR algorithm of [5] was extended in [29] to consider additional real-world requirements that are necessary to allow for an effective deployment of SR in practice. One of those requirements is the minimization of the number of SR policies required to implement a TE solution. Due to the overhead induced by the added segment lists (cf. Section II-B) and for the general sake of clarity, maintainability, and robustness [29] network operators often aim to deploy SR configurations with as few policies as possible. To address this issue, the Tunnel Limit Extension (TLE) concept was proposed that can be used as a follow up optimization step to an MLU optimization. It pursues the objective of minimizing the number of policies while not surpassing the optimal MLU of the previous optimization by more than a predefined margin. In the context of 2SR, the resulting algorithm is called 2TLE.

III. Related Work

As mentioned in Section II-A, there already are approaches to incorporate MPLS tunnels into the IGP routing. This can basically be interpreted as MO for MPLS. Ben-Ameur et al. [4] propose an offline TE methodology for computing MPLS tunnel configurations when utilizing IGP Shortcut. In [32], a simulated annealing heuristic using Basic IGP Shortcut is presented which is able to achieve near-optimal MLU with only a rather small number of tunnels. In [25], multiple heuristics for computing MPLS tunnel configurations for various hybrid IGP/MPLS routing schemes, such as IGP Shortcut and Basic IGP Shortcut, are proposed. In an exemplary evaluation they come to the conclusion that IGP Shortcut tends to perform best with regards to MLU minimization.

In contrast to the above approaches that all rely on standard MPLS tunnels, we implement MO with SR policies. MPLS tunnels can follow virtually arbitrary paths through the network, while SR is limited to concatenations of shortest paths when only using node segments. As a result, the traffic steering capabilities of SR are more restricted (especially if the number of segments is limited). However, SR offers significantly better scalability.

The general possibility of steering multiple different demands into a single SR policy is already briefly mentioned in some Internet-drafts (e.g., [11], [15]) and in [12, Ch. 5, Ch. 11] it is discussed from a technical perspective. Furthermore, multiple vendors of routing equipment (e.g., Cisco [8] and Juniper [22]) offer support for integrating SR policies into the IGP. However, the publicly available information on this topic is rather sparse. It mostly consists of brief notes of the new MO features in the respective user guides or product documentations. Information on how MO is implemented in detail or the MO optimization algorithm in Cisco's commercial WAN Automation Engine (WAE) [8] are not provided.

To the best of our knowledge, there are no scientific publications that explicitly deal with MO or a similar concept for SR and examine its use for TE purposes.

IV. Midpoint Optimization

In this section, we formally introduce and discuss the concept of MO for SR and explain what benefits it can offer compared to current end-to-end SR approaches.

A. Issues of End-to-End SR

While end-to-end SR enables a very precise, per-demand traffic control, there are scenarios in which it causes a high number of policies. For example, assume a highly utilized link that is traversed by a large number of demands of more or less equal size. To (sufficiently) reduce its load, a large subset of these traversing demands needs to be detoured. When using end-to-end SR, a dedicated SR policy has to be configured for each demand that needs to be redirected. Furthermore, there are scenarios in which the network structure requires a lot of policies that follow virtually the same path through the network and only differ on the first or last few hops. While there is no alternative when using end-to-end SR, configuring similar policies for large sets of demands is not very efficient.

These might sound like theoretic edge-cases but such scenarios can also be found in practice, especially in larger ISP backbone networks. The reason for this is the specific topology structure of these networks that often follows certain design patterns. The network *core* consists of high-capacity and high-speed but lower connectivity routers, accompanied by hierarchical, high-connectivity node structures at the *edges* of the network, which aggregate the traffic (cf. [2] or [10, Ch. 2.3]). As a result, ISP backbones often feature so called *Edge Points of Presence (PoPs)*. They consist of a set of edge routers that are redundantly connected to two or more core routers which, in turn, connect the PoP to the rest of the network. To redirect the traffic that enters the network at a specific PoP onto a predefined TE path, individual policies need to be installed for many of these edge routers when using end-to-end SR. Often, a large portion of these policies follow basically the same path through the network. In larger ISP backbones, there can be more than 100 edge routers per PoP, resulting in large numbers of policies needing to be configured.

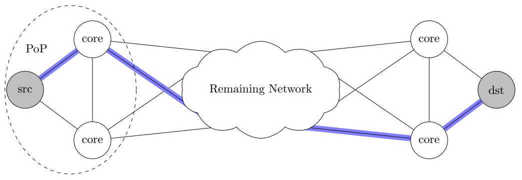
Another variant of this problem arises from the type of data that is often used for TE research. Calculations for large topologies, like ISP backbones, are rarely carried out on the original topology due to scalability issues. Especially approaches that try to solve for optimality (e.g., LP-based ones) scale rather poorly with network size. For this reason, the data used for optimization purposes often undergoes a compression or virtualization process. Those utilize the fact that Inter-PoP routing in the network core is of most interest for TE while intra-PoP routing is of lesser relevance. Therefore, the edge-routers at a PoP are often aggregated into a single virtual node (see e.g., [7] or [29]), which can significantly reduce the network size. In the ISP backbone considered later in this paper, for example, a similar preprocessing reduces the network size from over 3000 nodes to less than 200.

Publicly available datasets (e.g., *Repetita* [18] or the *Topology Zoo* [23]) that are used for TE research, often feature network data that is processed in similar fashion. Especially information on larger networks, like ISP backbones, is often only available on a PoP-level. While the use of this data for theoretic evaluations of TE algorithms is reasonable, it can result in a significant underestimation of the number of policies that are required to transfer solutions into practice. This is illustrated by the example in Figure 1. If TE solutions are computed on the virtualized topology at the top, the blue policy is considered as just one. However, its start- and endpoint are virtualized nodes that correspond to multiple edge-routers in practice. Hence, this one "virtual" policy corresponds to multiple real policies that have to be configured, one for each node-pair. In this example, with just four edge-routers per PoP, this results in a total of 16 policies instead of just one (see Figure 1b). In practice, however, the number of edge-nodes can be much higher with more than 100 nodes per PoP. In the worst case, this can result in multiple thousands of policies needing to be configured (see Section VII-B).

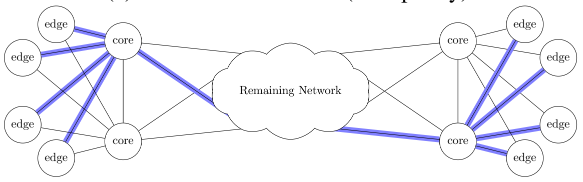
To the best of our knowledge, this problem has not been addressed in literature so far. Hence, current SR TE algorithms do not offer support for assessing the true number of policies when using virtualized data. Therefore, we propose an adaption for the 2TLE algorithm, that allows for a more accurate assessment of the required number of policies (Section VI-A).

B. The Concept of Midpoint Optimization

As seen in the previous section, relying solely on end-to-end SR policies that are bound to route just a single demand can result in a high number of policies. To deal with this issue, we suggest the use of MO for SR. MO itself is less of a concrete approach and more of a general idea that can be implemented



(a) Virtualized Solution (One policy).



(b) Deployed Solution (16 policies).

Figure 1: Policy multiplication example.

in many different ways. Overall, the idea of MO for SR can be summed up as integrating SR policies into the IGP to enable steering various demands into them. This allows for a single policy to route multiple demands which, offers the potential to significantly reduce the number of required policies, especially in scenarios like the ones described in Section IV-A.

C. Challenges of Midpoint Optimization

In the following, we take a look at problems and challenges that might arise when implementing MO.

1) Loss of Routing Expressiveness: One of the biggest advantages of SR is the detailed per-flow traffic control it offers. In theory, the routing path of each demand can be individually optimized with a dedicated policy, often enabling near perfect routing. By stepping away from this "demand-boundedness", the explicit, per-flow traffic control is lost and, hence, also some of the general routing expressiveness. This carries the risk of deteriorating achievable optimization results when deploying MO. However, MO might also be able to improve solution quality when it comes to practical implementations of SR. In current state-of-the-art SR TE algorithms (e.g., [17] or [29]) the number of segments is often limited to two or three. This is either done due to practical constraints or optimization-related reasons. While it has been shown multiple times that, in practice, three or even just two segments are often sufficient to obtain virtually optimal results, this is not always guaranteed. There are scenarios for which higher segment numbers are required. For end-to-end SR, this can result in infeasibly high computation times or resource demands of the respective optimization algorithms. It might even be impossible to sufficiently increase the segment number due to the respective Maximum Segment Depth (MSD) limit. MO, however, can benefit from the fact that it is able to (at least partially) mimic higher segment paths via a concatenation of multiple policies with fewer segments.

2) Routing Loops: When configuring a network, operators need to make sure there are no routing loops since those can significantly deteriorate performance. For end-to-end SR, infinite loops are virtually no issue and can be easily prevented. When deploying MO, however, scenarios can occur in which a policy that might be beneficial for one demand closes a loop for another. Hence, routing loops are a serious issue that has to be considered and dealt with. Adding dedicated loop-checks to optimization algorithms is probably not sufficient as long as they only consider the current network state. Any link or router failure can potentially alter the forwarding paths of packets inside and outside of policies. Such a change might result in a previously loop-free configuration to now contain a loop. Thus, solutions do not only need to be checked to be loop-free for the current network state, but also for any possible failure scenario. This, however, is probably not feasible as the number of possible failure scenarios grows exponentially with network size. Therefore, it is desirable to develop MO implementations which implicitly guarantee to never create loops.

3) Policy Nesting: For standard end-to-end SR, a demand follows a policy from the ingress to the egress node. It is technically not possible that it is affected by another policy along its path. This changes for MO. Here, traffic that already follows a policy might reach a node at which another applicable policy is configured. If that is the case, the traffic will be also routed through this policy. In theory, this *policy nesting* can occur an arbitrary number of times. This might not seem problematic at first glance. However, in practice, the maximum number of segments that can be applied to a packet is limited by the MSD of the used routing hardware. For end-to-end SR, this limitation is practically irrelevant since virtually optimal results can often be achieved with a very low number of segments (cf. [5]). However, this does not hold true for MO in the presence of policy nesting. The number of segments added by each policy varies depending on the exact MO implementation, but is always at least one. As a result, the MSD represents an upper limit for the feasible policy nesting depth that must not be exceeded. This could theoretically be ensured by explicitly checking the maximum nesting depth of computed solution. However, similar to loop-checks, this would also need to be ensured for every possible failure scenario. A more practical approach is to simply prohibit policy nesting. This can, for example, be done with so called *Strict Labels* [22, p. 639ff.] [26] [27]. This is a special type of label that guarantees that a packet is routed to the interim-destination referenced by the respective label strictly via the shortest path. Other policies or deviations will be ignored.

V. Implementation

In this section, we propose an LP-based SR TE optimization algorithm that utilizes the MO concept.

A. Selecting an MO Implementation

There are various possibilities to integrate TE tunnels (or policies) into the IGP. To not exceed the scope of this paper, we focus on just one of them for now. Based on discussions with experts from a Tier-1 ISP, we decided to follow an implementation similar to the IGP Shortcut approach for MPLS tunnels as it is defined in [4]. A packet will be steered into a policy if the policy endpoint lies on the IGP shortest path from the policy startpoint to the destination of the packet.

A crucial advantage of this approach is the fact that it makes it impossible to configure policy-induced loops if policy nesting is prohibited. With policy nesting enabled, there are scenarios in which a packet gets trapped indefinitely by nesting deeper and deeper between two policies. However, if policy nesting is prohibited, traffic entering a policy is guaranteed to also leave it again. Additionally, since packets are only steered into a policy if the policy endpoint lies on the shortest path towards their destination, they will always be closer to their destination after exiting a policy. The same holds true for SPR. Consequently, since the original distance to the destination is finite, the packet will reach it in a finite number of steps.[1]

Furthermore, IGP Shortcut strikes a good balance between other approaches like Basic IGP Shortcut or the advertisement of policies as IGP links. A policy can still be used by multiple demands with varying sources and destinations. And since it is only locally significant (cf. Section II-A), it can only influence traffic flows traversing the start-node of the policy. It will not draw additional traffic from nearby paths. In addition to that, a similar approach was found to perform best with regards to the integration of MPLS tunnels into the IGP [25].

B. Computational Challenges

Common SR TE LP formulations that aim to optimize MLU require information on how link utilizations change when adding or removing a policy. When using end-to-end SR, these values can be precomputed in a reasonable amount of time because it is perfectly clear which demand is routed through which policy, namely the one the policy is bound to.

This, however, changes when deploying MO. Now, policies can start and end on arbitrary nodes in the network and are no longer bound to a specific demand. As a result, various demands can be routed through a single policy. Furthermore, packets might be routed through multiple policies on their way to their destination. This introduces some kind of dependencies between policies (or at least their related link utilizations). Installing or removing a policy can impact the routing of various demands and, hence, alter the set of demands that is routed through other policies. For example, a policy might be configured that (normally) qualifies for routing a specific demand. However, if this demand's traffic never reaches the policies entry point due to other policies deviating it from its original path, traffic will not be routed through the former policy. As a result, the link utilizations induced by a specific policy cannot be computed without knowledge on what other policies are (and will be) installed in the network. However, this information is what we want to obtain from the optimization in the first place. Hence, it is not available at optimization time. This renders an (efficient) LP formulation (like, for example, 2TLE for end-to-end SR) virtually impossible. Note that basic end-to-end SR TE without any additional constraints is already NP-hard [19]. Introducing MO with its tunnel dependencies makes the optimization problem even harder.

Similar observations are made in [25] with regards to an IGP Shortcut approach for MPLS tunnels. There, a heuristic approach is chosen instead. The same holds true for other publications regarding the integration of MPLS tunnels into the IGP (cf. Section III). All utilize some form of heuristic.

C. Algorithmic Idea

The LP-based algorithm proposed in this paper, strictly speaking, also needs to be classified as a heuristic but it follows a quite different approach. Instead of utilizing meta-heuristics like simulated annealing or genetic algorithms, we optimally solve a (slightly) more restricted problem that does not suffer from the previously mentioned issue. The general idea is to limit the explored solution space to those solutions that do not incorporate policies that influence each other. For this, we add a constraint to our LP that prohibits the installation of policies that influence the amount of traffic that passes through already installed ones. By doing so, we circumvent the above mentioned issues regarding policy dependencies and make an efficient precomputation of resulting link loads feasible.

The general idea for computing the set of influencing policies for a specific policy p starting at node p_{start} and ending at node p_{end} is rather straight forward. For every other configurable policy x we compute the amount of traffic reaching p_{start} from every demand that is eligible for being steered into p according to the IGP Shortcut rules. If this traffic changes when x is installed, then x is an influencing policy for p. It is important to not take traffic into account while it is inside of a policy. Since policy nesting is prohibited, traffic that already is inside of a policy is not eligible to be steered into another one. Hence, the overall traffic that reaches p_{start} might be the same, but if portions of this traffic already are inside of a policy, they will not be steered into it anymore. As a result, the traffic passing through this policy and, hence, the resulting link utilizations will change. A useful property that can be exploited for a faster computation is the fact that the set of influencing policies only depends on the policy start- and endpoints due to the prohibition of policy nesting.

D. SC2SR Optimization Model

Since link loads can now be precomputed in reasonable time, this enables the formulation of an LP (see Problem 2). We refer to it as Shortcut 2SR (SC2SR) because it is based on IGP Shortcut and utilizes policies with up to two segments. The objective still is the minimization of the MLU θ. However, when compared to the 2SR formulation (Problem 1), completely different types of variables are used since policies are not bound to specific demands anymore. The binary decision variables x^l_{km} indicate whether a policy from node k over intermediate node l to m is installed. Similarly, the variables y_{km} indicate whether there is any policy installed between the nodes k and m, regardless of the respective intermediate

[1]While this approach prevents policy-induced routing loops, it still allows for structures referred to as *weak-loops* [6]. While those can still result in traffic visiting nodes multiple times, packets are not trapped infinitely in these loops but still reach their destination. Hence, those weak-loops are far less detrimental than standard routing loops. In fact, it was shown in [6] that they can even offer actual benefits with regards to certain TE objectives (e.g., MLU or policy number minimization).

$$\min \theta \tag{5}$$

$$\text{s.t.} \quad \sum_l x_{km}^l = y_{km} \quad \forall km \tag{6}$$

$$y_{km} + y_{ij} \leq 1 \quad \forall km,\ \forall ij \in \mathcal{I}_{km} \tag{7}$$

$$spr(e) + \sum_{ij} t_{ij} \sum_{klm} \mathit{diff}_{ij}^{klm}(e)\, x_{km}^l \leq \theta\, c(e) \quad \forall e \tag{8}$$

$$x_{km}^l \in \{0,1\} \quad \forall klm \tag{9}$$

$$y_{km} \in \{0,1\} \quad \forall km \tag{10}$$

Problem 2: SC2SR formulation.

$$\min \theta \tag{11}$$

$$\text{s.t.} \quad \sum_k x_{ij}^k = Z_{ij} \quad \forall ij \tag{12}$$

$$\sum_{ij} \frac{t_{ij}}{Z_{ij}} \sum_k g_{ij}^k(e) x_{ij}^k \leq \theta\, c(e) \quad \forall e \tag{13}$$

$$x_{ij}^k \in \mathbb{N}_0 \quad \forall ijk \tag{14}$$

Problem 3: Router-Level 2SR formulation.

segment[2]. The first constraint (Equation 6) connects the x_{km}^l to the corresponding y_{km} variable, while also limiting the number of policies that can be installed between any pair of nodes to at most one. With Equation 7, we ensure that if a policy between two nodes is set, none of its influencing policies $\mathcal{I}_{km}$ is installed as well. Finally, Equation 8 together with the objective function is responsible for minimizing the MLU. Its general idea is similar to Equation 3 of Problem 1. For each edge, we compute the amount of traffic that results from the current policy configuration and ensure that it does not exceed θ times the respective capacity. In this context, $spr(e)$ indicates the traffic load that is put on edge e when standard SPR is used and no policies are deployed. The $\mathit{diff}_{ij}^{klm}(e)$ values indicate the difference in the share of the demand from i to j that is put on edge e in the case of SPR and when a policy from k over l to m is installed. For example, if 70% of the demand from i to j would be routed over edge e if a policy is installed between nodes k and m with intermediate segment l, but in the SPR case (without this policy) it would only be 30%, then the diff_{ij}^{klm} value would be 0.4 (or $70\% - 30\% = 40\%$). If there is no difference between SPR and the use of the respective policy then the diff_{ij}^{klm} value is zero.

The LP as presented in Problem 2 does not consider any limitations regarding the configuration of policies on or towards specific nodes. For input data where each node just corresponds to exactly one router in practice this is fine. However, if we deal with virtualized topologies (cf. Section IV-A) in which (virtual) nodes correspond to many routers in practice, using these nodes as start- or endpoints or intermediate segments should be avoided. Otherwise, we run into the same policy multiplication problem as depicted in Figure 1. This can be done either explicitly via a dedicated constraint that fixes the corresponding x_{km}^l to zero, or implicitly by only setting up variables for non-virtual nodes. We implemented the second option because it reduces the number of variables and, hence, the overall problem size and computation times.

To allow for an effective minimization of the number of deployed policies, we developed an LP extension called SC2TLE. It is inspired by the TLE concept proposed in [29]. The general idea is to first compute the optimal MLU and then minimize the number of policies required to obtain it in a second, follow-up optimization step. The algorithm also allows for the specification of a maximum percentage MLU deterioration that is acceptable to further reduce the number of policies. This value is specified by the *trade-off coefficient* λ. The structure of the SC2TLE LP is similar to Problem 2, apart from two adaptions. First, the objective is changed from MLU to policy number minimization. This is done by replacing Equation 5 with the following one.

$$\min \sum_{km} y_{km} \tag{15}$$

Second, the following constraint is added to the LP to provide an upper limit for the MLU deterioration of the newly computed solution.

$$\theta \leq \lambda \theta' \tag{16}$$

It ensures that the MLU θ of the newly computed SC2TLE solution does not surpass the optimal MLU θ' of the preceding SC2SR optimization by more than the user-defined trade-off coefficient λ. If the MLU is not allowed to worsen at all, a trade-off coefficient of 0% ($\lambda = 1.0$) can be used.

VI. Evaluation Setup

This section introduces the algorithms and datasets used for the following evaluation. Computations are done on a computer with two AMD EPYC 7452 CPUs, 512GB of RAM and 64-bit Ubuntu 20.04.1. LPs are solved using CPLEX [20].

A. Algorithms

To assess the quality of SC2SR and SC2TLE, we use different algorithms depending on the examined objective. The first algorithm used for assessing the quality of the achieved MLUs is SPR. It is used to reflect the current state of routing that we want to improve on with our TE approaches. The second algorithm is Multicommodity Flow (MCF) [24, Ch. 4.4] which we use to compute a lower bound for the achievable MLU. To compute the MLU achievable with end-to-end SR we use the 2SR algorithm (Problem 1).

[2]Technically, the LP could also be formulated without the y_{km} variables using only x_{km}^l. However, utilizing the former allows for a smaller LP and, hence, a faster solving, at least with regards to the CPLEX solver used here.

To assess the minimal number of policies required by end-to-end SR it would be optimal to compute solutions on unvirtualized data. However, as mentioned in Section IV-A, this is often not feasible for larger networks for reasons of scalability. Another approach would be to simply add a weighting-factor to each of the policies in the virtualized solution that corresponds to the actual number of policies required to implement it in practice. For example, the blue policy in the virtualized solution in Figure 1 would get a weight of 16, as it actually requires 16 end-to-end polices in practice. Such an approach would provide information on the actual number of policies required to implement 2TLE solutions that were computed on virtualized data. However, it (most likely) does not provide information on the actually minimal number of policies that could be achieved if optimization would be carried out on the original, unvirtualized data. The reason for this is that it either installs a policy for all edge-routers resembled by a virtual node or for none of them. In practice, however, it might be optimal to only install the policy for some of the edge routers (or their respective demands).

To get a better approximation of the minimal number of required policies, we developed an adaption of the 2TLE algorithm, called Router-Level 2TLE (RL2TLE), that allows to split virtual demands into equal sub-demands according to the actual number of edge-routers grouped into the respective virtual node(s). The LP of the first optimization step is given in Problem 3. The Z_{ij} values denote the number of sub-demands in which the demand between nodes i and j can be split. For example, the virtual demand between the two virtual nodes in Figure 1 would get a Z-value of 16 because it resembles 16 real demands (one for each pair of nodes). For each of them a dedicated policy can be installed. The second optimization step uses a TLE similar to the one described in [29].

This approach assumes that traffic is distributed equally across the edge-routers of a PoP. While this might not always be true in practice, we still believe that it is a valid assumption since it is in the interest of operators to connect customers in a way that the traffic is distributed more or less equally.

B. Data

We carry out evaluations on three sets of data. The first consist of real topology and traffic data collected in the backbone network of a globally operating Tier-1 ISP. Snapshots of the respective network topology and the measured traffic matrix are provided on a quarter-hour basis. For our evaluations, we were given 19 snapshots from between March 2017 and January 2021 with traffic matrices located in the daily peak-hour. Due to network expansions, the respective topology varies between snapshots. On average it comprises around 143 nodes and 900 edges and has a diameter of about 7.

Our second set of data also features data from the ISP backbone. However, this time we tried to create instances that are more challenging to optimize by mapping back more recent traffic matrices (e.g., from 2021) to older expansion states of the network (e.g., 2020). Since network traffic is generally increasing over the years [9], this forces more traffic through a network with lower capacity. We carried out this operation for a large number of instances. Out of the resulting instances, we discarded all those for which the SPR MLU was still below 100% and those for which the MCF MLU was above 100%, as for those TE would be either not (strictly) necessary or not particularly helpful, respectively. Out of the remaining instances, we selected 10 to evaluate our algorithms on. Both of the above datasets feature data that is virtualized in a similar fashion as described in Section IV-A. Here, a single virtual node corresponds to 50 to 150 edge-routers in practice.

Table I: Overview over the selected Repetita instances.

Name	Nodes	Edges	Identifier
DeutscheTelekom	30	110	A
Forthnet	62	124	B
Globenet	67	226	C
GtsCzechRepublic	32	66	D
RedBestel	84	202	E
Renater2008	33	86	F
Renater2010	43	112	G
Ulaknet	82	164	H
Uninett2010	74	202	I
Uunet	49	168	J

To also evaluate MO on completely different networks, our third dataset features ten uniformly selected instances from the publicly available Repetita dataset [18] which are listed in Table I. Those instances are often provided on a PoP-level and, thus, likely also feature virtual nodes. However, there is no information on which nodes are virtualized and how many edge-routers are grouped into them. Hence, we just ignore possible virtualizations and treat every node as a normal, single router. This also allows us to evaluate the performance of our MO algorithm for smaller, "unvirtualized" networks.

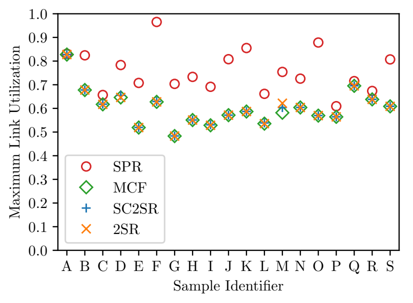
VII. Evaluation Results

In this section, we evaluate the performance of our newly developed MO algorithm with regards to the achievable MLU and the number of required policies.

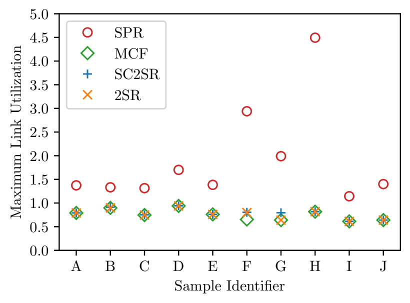
A. Maximum Link Utilization

To assess the MLU optimization capabilities of our new SC2SR algorithm we optimized each instance from our reference datasets with it. The results for the original ISP instances are depicted in Figure 2a together with the MLU values obtained with our reference algorithms (SPR, MCF, and 2SR). It can be seen that for nearly all of the 19 evaluation instances SC2SR performs as good as 2SR and is able to provide an optimal solution. Only for instance M SC2SR is not able to achieve the optimal MLU. However, it is still rather close to the optimum and also performs better than 2SR.

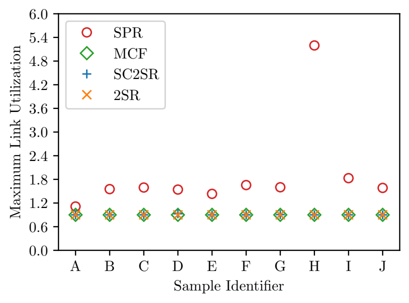
Similar results can be observed for the backmapped ISP dataset (see Figure 2b). The substantially higher SPR MLUs indicate that our traffic backmapping approach seems to have indeed created more challenging instances. Nonetheless, SC2SR is still able to achieve optimal results for 8 out of the 10 instances. For one of the two instances (G) that were not solved optimally, 2SR is able to achieve a better MLU, possibly due to its superior per-flow traffic control. However,



(a) Original ISP instances.

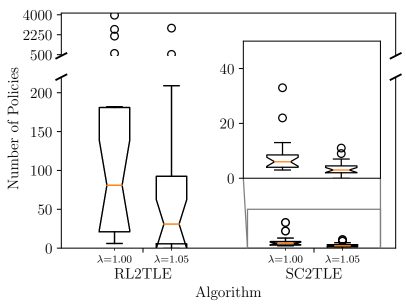


(b) Backmapped ISP instances.

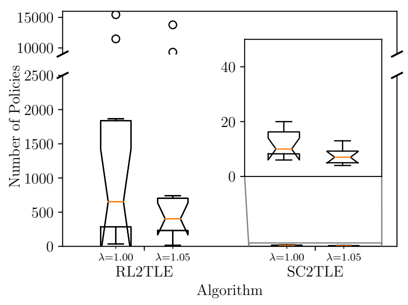


(c) Repetita instances.

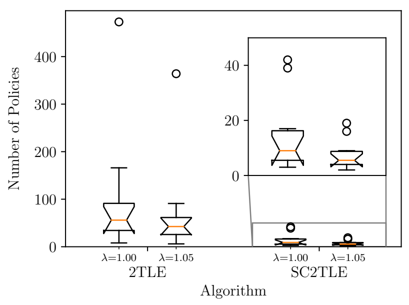
Figure 2: MLUs achieved with different algorithms.



(a) Original ISP instances.



(b) Backmapped ISP instances.



(c) Repetita instances.

Figure 3: Number of SR policies required by different algorithms.

it could also be a result of the limitations regarding *influencing* policies implemented into our algorithm (cf. Section V-C) that limit the explored solution space. Other MO algorithms without this constraint might be able to achieve the optimal MLU. For the second suboptimal instance (F), 2SR is not able to achieve a better MLU than SC2SR. Maybe, in this specific scenario, the theoretical lower bound that is presented by MCF is not reachable with TE approaches that have to adhere to practical limitations. Even though there are two instances for which SC2SR is not able to achieve the proven optimum, it is still able to substantially reduce the MLU when compared to SPR and prevents overutilization in all cases.

Results for the Repetita instances are depicted in Figure 2c. It can be seen that 2SR as well as SC2SR are able to achieve the optimal MLU of 0.9 for every instance. This confirms that our MO algorithm is also able to find optimal solutions for other (completely different) networks.

All in all, this evaluation shows that, regarding MLU, our SC2SR algorithm is able to keep up with end-to-end SR approaches. This is rather surprising because, as mentioned in Section IV-C, MO lacks the fine-grained, per-demand traffic control of end-to-end SR. While this can be a limiting factor in theory, it does not appear to be of much relevance in practice. It further needs to be remembered that SC2SR does not utilize the capabilities of the IGP Shortcut MO approach to their full extent. In order to enable an efficient LP formulation, we had to limit the explored solution space (cf. Section V-C). Theoretical examples can be constructed for which these limitations can result in arbitrarily bad results. However, these examples are carefully hand-crafted and far from a realistic network design. In our real-world evaluation instances, we are able to obtain optimal results in nearly every scenario.

B. Number of Required Policies

Following the evaluation of the primary objective of MLU minimization, we now take a look at how many SR policies are required to implement these solutions. We compare the number of policies required by our SC2TLE algorithm to the ones required by RL2TLE that utilizes end-to-end SR. This is done for a trade-off coefficient of 0% which resembles the number of policies required to obtain the best possible solutions as they were shown in Section VII-A, and also for a trade-off coefficient of 5%. The latter can be loosely understood as an upper bound of the MLU deterioration that is acceptable in practical scenarios to further reduce the number of policies.

The distributions of the results for the original ISP dataset are depicted in Figure 3a. First of all, it can be seen that the number of policies required by RL2TLE varies drastically between instances. Some can be solved with as little as 8 policies while others require multiple thousands. The reasons for this are those explained in Section IV-A. As soon as the optimal solution requires detouring demands between edge-PoPs (virtual nodes), the number of policies often increases drastically because individual policies need to be configured for most of the edge-routers. By using a trade-off coefficient of 5%, the number of policies can be further reduced but there are still instances that require hundreds and, in one case, more than 2800 policies. It has to be remembered that RL2TLE is no heuristic but provides the lowest possible number of policies required to obtain the respective MLUs when using end-to-end 2SR. With MO and our new SC2TLE algorithm, significant reductions in the number of policies can be achieved. Even for a trade-off coefficient of 0% it requires at most 33 policies and in most cases less than ten. When comparing these numbers to the ones of end-to-end SR the reductions are enormous. For some instances, they range up to more than 99% (e.g., instance B). Even for instances for which end-to-end SR requires rather low policy numbers, SC2TLE is often still able to undercut this number. In fact, it never requires more policies than the end-to-end approach. For the backmapped ISP instances (Figure 3b), the number of policies required by RL2TLE is even higher but the performance of SC2TLE and the qualitative results are similar to the ones obtained on the original ISP instances.

Since we assume that the Repetita data does not feature virtual nodes, we use 2TLE [29] to calculate the number of policies required by end-to-end SR. Overall, the number of policies is much smaller and, hence, the absolute reductions achieved by SC2TLE are also lower (see Figure 3c). However, it still enables substantial reductions of the policy numbers for most instances. For example, 2TLE requires over 350 policies for instance E, even when used with a trade-off coefficient of 5%, while SC2TLE achieves the optimal MLU with just 9 policies. This illustrates that the benefits of our SC2TLE algorithm are not limited to large networks and virtualized data, but also apply to smaller, unvirtualized networks as well.

Detailed policy numbers for each instance are listed in Table II in the appendix. For better readability they are sorted in descending order with respect to the number of policies required by RL2TLE. It also features information on the actual number of policies that would be required to implement solutions into practice that were obtained with the standard 2TLE algorithm. Those are computed by applying a weighting to virtual policies as described in Section VI-A. The results are listed in the column labeled "w2TLE" (weighted 2TLE).

C. Computation Times and Resource Demands

Like most LP-based approaches, SC2TLE is quite demanding with regards to resources and computation times. For some larger networks, it can take multiple hours and require a couple hundred gigabytes of RAM to find the optimal solution. This is perfectly acceptable when planning to use this algorithm as intended by us, namely to optimize a network on a weekly (or daily) basis. However, it is not designed and, hence, not suited for sub-second optimization or quick, tactical re-optimizations in failure scenarios. For such scenarios, dedicated heuristics should be developed, which we plan to do in the future.

VIII. Conclusion

In this paper, we discussed the concept of MO for SR. It is based on the idea of integrating SR policies into the IGP to steer traffic into them. Contrary to the current end-to-end SR approaches in which a dedicated policy has to be installed for each demand that needs to be detoured, MO allows a single policy to route multiple different demands. This enables TE solutions with a substantially lower number of policies.

Besides a formal description of the MO concept and a discussion of implementation related challenges, we developed an optimization algorithm to assess the TE capabilities of MO. Based on data from a Tier-1 ISP and the publicly available Repetita [18] dataset, we showed that our MO algorithm is able to achieve virtually optimal MLUs that are on par with current end-to-end SR approaches. However, while the latter often require multiple hundreds (if not thousands) of policies, our algorithm achieves solutions of similar quality with a single-digit number of policies in many cases, sometimes corresponding to a reduction of more than 99%. This impressively demonstrates the capabilities and benefits of MO for SR.

In the future, we plan to implement other approaches to the MO concept to compare them to the one proposed here. Furthermore, it might be worthwhile to examine the potential of hybrid approaches that allow for a local activation of MO capabilities on a per-router basis. This way, it might be possible to combine the per-demand traffic control of end-to-end SR with the exceptional low policy numbers of MO.

Appendix

Table II: Number of SR policies required by different algorithms for the three datasets.

	RL2TLE		SC2TLE		w2TLE	
$\lambda =$	1.0	1.05	1.0	1.05	1.0	1.05
Instance						
ISP Original						
B	3961	2829	6	3	22653	22502
M	2722	51	7	2	2750	100
H	2128	209	8	6	22807	453
F	610	526	33*	11	783	756
K	182	148	6	3	254	153
J	180	131	11	9	263	162
L	106	54	4	3	205	54
D	89	46	22	4	302	152
O	84	35	13	7	261	106
I	81	31	4	3	154	53
N	47	12	9	5	106	56
S	46	23	4	2	252	102
R	32	3	3	1	250	50
P	30	8	5	3	151	58
G	12	9	7	4	160	9
Q	11	0	3	0	200	0
C	7	2	5	2	155	2
A	7	0	3	0	150	0
E	6	2	5	2	153	2
ISP Backmapped						
A	15444	13782	8	7	68264	45006
B	11482	9300	8	7	45305	45302
F	1866*	592	18	10	2016	636
H	1757*	425	18	12	1765	535
J	860	741	20*	13	1274	965
C	448	380	10	7	611	458
G	368	259	10	4	723	524
D	257	223	9	5	310	304
E	112	71	6	4	160	118
I	35	15	11	5	208	105
Repetita						
E	473	364	9	8	-	-
G	166	91	42	16	-	-
C	99	65	7	4	-	-
D	68	34	17	6	-	-
F	57	48	14	9	-	-
H	55	51	3	2	-	-
J	53	37	39*	19	-	-
B	28	23	5	4	-	-
I	24	12	9	5	-	-
A	8	6	4	4	-	-

* The asterisks mark optimizations that were aborted due to memory or time limit exceedance. The respective values correspond to the currently best lower and upper bound for RL2TLE and SC2TLE, respectively.

References

[1] 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.

[2] D. Alderson, L. Li, W. Willinger, and J. Doyle, "Understanding Internet Topology: Principles, Models, and Validation," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1205–1218, 2005.

[3] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC Editor, RFC 3209, 2001.

[4] W. Ben-Ameur, N. Michel, B. Liau, J. Geffard, and E. Gourdin, "Routing Strategies for IP-Networks," *Telektronikk Magazine*, vol. 97, pp. 145–158, 2001.

[5] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, "Optimized Network Traffic Engineering using Segment Routing," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2015, pp. 657–665.

[6] A. Brundiers, T. Schüller, and N. Aschenbruck, "On the Benefits of Loops for Segment Routing Traffic Engineering," in *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, 2021, pp. 32–40.

[7] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP Network Energy Cost: Formulation and Solutions," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 463–476, 2012.

[8] Cisco Systems. Cisco WAN Automation Engine (WAE). [Online]. Available: https://www.cisco.com/c/en/us/products/routers/wan-automation-engine

[9] Cisco Systems Inc., "Cisco Annual Internet Report (2018–2023)," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020.

[10] R. D. Doverspike, K. K. Ramakrishnan, and C. Chase, "Structural Overview of ISP Networks," in *Guide to Reliable Internet Services and Applications*, C. R. Kalmanek, S. Misra, and Y. Yang, Eds. Springer London, 2010, pp. 19–93.

[11] C. Filsfils *et al.*, "Segment Routing Policy for Traffic Engineering," Internet Draft draft-filsfils-spring-segment-routing-policy-04, 2017.

[12] C. Filsfils, K. Michielsen, F. Clad, and D. Voyer, *Segment Routing Part II - Traffic Engineering*, 2019.

[13] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *Proc. of the IEEE Global Communications Conf. (GLOBECOM)*, 2015.

[14] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture ," RFC Editor, RFC 8402, 2018.

[15] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, and P. Mattes, "Segment Routing Policy Architecture," Internet Draft draft-ietf-spring-segment-routing-policy-08, 2020.

[16] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 756–767, 2002.

[17] S. Gay, R. Hartert, and S. Vissicchio, "Expect the Unexpected: Sub-Second Optimization for Segment Routing," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2017.

[18] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms," *ArXiv e-prints*, 2017.

[19] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving Segment Routing Problems with Hybrid Constraint Programming Techniques," in *Proc. of the Int. Conf. on Principles and Practice of Constraint Programming (CP)*, 2015, pp. 592–608.

[20] IBM, "IBM ILOG CPLEX Optimization Studio 20.1.0," https://www.ibm.com/docs/en/icos/20.1.0, 2020.

[21] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "CG4SR: Near Optimal Traffic Engineering for Segment Routing with Column Generation," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2019, pp. 1333–1341.

[22] Juniper Networks, "Junos OS IS-IS User Guide," Tech. Rep., 2021. [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/is-is/is-is.pdf

[23] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[24] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017.

[25] E. Mulyana and U. Killat, "Optimization of IP Networks in Various Hybrid IGP/MPLS Routing Schemes," in *Proc. of the GI/ITG Conf. on Measuring and Evaluation of Computer and Communication Systems (MMB) together with 3rd Polish-German Teletraffic Symposium (PGTS)*, 2004, pp. 295–304.

[26] P. Psenak, R. Hanzl, C. Filsfils, and K. J. Talaulikar, "Enforcing Strict Shortest Path Forwarding Using Strict Segment Identifiers," US Patent 10 263 881, 2017. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=US206620949

[27] ——, "Enforcing Strict Shortest Path Forwarding Using Strict Segment Identifiers," US Patent 10 742 537, 2019. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=US254129770

[28] J.-L. L. Roux, J.-P. Vasseur, and J. Boyle, "Requirements for Inter-Area MPLS Traffic Engineering ," RFC Editor, RFC 4105, 2005.

[29] 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, pp. 1851–1864, 2018.

[30] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, "Failure Resiliency With Only a Few Tunnels – Enabling Segment Routing for Traffic Engineering," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 262–274, 2021.

[31] J. Shen and H. Smit, "Calculating Interior Gateway Protocol (IGP) Routes Over Traffic Engineering Tunnels," RFC Editor, RFC 3906, 2004.

[32] F. Skivée, S. Balon, and G. Leduc, "A Scalable Heuristic for Hybrid IGP/MPLS Traffic Engineering - Case Study on an Operational Network," in *Proc. of the IEEE Int. Conf. on Networks (ICON)*. IEEE, 2006, pp. 1–6.

[33] P. Tune and M. Roughan, "Internet Traffic Matrices: A Primer," in *Recent Advances in Networking, Vol. 1*, H. Haddadi and O. Bonaventure, Eds. ACM SIGCOMM, 2013, pp. 108–163.

[34] 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.

[35] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An Overview of Routing Optimization for Internet Traffic Engineering," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.