Received 29 January 2024; accepted 22 February 2024. Date of publication 26 February 2024; date of current version 12 March 2024.

*Digital Object Identifier 10.1109/OJCOMS.2024.3369873*

# An Extended Look at Midpoint Optimization for Segment Routing

**ALEXANDER BRUNDIERS (Graduate Student Member, IEEE), TIMMY SCHÜLLER, AND NILS ASCHENBRUCK (Member, IEEE)**

Institute of Computer Science, Osnabrück University, 49069 Osnabrück, Germany

CORRESPONDING AUTHOR: A. BRUNDIERS (e-mail: brundiers@uos.de)

This work is an extended version of [1] [DOI: 10.1109/INFOCOM48880.2022.9796747].

**ABSTRACT** In this paper, we discuss and examine 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 a Linear Program-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, for most problem instances, this algorithm is able to achieve (close to) 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%. Furthermore, we show that latency bounds for individual demands can be incorporated into our algorithm without significantly worsening the quality of solutions. This is a crucial finding as the inclusion of latency bounds is a basically mandatory requirement for traffic engineering algorithms to be used in many real-world networks.

**INDEX TERMS** Segment routing (SR), traffic engineering, midpoint optimization.

## 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 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 [2].

Various research (e.g., [3], [4], [5]) 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 publications focus solely on what we refer to as "*end-to-end*" SR. This means that each SR policy is dedicated to 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 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 to be configured in a network. We refer to this idea as 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 and takes further important practical requirements (e.g., latency bounds) into consideration. 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[1] 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.

The remainder of this paper is structured as follows. In Section II, we briefly introduce mandatory background information, followed by a discussion of related work (Section III). Based on this, we then introduce and discuss the concept of MO for SR in Section IV and present our LP-based optimization algorithm in Section V. After this, we describe our evaluation setup (Section VI) before presenting and discussing our evaluation results in Section VII. In Section VIII, we introduce another, practically motivated constraint (latency bounds for individual demands) into the optimization problem and discuss how our algorithms can be extended to also take this new constraint into account. Finally, Section IX concludes this paper by recapitulating on the most important findings and contributions and discussing possible future work.

## 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 [6].

[1]In the context of this paper, the term "*virtually optimal*" refers to solutions that might not be absolutely optimal regarding the strict mathematical definition of the term "optimal" as there is another, ever so slightly better solution. However, the difference in solution quality to the true optimal solution is negligibly small. As a result, for any practical real-world application, there is basically no detectable quality difference between the two solutions. For example, imagine a TE scenario where the truly optimal solution has an Maximum Link Utilization (MLU) of 0.9 but our algorithm "only" finds a solution with an MLU of 0.905. Then this solution is, strictly speaking, not optimal but for every reasonable application scenario, it can be considered basically as good as the truly optimal solution, since such subtle differences are lost in the general noise and variation of traffic anyway.

### 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. [7], [8], [9]) 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 is a downstream router of the tunnel endpoint [10]. In general, the term downstream denotes a router lying "behind" the tunnel endpoint with respect to Shortest Path Routing (SPR), but exact definitions can vary [7]. The definition followed in this paper is this one (cf. [7]): 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 this choice 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. Hence, 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 [7]. This enables the IGP and, thus, all nodes to consider these tunnels in their shortest path computations. While this can be beneficial to some extent, it also introduces problems similar to the limitations of metric optimization TE approaches (cf. e.g., [11], [12]).

### 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. A policy can basically be interpreted as a rule specifying which segments to apply to a packet that is steered into it [2].

Another network tunneling technique that provides even better traffic steering capabilities is Resource Reservation Protocol (RSVP)-TE [14] 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, significantly reducing 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., [15] or [16]).

### 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. [16]) but a large portion focuses on the minimization of the MLU. This objective will also be the main focus of this paper.

One of the first publications regarding SR TE is [3]. It proposes an LP-based optimization algorithm called 2SR and demonstrates that two segments are often sufficient to achieve near-optimal results. The respective optimization problem can be formulated as follows[2]:

$$\text{P1}: \min \quad \theta \tag{1a}$$

$$\text{s.t.} \quad \sum_{k} x_{ij}^{k} = 1 \qquad \forall ij \tag{1b}$$

$$\sum_{ij} t_{ij} \sum_{k} g_{ij}^{k}(e) x_{ij}^{k} \leq \theta\, c(e) \qquad \forall e \tag{1c}$$

$$x_{ij}^{k} \geq 0 \qquad \forall ijk \tag{1d}$$

The objective is to minimize the MLU denoted by $\theta$. 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 (1b) ensures that each demand is satisfied. Equation (1c), 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 $\theta$. By minimizing this scaling factor, a SR configuration with minimal MLU is computed. We note that all constraints are of linear nature. Hence, P1 is an LP and can be efficiently solved with software like CPLEX [17].

A major issue of the above 2SR algorithm as proposed in [3] is the fact that it does not take into account crucial real-world constraints that result from either hard- and software limitations or other important operational requirements. Thus, the results computed with this algorithm are basically of theoretical nature only and (in most cases) do not yield practically usable SR configurations. One of those crucial operational 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 network operators often aim to deploy SR configurations with as few policies as possible [5], [18]. The original 2SR algorithm of [3], however, often uses multiple thousands of SR policies in its solutions (cf. [18]). These issues are addressed in [18] where the original 2SR algorithm of [3] is extended to also consider those additional real-world requirements that are necessary to allow for an effective deployment of SR in practice. In particular, to address the objective of minimizing policy numbers, the Tunnel Limit Extension (TLE) concept is 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.

[2]A list of the most important notations used throughout this paper is given in Table 5 in Appendix C.

## III. RELATED WORK

In this section, we discuss related work regarding the concept of MO for SR and also present a short primer on why most SR TE algorithms in literature (most likely) underestimate required policy numbers on certain datasets.

### A. MIDPOINT OPTIMIZATION-LIKE CONCEPTS

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. [7] propose an offline TE methodology for computing MPLS tunnel configurations when utilizing IGP Shortcut. In [19], 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 [8], 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 regarding MLU minimization.

Contrary 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. Hence, the traffic steering capabilities of SR are more restricted (when limiting the segment number) but it offers significantly better scalability.

The general possibility of steering multiple different demands into a single SR policy is already briefly mentioned in some RFCs or Internet drafts (e.g., [20], [21], or [22]) and in [23, Ch. 5, Ch. 11] it is discussed from a technical perspective. Furthermore, multiple vendors of routing equipment (e.g., Cisco [24] and Juniper [25]) offer support for integrating SR policies into the IGP. Depending on the respective vendor, such approaches also go under the name of *IGP Shortcut* or *autoroute* features. 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) [24] 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, apart from our own work [26], [27] building up on the original conference version of this paper [1].

### B. A PRIMER ON ISSUES REGARDING SR POLICY NUMBER CALCULATION ON CERTAIN DATASETS

TE optimization 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-Point of Presence (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., [18] or [28]), 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* [29] or the *Topology Zoo* [30]) that are commonly 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. The reason for this is as follows. If a SR policy featuring a virtualized node as its start- or endpoint is chosen in a TE solution, most algorithms (e.g., [18]) count this policy as just a single one. In a practical deployment, however, the number of real policies required to implement such a "virtual" policy can be much higher. The reason for this is that the respective virtual start- or endpoint corresponds not to a single but to multiple real routers in practice and the respective policy needs to be installed on each of those individually. For large ISP backbones with multiple tens of edge-routers per PoP, this can result in a single policy between two virtual nodes corresponding to multiple hundred policies in practice, increasing the total number of policies to multiple thousands in the worst case (see Section VII-B).

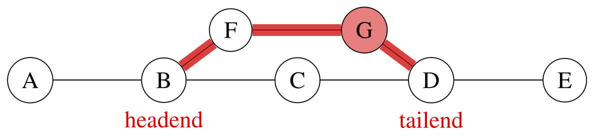
To the best of our knowledge, this problem has not been addressed in literature so far. Hence, current SR TE algorithms do not offer support for assessing the true number of policies when using virtualized data and, thus, feature the risk of heavily underestimating the actual number of SR policies required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

## IV. MIDPOINT OPTIMIZATION

In this section, we introduce and discuss the concept of MO for SR and provide a general mathematical problem formulation. Furthermore, we explain what benefits MO can offer compared to current end-to-end SR approaches and discuss potential challenges that need to be considered when implementing MO in practice.

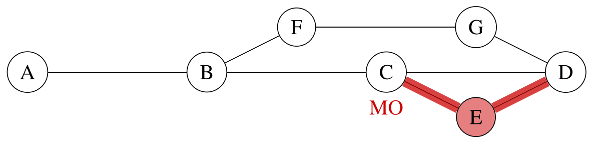
### A. THE GENERAL MIDPOINT OPTIMIZATION CONCEPT

Before formulating a mathematical model, a general understanding of the MO concept is required. Overall, the idea of MO for SR can be summed up as stepping away from the end-to-end nature of conventional SR and "freeing up" SR policies from being bound to a specific demand, thus enabling them to influence the paths of multiple demands at once. This is done by changing the mechanism of how a demand is steered onto a SR policy. In the conventional, end-to-end SR case, a demand is only steered onto a policy if the policy is configured on the source-node of the demand and ends at its destination node. So, here, SR policies always function like some form of end-to-end tunnels for a specific demand. In the context of MO, however, different approaches can be used to determine whether a demand is steered onto an existing SR policy. As a result, policies basically are no longer bound to specific demands. Instead, they just "exist" within the network together with a given rule-set on how to determine whether a demand that passes over the start (*headend*) of a policy is steered onto it. If a demand is not eligible to be steered onto the respective policy or if there are no policies configured at the respective node, it is forwarded to the next hop based on the standard IGP routing (i.e., SPR). The selected rule-set distinguishes the different variations or implementations of the more abstract general MO concept. An example for possible MO variations and the resulting traffic steering decisions is depicted in Figure 1. Assuming the given topology where a simple hop-count metric is used and a SR policy configured between nodes $B$ and $D$ using node $G$ as intermediate segment (indicated in red). In the case of end-to-end SR, only the demand $B \rightarrow D$ would be steered onto this policy. In the context of MO, a rule-set could be used that steers a demand onto a policy only if the policy end-node (*tailend*) corresponds to the demands destination. We refer to this MO variation as *destination-bound*. In this scenario, the demands $B \rightarrow D$ and $A \rightarrow D$ would both be steered onto the policy as they both pass over the headend (node $B$) and have $D$ as destination. When using the *IGP Shortcut* rule-set (cf. Section II-A), even more demands can be detoured with this single policy.

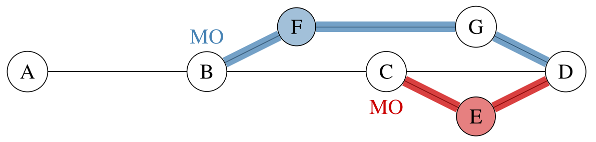


| MO variation | Demands routed through policy $B \rightarrow G \rightarrow D$ |
|---|---|
| End-to-End SR | $B \rightarrow D$ |
| MO (dst-bound) | $B \rightarrow D$, $A \rightarrow D$ |
| MO (IGP Shortcut) | $B \rightarrow D$, $A \rightarrow D$, $A \rightarrow E$, $B \rightarrow E$ |

**FIGURE 1.** **Example for different MO rule-set variations and the resulting routing (listed in the table).**



(a) Demand $A \rightarrow D$ is routed through the red policy.



(b) If the blue policy is additionally installed, traffic from demand $A \rightarrow D$ no longer enters the red policy.

**FIGURE 2.** **Simplified example for policy dependency when deploying MO-capable SR policies (inspired by [27]).**

## MATHEMATICAL PROBLEM FORMULATION

With the general concept of MO being established, we can now formulate the respective mathematical optimization problem. Given a directed graph $G = (V, E)$ where $V$ is the set of nodes and $E$ the set of edges, representing the network's routers and edges, respectively. Each edge $e \in E$ is associated with a capacity $c(e)$ and an IGP metric value $m(e)$. Additionally, for each pair of nodes $(i, j) \in N \times N$ a traffic demand of size $t_{ij}$ is given that has to be routed from node $i$ to node $j$. $\mathcal{R}$ defines the *rule-set* that is used to determine whether a demand that passes over the start (*headend*) of a policy is steered onto it. The objective is to find a set of SR policies $\mathcal{P}$ that, when deployed in the network, minimizes the MLU when routing the given set of traffic demands while adhering to the given MO rule-set $\mathcal{R}$. This can be formulated as follows:

$$\text{P2: } \min \ \theta \qquad (2a)$$

$$s.t. \ \ load(e) \ \leq \ \theta \, c(e) \ \forall e \qquad (2b)$$

where $\theta$ denotes the MLU and $load(e)$ the traffic load on edge $e$ which is made up of the traffic put on the respective edge by each individual demand:

$$load(e) = \sum_{i,j} t_{ij} \ FP_{ij}(\mathcal{P}, e) \qquad (2c)$$

In this context, $FP_{ij}(\mathcal{P})$ denotes the forwarding path (including Equal Cost Multipath (ECMP)) of the demand from $i$ to $j$ when using the policy configuration $\mathcal{P}$ under the given MO rule-set $\mathcal{R}$. Analogously, $FP_{ij}(\mathcal{P}, e)$ denotes the share of traffic that is put on edge $e$ when routing a uniform traffic flow from $i$ to $j$ according to the respective forwarding path (similar to the $g_{ij}^k(e)$ in problem P1).

While the former problem formulation might look rather simple, it inherits one major issue when it comes to actually solving it in practice. We further elaborate this in the following section.

## POLICY DEPENDENCIES AND RESULTING COMPUTATIONAL PROBLEMS

As shown in Equation (2c), the utilization of each edge depends on the share of traffic that is routed over it. To determine which demand is steered over which edges, information on the respective forwarding path $FP_{ij}$ is required. These forwarding paths depend on the set of policies $\mathcal{P}$ that are configured in the network. This information, however, is not known prior to optimization. In fact, it is what we want to compute in the first place. For end-to-end SR, this issue can be circumvented since a demand is only steered onto policies configured between its respective source and destination node. Hence, the forwarding path for a demand $i \rightarrow j$ only depends on policies starting at node $i$ and ending at node $j$, but is completely independent of further policies configured between other nodes in the network. This can be exploited to efficiently precompute the required information on how adding or removing a policy changes link utilizations (i.e., the $g_{ij}^k(e)$ values of problem P1) and, thus, allows for an efficient LP formulation.

This, however, is no longer possible when using MO instead of end-to-end SR. Now, policies are no longer bound to a specific demand and, as a result, various demands can be routed through a single policy, depending on the respective MO rule-set $\mathcal{R}$. 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 startpoint due to other policies deviating it from its original path, traffic will not be routed through the former policy. This is exemplary depicted in Figure 2. If only the red policy is installed at node $C$ (the colored node marks the respective intermediate segment), the traffic of demand $A \rightarrow D$ will be routed through it. However, if the blue policy is installed on node $B$ as well, the traffic will be routed through it instead. As a result, the link utilizations induced by a specific policy cannot be computed without knowledge on what other policies are 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 end-to-end SR TE without any additional constraints is already NP-hard [31]. Introducing MO with its policy dependencies makes the optimization problem even harder.

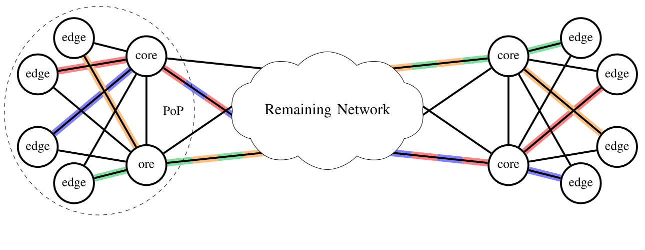
Similar observations are made in [8] with regards to an IGP Shortcut approach for MPLS tunnels. There, a heuristic approach is chosen instead. The same holds true for other publications regarding the integration of MPLS tunnels into the IGP (cf. Section III-A). All utilize some form of heuristic.

## B. BENEFITS OF MO COMPARED TO E2E SR

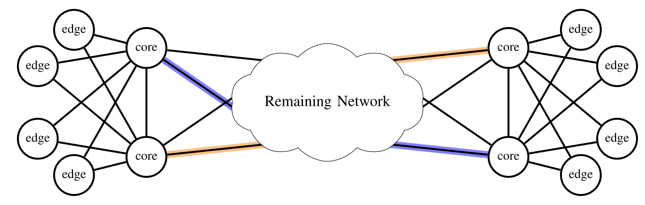
In the following, we discuss several use cases in which the use of MO for SR holds the potential to overcome shortcomings and limitations of end-to-end SR.

### 1) POLICY NUMBER REDUCTION

While the end-to-end nature of conventional SR enables a very precise, per-demand traffic control, it is also the cause for its main issue: The rather high policy numbers required to implement TE solutions. Since each end-to-end SR policy is designated to only route one specific demand (the one between its start and endpoint), this means that a TE solution requires at least as many policies as there are demands to be detoured. Especially for larger networks (e.g., ISP backbones or large WANs) that often feature several tens of thousands of demands, this can result in rather high policy numbers even if only a small fraction of those demands needs to be detoured (cf. Section VII-B). However, deploying MO in the context of ISP backbone networks does not only offer benefits because of their high demand numbers, but also because of their rather distinct topological structure, originating from common network 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. [32] or [33, Ch. 2.3]). As a result, ISP backbones often feature so called *Edge 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, multiple of these policies follow basically the same path through the network. When using MO instead, the same routing can often be achieved by installing only a few policies between the respective core routers, that each detour multiple demands at once. For a better understanding, the previously described general ISP topology structure is depicted in Figure 3(c), together with illustrations of exemplary policy configurations using either end-to-end SR or MO. It can be seen that deploying just a couple of MO policies between core routers holds the potential of reducing the number of policies to



(a) Using end-to-end SR: Individual policy for each detoured demand.



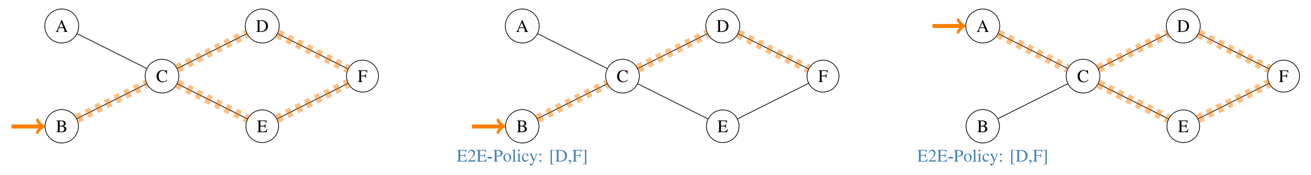
(b) Using MO: Only a few policies between core routers.

**FIGURE 3. Illustration of the capability of MO to reduce policy numbers in ISP backbone-like structured networks.**

configure in the network while achieving comparable traffic steering. This is only a scaled down example, but it is easily imaginable that, for larger ISP backbones with multiple tens of edge routers per PoP, the reduction in policy numbers can be substantially larger. Finally, we want to stress again that the topology structure depicted in Figure 3(c) is not a constructed, theoretical example. It, in fact, results from common network design principles. Hence, most modern ISP backbones probably feature comparable topology structures which, in turn, allows them to benefit from the use of MO over end-to-end SR when it comes to reducing the number of SR policies required to implement TE solutions. This is also confirmed in our following evaluations (cf. Section VII).

### 2) TACTICAL TE

As already pointed out, a major advantage of MO compared to end-to-end SR is its ability to greatly decrease the number of policies that need to be configured in a network to implement TE solutions. This is of great interest for ISPs and operators of other large networks as it reduces configuration effort and improves clarity, maintainability, and robustness of the network (cf. Section II-C). However, what has not been discussed yet, is the fact that being able to compute TE solutions with small policy numbers could be especially beneficial in the use case of *tactical TE*. In this branch of TE, the main objective is to provide and roll out reasonably good TE solutions within rather tight time constraints to quickly fix bad network states (i.e., resulting from failures or unexpected traffic changes) [34], [35]. In such scenarios, substantially reducing the number of policies that need to be configured to “repair” the network could be of great benefit. Configurations with lower numbers of policies are (most likely) also faster to roll out into the network and, thus, can reduce the overall “time-to-repair”.



(a) A certain traffic flow enters the network at node $B$ with destination $F$. Using standard SPR (with ECMP), it follows the path indicated in orange.

(b) Assume that traffic needs to be steered away from the bottom path ($C \rightarrow E \rightarrow F$) for operational reasons. Using end-to-end SR, this can be done with policy $B \rightarrow D \rightarrow F$.

(c) If the ingress point of the flow changes from $B$ to $A$, the dedicated E2E policy on $B$ is bypassed. Reconfiguration is required to steer traffic away from the bottom path again.

**FIGURE 4.** **Simplified example for a scenario in which the use of MO instead of end-to-end SR results in a more "stable" TE configuration that does not require reconfiguration even if the ingress point of traffic changes (within a reasonable extent, e.g., due to peering or BGP changes outside of the network). The MO policy $C \rightarrow D \rightarrow F$ would steer the traffic away from the bottom path, irrespective of whether it enters the network at node $A$ or $B$.**

### 3) SR PATH ENCODING

Each SR label in the segment list takes up additional space (e.g., 32bit for SR via MPLS). As a result, the longer the segment list, the larger the overhead added to each packet. There is a whole subfield of SR research focusing on efficient SR *path encoding* and reducing segment list sizes (cf. e.g., [16]). MO, however, might offer a somewhat implicit solution to this problem as it is no longer required to push the full segment list onto a packet at the ingress node, as it is done by end-to-end SR. Instead, we can use a concatenation of multiple MO policies, each adding just one or two labels after the previous ones are already handled and removed. This can be utilized to lower the maximum number of segments added to a packet along its path, reducing the size of the SR header and, this, also the overhead resulting from it. So instead of, for example, adding a large 6-label long list as a whole at the ingress node, we can iteratively add labels at later stages of the path. This could keep the maximum size of the segment list along the whole path at just three segments. So instead of reserving $6 \times 32bit$ for an SR header with six labels, we only need to reserve $3 \times 32bit$ instead, basically cutting the overhead more or less in half.

### 4) SOLUTION LONGEVITY AND ROBUSTNESS

Lastly, we expect that using MO can also have beneficial effects on the longevity of TE configurations. Some operators run their networks in a highly dynamic and automated fashion, changing and adapting TE configurations every couple minutes. Others, however, are more hesitant when it comes to continuously changing network configurations, especially in fully automated fashion (cf. e.g., [26], [35]). Instead, they prefer more stable configurations lasting longer periods of time (i.e., multiple days or even weeks) that will be tweaked and adapted only when necessary. Deploying MO policies instead of conventional, end-to-end SR holds the potential to further improve the longevity and robustness of TE configurations which we hope to demonstrate with the following example scenario. As most traffic traversing a carrier or ISP network is transit traffic originating and ending in other autonomous systems, interdomain-routing decisions (influenced by e.g., BGP or peering) can have a non-negligible impact on the resulting traffic matrix that an ISP network has to handle [36], [37]. For example, large traffic portions (i.e., originating from large content providers) might normally enter the ISP network at a certain PoP $B$. The ISP knows this and installs dedicated TE policies on this PoP (i.e., based on SR) to send this traffic on the desired intra-domain path. If now, due to changes in the inter-domain routing (BGP), the traffic no longer enters the ISP network at PoP $B$ but instead at another (geographically probably still rather close) PoP $A$, the aforementioned TE configurations on PoP $B$ are bypassed and traffic no longer follows the desired route. As a consequence, reconfiguration is required. For a better understanding, such a scenario is schematically depicted in Figure 4. With end-to-end SR policies, every change in the ingress node requires an adaption or reconfiguration of the network. When deploying MO, however, a configuration can be found that sends the traffic along the desired path, irrespective of it entering the network at node $A$ or $B$. As a result, the configuration is more robust against certain traffic changes and its general longevity is improved.

## C. POTENTIAL CHALLENGES OF MIDPOINT OPTIMIZATION

After having discussed the benefits that MO can offer compared to end-to-end SR, we now take a look at problems and challenges that have to be taken into account when implementing MO in practice, together with suggestions on how those can be circumvented.

### 1) 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 Maximum Segment Depth (MSD) of the used routing hardware. If a packet exceeds the MSD the resulting behavior is undefined and depends on the used hard- and software. However, the packet will most likely be dropped. Hence, it should always be ensured that a SR configuration does not violate the MSD as this could substantially degrade network performance. For end-to-end SR, the MSD limitation is practically irrelevant since virtually optimal results can often be achieved with a very low number of segments (cf. [3]). 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 solutions. However, such checks are 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 SR configuration previously complying with the networks MSD to now violate it. Thus, solutions do not only need to be checked to comply to the MSD 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. A more practical approach to resolve these issues is to simply prohibit policy nesting. This can, for example, be done with so called *Strict Labels* [25, p. 634ff.] [38]. This is a special type of label that guarantees a packet is routed to the interim-destination referenced by the respective label strictly via the shortest path. Other policies along the path will be ignored.

### 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 basically no issue and can be easily prevented. When using MO, however, it becomes possible to build policy (mis-)configurations that result in routing loops, permanently trapping packets of certain demands. Such loops can be configured in two different ways. The first one is rather straight forward and can only occur, if an MO rule set $\mathcal{R}$ is used that allows packets to be steered onto policies whose endpoints are further away from the packets destination (w.r.t. the IGP metric distance) than the respective policy startpoint. This enables the (accidental) configuration of policies that route a packet back to a node on the path it already traversed earlier (on the way from its source to the respective policy startpoint). From there, it follows the same forwarding path as before until it reaches the respective policy's startpoint again. There it is, again, routed back on its path, effectively closing a loop. While this first type of loop is rather easy to detect and prevent (e.g., be choosing a well-suited MO rule set $\mathcal{R}$), the second class of loops is more complex. It can only occur if policy nesting is allowed. If that is the case, there are scenarios in which a packet that already is inside of a policy $p_1$ is steered onto another policy $p_2$ whose associated SR path traverses the startpoint of $p_1$ again. There the packet is steered onto $p_1$ again that brings it back to the startpoint of $p_2$, and so on. As a result, the packet keeps nesting deeper and deeper between those two policies, effectively being trapped in an infinite loop. Similar has also been described in [25, p. 634f.] together with a simple example for such a policy (mis-)configuration.

It should also be noted that the resulting loops are no temporary or *transient* loops that can, for example, occur during the network convergence stage (e.g., after a failure of IGP weight reconfiguration) and basically resolve themselves after the network converged. Instead, they are permanent and need to be manually resolved. Thus, their impact on the network performance can be expected to be substantially more severe as basically all packets of the impacted demands will be trapped in the loop (and eventually be dropped) until the issue is resolved by altering the SR configuration. Hence, routing loops are a serious issue that has to be considered and dealt with when deploying MO in practice.

Similar to checking SR configurations for MSD violations, using dedicated loop-checks to ensure a given SR configuration does not contain loops is not viable in practice, as such a check would not only need to be carried out for the current network state, but for all possible failure scenarios, as well. Therefore, it is desirable to use MO implementations (or rule sets) which implicitly guarantee to never create loops. Fortunately, this is rather easy to ensure. For this, basically only two properties are required: First, policy nesting needs to be prohibited[3] and, second, an MO implementation (with a rule set $\mathcal{R}$) has to be used, that ensures that a packet will only be steered onto a policy if the policy endpoint is closer to the packets destination than its startpoint. The IGP Shortcut approach (cf. Section II-A), for example, inherits this property. Prohibiting policy nesting ensures that a packet entering a policy is guaranteed to also always leave it again. The second property ensures that a packet will always be closer to its destination after exiting a policy. The same holds true for SPR. Consequently, if a packet is guaranteed to not be "trapped" inside a policy and each step (either SPR or a SR policy) brings it closer to its destination, the packet will reach it in a finite number of steps since the original distance to the destination is also finite. Thus, permanent loops are effectively prevented.

### 3) TRAFFIC STEERING CAPABILITIES

One of the biggest advantages of end-to-end SR is the detailed per-flow traffic control it offers. Since each policy is dedicated to route exactly one demand, end-to-end SR offers the possibility to individually optimize the routing path of each demand, often enabling near perfect routing. When using MO, however, policies are no longer dedicated to route just a single demand but, instead, potentially influence the path of multiple demands at once. Intuitively, one would

[3]This is also suggested in [25, p. 634f].

expect that this results in a deterioration of the overall traffic steering capabilities and thus also the achievable TE solution quality, as demands can no longer be individually controlled with dedicated policies. This intuition, however, is misleading. From a theoretical perspective, there is actually no deterioration in traffic steering capabilities when using MO instead of end-to-end SR. The reason for this is that, if needed, the routing of every end-to-end SR solution can be mimicked with a respective MO policy configuration if policy nesting is prohibited. Then, a "full-mesh" policy configuration that installs a MO policy between each pair of nodes in the network can be deployed. This results in each demand being steered onto an (implicitly "dedicated") MO policy between its source and destination. Of course, such a full-mesh configuration would require an immense number of MO policies ($\mathcal{O}(|V|^2)$) to be configured. This (at least to a certain extent) contradicts the general idea of MO being able to reduce policy numbers compared to end-to-end SR (cf. Section IV-B). However, this shows that, from a theoretical perspective, using MO instead of end-to-end SR does not result in a deterioration of traffic steering capabilities, per se. As a result, solutions of similar quality are achievable when using MO (also shown in Section VII-A).

In fact, MO actually even offers the possibility to improve solution quality when it comes to practical implementations of SR. In current state-of-the-art SR TE algorithms (e.g., [39] or [18]) the number of segments is often limited to two or three. This is either done due to practical constraints (i.e., MSD) 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 can be required (cf. [18, Fig. 2]). 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 MSD limit. MO, however, can benefit from the fact that it is able to mimic higher segment paths via a concatenation of multiple policies with fewer segments.

An example for such a scenario is given in Figure 5. The topology contains two types of edges. The thinly drawn edges denote links that have a low metric and low capacity and the thick edges denote links that have high capacity but also a high metric. Assume that a traffic flow has to be routed from node $A$ to node $C$ and that this flow is at least more than twice as large as the capacity of the thin edges. To prevent congestion, it has to be rerouted over the thick edges instead, which offer sufficient capacity. With conventional SR limited to policies with at most two segments (2SR), every possible policy configuration would still result in overutilization, since traffic will always pass over at least one of the low-capacity edges (assuming link metrics are not adopted). However, when using MO, a concatenation of two 2SR MO policies ($A \rightarrow D \rightarrow B$ and $B \rightarrow E \rightarrow C$) can be used to ensure that the traffic flow is routed only over the high-capacity

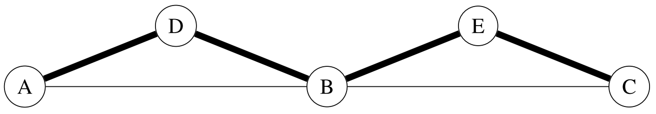


**FIGURE 5.** Example for a simple topology where the optimal routing can be achieved with a concatenation of MO-capable 2SR policies, but not with conventional 2SR.

edges. As a result, the optimal MLU can only be achieved when using MO, while the conventional SR solution can be arbitrarily worse depending on the chosen link capacities and traffic volumes. This worst-case example can be extended for k-SR with arbitrary many segments by simply concatenating more triangles to the above topology.

## V. LP-BASED OPTIMIZATION ALGORITHM

In this section, we propose a new LP-based SR TE optimization algorithm that utilizes the MO concept and aims at minimizing the network MLU while simultaneously minimizing the number of MO policies required to do so.

### 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: The IGP Shortcut approach as defined for MPLS in [7]. The respective rule set $\mathcal{R}$ for deciding whether a packet that traverses a router on which an MO policy is configured will be steered onto the respective policy comprises of just one rule: *A packet will be steered into the policy if the policy endpoint lies on the IGP shortest path from the policy startpoint to the destination of the packet.* Our reasoning for choosing this specific MO variation is explained in the following.

First, a crucial advantage of this approach is the fact that it makes it impossible to configure policy-induced loops[4] if policy nesting is prohibited, as already discussed in Section IV-C. Second, 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. Third, a similar approach was found to perform best with regards to the integration of MPLS tunnels into the IGP [8]. And lastly, according to the respective documentations (e.g., [25]), the IGP Shortcut approach seems to be the standard MO approach that is supported in recent routing hard- and software by some of the large vendors. This makes our optimization algorithms actually usable in practice.

[4]While this prevents policy-induced routing loops, it still allows for structures referred to as *weak-loops* [40]. 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 [40] that they can even offer actual benefits with regards to certain TE objectives (e.g., MLU or policy number minimization).

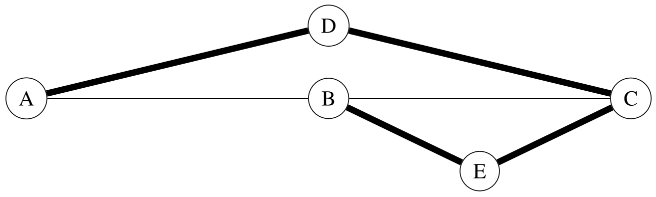


FIGURE 6. A worst-case example in which the best MLU obtained with the SC2SR algorithm is arbitrarily worse than the actual optimal MLU that is theoretically achievable with MO, resulting from the prohibition of "influencing" policies.

## B. CIRCUMVENTING THE POLICY DEPENDENCY PROBLEM

As already explained at the end of Section IV-A, solving the general MO optimization problem (cf., problem P2) optimally with linear programming is not feasible due to the policy dependency issue. Therefore, in this section, we present a new algorithmic approach that aims to circumvent this issue by adding additional restrictions to the explored solution space, enabling the use of linear programming to solve a slightly more restricted version of the problem. 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.

The above limitation of the solution space allows for an efficient problem formulation. However, the downside of such an approach is that it, strictly speaking, makes our algorithm a heuristic one as solutions can (in theory) become arbitrarily worse than solutions to the unrestricted MO optimization problem (problem P2). An example for such a worst-case scenario is given in Figure 6. It depicts an exemplary topology for which the MLU obtained with our algorithm can be arbitrarily worse than the actual optimal MLU achievable with the MO approach. Analogously to the previous examples, the thinly drawn edges denote links with low metric and capacity while the thicker ones denote links with high metric and capacity. Assume that two traffic flows have to be routed: One from $A$ to $C$ and one from $B$ to $C$. Further assume that each of these flows is larger than the capacity of the thin edges. Hence, we need to detour these flows from their standard shortest paths in order to prevent congestion. When utilizing MO with at most two segments per policy, the only way of doing this is installing the policy $A \rightarrow D \rightarrow C$ to reroute the traffic flow between $A$ and $C$ over the two upper thick-drawn links, and also installing the policy $B \rightarrow E \rightarrow C$ to reroute the traffic flow between $B$ and $C$ over node $E$. However, since, without the policy $A \rightarrow D \rightarrow C$ the traffic between $A$ and $C$ would also be routed over policy $B \rightarrow E \rightarrow C$, the former policy "influences" the traffic that passes through the latter. Hence, it is included in its set of influencing policies $\mathcal{I}_{BC}$. As a result, our algorithm is prohibited to configure both policies at the same time and, therefore, would not be able to find the above optimal solution. For this example, this configuration is also the only one to obtain the optimal MLU. Thus, our algorithm finds a sub-optimal solution. By setting the respective traffic and capacity values accordingly, the difference between the optimal solution and the solution obtained by our algorithm can be made arbitrarily large.

While the above considerations are very important for a theoretical, worst-case analysis of our algorithm, we believe that they are of lesser relevance in practice. This is backed up by the fact that we are able to show that our algorithm is still able to obtain virtual optimal MLUs on nearly all of our (real-world) evaluation instances (see Section VII-A), irrespective of these theoretical limitations.

## C. SC2SR OPTIMIZATION MODEL

Since link loads can now be precomputed in reasonable time, this enables the formulation of an LP. We refer to it as Shortcut 2SR (SC2SR) because it is based on IGP Shortcut and utilizes policies with up to two segments. The optimization problem is expressed as follows:

$$\begin{aligned}
\text{P3}: \min \quad & \theta && && \text{(3a)}\\
\text{s.t.} \quad & \sum_{l} x^{l}_{km} = y_{km} && \forall km && \text{(3b)}\\
& y_{km} + y_{ij} \leq 1 && \forall km && \\
& && \forall ij \in \mathcal{I}_{km} && \text{(3c)}\\
& \left( \sum_{ij} t_{ij} \sum_{klm} \mathit{diff}^{klm}_{ij}(e)\, x^{l}_{km} \right) && \forall e && \\
& + \mathit{spr}(e) \leq \theta\, c(e) && && \text{(3d)}\\
& x^{l}_{km} \in \{0,1\} && \forall klm && \text{(3e)}\\
& y_{km} \in \{0,1\} && \forall km && \text{(3f)}
\end{aligned}$$

The objective still is the minimization of the MLU $\theta$. However, when compared to the 2SR formulation (problem P1), completely different types of variables are used since policies are not bound to specific demands anymore. The binary decision variables $x_{km}^{l}$ indicate whether a 2SR 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 segment.[5] The first constraint (Equation (3b)) 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 (3c), 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 (3d) together with the objective function is responsible for minimizing the MLU. Its general idea is similar to Equation (1c) of problem P1 (2SR). For each edge, we compute the amount of traffic that results from the current policy configuration and ensure that it does not exceed $\theta$ 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 $diff_{ij}^{klm}(e)$ 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. Again, all constraints of the optimization problem are of linear nature, allowing it to be solved with commercial LP solver like CPLEX [17].

The LP as presented in problem P3 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 III-B) in which (virtual) nodes correspond to many routers in practice, using these nodes as start- or endpoints or intermediate segments should be avoided as, otherwise, we run into the same policy multiplication problem. 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 [18]. 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* $\lambda$. The structure of the SC2TLE LP is similar to problem P3, apart from two adaptions. First, the objective is changed from MLU to policy number minimization. This is done by replacing Equation (3a) with the following one.

$$\min \sum_{km} y_{km} \tag{4}$$

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

$$\theta \leq \lambda\theta' \tag{5}$$

It ensures that the MLU $\theta$ of the newly computed SC2TLE solution does not surpass the optimal MLU $\theta'$ of the preceding SC2SR optimization by more than the user-defined trade-off coefficient $\lambda$. 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 version 20.1.0 [17].

### 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 in many networks that we want to improve on with our TE approaches. The second algorithm is Multicommodity Flow (MCF) [41, Ch. 4.4]. It can be used to compute the theoretically best achievable MLU that can be realized with any kind of traffic steering. However, it has to be noted that MCF solutions are generally not really deployable in practice since it ignores indispensible real-world constraints and restrictions (e.g., the infeasibility of splitting traffic flows into arbitrary fractions). Nonetheless, MCF is useful in that sense that it allows to obtain a lower bound for the best achievable MLU that we can compare our algorithms against. If they achieve solution qualities comparable to MCF, we know that these solutions are virtually optimal. Finally, to compute the MLU achievable with end-to-end SR we use the 2SR algorithm (problem P1).

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 III-B, this is often not feasible for larger networks for reasons of scalability. To nonetheless get a better approximation of

[5]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.

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 formulated as

$$\text{P4}: \min \ \theta \tag{6a}$$

$$\text{s.t.} \ \sum_k x_{ij}^k = Z_{ij} \qquad \forall ij \tag{6b}$$

$$\sum_{ij} \frac{t_{ij}}{Z_{ij}} \sum_k g_{ij}^k(e) x_{ij}^k \le \theta\, c(e) \qquad \forall e \tag{6c}$$

$$x_{ij}^k \in \mathbb{N}_0 \qquad \forall ijk \tag{6d}$$

The $Z_{ij}$ values denote the number of sub-demands in which the demand between nodes $i$ and $j$ can be split. For example, a demand between two virtual nodes corresponding to four real routers each, 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 [18]. 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. IGP metrics are set according to a preceding metric optimization carried out by the operator. For our evaluations, we were given 19 snapshots from between March 2017 and January 2021 with traffic matrices located in the daily peak-hour (generally between 9 and 10 o'clock in the evening). Due to continuous expansions of the network, the respective topology varies between snapshots but, on average, it comprises around 143 nodes and 900 edges and has a diameter of about 7 (see Table 1 for more topology characteristics).

It is common knowledge that ISPs generally tend to over-provision their networks in order to preemptively account for changes in traffic characteristics or other unexpected events (cf. e.g., [42]). As a result, overutilization rarely occurs even when not using sophisticated TE approaches but solely relying on SPR. The same can be observed for our aforementioned ISP dataset. Hence, while they represent real use cases in a real ISP backbone, it could be argued that, from a more theoretical perspective, these instances might not be that challenging since overutilization can be prevented without using SR at all. Therefore, with our second dataset, we aim to adapt and alter problem instances from our aformentioned ISP dataset to create TE instances that are more challenging to optimize. The basic idea is to map 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 [43], this forces more traffic through a network with lower capacity. For this, topology information is again taken from different network snapshots of the ISP backbone network between 2017 and 2021, but we change the related traffic matrix to a more recent one. 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 ISP datasets feature data that is virtualized in a similar fashion as described in Section III-B. Here, a single virtual node corresponds to 50 to 150 edge routers in practice.

While our first two datasets feature different expansion states of the ISP backbone, they are nonetheless both based on the same singular network. To also evaluate MO on a plethora of different networks, our third dataset features instances from the publicly available Repetita dataset [29]. This dataset contains topologies of real-world networks (mostly WANs or ISP backbones) collected in the *Internet Topology Zoo* [30] and artificially generated[6] traffic matrices for each topology. All of these traffic matrices are intentionally designed in a way that the optimal MLU (e.g., obtainable with MCF) for each instance is always 0.9 (corresponding to 90% utilization). Most instances also feature SPR MLUs grater than 1.0, corresponding to network overutilization. This is done to create especially challenging instances for the evaluation of TE approaches. However, it has to be noted that such high utilization scenarios very rarely occur in practice (basically only in the case of multiple crucial hardware failures or outages or unexpected traffic surges). Hence, these instances (similar to our backmapped ISP dataset) can be considered "harder" than most TE scenarios that are encountered in a practical, real-world deployment. The Repetita dataset contains a large portion of rather small topologies with just a couple of nodes, mostly from fairly old networks like the Arpanet. Since we see the main use case of MO and related approaches in substantially larger networks (e.g., modern ISP backbones and WANs), we argue that such small instances are of lesser interest for us and modern TE in general. Therefore, we limit our evaluations to networks that have at least 40 nodes.[7] Furthermore, we also exclude those instances that are already solved optimally by SPR as they are of no interest for more sophisticated TE approaches. Overall, this leaves us with a total of 72 instances in this dataset. Those instances are often provided on a PoP-level and, thus, likely also feature virtual nodes. However, there is

[6]Traffic matrices were generated using a *random gravity model* [44].
[7]Similar was, for example, also done in [39] or [40].

TABLE 1. Graph properties of the topologies in the three datasets used for evaluation.

| | Original ISP (19 Topologies) | | | | Backmapped ISP (10 Topologies) | | | | Repetita (72 Topologies) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | min | max | avg | stdDev | min | max | avg | stdDev | min | max | avg | stdDev |
| Nodes | 108 | 186 | 143.11 | 29.90 | 108 | 165 | 141.9 | 25.95 | 40 | 197 | 68.69 | 31.81 |
| Edges | 660 | 1064 | 897.16 | 136.25 | 666 | 1024 | 901.0 | 134.94 | 86 | 486 | 171.94 | 77.31 |
| Density [%] | 3.09 | 6.57 | 4.73 | 1.35 | 3.71 | 6.22 | 4.72 | 1.15 | 1.26 | 7.82 | 4.3 | 1.48 |
| Diameter | 6 | 8 | 7.32 | 0.58 | 6 | 8 | 7.3 | 0.67 | 4 | 35 | 11.79 | 7.57 |

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.

Table 1 lists the most important graph properties across the respective topologies for each of the three dataset used in our evaluation. Regarding the number of edges, parallel links are counted as just one edge and the density characterizes the ratio of (non-parallel) edges in the graph relative to a complete graph with the same number of nodes.

## 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 7(a) 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 solutions of virtual optimal quality (as it matches MCF results). 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 7(b)). 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, it could also be a result of the limitations regarding *influencing* policies implemented into our algorithm (cf. Section V-B) 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 7(c). Since the Repetita dataset features substantially more instances than the other two, visualizing results for each individual instance is not really feasible anymore. Hence, for the sake of readability, Figure 7(c) instead depicts the distributions of the MLUs across all instances for the respective algorithms. The dashed green line marks the theoretical optimal solution value of 0.9 obtained by MCF.[8] It can be seen that 2SR as well as SC2SR are able to obtain (near-)optimal MLUs close to 0.9 for nearly all instances, apart from a few outliers. Nonetheless, it also has to be noted that SC2SR tends to generally achieve slightly worse MLUs than 2SR. However, most of the latter differences are marginally small (often $< 1\%$). In fact, such subtle differences are probably not even noticeable in a practical deployment. Traffic, while mostly being quite stable and predictable, is still subject to small ongoing variations which (most likely) cover up such marginal MLU differences. Furthermore, it has to be remembered that the Repetita instances resemble intentionally challenging TE scenarios that are (most likely) harder than most scenarios that would occur in a practical deployment (cf. Section VI-B). And even in such worst-case scenarios, our SC2SR algorithm is able to find near-optimal solutions and prevent overutilization in virtually all cases. Overall, the evaluation on the Repetita dataset confirms our previous findings that our MO algorithm is able to find (near-)optimal solutions comparable to those of 2SR, for a wide range of 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

[8]The traffic matrices of the Repetita dataset are all designed in a way that for each instance the optimal MLU obtained by MCF is exactly 0.9.

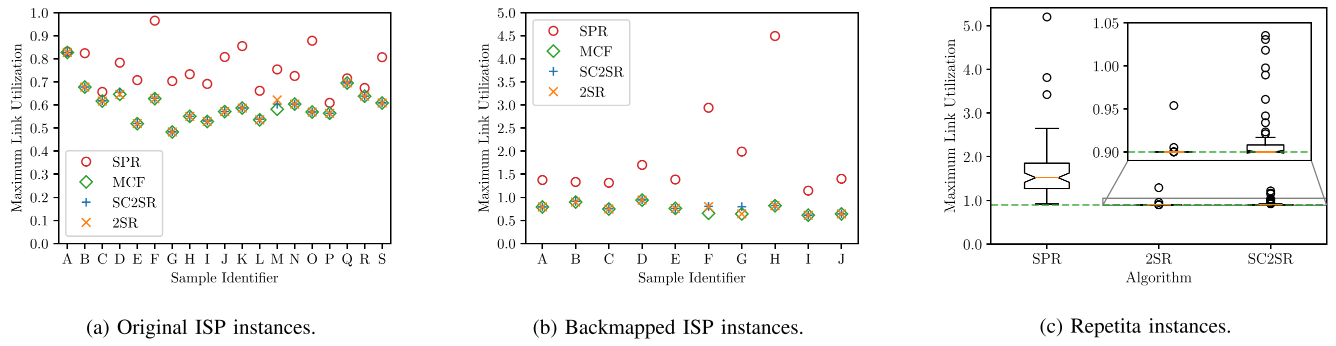


**FIGURE 7.** **MLUs achieved with different algorithms.**

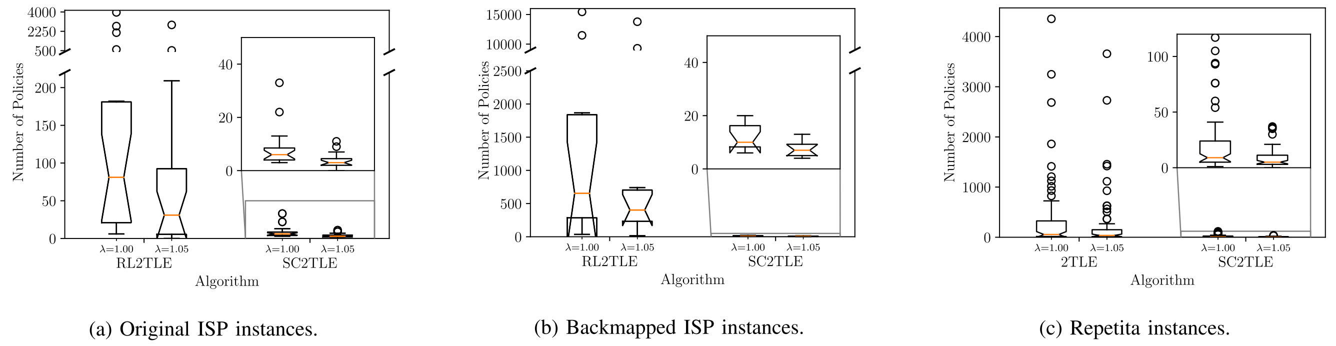


**FIGURE 8.** **Number of SR policies required by different algorithms.**

(cf. Section V-B). 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 8(a). 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-B. 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 8(b)), 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 [18] to calculate the number of policies required by end-to-end SR. The respective results are depicted in Figure 8(c). It can be seen that, even without virtual nodes, our SC2TLE algorithm is still able to achieve a substantial reduction in the number of required policies, from multiple hundreds or even thousands to less than 50 for nearly all instances. For example, for the `Cogentco` instance (the largest instance in the Repetita dataset) 2TLE requires over 1400 policies, even when used with a trade-off coefficient of 5%, while SC2TLE achieves an optimal MLU of around 0.9 with just 19 policies. This impressively illustrates that the benefits of our SC2TLE algorithm or MO in general are not limited to large networks and virtualized data, but apply to smaller, unvirtualized networks as well.

Since the respective topology and traffic data of our ISP datasets cannot be made publicly available, we instead provide detailed information on policy numbers for each of our ISP instances in Table 4 in Appendix A. 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 that were obtained with the standard 2TLE algorithm. Those are computed by applying a weighting to virtual policies corresponding to the actual number of policies required to implement them in a practical deployment. The respective LP formulation is given in problem P5 in Appendix B and the results are listed in the column labeled "w2TLE" (weighted 2TLE).

## C. IMPACT OF THE POP VIRTUALIZATION FACTOR

As shown in the previous section, SC2TLE is able to substantially reduce the number of policies required for TE solutions. An interesting question that remains to be answered, however, is how the number of edge routers per PoP influences the overall reduction of policies. For our ISP instances, a single virtual node corresponds to 50 to 150 edge routers in practice (cf. Section VI-B). This is a quite high number as we are dealing with the backbone network of one of the world's largest ISPs. In other, smaller networks this *virtualization factor* might be much lower with a virtual node corresponding to just a couple edge routers. Hence, the number of policies required by end-to-end SR will (presumably) be lower as well.

To a certain extent, this is already covered by the evaluations carried out on the Repetita instances. For those, a virtualization factor of 1 was assumed (meaning that each node in the topology does correspond to just one router in practice). The results show that even for very small virtualization factors (or in this case no virtualization at all) SC2TLE is still able to achieve substantial reductions in policy numbers compared to end-to-end SR approaches.

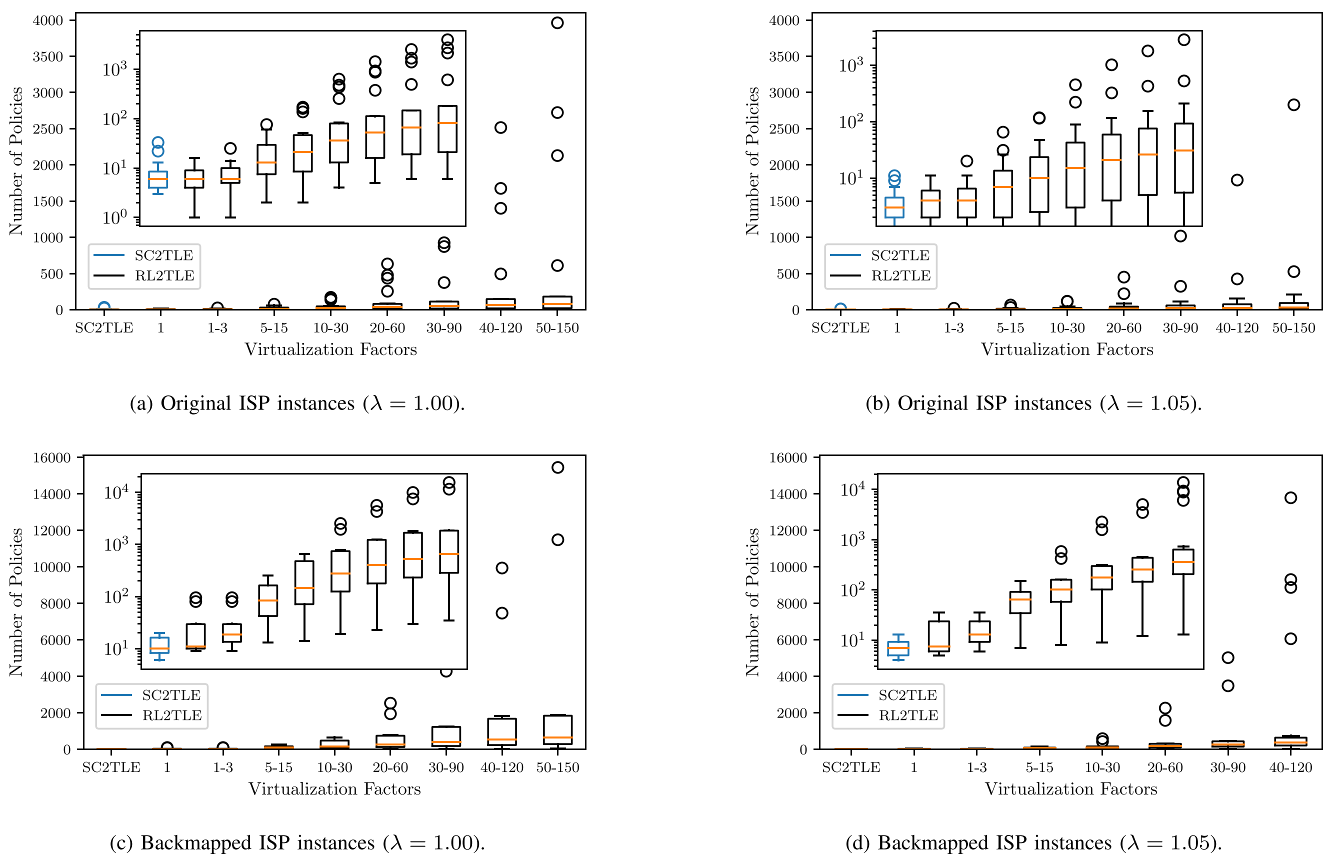
Nonetheless, to allow for a more substantiated conclusion, we conduct further evaluations based on our ISP data. For this, we repeat the RL2TLE computations for these instances with a variety of different virtualization factors ranging from 1 to the original value of 50 to 150 edge routers per virtual node. The results are depicted in Figure 9. Each subplot shows the distribution of the number of policies required by the RL2TLE algorithm across all instances of the respective dataset for different virtualization factors. For reference, each plot also contains the distribution of the number of policies required by SC2TLE (blue boxplot). Since this number is independent of the virtualization factor it stays the same across all runs and is, therefore, only included once. Across all four evaluations (irrespective of the used dataset and trade-off coefficient), it can be seen that, as expected, the number of policies required by RL2TLE decreases if we lower the virtualization factors. However, even if we compare it to the policy numbers of these lower virtualization factors, SC2TLE is still able to achieve substantial reductions for many of them. For example, even if we reduce the virtualization factor to just a tenth of its original value (5–15 edge routers per virtual node), RL2TLE still requires substantially more policies than SC2TLE, especially on the backmapped ISP instances. In fact, even for a virtualization factor of just 1 (meaning no virtualization at all) the number of policies required by SC2TLE is either basically on-par with end-to-end SR (Figures 9(a) and 9(b)) or already quite a bit lower (Figures 9(c) and 9(d)). Together with our results obtained on the Repetita instances, this further supports the observation that, while SC2TLE (and MO in general) are most beneficial in larger networks with many edge routers per PoP, they can also provide substantial benefits for smaller networks with less hardware at the edge as well.

## D. COMPUTATION TIMES AND RESOURCE DEMANDS

Like most LP-based approaches, SC2TLE is quite demanding with regards to resources and computation times. While for smaller to medium sized networks, optimization can still be done within seconds or at most a couple of minutes, for some of the largest 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. Additionally, we recently were able to show that a large portion (up to over 97%) of the theoretically configurable SR policies can be ruled out prior to optimization without resulting in a substantial deterioration in solution quality [45] and that this can be utilized to reduce the computation time of LP-based optimization algorithms (including SC2SR) by a factor of ten or more. However, while this further facilitates the applicability of our algorithm for different scenarios, the latter is still not suited for quick, tactical re-optimizations (e.g., in failure scenarios) that require solutions to be computed within seconds. For such scenarios, dedicated heuristics need to be developed. For completeness, it should be noted that we propose such a heuristic in [26], showing that MO configurations of very good quality can also be computed in substantially less time (i.e., a couple of seconds). While this, in our opinion, further emphasizes the usability of MO for SR for various use cases, reporting on these results in detail would exceed the scope of this paper. Hence, we refer the interested reader to [26] for detailed information, instead.

## VIII. INTEGRATION OF LATENCY BOUNDS

As seen in the previous sections, our MO-based optimization algorithm is able to achieve exceptional results regarding MLU and the number of policies. However, in practice,



(a) Original ISP instances ($\lambda = 1.00$).

(b) Original ISP instances ($\lambda = 1.05$).

(c) Backmapped ISP instances ($\lambda = 1.00$).

(d) Backmapped ISP instances ($\lambda = 1.05$).

**FIGURE 9. Evaluation of the impact of the virtualization factor on the number of policies required by the RL2TLE algorithm on the two ISP datasets (*original* at the top and *backmapped* at the bottom). Policy numbers of SC2TLE are also included for reference (blue boxplot). For better readability, the smaller inset plots show the same results with a logarithmic scale.**

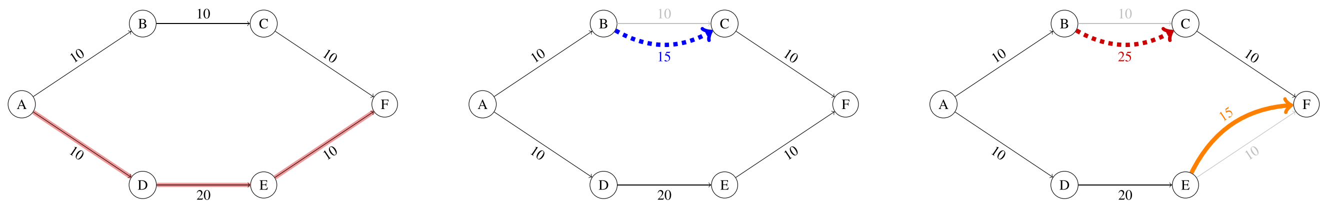
there often are further requirements arising from the side of network operation and management that TE solutions must adhere to (see e.g., [18]). Often, contracts between ISPs and their business customers feature so called Service Level Agreements (SLAs), in which the ISP guarantees a certain quality of service regarding various aspects like availability or time-to-repair.[9] A crucial TE-related requirement in many of these SLAs is the fulfillment of certain *delay constraints* or *latency bounds* for specific traffic demands. With these the ISP guarantees to deliver traffic within a certain specified timelimit. They can range from quite loosely requirements (e.g., a certain average network delay) to very rigorous ones like guaranteeing that traffic between two sites will always be transferred within a certain maximum delay. For ISPs, it is of utmost importance to fulfill these agreements, as otherwise they violate their contracts, resulting in a potential loss of reputation and money. Hence, in order to provide practically deployable solutions, TE algorithms have to incorporate additional constraints resulting from SLAs into their computations. Therefore, in the following, we present and evaluate an approach to incorporate latency bounds into our optimization algorithm to prevent the computed SR detours to exceed such bounds.

[9]e.g., https://www.verizon.com/business/terms/us/products/internet/sla/

## A. PROBLEM STATEMENT

Before talking about minimizing path delays or ensuring that path delays adhere to certain latency constraints, we first need to define the term *path delay* for a demand. When there are only simple shortest path (i.e., no ECMP), this definition is rather straightforward. We simply sum up the individual link delays[10] across all the links traversed by the path from the demand's source to the sink. In the context of ECMP and multi-path, however, this becomes a bit more complicated. Here, we potentially have multiple different equal-cost shortest paths between two nodes. This is no issue if the metric based on which the shortest paths are computed corresponds to the link-delays. In this case, all resulting individual simple paths, by definition, have the same delay. If, however, the used metric does not resemble the link-delays, the simple paths in the ECMP forwarding multi-graph between two nodes might all have different delays. Hence, the definition of the end-to-end delay for a demand is not that straightforward anymore. Depending on the use case, arguments can be made for various kinds of delay definitions. For this paper, the objective is to make sure

[10]For simplicity, we assume that the delay assigned to a link already incorporates all relevant delay factors (e.g., processing or queuing delays arising on router-level, etc.). Hence, those are not taken into account individually in our calculations.



(a) ECMP forwarding graph for SPR. Demand delay for $A \rightarrow F$: 40.

(b) Single policy on non max-delay path. Delay change for $A \rightarrow F$: $\pm 0$.

(c) Multiple Policies. Delay change for $A \rightarrow F$: $\pm 0$.

**FIGURE 10. Example for the policy dependency problem with respect to delay changes. Intermediate segments of policies are not relevant here and, hence, omitted for clarity of presentation.**

that our SR solutions do not exceed certain latency bounds. In this context, the most sensible definition for the delay of an ECMP path between two nodes is using the maximum delay across all the simple paths in the respective ECMP forwarding graph. If this maximum adheres to the latency bounds, we can be sure that irrespective of which specific simple path traffic will be routed over, it will never exceed the specified bound.

Now that we have defined a demand's delay in the context of ECMP, we can finally provide a proper definition of the latency bound problem. Assume that for each link in the network we have information on its delay and for each demand $d \in D$ we are given a latency bound $lb_d$. Now, the objective is to ensure that with our computed TE solutions the selected forwarding path for each demand complies with these bounds, meaning that the end-to-end delay of demand $d$ is at most $lb_d$:

$$delay(i \rightarrow j) \leq lb_d \qquad \forall d = (i, j) \in D \tag{7}$$

In the context of end-to-end SR, this can be implemented into an LP in a rather straightforward fashion. A demand is either routed via the shortest path whose delay can be easily precomputed or via a configured end-to-end policy. If the number of intermediate segments is limited and traffic splitting over multiple policies is prohibited (like it is for the 2TLE LP), the delay resulting from such a policy can also be precomputed. The constraint that has to be added to the 2TLE LP could then, for example, look like this:

$$\sum_{k} delay(i, j, k)\, x_{ij}^{k} \leq lb_d \qquad \forall d = (i, j) \in D \tag{8}$$

Here, $delay(i, j, k)$ denotes the end-to-end delay of the demand between $i$ and $j$ if node $k$ is used as intermediate segment. These values can be efficiently precomputed as they are independent of other policies configured in the network due to the end-to-end nature of conventional SR policies.

## B. COMPUTATIONAL CHALLENGES IN THE CONTEXT OF MO

Unfortunately, computing the impact of a policy on a demand's delay is significantly more complicated in the context of MO. To correctly compute the delay changes resulting from inserting/removing a policy, we need to compute the delay difference relative to the maximum delay subpath. This subpath, however, can be changed by other policies configured in the network. As a result, we run into a similar problem as the policy dependency problem discussed earlier in Section IV-A. To be able to precompute the delay changes resulting from individual policies for using them in our LP formulation, information on (potentially all) other configured policies is required. However, since this information is not available prior to optimization and iterating over all possible policy variations is not feasible, we cannot efficiently precompute these values to use them in our LP formulation.

For a better understanding, an example for these issues is given in Figure 10. On the left, the ECMP forwarding graph for the demand $A \rightarrow F$ is depicted. For this demand, there are two equal-cost shortest paths that can be used for routing. The upper path has a delay of 30ms and the lower one a delay of 40ms. As explained earlier, the overall path delay for the demand $A \rightarrow F$ is the maximum delay value across all simple forwarding paths, in this case 40ms (the respective path is indicated in red). Now assume we want to add a SR policy to the network, e.g., between nodes $B$ and $C$ (see Figure 10(b)). To determine the impact of this policy on the end-to-end delay of demand $A \rightarrow F$, the first intuition might be to simply subtract the delay of the sub-path that is "bridged" by the policy (in this example the path from $B$ to $C$ with a delay of 10ms) and to then add the path delay resulting from the new policy (15ms). This, however, is not correct as it would yield a delay increase of 5ms to a total delay of 45ms instead of the correct value of still 40ms. To correctly compute the resulting delay changes, we need to compute the delay difference relative to the maximum delay subpath, not to the subpath the policy is added to. This is more complex but still doable in reasonable time when focusing on individual policies and their differences to the maximum delay subpaths resulting from SPR.

However, in the context of MO, a demand might not only be routed through a single policy but through multiple ones. As as result, the current maximum delay subpath might not correspond to the one from the SPR "base case", since it can be altered by other, already installed policies. An example for this is given in Figure 10(c). Here, we want to add the red policy between nodes $B$ and $C$ with a delay of 25ms. Compared to the standard SPR case (cf. Figure 10(a)) this would normally increase the overall delay of the demand

$A \to F$ by 5ms. However, in this case there already is another policy installed (indicated by the orange arrow) that further increased the delay of the lower, maximum delay subpath. Thus, adding the red policy does not increase the overall delay for demand $A \to F$, but would do so if the orange policy was not there.

In the above examples, we assumed that the metric used for SPR calculations does not correspond to the link delays. However, even if those two are equivalent, scenarios can be constructed for which we run into the same problems as depicted in Figure 10. For example, as soon as there is at least one policy configured on one of the ECMP paths, we potentially have different delays across them, even though link metrics correspond to the delay. This basically leads to the same "starting situation" as it is depicted in Figure 10(a). The only difference is the higher delay of the bottom path does not result from link $D \to E$ having a higher delay by default. Instead, it comes from an already configured policy between nodes $D$ and $E$ that increases the delay between these two nodes from 10 to 20. Then, we can basically follow the same argumentation as earlier even though now we actually fulfilled the premise of having the link metrics correspond to the delays.

### C. INCORPORATING LATENCY BOUNDS INTO THE SC2TLE ALGORITHM

As seen above, incorporating latency bounds into the SC2SR LP is no trivial task. In fact, it (most likely) is not possible to fully incorporate them in an efficient way for reasons similar to those that prevent an efficient LP formulation of the complete IGP Shortcut optimization problem in the first place (cf. Section IV-A). Since both problems suffer from the same underlying issue (i.e., dependencies between policies), a straightforward approach would be to also circumvent this issue in similar fashion. If we, again, prohibit the simultaneous configuration of policies that influence the delay changes resulting from one another, we would prevent all issues arising from policy dependencies. As a result, we could efficiently precompute the delay changes that arise from individual policies and use them in our LP without having to consider the sideeffects of other policies in the network. Such an approach, however, would further limit the searched solution space by prohibiting even more technically feasible solutions for the sake of efficient computation. We have shown in Section VII-A that the first set of artificial limitations does not seem to have a significant impact on the overall solution quality in practice. However, imposing further limitations onto the searched solution space might "push things over the limits", resulting in a deterioration of solution quality. This assumption is further backed up by the fact that the set of influencing policies with respect to the delay computations is somewhat complementary to the set of influencing policies with respect to link utilization. While the latter basically consists of policies that lie "on the same path" (cf. Figure 2), the newly added set of influencing policies mostly consists of "parallel" policies that lie on different ECMP sub-paths (e.g., like the red and orange policies in Figure 10(c)). As a result, both sets are probably mostly disjoint and, hence, adding this second set of prohibitory constraints would impose a substantial number of new restrictions onto the problem. For these reasons, we decided to, instead, follow a different approach to incorporate latency bounds into our LP formulation.

This approach is based on the observation that the main problem is as follows: Without knowledge of the other policies in the network, we do not know whether a policy is added into the simple ECMP (sub-)path with maximum delay across all alternative ECMP paths for a demand or into one of the other paths with lower delay. Hence, we also do not know the effect of the policy onto the overall end-to-end delay for the demand. However, exact knowledge on this is actually not strictly necessary for our use case. We do not need solutions that minimize demand delays. Instead, it is sufficient to find any solution that adheres to the specified latency bounds. Therefore, we decided to not rely on exact delay computations, but to simply use an upper bound estimation of the delay changes instead. For every policy for which we cannot exactly determine its impact on the end-to-end delay for a demand because of dependencies towards other policies, we simply assume that this policy is actually added to the current maximum delay ECMP path. Hence, we can simply add the resulting delay difference between the path of the newly added policy and the "bridged" subpath between its start- and endpoint to the current end-to-end delay of the respective demand. This way, we might overestimate some of the demand delays, but if a solution is found for which even those potentially overestimated delays adhere to the latency constraints, the actual delays will do so, too.

For a better understanding of this upper bound estimation, we can go back to the example shown in Figure 10(b). The blue policy is not added on the maximum delay ECMP path (with a delay of 40ms) but on a path with lower delay (30ms) instead. The former delay between nodes $B$ and $C$ (the policies start- and endpoint) is 10ms but the newly added policy has a delay of 15ms. Hence, the overall delay of the top path increased by 5ms, but the maximum demand delay between nodes $A$ and $F$ stayed the same because the delay increase actually happened on a non-maximum ECMP path. However, without information on other policies in the network, we cannot know this. Hence, we have to assume that the delay increase actually happened on the maximum delay ECMP path. Therefore, we add the 5ms increase to the previous demand delay of 40ms. This results in a new upper bound estimation of the demand delay of 45ms, even though the actual delay still is 40ms.

All in all, we add the following constraint into our SC2SR formulation (problem P3) to incorporate latency bounds.

$$delay_d + \sum_{klm \in \mathcal{E}_d} delay\_diff_d^{klm} \, x_{km}^l \leq lb_d \quad \forall d \in D \qquad (9)$$

Here, $delay_d$ corresponds to the SPR delay of demand $d$, if no SR policies are installed. For each MO policy that is in the set $\mathcal{E}_d$ of eligible policies for demand $d$ (i.e., those policies that are eligible to route the demand with respect to the IGP Shortcut rule given in Section V-A), the respective upper bound estimation of the delay difference (*delay_diff*) resulting from the respective policy is added. This delay difference can be calculated in two different ways depending on the scenario. The first scenario is the one explained earlier, in which the exact influence of a policy on a demand's delay cannot be computed without the knowledge of other policies in the network. In this case, we use the upper bound estimation for the delay. However, there is a second scenario in which we are actually able to exactly compute the influence of certain policies on a demand's delay, even potential delay reductions. This is feasible for all those policies between nodes that route 100% of this demands traffic as this implicates that there is no other parallel ECMP path that needs to be taken into account. Hence, the resulting delay change is not dependent on other policies (apart from those influencing ones that are already prohibited in the original SC2SR formulation) and can be efficiently precomputed. This leaves us with the following equation for calculating the delay differences in which $\Delta^l_{km}$ denotes the delay difference between the SPR path from $k$ to $m$ and the 2SR path from $k$ over $l$ to $m$.

$$delay\_diff^{klm}_d = \begin{cases} max(0, \Delta^l_{km}) & \text{if } klm \notin \mathcal{P}_d \\ \Delta^l_{km} & \text{if } klm \in \mathcal{P}_d \end{cases} \quad (10)$$

## D. EVALUATION RESULTS

Unfortunately, we have to limit the evaluation of our SC2SR algorithm with latency bounds to the original ISP data (cf. Section VI-B). For the other two datasets we were not able to obtain the respective latency bound information. For the same reason, instance $Q$ of the original dataset has to be omitted as for this instance latency bound information was also not available. This leaves us with a total of 18 instances to evaluate our adapted algorithm on. Furthermore, the main objective of the latency constraint is to prevent the SR detours introduced by MO from increasing the normal SPR delays to more than the specified limit. It is not designed to "repair" latency bounds that are already violated by standard SPR as this is a different problem that is beyond the scope of this paper. For this reason, in the following evaluations all latency bound constraints are set to the maximum of the specified latency bound and the SPR path delay of the respective demand.

In a first step, we assess to what extent the original SC2SR algorithm without the new latency constraint violates these bounds. For this, we simply take the computed SR configurations and compute for each demand the resulting end-to-end delay and compare it to the respective bound. The results of this evaluation are shown in Table 2. For each instance, it lists the number of latency bounds violated by the SC2SR solution as well as the average percentage value by which each bound is exceeded to assess the severity of the violation. The first row in Table 2, for example, shows that for instance $A$ a total of 98 latency bounds were violated and on average each bound is exceeded around 25%. Overall, it can be seen that the results tend to vary quite heavily. For some instances only a few or even no bounds are violated. For others there are close to 100 violations of quite significant severity (e.g., instance $D$ with an avg. exceeding of nearly 400%). These results show that, in order for our algorithm to be usable in practice, the addition of a latency bound constraints is, in fact, required. Otherwise important operational requirements are not met by the solutions, rendering them basically undeployable.

TABLE 2. Overview over the number of latency bound violations after optimization and the average latency-bound exceeding per ISP instance for the SC2SR algorithm when latency-bound constraints are not included in the LP.

| Instance | # of Violations | Avg. Bound Exceeding [%] |
|---|---|---|
| A | 98 | 24.78 |
| B | 73 | 76.59 |
| C | 11 | 54.71 |
| D | 39 | 385.37 |
| E | 24 | 53.78 |
| F | 32 | 55.23 |
| G | 0 | 0.00 |
| H | 2 | 55.00 |
| I | 2 | 8.44 |
| J | 0 | 0.00 |
| K | 22 | 34.17 |
| L | 0 | 0.00 |
| M | 6 | 27.17 |
| N | 1 | 44.00 |
| O | 98 | 105.30 |
| P | 2 | 45.00 |
| R | 89 | 64.47 |
| S | 75 | 153.82 |

However, adding the latency bound constraint (Equation (9)) to the LP further restricts the set of feasible solutions and, hence, might have a negative impact on the achievable solution quality. To evaluate this, we ran the SC2SR optimizations once with latency bounds disabled and once with them being enabled to compare the achieved MLUs. The results are depicted in Figure 11. It can be seen that for all but two instances ($C$ and $F$) basically the same MLUs are achieved. For the latter two instances, the MLU worsens slightly but they are still quite close to the optimal MCF value and, in the case of instance $F$, still offer a substantial improvement compared to SPR. This shows that adding these latency constraints to the optimization problem can have a slightly negative impact on the solution quality in rare occasions but for the majority of instances the results remain virtually optimal.

One further aspect we evaluate is whether the addition of the latency constraint has a negative impact on the number of required policies when used with the SC2TLE extension. So far we have shown that it (mostly) does not impact the achievable MLU, but it might results in a significant increase

**TABLE 3. Comparison of the number of policies required by the SC2TLE algorithm ($\lambda = 1.0$) with and without enforcement of latency bounds. The lowest number of policies for each instance is highlighted.**

| Instance | # of Policies | |
|---|---|---|
| | no Latency Bounds | Latency Bounds |
| A | 3 | **1** |
| B | 6 | **5** |
| C | 5 | **3** |
| D | **22** | **22** |
| E | **5** | 6 |
| F | 33 | **27** |
| G | **7** | **7** |
| H | **8** | **8** |
| I | **4** | 5 |
| J | **11** | **11** |
| K | 6 | **5** |
| L | **4** | **4** |
| M | 7 | **6** |
| N | **9** | 10 |
| O | **13** | 15 |
| P | **5** | **5** |
| R | **3** | 4 |
| S | **4** | 5 |

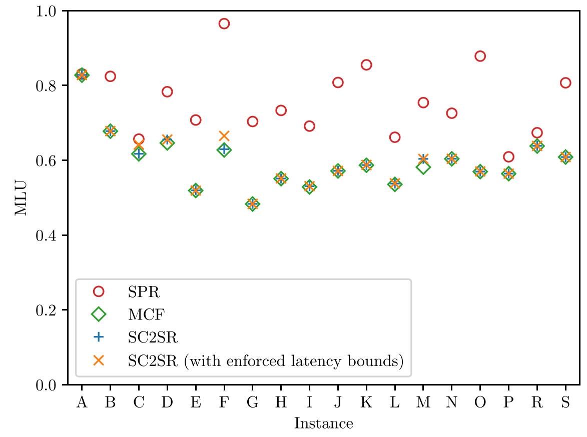


**FIGURE 11. Comparison of the MLUs achievable with the SC2SR algorithm with and without enforcing latency bounds.**

in the number of policies. To analyze this, we reran the SC2TLE optimization with added latency constraints and compare them to the policy numbers of standard SC2TLE. The results are shown in Table 3. It can be seen that, while there are slight variations, on the greater scale the policy numbers stay more or less the same with a variance of just one or two policies for most instances. An interesting observation is that for instance *F* (and on a smaller scale for some other instances as well) the number of policies decreases when adhering to the latency bounds. On the first look, this is somewhat counterintuitive as these presumably "better" solutions should have also been found by the original algorithm without latency bounds as it is less restricted. However, the rather simple explanation for this is that these solutions have an (at least slightly) worse MLU than their counterparts that do not have to respect latency bounds and, hence, also tend to use fewer policies. This can be seen best at the example of instance *F*. Here, the solution with latency bounds requires six less policies. However, if we go back to Figure 11, we see that for this instance the achieved MLU is noticeably higher than the MLU of the solution that does not respect latency bounds. Hence, it is expected that it also requires fewer policies.

**TABLE 4. Number of SR policies required by different algorithms for the two ISP datasets. (Policy numbers for the Repetita dataset had to be omitted for the sake of readability, since the number of featured instances is too large.).**

| | 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 |

* 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.

All in all, these evaluations have shown that it is possible to incorporate latency bound constraints into the SC2SR and SC2TLE algorithms without significantly impacting the solution quality. This is an important finding as these constraints often are a crucial operational requirement that has to be respected for solutions to be actually deployable in practice. Hence, this can be seen as an important extension of these algorithms that facilitates their use in a practical deployment.

## IX. 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.

TABLE 5. Summary of important notations (for details, we refer to the respective in-text problem descriptions).

| Symbol | Meaning | Problem |
|---|---|---|
| $\mathcal{R}$ | MO rule-set determining traffic steering | 2 |
| $\mathcal{P}$ | SR policy configuration | 2 |
| $FP_{ij}(\mathcal{P})$ | Forwarding path of demand $i \rightarrow j$ when using policy configuration $\mathcal{P}$ | 2 |
| $\theta$ | MLU | 1,2,3,4 |
| $c(e)$ | Capacity of edge $e$ | 1,2,3,4 |
| $t_{ij}$ | Size of traffic demand $i \rightarrow j$ | 1,2,3,4 |
| $x_{ij}^k$ | LP variables representing the SR policy configuration (Note: exact meaning varies between optimization problems!) | 1,3,4 |
| $y_{km}$ | Binary variable indicating whether there is any MO policy installed between nodes $k$ and $m$ (irrespective of the intermediate segment) | 3 |
| $g_{ij}^k(e)$ | Load put on edge $e$ if a uniform demand is routed from node $i$ to $j$ over intermediate segment $k$ | 1,4 |
| $diff_{ij}^{klm}(e)$ | Difference in the share of the demand from $i$ to $j$ that is put on edge $e$ in the case of SPR and when an MO policy from $k$ over $l$ to $m$ is installed | 3 |
| $spr(e)$ | Traffic load on edge $e$ when using SPR | 3 |
| $\mathcal{I}_{km}$ | Set of all policies that influence the traffic that is steered onto a policy between nodes $k$ and $m$ | 3 |
| $Z_{ij}$ | Number of sub-demands the demand between nodes $i$ and $j$ can be split into | 4 |

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 [29] 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. Furthermore, we have shown that latency bounds for individual demands can be integrated into the optimization algorithms without significantly deteriorating the quality of solutions. This is an important finding since these bounds often are important practical requirements that need to be respected by TE solutions.

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 A
See the Table 4.

## APPENDIX B
Weighted 2TLE (w2TLE) problem formulation (adapted form the 2TLE LP formulation of [18]):

$$\text{P5}: \min \ \theta' \frac{1}{2\lambda\theta} + \sum_{k \neq j} w_{ijk} x_{ij}^k \tag{11a}$$

$$\text{s.t.} \sum_k x_{ij}^k = 1 \qquad \forall ij \tag{11b}$$

$$\sum_{ij} t_{ij} \sum_k g_{ij}^k(e)\, x_{ij}^k \leq \theta' c(e) \qquad \forall e \tag{11c}$$

$$\theta' \leq \lambda\theta \tag{11d}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall ijk \tag{11e}$$

## APPENDIX C
See the Table 5.

## ACKNOWLEDGMENT
We would like to thank the anonymous reviewers for their thorough an constructive feedback that helped us to improve the quality of this paper.

## REFERENCES

[1] A. Brundiers, T. Schüller, and N. Aschenbruck, "Midpoint optimization for segment routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2022, pp. 1579–1588.

[2] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," Internet Eng. Task Force, RFC 8402, 2018.

[3] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 657–665.

[4] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "CG4SR: Near optimal traffic engineering for segment routing with column generation," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2019, pp. 1333–1341.

[5] 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. Netw.*, vol. 29, no. 1, pp. 262–274, Feb. 2021.

[6] P. Tune, M. Roughan, H. Haddadi, and O. Bonaventure, "Internet traffic matrices: A primer," *Recent Adv. Netw.*, vol. 1, pp. 108–163, Aug. 2013.

[7] W. Ben-Ameur, N. Michel, B. Liau, and E. Gourdin, "Routing strategies for IP-networks," *Telektronikk Mag.*, vol. 97, pp. 145–158, Mar. 2001.

[8] E. Mulyana and U. Killat, "Optimization of IP networks in various hybrid IGP/MPLS routing schemes," in *Proc. GI/ITG Conf. Meas. Eval. Comput. Commun. Syst. (MMB) Together 3rd Polish-German Teletraffic Symp. (PGTS)*, 2004, pp. 295–304.

[9] J.-L. L. Roux, J.-P. Vasseur, and J. Boyle, "Requirements for inter-area MPLS traffic engineering," Internet Eng. Task Force, RFC 4105, 2005.

[10] J. Shen and H. Smit, "Calculating interior gateway protocol (IGP) routes over traffic engineering tunnels," Internet Eng. Task Force, RFC 3906, 2004.

[11] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.

[12] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for Internet traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 36–56, 1st Quart., 2008.

[13] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2015, pp. 1–6.

[14] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," Internet Eng. Task Force, RFC 3209, 2001.
[15] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 464–486, 1st Quart., 2019.
[16] P. L. Ventre et al., "Segment routing: A comprehensive survey of research activities, Standardization efforts, and implementation results," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 182–221, 1st Quart., 2021.
[17] (IBM, Armonk, NY, USA). *IBM ILOG CPLEX Optimization Studio 20.1.0*. (2020). [Online]. Available: https://www.ibm.com/docs/en/icos/20.1.0
[18] 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 Trans. Netw.*, vol. 26, no. 4, pp. 1851–1864, Jul. 2018.
[19] F. Skivée, S. Balon, and G. Leduc, "A scalable heuristic for hybrid IGP/MPLS traffic engineering—Case study on an operational network," in *Proc. IEEE 14th Int. Conf. Netw. (ICON)*, 2006, pp. 1–6.
[20] C. Filsfils et al., "Segment routing policy for traffic engineering," Internet Eng. Task Force, Fremont, CA, USA, Internet Draft draft-filsfils-spring-segment-routing-policy-04, 2017.
[21] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, and P. Mattes, "Segment routing policy architecture," Internet Eng. Task Force, Fremont, CA, USA, Internet Draft draft-ietf-spring-segment-routing-policy-08, 2020.
[22] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, and P. Mattes, "Segment routing policy architecture," Internet Eng. Task Force, RFC 9256, 2022.
[23] C. Filsfils, K. Michielsen, F. Clad, and D. Voyer, *Segment Routing Part II—Traffic Engineering*. Scotts Valley, CA, USA: CreateSpace, 2019.
[24] (Cisco Technol. Co., San Jose CA, USA). *Cisco WAN Automation Engine (WAE)*. Accessed: Mar. 5, 2024. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/net_mgmt/wae/7-6-0/design_user_guide/cisco-wae-design-76-user-guide.pdf
[25] (Juniper Netw. Corp., Sunnyvale, CA, USA). *Junos OS IS-IS user Guide*. (2023). [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/is-is/is-is.pdf
[26] A. Brundiers, T. Schüller, and N. Aschenbruck, "Tactical traffic engineering with segment routing midpoint optimization," in *Proc. IFIP Netw. Conf. (NETWORKING)*, 2023, pp. 1–9.
[27] A. Brundiers, T. Schüller, and N. Aschenbruck, "Combining midpoint optimization and conventional end-to-end segment routing for traffic engineering," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, 2023, pp. 1–9.
[28] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP network energy cost: Formulation and solutions," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 463–476, Apr. 2012.
[29] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017, *arXiv:1710.08665*.
[30] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[31] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving segment routing problems with hybrid constraint programming techniques," in *Proc. 21st Int. Conf. Princ. Pract. Constr. Program. (CP)*, 2015, pp. 592–608.
[32] D. Alderson, L. Li, W. Willinger, and J. Doyle, "Understanding Internet topology: Principles, models, and validation," *IEEE/ACM Trans. Netw.*, vol. 13, no. 6, pp. 1205–1218, Dec. 2005.
[33] 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. London, U.K.: Springer, 2010, pp. 19–93, doi: 10.1007/978-1-84882-828-5_2.
[34] T. Li, C. Barth, A. Smith, and B. Wen, "Tactical traffic engineering (TTE)," Internet Eng. Task Force, Fremont, CA, USA, Internet Draft draft-li-rtgwg-tte-00, 2023.
[35] Tactical traffic engineering based on segment routing policies, by T. LaBerge, C. Filsfils, and P. Francois. (2020). U.S. Patent 10 742 556 B2. [Online]. Available: https://patents.google.com/patent/US10742556B2
[36] S. Agarwal, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "The impact of BGP dynamics on intra-domain traffic," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 319–330, 2004.
[37] J. Wu, Z. M. Mao, J. Rexford, and J. Wang, "Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network," in *Proc. Symp. Netw. Syst. Design Implement. (NSDI)*, 2005, pp. 1–14.
[38] Enforcing strict shortest path forwarding using strict segment identifiers, by P. Psenak, R. Hanzl, C. Filsfils, and K. J. Talaulikar. (2019). U.S. Patent 10 742 537. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=US254129770
[39] S. Gay, R. Hartert, and S. Vissicchio, "Expect the unexpected: Sub-second optimization for segment routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2017, pp. 1–9.
[40] A. Brundiers, T. Schüller, and N. Aschenbruck, "On the benefits of loops for segment routing traffic engineering," in *Proc. IEEE 46th Conf. Local Comput. Netw. (LCN)*, 2021, pp. 32–40.
[41] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, 2nd ed., San Francisco, CA, USA: Morgan Kaufmann Publ., 2017.
[42] (Cisco Technol. Co., San Jose, CA, USA). *Best Practices in Core Network Capacity Planning White Paper*. (2020). [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/routers/wan-automation-engine/white_paper_c11-728551.html
[43] (Cisco Technol. Co., San Jose, CA, USA). *Cisco Annual Internet Report (2018–2023)*. (2020). [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
[44] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 93–96, Oct. 2005.
[45] A. Brundiers, T. Schüller, and N. Aschenbruck, "Preprocess your paths—Speeding up linear programming-based optimization for segment routing traffic engineering," 2023, *arXiv:2312.00518*.

**ALEXANDER BRUNDIERS** (Graduate Student Member, IEEE) received the master's degree in computer science from Osnabrück University, Germany, in 2020, where he is currently pursuing the Ph.D. degree with the Distributed Systems Group, Institute of Computer Science. His research focuses mainly on segment routing and its applications for intra-domain traffic engineering, but his interests also encompass various areas in the broader field of Internet routing.

**TIMMY SCHÜLLER** received the master's and Ph.D. degrees in computer science from Osnabrück University, Germany, in 2015 and 2020, respectively. From 2015 to 2019, he worked in a Joint Project with Detecon International GmbH and Deutsche Telekom Technik GmbH. Since then, he has been working as a DevOps Engineer with Deutsche Telekom Technik GmbH. As such, he works toward developing and deploying next-gen traffic engineering strategies in a global IP backbone network.

**NILS ASCHENBRUCK** (Member, IEEE) received the Graduate Diploma and Ph.D. degrees in computer science from Bonn University, Germany, in 2003 and 2008, respectively. He was a Senior Researcher and the Head of the Research Area "Tactical Wireless Multi-Hop Networks" with the Communication Systems Group, Bonn University. Since 2012, he has been holding a Tenured Professorship for Distributed Systems with the Osnabrück University. His research focus is on dependable and robust networked systems including scenario modeling, traffic engineering, and network security.