Combining Midpoint Optimization and Conventional End-to-End Segment Routing for Traffic Engineering

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

°Osnabrück University, Institute of Computer Science
Friedrich-Janssen-Str. 1, 49076 Osnabrück, Germany
Email: {brundiers, schueller, aschenbruck}@uos.de

•Deutsche Telekom Technik GmbH
Gartenstraße 217, 48147 Münster, Germany
Email: timmy.schueller@telekom.de

***Abstract*—A recent innovation in the context of Segment Routing (SR) Traffic Engineering (TE) is the use of the Midpoint Optimization (MO) concept to substantially reduce the number of SR policies required to implement TE solutions. However, these benefits come at the price of a potential deterioration of traffic steering capabilities as the individual, per-demand traffic control of end-to-end (E2E) SR is lost. In this paper, we show that a hybrid SR approach utilizing both MO as well as E2E SR allows to combine the individual benefits of both approaches. It results in improved TE capabilities and, thus, enables better TE solutions. Besides discussing this with theoretical examples, we also confirm our findings in an extensive evaluation on real-world network topologies. For this, we propose a TE algorithm based on the hybrid SR approach and show that it outperforms state-of-the-art algorithms that rely solely on MO or E2E SR. Furthermore, we demonstrate that the hybrid SR approach is also suitable for the use-case of time-constrained, tactical TE.**

I. Introduction

Over the recent years, Segment Routing (SR) [6] has become a premier tool for Traffic Engineering (TE) as it offers great traffic steering capabilities while introducing very little overhead into the network. Most of the SR literature focuses on the use of conventional SR which deploys SR policies as end-to-end (E2E) tunnels to alter the forwarding paths of individual demands. Recently, however, Brundiers et al. [3] studied the benefits of applying the Midpoint Optimization (MO) concept to SR TE. Instead of using SR policies in an E2E fashion, MO allows for a single policy to route multiple demands at once. It was shown that such an approach can substantially reduce the number of SR policies required to implement TE solutions while still achieving optimization quality that is on-par with conventional E2E SR algorithms.

In this paper, we show that there are scenarios in which neither of these two approaches (MO and E2E SR) alone are sufficient to find the best possible TE solutions, both with respect to the number of policies but also the achievable Maximum Link Utilization (MLU). To overcome these issues, we utilize a *hybrid SR* approach which can deploy both MO as well as E2E policies. It allows to utilize the per-demand traffic control of E2E SR while relying on MO to reduce the overall number of policies, which allows for better TE solutions. We verify our theoretical findings with an extensive evaluation on real-world network topologies. For this, we develop the first TE algorithm that utilizes a hybrid SR approach and show that it outperforms state-of-the-art algorithms that solely rely on either MO or E2E SR. We further show that the hybrid SR approach is also suitable for fast, tactical TE and allows for the computation of very good solutions within seconds.

The remainder of the paper is structured as follows. First, we introduce required background information on the topic of SR and TE (Section II), followed by a discussion of related work (Section III). After this, Section IV describes the general concept of the hybrid SR approach and illustrates its advantages by means of a small example. This is followed by the introduction of our new Linear Program (LP)-based optimization algorithm (Section V) and the evaluation of the latter (Section VI). In Section VII, we demonstrate that the hybrid SR approach is also suitable for the use-case of time-constrained, tactical TE. Finally, the paper is concluded in Section VIII with a recapitulation of our key contributions and findings and a short discussion of possible future work.

II. Background

This section provides background knowledge on concepts fundamental for the understanding of this paper. This includes Segment Routing (SR), its use for Traffic Engineering (TE) and the concept of Midpoint Optimization (MO) for SR.

A. Segment Routing

Segment Routing (SR) [6], [7] is a network tunneling technique. It implements the source routing paradigm by adding a list of labels (so called *segments*) to a packet, which determines the packets path through the network. These labels can be interpreted as waypoints that have to be visited in the given order before forwarding the packet to its original destination. To determine the forwarding paths between these waypoints, the Interior Gateway Protocol (IGP) is consulted. Labels are applied to packets by configuring so called SR *policies* on individual nodes in the network. These policies can be understood as a form of "rules" that specify which labels to add to which packets on this specific node. Such policies are generally used in a demand-bound, E2E fashion. For a traffic demand between two nodes A and B, a dedicated policy is installed on node A that specifies the forwarding path for this demand. However, traffic of demands with other sources that also traverse A and head towards B will not be routed along the TE path specified in this policy. Instead, it will follow the standard shortest path. This enables a fine-grained traffic control on the level of individual demands and the definition of virtually arbitrary forwarding paths, comparable to other established traffic steering approaches like Multiprotocol Label Switching (MPLS) with Resource Reservation Protocol (RSVP)-TE [1]. However, SR introduces lower overhead and, hence, scales better than the latter.

B. Traffic Engineering with Segment Routing

Its fine grained traffic control combined with a reduced overhead have rendered SR one of the premier choices for TE over the recent years. As a result, there is a wide variety of publications dealing with SR and its applications for various TE purposes (for an overview see e.g., [16]).

One of the first works that deals with SR TE is [2]. It proposes a LP-based optimization approach for computing optimal SR configurations with respect to the MLU. While, in theory, arbitrarily many segments can be applied to a packet, in practice this number is limited by the so called Maximum Segment Depth (MSD) of the used hard- and software. For this reason, the algorithm proposed in [2] is limited to SR paths with at most two segments (one intermediate one and one for the destination). Hence, it is also referred to as 2SR. It has been shown in [2] that even when using at most two segments, virtually optimal MLUs can be achieved. However, those results are of a more theoretical nature since the respective SR configurations do not adhere to various other real-world requirements and, thus, are generally not deployable in practice. For example, current routing hardware does not allow for traffic to be split into arbitrary fractions. Furthermore, the computed solutions often require thousands of SR policies to be implemented, while operators prefer solutions with as few policies as possible (i.e. to reduce overhead or for the sake of clarity and maintainability). This is addressed in [14] in which the original 2SR formulation is extended to also include those real-world requirements. It is shown that the resulting 2TLE algorithm is still able to obtain virtually optimal results on real-world instances.

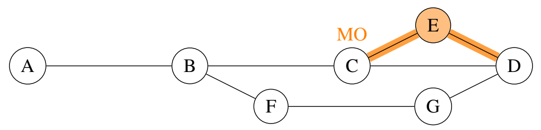
C. Midpoint Optimization for Segment Routing

A recently studied innovation in the context of SR is the concept of Midpoint Optimization (MO) [3]. Instead of using SR policies as E2E "tunnels" for single demands, MO builds onto the idea of allowing a single policy to route multiple demands. This holds the potential to substantially reduce the number of policies to be configured, resulting in reduced overhead and improved clarity and maintainability. MO itself is an abstract concept that can be implemented in various ways, depending on the chosen rule-set on how traffic is steered onto policies. The MO implementation studied in [3], for example, is based on the *IGP Shortcut* approach. A packet is only steered onto a policy if the policy endpoint lies on the IGP shortest path from the policy startpoint to the packets destination, offering some sort of "shortcut". Furthermore, traffic that is currently detoured by a policy is not steered onto other policies encountered along this detour. This implicitly prevents the accidental configuration of forwarding loops [3].

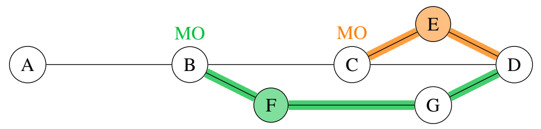
A popular first approach to TE problems is formulating them as LPs (cf. e.g., [2] or [14]). Those can be used to find provably optimal solutions that allow to assess the overall potential of the new TE concepts. In the context of MO, however, such an approach proves to be difficult (cf. [3]). The main problem is that, for an efficient LP formulation, information on how the insertion (or removal) of policies impacts the link utilizations in the network is needed. For E2E SR, this information can be efficiently precomputed as each policy routes exactly one demand in an E2E fashion. When utilizing MO, however, policies are no longer bound to a specific demand and, hence, can route a multitude of demands. Furthermore, since policies are no longer installed in an E2E fashion, a demand can be routed sequentially through multiple policies, one after the other. This results in what is defined as *policy dependencies* in [3]: The addition or removal of a policy can (potentially) influence the traffic that is routed through other policies in the network. An example for this is given in Figure 1. If only the orange policy is installed at node C (the colored node marks the respective intermediate segment), the traffic of demand $A \to D$ will be routed through it. However, if the green policy is installed on node B as well, the traffic will be routed through it instead. As a result of these dependencies, the impact of an individual policy can no longer be determined independently of the other policies. In the worst case, information on (virtually) all other policies in the network can be necessary to determine the link utilizations resulting from a policy. This renders the efficient calculation of such an LP formulation virtually impossible [3].

To (at least partially) circumvent this problem, [3] proposes the idea of prohibiting the simultaneous configuration of SR policies that "influence each other", to facilitate an efficient LP formulation. This artificially limits the solution space and, thus, can (theoretically) deteriorate the quality of the solutions found. However, it is shown in [3] that the Shortcut 2SR (SC2SR) algorithm that utilizes this idea is still able to achieve optimization results of similar quality as conventional E2E SR while requiring substantially less policies to do so.

Since the optimization algorithm proposed later on in this paper reuses parts of the SC2SR algorithm, the latter one is vital for the understanding of this paper. Hence, the respective LP is given in Problem 1. The binary x_{km}^{l} variables denote whether an MO policy is between nodes k and m with intermediate segment l. For the sake of a smaller and, thus, faster solvable LP (cf. [3]) a second set of variables is introduced. These y_{km} basically aggregate the respective x_{km}^{l} for nodes k and m and indicate whether a policy is installed between these nodes, irrespective of the intermediate segment. With Equation 2, the y_{km} are tied to their respective x_{km}^{l}. It also limits the number of policies between any pair of nodes to at most one. The second constraint (Equation 3) ensures that for any installed policy none of its influencing



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



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

Figure 1: Simplified example for policy dependency when deploying MO-capable SR policies.

$$\min \ \theta \tag{1}$$

$$\text{s.t.} \quad \sum_l x^l_{km} = y_{km} \quad \forall km \tag{2}$$

$$y_{km} + y_{ij} \leq 1 \quad \forall km, \ \forall ij \in \mathcal{I}_{km} \tag{3}$$

$$spr(e) + \sum_{ij} t_{ij} \sum_{klm} \mathit{diff}^{klm}_{ij}(e)\, x^l_{km} \leq \theta\, c(e) \quad \forall e \tag{4}$$

$$x^l_{km} \in \{0,1\} \quad \forall klm \tag{5}$$

$$y_{km} \in \{0,1\} \quad \forall km \tag{6}$$

Problem 1: SC2SR formulation [3].

policies will be installed as well. Equation 4 is the *capacity constraint*. It ensures that, for each edge e, the load put on this edge does not exceed θ times its capacity, with θ, again, resembling the overall MLU that has to be minimized. Here, $spr(e)$ corresponds to the amount of traffic that is put on edge e in the standard Shortest Path Routing (SPR) case without any policies installed. From this "default" value, we then add or subtract the traffic differences that arise from the configured policies. The $\mathit{diff}^{klm}_{ij}(e)$ values indicate the difference (compared to SPR) in the traffic share of demand $i \rightarrow j$ that is put on edge e when a policy between k and m over intermediate segment l is installed.

For policy minimization, a two-step Tunnel Limit Extension (TLE) approach like the one proposed in [14] can be used. It first minimizes the MLU with the LP of Problem 1 and then carries out a second optimization step that minimizes the number of policies required to obtain this MLU. For this second step, the objective function is changed to

$$\min \sum_{km} y_{km} \tag{7}$$

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

$$\theta \leq \lambda \theta' \tag{8}$$

It ensures that the MLU θ of the new solution does not surpass the optimal MLU θ' of the preceding MLU optimization step by more than the user-defined trade-off coefficient λ. The resulting algorithm is then called SC2TLE [3].

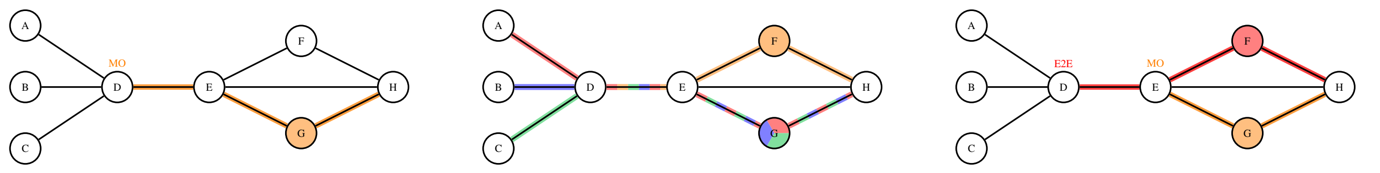
III. Related Work

MO for SR is a very recent innovation with the already mentioned study of Brundiers et al. [3], published in 2022, being the first scientific paper on this topic. It formally describes the general concept of MO for SR and discusses its advantages and disadvantages compared to conventional SR. Furthermore, the already mentioned SC2SR algorithm (cf. Problem 1) is proposed. The only other paper that deals with MO for SR is [4]. It studies the applicability of MO for the context of tactical TE (i.e. fast reoptimization in failure scenarios). A Local Search (LS)-based heuristic called Midpoint Optimization Local Search (MOLS) is proposed and it is shown that, within very limited time, it is able to find solutions that are on-par with tactical TE algorithms that rely on conventional SR but require substantially less policies.

The idea to allow for the activation of the IGP Shortcut feature on individual nodes instead of all routers to realize some form of hybrid TE approach is not completely new. In the MPLS context, there are IGP Shortcut implementations from large routing manufacturers for which the functionality can be activated on a per-node basis (cf. e.g., [10], [13]). For SR itself, we were not able to find information on this in the respective documentations. However, even if this functionality does not exist (yet), we believe that it could be implemented in the foreseeable future. Especially, since one way to implement SR in practice is based on MPLS, potentially allowing for a reuse of the existing MPLS feature implementation. However, while the technical foundations for a per-node activation of the IGP Shortcut feature (more or less) already exist, there are, to the best of our knowledge, no scientific publications that deal with such an approach in the context of TE or evaluate its optimization capabilities. Hence, this paper is the first to deal with an hybrid approach between conventional E2E SR and MO and also the first one to propose and evaluated a corresponding optimization algorithm.

IV. A Hybrid Approach between MO and E2E SR

One of the key benefits of MO compared to conventional E2E SR is its ability to substantially reduce the number of SR policies required to implement TE solutions. The downside of this approach, however, is that it gives up the individual, per-demand traffic control of E2E SR, impacting its overall TE capabilities (cf. [3]). Strictly speaking, there is, in fact, no loss of routing expressiveness when using MO instead of E2E SR. To mimic the per-demand traffic control of E2E SR, we can simply deploy a (more or less) "full-mesh" policy configuration in which a MO policy is configured between each pair of nodes in the network. As a result, each demand will be steered onto a dedicated policy between its source and destination. This is necessary to guarantee that the paths of



(a) The desired routing cannot be fully implemented when exclusively using MO or E2E policies and policy configuration on A, B, and C is prohibited.

(b) If policy configuration is allowed on all nodes, using either exclusively MO or E2E policies still results in a total of four policies.

(c) A hybrid use of E2E and MO policies requires just two policies and avoids policies on A, B, or C.

Figure 2: Example scenario for which a hybrid use of MO and E2E policies is necessary to implement the desired routing under the assumption that the configuration of SR policies is prohibited on certain nodes (A, B, and C). Even if policy configuration on those nodes is allowed, the hybrid approach is able to further reduce the number of required policies.

demands that should be forwarded along their normal shortest path are not influenced by other MO policies along the way. This basically means that it is not only required to install policies for demands that should be detoured, but also for those that should not, to ensure the latter are following their standard shortest path. At this point, it has to be remembered that policy number minimization is one of the main reasons for deploying MO, in the first place. Hence, relying on a full-mesh configuration, which basically requires the highest possible number of policies ($\mathcal{O}(|V|^2)$), is not really feasible in practice. Therefore, MO might (theoretically) offer the same traffic steering capabilities as E2E SR but this does not carry over into practice where the number of policies is a limiting factor as well. As a result, solely relying on MO can lead to sub-optimal solutions in the same way as using only E2E SR.

Therefore, we propose the use of a *hybrid SR* approach, similar to those that already exist for MPLS (cf. Section III). Each node can be individually configured to either interpret SR policies as MO or E2E policies. Such an approach allows to utilize the per-demand traffic control of E2E SR wherever necessary while also benefiting from the ability of MO to substantially reduce policy numbers. This can further reduce the number of policies required to implement TE solutions compared to a deployment that uses either only MO or E2E SR. Furthermore, there are practical scenarios in which SR policies can only be configured on a subset of nodes (c.f., [3] or [5]). In this case, there are routings and even optimal solutions that can only be implemented with hybrid SR.

The example in Figure 2 illustrates the benefits of the hybrid SR approach. Given the depicted topology with a simple hop-count metric, traffic has to be routed from each of the nodes A, B, C, and D to node H. However, to adhere to a given TE objective, traffic from A, B, and C has to be routed over G and traffic from D over F. We further assume that it is not possible to configure SR policies of any kind on A, B, and C. Under those assumptions, the desired routing cannot be implemented when exclusively relying on either E2E or MO policies. This is illustrated in Figure 2a. To detour the traffic from A, B, and C, we need to install an MO-capable policy either on D (as depicted) or, alternatively, on E. However, the traffic from D also has to pass over both of these nodes and shares the same destination. Thus, it would also be steered into the respective policy, following the same path. As a result, only one of the desired detours can be implemented (either for traffic from A, B, and C or for traffic from D), but not both. Hence, under the given circumstances, the desired routing is not feasible using either exclusively MO or E2E policies.

This changes, if we allow policies to be installed on all nodes. In this scenario, a possible solution with the minimal number of policies requires four policies: $[A, B, C] \rightarrow G \rightarrow H$ and $D \rightarrow F \rightarrow H$ (cf. Figure 2a). Here, we cannot use MO to detour multiple demands with a single policy as each MO policy that would be applicable for traffic from A, B, and C would also be applicable for traffic originating at D (as explained in detail in the previous paragraph). As a result, all four demands will always be steered along the same path, which is not a valid solution. For this scenario, there is, in fact, no real difference between using MO or E2E policies. Even when technically using MO-capable policies, each of them only routes exactly one traffic flow. As a result, we need as many policies as there are demands to be rerouted, which is sub-optimal regarding the number of installed policies.

Both of the above issues can be resolved when deploying the hybrid SR approach as it allows for a solution that requires less than four policies and does not require the installation of policies on nodes A, B, or C. The respective hybrid configuration is depicted in Figure 2c. It requires an E2E policy $D \rightarrow F \rightarrow H$ and an MO-capable policy $E \rightarrow G \rightarrow H$. The first one is responsible for detouring the demand between D and H. Since it is an E2E policy, it does not affect traffic coming from other nodes. As a result, traffic from A, B, and C can pass over D without being detoured and will be steered onto the MO-capable policy on node E that routes it over node G to H. All in all, this example shows that a hybrid SR approach can 1) reduce the number of policies compared to an exclusive use of either MO or E2E SR and 2) can actually be necessary to implement a desired routing if there are nodes on which policies cannot be installed.

V. LP-based Optimization Algorithm

Since the "pure" MO problem is a sub-problem of the hybrid SR optimization scenario, the latter one also suffers

$$
\begin{aligned}
&\min \theta && && (9)\\
&\text{s.t.} \quad \sum_l x^l_{km} = y_{km} && \forall km && (10)\\
&\sum_l u^l_{km} = v_{km} && \forall km && (11)\\
&z_k + v_{km} \leq 1 && \forall km && (12)\\
&y_{km} \leq z_k && \forall km && (13)\\
&y_{km} + y_{ij} \leq 1 && \forall km \quad \forall ij \in \mathcal{I}_{km} && (14)\\
&v_{km} + y_{ij} \leq 1 && \forall km \quad \forall ij \in \mathcal{J}_{km} && (15)\\
&spr(e) + \sum_{ij} t_{ij} \left(\sum_{klm} \mathit{diff}^{klm}_{ij}(e)\, x^l_{km} + \sum_k (g^k_{ij}(e) - f_{ij}(e))\, u^k_{ij} \right) \leq \theta\, c(e) && \forall e && (16)\\
&z_k \in \{0,1\} && \forall k && (17)\\
&y_{km}\ ,\ v_{km} \in \{0,1\} && \forall km && (18)\\
&x^l_{km}\ ,\ u^l_{km} \in \{0,1\} && \forall klm && (19)
\end{aligned}
$$

Problem 2: Hybrid 2SR (H2SR) formulation.

from the *policy dependency* issues explained in Section II-C. For this reason, finding an efficient LP formulation is equally difficult here. However, it was shown in [3] that prohibiting the configuration of *influencing* policies enables an efficient LP formulation while still finding (virtually) optimal solutions in most cases. Thus, we decided to follow a similar approach.

The LP formulation of our hybrid optimization algorithm called H2SR is given in Problem 2. First and foremost, we need a way to distinguish between nodes which use conventional SR and those for which the MO (or IGP Shortcut) functionality is activated. This is done with the binary z_k variables. For each node k, those indicate whether the MO feature is activated ($z_k = 1$) or deactivated ($z_k = 0$). Like in the SC2SR LP, the x^l_{km} and y_{km} variables indicate whether an MO-capable policy is configured on node k with node m as endpoint (and node l as intermediate segment). Equations 10 and 14 are also identical to those of SC2SR and fulfill the same purposes (cf. Section II-C). Analogously, the u^l_{km} and v_{km} indicate whether an E2E SR policy is installed on node k towards node m (via intermediate segment l) and Equation 11 connects the u and v variables in the same fashion as Equation 10 does for the x and y variables. Equations 12 and 13 ensure that if a node is set to "MO-mode" only MO-capable policies are configured on it and vice versa for the conventional SR case. When utilizing the hybrid SR approach, traffic routed through an MO policy cannot only be influenced by other MO policies but also by E2E policies that prevent a demand from entering it. Thus, we do not only need a constraint ensuring that for each installed policy none of its influencing MO policies is installed (Equation 14), but also one that does the same for influencing E2E policies. This is done with Equation 15 where the set $\mathcal{J}_{km}$ contains all those start- and endpoints of E2E policies that influence the traffic that is routed through MO policies between nodes k and m. Similar to the $\mathcal{I}_{km}$, these sets can be precomputed in advance. Lastly, Equation 16 realizes the capacity constraint and, together with the objective function, is responsible for minimizing the MLU. This constraint is similar to the one used in Problem 1. However, it now also includes the traffic loads resulting from E2E policies. These are included by the second sum within the large parentheses. Here, $f_{ij}(e)$ corresponds to the share of demand $i \to j$ that is put on edge e when it is routed via SPR. Analogously, the $g^k_{ij}(e)$ resemble the respective share of traffic if demand $i \to j$ is routed through an E2E policy with intermediate segment k. Hence, the second sum in the large parentheses adds, for each E2E policy, the respective difference to the SPR case to the link utilization, similar to the diff^{klm}_{ij} values for the MO case.

To not only optimize the MLU but also minimize the number of policies required to implement the optimal configuration, we utilize the TLE concept which is also used in the 2TLE and SC2TLE algorithms (cf. Section II-C). After minimizing the MLU with Problem 2, a second optimization step is carried out which minimizes the number of policies required to obtain this MLU. The second step uses an adapted version of Problem 2. Equation 8 is added to the LP to ensure that the MLU calculated in the first optimization step is not exceeded by more than a user-defined margin λ. Additionally, the objective is changed from MLU to policy number minimization by replacing the objective function with the following one:

$$\min \sum_{km} y_{km} + v_{km} \qquad (20)$$

The final, two-stage algorithm is then called H2TLE.

Regarding complexity, the bottleneck of the H2SR algorithm and its H2TLE extension lies in the *capacity constraint* (Equation 16). It summarizes over $\mathcal{O}(N^5)$ variables and has to be set up for every edge in the network (potentially $\mathcal{O}(N^2)$), resulting in an overall complexity of $\mathcal{O}(N^7)$ with N being the number of nodes in the network. However, it has to be remembered that this is the worst-case complexity. Real-world problem instances often feature certain topology and traffic characteristics that can be exploited to substantially lower the problem size. For example, ISP backbones tend to have a fairly low graph density, nowhere near the worst case $\mathcal{O}(N^2)$ edges, and often there isn't a traffic demand for every single node pair. Furthermore, not all theoretically feasible intermediate segments need to be considered to guarantee optimal solutions. This can be exploited to further reduce the problem size, resulting in a substantially lower complexity for real-world instances.

VI. Evaluation

In this section, we evaluate the performance of our H2TLE algorithm to assess whether the theoretical benefits of the hybrid SR approach described in Section IV carry over into practice. For this, we use it to optimize real-world network topologies and compare these results to state-of-the-art TE algorithms that solely rely on either MO or E2E SR.

A. Data and Setup

Our evaluations are carried out on instances from the publicly available *Repetita* dataset [8]. It is based on real-world network topologies (mostly of WANs and/or ISP backbones) collected in the *Internet Topology Zoo* [11] (ranging from just 4 to up to 197 nodes in size) for which five artificially generated traffic matrices are given. There are two sets of metrics for each topology: One with unary metrics and one with the metric corresponding to the inverse capacity of the respective link. For reasons of time and space, we limit our evaluations to one traffic matrix per instance (matrix set 0) with unary metrics. We do not expect the results to differ significantly for the inverse capacity metrics since it has been shown in [15] that, as long as a sensible metric is selected, the results obtainable with SR are rather independent of the exact metric choice. Furthermore, we exclude all those instances for which the optimal MLU is already achieved with SPR as they are of no interest for TE. This leaves us with a total of 209 instances, which we further categorize by their number of nodes, dividing them into three categories: *small* ($|V| < 20$), *medium* ($20 \leq |V| < 40$), and *large* ($40 \leq |V|$).

For each instance, we use the 2TLE algorithm to compute the optimal MLU obtainable with E2E 2SR as well as the minimal required number of policies. The same is done with the SC2TLE algorithm for MO. These results are then used as references to compare our new hybrid SR approach against and to assess its optimization capabilities and its potential to improve upon those established SR TE approaches. For the assessment of the following results, it should also be noted that all Repetita instances are designed to have a theoretically achievable optimal MLU of 0.9 (or 90%). All computations are carried out on a node with two AMD EPYC 7452 CPUs and 512GB of RAM. LPs are solved with CPLEX [9].

B. Results

The MLUs achieved by the different optimization algorithms are depicted in Figure 3. The dashed green line marks

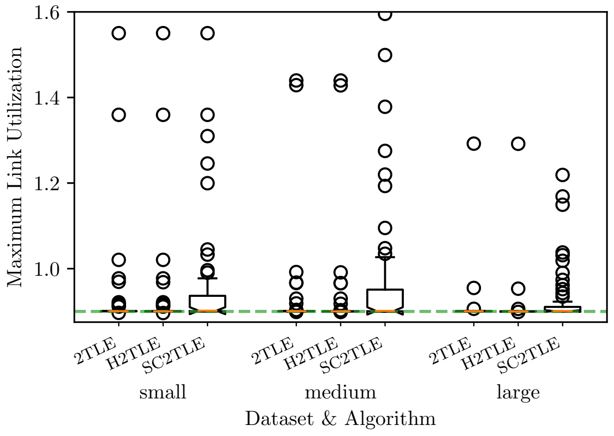
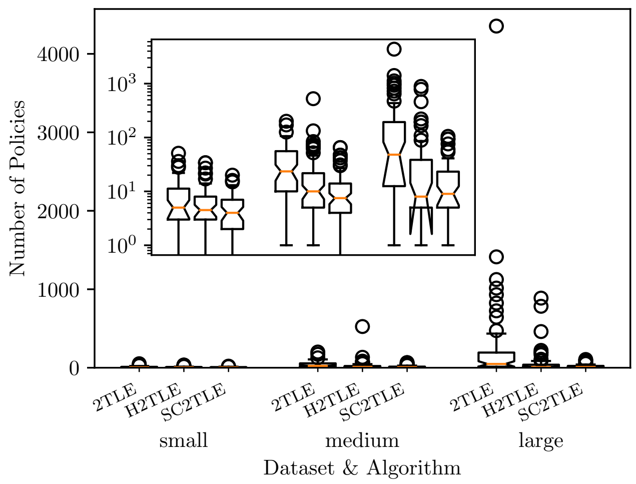


Figure 3: Comparison of the achievable MLUs of LP-based optimization algorithms based on the three different SR concepts: E2E (2TLE), MO (SC2TLE), and the hybrid approach (H2TLE). (Five outlier instances with very high SC2TLE MLUs are not included in the plot to improve readability.)

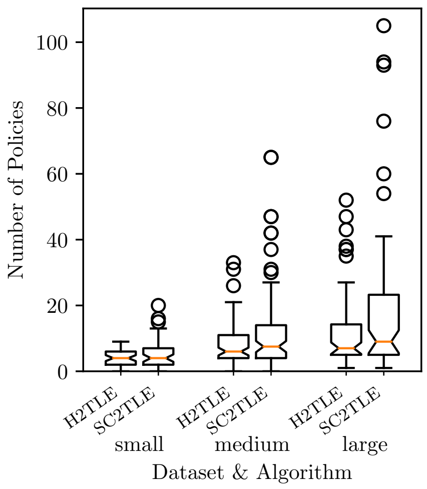
the optimal MLU of 0.9. It can be seen that while SC2TLE solves the major part of instances more or less optimally, there is also a noticeable number of (especially small and medium sized) instances that leave room for improvements. For nearly all of the latter instances, 2TLE and H2TLE are able to find optimal solutions. Since H2TLE is guaranteed to find solutions that are at least as good as those of 2TLE and SC2TLE it is no surprise that it performs on the same level as 2TLE. However, the fact that the MLU distributions of 2TLE and H2TLE are virtually identical indicates that there is no additional benefit from combining MO and E2E in terms of MLU (at least for the considered instances).

Such a benefit, however, can be seen when looking at the number of policies required to implement the respective solutions (Figure 4a). Even though H2TLE achieves the same MLUs as 2TLE, it requires less policies to do so. For the small instances, the differences are neglectable as most instances are solved with less than 10 policies which does not leave much room for improvements. However, already for mediums sized instances, on average, 2TLE tends to require more than twice as many policies as H2TLE. For the large instances, the differences become even more noticeable. Here, 2TLE often requires hundred or more policies and in worst case scenarios even up to around a thousand. In contrast, H2TLE achieves the same MLUs with substantially fewer policies, mostly within the range of a low two-digit number of policies.

An observation that might seem odd at first glance is the fact that, in Figure 4a, the number of policies required by SC2TLE are even lower than the ones of H2TLE. However, the solution space of the former algorithm is a subset of the solution space of the latter. Thus, all solutions found by SC2TLE should also be available to the H2TLE algorithm. The straight forward explanation for this observation is the fact that we carry out a (more or less) unfair comparison here. As shown in Figure 3, H2TLE tends to find better MLUs



(a) Distribution of the number of policies required to implement the solutions depicted in Figure 3. For better readability, the inset plot shows the same data but on a logarithmic scale.



(b) Distribution of policy numbers when using the same MLU target values for H2TLE and SC2TLE.

Figure 4: Comparison of the number of policies required by the different LP-based optimization algorithms.

than SC2TLE for quite a large set of instances, and solutions with better MLUs generally tend to require more policies to implement. Hence, for a fair comparison, it is necessary to compare the number of policies required by both algorithms to implement solutions of similar quality (with equal MLU). We carried out such an evaluation by running the H2TLE optimization with an MLU *target value* corresponding to the best MLU found by SC2TLE for the respective instance. As soon as H2TLE finds a solution with an equally good or better MLU, it will not try to further reduce the MLU but switch to the policy minimization instead. The results of this optimization are depicted in Figure 4b. It can be seen that, when solutions of similar quality (wrt. MLU) are compared, H2TLE requires even less policies than SC2TLE.

All in all, these findings demonstrate that our theoretical considerations from Section IV actually carry over into practice. Deploying the hybrid SR approach further improves TE capabilities, resulting in improved solution quality regarding both MLU as well as policy numbers. If required, conventional E2E SR enables individual, per-flow traffic control while the the use of MO allows for a substantial reduction in the number of policies. In fact, the finding that H2TLE requires even less policies than SC2TLE shows that the hybrid SR approach is not just the “sum of its two parts” but that the combination of both approaches yields additional benefits.

VII. Tactical Traffic Engineering with Hybrid SR

As shown in the previous Section, a hybrid SR approach can be used to further improve TE capabilities and, thus, quality of TE solutions. While the obtained results undeniably demonstrate the benefits of the hybrid SR approach, the H2TLE algorithm itself suffers from an inherent weakness. It is an LP-based algorithm and as such it does not scale well with increasing network size. The larger the network,

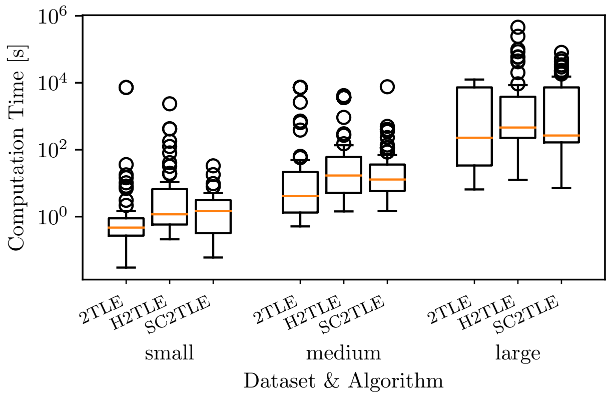


Figure 5: Computation times required by the different LP-based algorithms to compute the results presented in Figure 3. (Optimization of some (mostly larger) instances was aborted before finding a provable optimal solution if no further progress in solution quality was observable.)

the larger and more complex the related LP and the more memory and computation time is required for optimization. For small to medium sized instances, this is not really an issue as those instances can still (mostly) be solved within minutes or even seconds (cf. Figure 5). However, for very large networks this quickly becomes problematic. For example, for the second largest topology in the Repetita dataset (`UsCarrier`), H2TLE took multiple days to compute and for the largest instance (`Cogentco`) it was not able to find a solution on our hardware. Such high computation times are not-optimal but (arguably) tolerable if the use case is strategic TE that is done in a planned fashion on a weekly or even monthly basis. However, they are definitely not suitable for tactical TE. For this, it is of utmost importance to find sufficient solutions within short time (e.g., to quickly react to failures or sudden changes in traffic characteristics) [12].

Since fast, tactical TE is often as important to network operators as strategic TE, we want to demonstrate that the hybrid SR approach is also suitable for this use case. For this, we extended the LS-based MOLS algorithm proposed in [4], which previously just supported pure SR-MO, to utilize the hybrid SR approach instead and evaluate its performance. The resulting algorithm is called Hybrid SR LS (HSLS).

A. Adaption of the MOLS Algorithm

Since our HSLS algorithm is an extension of the MOLS algorithm of Brundiers et al. [4], it follows the same structure and reuses parts of the latter algorithm. For reasons of space, we cannot describe all these functionalities of the MOLS algorithm in detail here. Therefore, this section focuses mainly on changes and adaptions made by us to the original MOLS algorithm in order to support the hybrid SR approach. All other algorithmic components not mentioned here were basically left untouched and are implemented and used as described in [4]. For a more in-depth description of the MOLS algorithm (and, thus, also our HSLS adaption), we refer to [4].

In order to adapt the MOLS algorithm to support the hybrid SR approach, we had to make two major changes. The first difference is the representation of the solutions. In the MOLS algorithm, a solution is represented by the set of installed policies. To support the hybrid SR approach, this representation has to be extended to also feature information on the *mode* of each node in the network (either MO or E2E). This is done with a boolean array that, for each node, indicates whether it is set to MO-mode (which means all policies installed on it will be MO policies) or not. As the second major change, we also had to adapt the set of eligible moves and the respective candidate/move evaluation procedure. As MOLS relies solely on MO, it has basically just two types of moves: *Insertion* and *removal* of a MO policy. When utilizing hybrid SR, we also have the option to add (or remove) E2E policies. Furthermore, there also has to be the option to change the mode of nodes, either from MO to E2E or vice versa. We combine this mode change with the insertion of policies. As a result, HSLS features four different insertion options: Insertion of a MO or E2E policy either on a node that already is in the respective mode or needs to be changed accordingly. Insertions in which a mode change is not required are rather straight forward. For an MO insertion, we can reuse the implementation of MOLS and the insertion of an E2E policy only requires the rerouting of a single demand. Insertions that also require a mode change, however, are inherently more complex, since it also alters the behavior of all other policies configured on this node. Hence, we do not only have to recompute the paths (and resulting link utilizations) of demands that are impacted by the newly inserted policy, but also of all those demands that are (or were) impacted by the other policies on this node. Fortunately, we can apply a similar approach to what is proposed in [4] in order to reduce the complexity of this procedure. It is not required to reroute demands all the way from their source to their destination. Instead, we can limit computations to the subpath starting at the node where the policy is installed and ending at the demands destination. This also allows for demands that share the same destination to be grouped and rerouted together. For example, instead of individually rerouting all demands towards destination X, we can simply sum up their traffic into a single, "merged demand"' for which the new forwarding path towards X has to be computed only once. Overall, this reduces the number of potential path recomputations per move from $\mathcal{O}(N^2)$ to just $\mathcal{O}(N)$, resulting in a substantial computation speed-up.

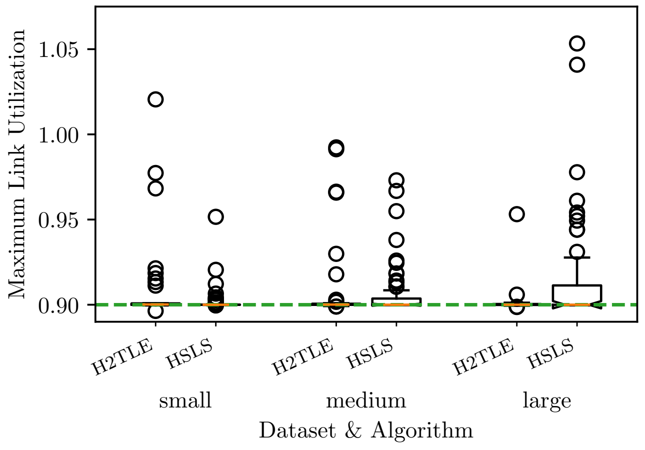


Figure 6: Distribution of the MLUs achieved by HSLS in comparison to those achieved by H2TLE. The dashed green line marks the theoretical optimal MLU. (Six outlier instances with very high MLUs are excluded to improve readability.)

B. Evaluation

To assess the overall optimization quality of our HSLS algorithm, we compare the achieved MLUs to those computed with our H2TLE algorithm. HSLS is run with a time limit of two minutes. Since it features non-deterministic components, we also repeat each experiment five times and average the results across these five runs. The results are shown in Figure 6, once as a whole and once with a zoomed view for better readability. It can be seen that, for the small and medium sized instances, HSLS finds solutions of virtually the same quality as the H2TLE algorithm. For some of the larger instances, results are slightly worse than those of H2TLE. However, those differences are rather small (mostly within 1–3%) and still very close to the theoretical optimum. When interpreting these results, it has to be remembered that H2TLE can require multiple hours to compute while HSLS finds solutions mostly within seconds. In the context of tactical TE, such minor deteriorations of MLUs in exchange for a substantial reduction in computation time are perfectly acceptable.

Furthermore, we also look at the ability of HSLS to remove congestion in a network and the time required to do so, as this is one of the most important use cases of tactical TE

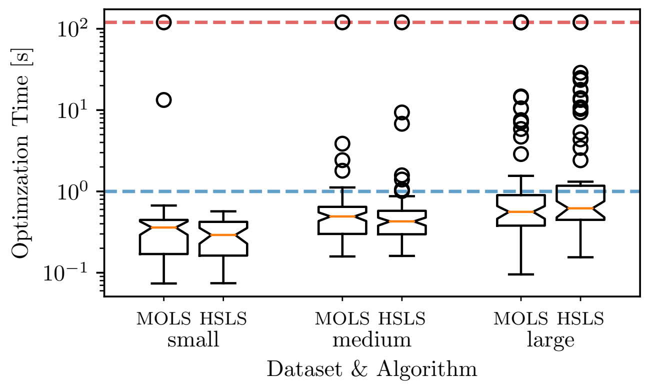


Figure 7: Log-scale distribution of the optimization times required by HSLS to remove congestion compared to MOLS.

(cf. [4]). To better assess the quality of the results, we also compare them to those of the MOLS algorithm. Again, we use a time-limit of two minutes and each experiment is repeated five times. Overall, HSLS is able to remove congestion for 187 of the 190 instances (98.4%). This is slightly better than MOLS which only removes congestion for 185 instances. The optimization times required to remove congestion are depicted in Figure 7, with the dashed blue line marking the sub-second threshold and the dashed red line denoting the two minute time-limit. All instances for which congestion could not be removed within this limit are placed on the latter line. It can be seen that for small and medium sized instances, HSLS tends to be slightly faster than MOLS, probably because it can also utilize E2E policies which provides it with more options for a single move. However, this comes at a price regarding complexity that becomes visible for the larger instances. Here, HSLS is slower than MOLS because in each iteration, we do not only need to check MO moves but also E2E moves and mode switches, as well. While, for the smaller instances, this increased complexity is compensated by the improved expressiveness, it starts to dominate for larger networks resulting in an increase in runtime. However, the overall optimization times of HSLS are still very close to those of MOLS and perfectly suitable for tactical TE as congestion can still be removed mostly in sub-second fashion.

For reasons of space, we cannot conduct a more in-depth evaluation of the HSLS algorithm here. However, the shown results should function as a proof-of-concept that shows that the hybrid SR approach can also be used for fast, tactical TE within limited time constraints and can even improve on the current state-of-the-art in certain aspects.

VIII. Conclusion

In this paper, we studied the benefits of utilizing a *hybrid SR* approach for TE purposes. We have shown that there are TE scenarios in which relying solely on either MO or E2E SR is not sufficient to find optimal solutions. However, when combining both approaches and allowing for the configuration of MO as well as E2E policies, we are able to harness their individual benefits and further improve the quality of TE solutions. This is not only shown at theoretical examples but also confirmed with an extensive analysis on real-world network topologies. It shows that, by utilizing the hybrid SR approach, our proposed H2TLE algorithm is able to outperform comparable state-of-the-art TE algorithms that solely rely on either MO or E2E SR. Furthermore, we have shown that it is possible to develop a fast, heuristic algorithm to facilitate tactical TE based on the hybrid SR approach and that this algorithm performs on-par with current state-of-the-art algorithms that rely solely on MO. As future work, we plan to evaluate our HSLS algorithm in different scenarios and use cases related to the objective of tactical TE (i.e., fast reconfiguration in case of traffic changes and different failure scenarios). In addition to that, we also want to extend our hybrid SR algorithms to support further real-world requirements like latency-bounds or other service-related constraints.

References

[1] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, “RSVP-TE: Extensions to RSVP for LSP Tunnels,” RFC Editor, RFC 3209, 2001.
[2] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, “Optimized Network Traffic Engineering using Segment Routing,” in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2015, pp. 657–665.
[3] A. Brundiers, T. Schüller, and N. Aschenbruck, “Midpoint Optimization for Segment Routing,” in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2022, pp. 1579–1588.
[4] ——, “Tactical Traffic Engineering with Segment Routing Midpoint Optimization,” in *Proc. of the IFIP Netw. Conf. (NETWORKING)*, 2023.
[5] A. Cianfrani, M. Listanti, and M. Polverini, “Incremental Deployment of Segment Routing Into an ISP Network: A Traffic Engineering Perspective,” *IEEE/ACM Transactions on Networking*, vol. 25, pp. 3146–3160, 2017.
[6] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, “The Segment Routing Architecture,” in *Proc. of the IEEE Global Communications Conf. (GLOBECOM)*, 2015.
[7] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, “Segment Routing Architecture ,” RFC Editor, RFC 8402, 2018.
[8] S. Gay, P. Schaus, and S. Vissicchio, “REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms,” *ArXiv e-prints*, 2017.
[9] IBM, “IBM ILOG CPLEX Optimization Studio 20.1.0,” https://www.ibm.com/docs/en/icos/20.1.0, 2020.
[10] Juniper Networks. Junos OS - Enabling IGP Shortcuts. [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/concept/mpls-igp-enabling-shortcuts.html
[11] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet Topology Zoo,” *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.
[12] T. Li, C. Barth, A. Smith, and B. Wen, “Tactical Traffic Engineering (TTE),” Internet Draft draft-li-rtgwg-tte-00, 2023.
[13] Nokia, “7X50 Advanced Configuration Guide,” Tech. Rep., 2015. [Online]. Available: https://documentation.nokia.com/html/0_add-h-f/93-0267-HTML/7X50_Advanced_Configuration_Guide/
[14] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, “Traffic Engineering using Segment Routing and Considering Requirements of a Carrier IP Network,” *IEEE/ACM Transactions on Networking*, vol. 26, pp. 1851–1864, 2018.
[15] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, “On the Practical Irrelevance of Metrics on Segment Routing Traffic Engineering optimization,” in *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, 2018, pp. 640–647.
[16] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, “Segment Routing: A Comprehensive Survey of Research Activities, Standardization Efforts, and Implementation Results,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 182–221, 2021.