Fast Reoptimization with only a few Changes – Enhancing Tactical Traffic Engineering with Segment Routing Midpoint Optimization

Alexander Brundiers *Student Member, IEEE*, Timmy Schüller, Nils Aschenbruck *Member, IEEE*

***Abstract*—Recent advancements in the context of Segment Routing (SR) have shown that the Midpoint Optimization (MO) concept enables a substantial reduction in the number of SR policies required to implement Traffic Engineering (TE) configurations. In this paper, we demonstrate that this concept can also be applied to the use case of tactical TE to considerably reduce the number of network changes required to react to critical events, thereby facilitating lower provisioning times and a generally improved time-to-repair. For this, we develop MOLS, a Local Search-based optimization routine that is able to provide close to optimal solutions within just a couple of seconds. The latter is shown based on extensive evaluations featuring various real-world topologies, including data from the backbone of a Tier-1 Internet Service Provider. Compared to state-of-the-art approaches relying on conventional SR, MOLS achieves similar or better solution quality while requiring substantially fewer configuration changes to implement the respective solutions. Furthermore, MOLS is able to resolve over 99% of overutilization scenarios resulting from different failure types, mostly within sub-second fashion and with an exceptionally small number of changes. Lastly, we also extend MOLS to adhere to specified latency bounds while even fixing initially violated ones.**

***Index Terms*—Segment Routing (SR), traffic engineering, optimization, failure recovery**

I. Introduction

Being able to quickly address and resolve critical network events resulting from failures or unforeseen changes in traffic is a crucial requirement for operators. The lower the overall time-to-repair the better, as even rather short disruptions in service and availability can result in a deterioration of customer satisfaction or considerable monetary losses [2]. With this in mind, a family of Traffic Engineering (TE) approaches [3], [4], [5], [6], [7], [8], [9], [10] aims to *proactively* hedge networks against such events by computing TE configurations that are intrinsically resilient against a specified set of failures or traffic changes. This reduces the time-to-repair to basically zero (or only the respective network convergence time) since no changes or adaptions are required. However, such approaches feature one key weakness: It is basically infeasible to find configurations that are sufficiently resilient against all possible scenarios, both due to the enormous computational complexity but also since such solutions simply do not exist in most cases. Hence, such approaches can only be utilized to hedge a network against a selected subset of scenarios (i.e. the most probable or most severe ones). In order to deal with scenarios not covered by such a preemptive optimization, other *reactive* approaches are required that are able to quickly come up with a set of changes that restore a proper network state. In this context, the primary focus is often put on fast *computation* of the respective solution. While this is undeniably important, the time required to actually roll them out into the network should be considered as well. Obtaining a solution in sub-second fashion is less helpful if it takes minutes to deploy due to an (unnecessarily) high number of required changes. Thus, approaches developed for an application in such *tactical TE* [11] scenarios often aim to also keep the number of required changes to a minimum [12], [13], [14]. This is not only helpful when it comes to raw deployment times, but also makes it easier for human experts to verify the correctness and validity of said changes before rolling them out. The latter is highly relevant for many operators that often are hesitant to carry out updates without such a sanity check [12], [15] since accidental misconfigurations can have even more detrimental effects than the respective failure or traffic change event.

Over the recent years, Segment Routing (SR) has become the premier technology choice across many networks [16] and received a lot of attention, both from research [17] and industry. As a result, there already are multiple SR-based algorithms addressing the use case of tactical TE by allowing to compute suitable TE configurations within a couple minutes [13] or even in sub-second fashion [14]. However, all of them rely on conventional end-to-end (E2E) SR that only utilizes SR policies as end-to-end "tunnels" for individual demands. Recent studies [18], [19] have shown that, in the context of strategic TE, the latter often results in unnecessarily high policy numbers which can be substantially reduced when utilizing the Midpoint Optimization (MO) concept. This is enabled by the latter allowing multiple

This is an extended version of a paper previously presented at the IFIP Networking Conference in June 2023 [1]. *(Corresponding author: A. Brundiers)*

A. Brundiers and N. Aschenbruck are with the Institute of Computer Science, Osnabrück University, 49069 Osnabrück, Germany (e-mail: brundiers@uos.de; aschenbruck@uos.de)

T. Schüller is with Deutsche Telekom Technik GmbH, 48155 Münster, Germany (e-mail: timmy.schueller@telekom.de)

demands to be routed via a single SR policy (i.e. using well-known steering approaches like *IGP Shortcut* [20]) instead of having to configure a dedicated SR policy for each demand that has to be detoured. Its ability to substantially reduce the number of policies required to implement TE solutions render MO an interesting option for tactical TE use cases, as well. However, until now, it has only been studied in the context of strategic TE and long term network optimization. Thus, existing algorithms generally require up to multiple hours to compute solutions, especially for larger networks. This renders them completely unsuitable for the use case of tactical TE and its tight time constraints.

In this paper, we are the first to address this research gap by studying how to apply the concept of SR MO to the use case of tactical TE within tight time constraints, demonstrating that it can provide considerable benefits for the latter. Thereby, our main contributions can be summarized as follows:

- We utilize the well-known concept of Local Search (LS) to develop MOLS, the first MO-capable SR optimization algorithm suitable for the use case of tactical TE.
- In an extensive evaluation featuring various real-world topologies, including those of a Tier-1 Internet Service Provider (ISP), we show that MOLS is able to achieve close to optimal solutions within seconds, that come close to those of state-of-the-art MO algorithms that require magnitudes higher computation times.
- Comparing MOLS to state-of-the-art tactical TE approaches relying on E2E SR reveals that it is able to achieve solutions of similar or even better quality while simultaneously facilitating substantial reductions (of up to 99%) in the number of required policies. The latter facilitates faster deployment and verification times and, thereby, an improvement of the overall time-to-repair.
- We also demonstrate that MOLS resolves over 99% of overutilization scenarios resulting from different failure types or traffic changes, with most solutions being computed in sub-second fashion and requiring only a double or even single digit number of changes to the network.
- We propose an extension for the MOLS algorithm that allows further operational requirements (i.e. latency bounds) to be taken into account during optimization. This extension not only prevents the introduction of new latency bound violations but is also able to fix the majority of initially existing ones while having negligible impact on solution quality and policy numbers.

The remainder of the paper is structured as follows. First, Section II introduces relevant fundamental concepts and background information required for understanding this paper, followed by a discussion of related work in Section III. The MO variant of the SR TE Problem (SRTEP) that we aim to solve in this paper is formally described in Section IV, together with further relevant operational requirements to consider. Section V describes our MOLS algorithm proposed to address the above problem. After this, Section VI introduces our evaluation setup and evaluation results are presented in Section VII. Finally, we also examine how to integrate further real-world constraints into the MOLS algorithm, before concluding the paper in Section IX.

II. Segment Routing & Midpoint Optimization

Segment Routing (SR) [21], is a modern source-routing architecture that allows to control a packets path through the network. This is done with so called *SR policies* [22] which apply a stack of labels (so called *segments*) to packets steered onto them. These labels function as waypoints that have to be visited in the given order, with the paths between them being determined by the respective Interior Gateway Protocol (IGP). There are different types of segments. Originally, they mostly corresponded to the nature of the related waypoint (i.e. *node*, *adjacency*, and *service* segments) but, over time, this list was extended with segments referring to more complex instructions as well [22]. Despite the plethora of available segment types, most of the SR TE literature (e.g., [3], [13], [14], [23], [18]) focuses solely on the use of a limited number of node segments. While this, in theory, restricts traffic steering capabilities, it has been shown that near-optimal results, in many cases, can already be achieved with just two node segments [3], [23], [19]. Additionally, not relying on other types, like adjacency segments, yields other benefits as well (e.g., implicit support of Equal Cost Multipath (ECMP) and generally lower optimization complexity). For these reasons, we also focus our considerations on SR with at most two node segments in the remainder of this paper. Furthermore, similar to other works [23], [13], we also prohibit arbitrary traffic splitting over multiple SR paths, as this is generally not implementable in practice due to hardware limitations [23]. Overall, SR can be used to implement virtually arbitrary forwarding path while providing considerable benefits regarding overhead and scalability over comparable expressive traffic steering techniques like Multiprotocol Label Switching (MPLS) with Resource Reservation Protocol (RSVP)-TE [24] This has lead to SR becoming the premier technology choice for many operators [16] and a plethora of research being conducted in this area. For an overview of the latter, we refer to corresponding surveys like [25] or [17].

In literature, SR is almost exclusively considered in an E2E fashion, with each SR policy being dedicated to route the traffic between just one pair of nodes, 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 onto the policy. However, from a technical perspective, SR can actually be used in conjunction with other steering mechanisms as well. For example, [22, Sec. 8.7] suggest the use of *IGP Shortcut* [20] to determine whether a packet is steered onto an existing SR policy, which is already supported in the most recent hard- and software releases of some of the

large vendors [26], [27]. This overall concept of stepping away from the E2E nature of conventional SR and allowing other steering mechanisms to be used is often referred to as *Midpoint Optimization (MO)* [18], [19], [26], as it allows traffic to be detoured (or "*optimized*") at arbitrary *midpoints* along its path through the network, instead of only its ingress node. While this gives up on the fine-grained, per-flow traffic control of E2E SR, it has been shown that MO still allows for virtually optimal TE solutions that are on-par with those of conventional E2E SR approaches [18], [19]. In fact, if the number of segments is limited, MO even bears the potential to improve solution quality by "mimicking" higher segment-number paths via a concatenation of multiple MO policies. Furthermore, MO can also facilitate TE configurations that are more robust against failure or traffic change events than those obtainable with E2E SR [10]. Most importantly, however, utilizing MO allows for a substantial reduction in the number of SR policies that are required to implement TE solutions as it eliminates the need to configure individual policies for each demand that has to be detoured. The latter is particularly important for many operators as it reduces the introduced overhead while also generally improving clarity and maintainability of the network.

The MO concept can be implemented using different traffic steering mechanisms. However, in this paper, we limit our considerations to the *IGP Shortcut* approach, as it is already supported in recent router hard- and software and considered in related work [18], [19]. Using IGP Shortcut, a packet is steered onto an SR policy if the policy tailend is a downstream router with respect to the IGP path from the policy headend to the packets destination. In order to efficiently prevent loops, packets that already are "inside" of a policy will also not be steered into other ones encountered along the way [27].

III. Related Work

As already mentioned in the introduction, there are basically two categories of approaches when it comes to dealing with failures or changes in traffic: *proactive* and *reactive* ones. Since it is generally not feasible to proactively hedge a network against all possible scenarios, the former can only be applied to cover a certain subset of events, leaving a need for reactive approaches to be able to deal with the uncovered ones. This also applies to approaches like *Sentinel* [28] that rely on precomputing backup paths for certain failures that are automatically activated if the respective failure occurs. As there are exponentially many failure scenarios, it is generally infeasible to precompute individual backup paths for all of them. Nonetheless, this general concept can be used to establish *fast-reroute* mechanisms like Topology Independent Loop Free Alternate (TI-LFA) [29] (or more historically [30]) which allow to locally steer traffic away from a failed link and onto the expected post-convergence path until the network reconverges. Such approaches, however, only aim to minimize disruption during the network convergence stage and do not provide any benefits or protection against overutilization afterwards. Thus, such fast-reroute mechanisms do not replace TE algorithms that allow for a fast (global) re-optimization of a network but can be applied complementary to them.

When it comes to fast re-optimization of segment-routed networks, there are two prevalent state-of-the-art approaches. The first one is the Declarative and Expressive Forwarding Optimizer (DEFO) [13]. It is an optimization architecture designed for the use in large carrier-grade networks. It allows operators to specify various goals or cost-functions that the network should be optimized for. Those optimization problems are then solved by a heuristic optimization algorithm based on Constraint Programming (CP) [31]. While the CP approach itself could theoretically provide truly optimal solutions, it often requires a lot of time to do so (especially on large networks). For this reason, DEFO does not solve the CP problem to optimality but uses an LS-based heuristic to explore the search space more efficiently. This allows DEFO to adhere to the timing constraints of tactical re-optimization by providing reasonably good solutions within minutes or seconds instead of hours. The second algorithm is Segment Routing Local Search (SRLS) [14]. It aims for true sub-second optimization to enable an immediate, automated re-optimization of the network in the case of failures or unexpected traffic changes. The required, exceptionally low computation times are achieved by utilizing a heuristic approach based on LS that iteratively inserts new SR policies into the network to bring down the Maximum Link Utilization (MLU). This enables SRLS to remove congestion in a sub-second fashion for many networks, being significantly faster than DEFO and related Linear Program (LP)-based approaches.

While both of the above approaches achieve very good optimization results within short amounts of time, a substantial number of policies is often required to implement their solutions (cf. Section VII-A), which negatively impacts provisioning times. The high policy numbers mainly result from the fact that both approaches rely on conventional E2E SR which requires a dedicated policy to be installed for each demand that has to be detoured. Especially in large networks with multiple tens of thousands of demands, this can quickly lead to rather high policy numbers even if only a small percentage of demands has to be rerouted. Recent results [18], [19] have shown that this issue can be addressed by stepping away from the use of E2E SR and instead utilizing the concept of MO. However, the only existing optimization algorithm that currently supports MO is Shortcut 2SR (SC2SR) [18], which is designed for strategic, long-term optimization. As such, it often exhibits computation times in the range of multiple hours, especially for larger networks, even when deploying certain *preprocessing* approaches to further speed up computations [32]. This renders it completely unsuitable for the use case of tactical TE and fast re-optimization as we consider them them in this paper. Additionally, SC2SR only

solves a restricted version of the MO optimization problem in which certain practically feasible policy configurations are artificially prohibited in order to allow for an efficient LP formulation (cf. [18], [19, Sections IV-A and V-B]). There are scenarios in which these artificial restrictions result in the algorithm not being able to find the optimal solution but an arbitrarily worse one instead. Contrary to this, the MOLS algorithm proposed in this paper does not suffer from such limitations and is the first one in literature to fully utilize the capabilities of MO.

IV. Problem Statement

In this section, we formally define the optimization problem addressed in this paper while also specifying operational requirements for an optimization algorithm to be applicable for the use case of tactical TE.

A. The MO SR Traffic Engineering Problem

The MO variant of the SRTEP considered in this paper is rather similar to the E2E variant [33] with the only major difference lying in the deployed traffic steering approach. Instead of using SR policies in E2E fashion, we utilize the IGP Shortcut mechanism (cf. Section II) to determine whether a demand is steered onto a policy or not. Thus, the general optimization problem can be defined as follows. Given a network topology as a directed graph $G = (V, E)$ with V and E being the sets of nodes and edges, respectively. Each edge $e \in E$ is annotated with a capacity c_e and an IGP metric value m_e, the latter being used to determine the shortest paths between nodes. Furthermore, a set of demands D is given, with each individual demand $d = (i, j)$ specifying the amount of traffic t_{ij} that has to be routed from node i to j. The goal is to find a set $\mathcal{P}$ of SR policies that, assuming traffic is steered onto them according to the IGP Shortcut mechanism, minimizes the resulting MLU in the network:

$$\min \max_{e \in \mathbf{E}} \left(\frac{\text{load}_e(\mathcal{P})}{c_e} \right) \tag{1}$$

Like the E2E version of the SRTEP [33], this problem is NP-hard when imposing the *integrality constraint*[1] for traffic flows, as we show in Appendix B.

B. Operational Requirements for the Use Case of Tactical TE

The previous definition only describes the general optimization problem of minimizing the MLU using SR MO. Regarding the use case of tactical TE considered in this paper, there are additional peculiarities and (operational) requirements to consider. First of all, full-on optimality of a solution is generally not required as long as it is sufficient to resolve the respective issue (i.e. overutilization). Instead, *timeliness* of a solution basically becomes the most important factor, as issues (i.e., resulting from failures or unexpected traffic changes) need to be resolved *as fast as possible* to quickly restore proper operability of the network [11], [15].

In this context, the primary focus is generally put on fast *computation* of the respective solutions. While this is undeniably important, the time required to actually roll them out into the network must be considered as well. Obtaining a solution in sub-second fashion is less helpful if it takes minutes to deploy due to an (unnecessarily) high number of required changes. Thus, approaches developed for tactical TE applications should aim to also keep the number of required changes to a minimum [12], [13], [14]. This not only keeps provisioning times low, but also makes it easier to *verify* the correctness and validity of said changes (i.e. regarding their conformance to the respective design and operation principles of the network) before rolling them out [12] [15, col. 9].

The latter is especially important in the context of ISP networks which can (at least to a certain extent) be considered as critical infrastructure and, thus, are run with particularly strict requirements and regulations regarding availability and reliability of service. As a result, their operators are often rather hesitant to deploy automated TE systems that dynamically reconfigure the network in fully autonomous fashion, since an accidental network misconfiguration carried out by such a system can potentially have much more detrimental effects than the initial failure or traffic change event. Instead, they often rely on a *human-in-the-loop* for checking and validating the recommended changes prior to rolling them out into the network [12] [15, col. 9].

So, in the context of tactical TE, reasonably good solutions do not only need to be *computed* quickly but also have to implementable with only a small number of changes to the network, in order to ensure that they can be *verified* and *rolled out* fast as well, as the latter aspects also contribute to the overall *time-to-repair*.

V. Algorithmic Approach

The MO SRTEP being NP-hard means that there (most likely) is no efficient way to optimally solve this problem in reasonable time, especially not within the tight time constraints of tactical TE. Therefore, we base our algorithmic approach on the well known heuristic concept of Local Search (LS) [34], which has already been shown to provide exceptional results in combination with other TE approaches like IGP metric tuning [12] and even conventional E2E SR [14], [13]. In the following, we describe the relevant aspects of our proposed Midpoint Optimization Local Search (MOLS) algorithm, like the chosen neighborhood, its exploration strategy and how we carry out an efficient move evaluation. For a better understanding, Fig. 1 also provides a simplified high-level visualization of the general MOLS optimization procedure and where in this section to find details on the respective aspect of the algorithm.

[1] Meaning that a demand is not allowed to be split over multiple "parallel" policies. ECMP-related splitting, however, is allowed.

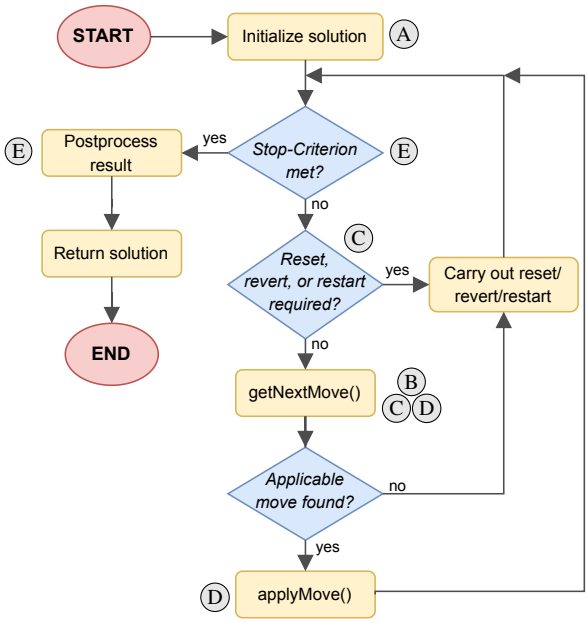


Figure 1: High-level visualization of the general Midpoint Optimization Local Search (MOLS) optimization procedure. The annotated letters in the gray circles indicate in which subsection of Section V details to the respective part are explained (e.g., (D) for Section V-D).

A. Solution Representation & Neighborhood

A solution is represented by the chosen set $\mathcal{P}$ of SR policies to be configured in the network and optimization starts from an empty policy configuration ($\mathcal{P} = \emptyset$) in which all demands are simply routed via Shortest Path Routing (SPR) according to the IGP. The objective function value associated with a solution that we aim to minimize is the MLU resulting from the respective configuration $f(\mathcal{P}) = \text{MLU}(\mathcal{P})$.

The considered neighborhood consists of two types of moves, the *insertion* and *removal* of policy. These are sufficient to connect the solution space (i.e. guaranteeing theoretical reachability of an optimal solution) as every solution can be obtained from any starting solution by carrying out a sequence of insertion and removal moves. Since we are limiting our considerations to policies with at most two segments (identifiable by the triple of start-, intermediate-, and end-node), the size of this neighborhood is within $\mathcal{O}(|V|^3)$.

B. Neighborhood Exploration Strategy

To explore the above neighborhood more efficiently, we follow an approach similar to the one proposed in [14] to focus our exploration on candidates that tend to have a higher chance of resulting in an improvement of the MLU. This is done by selecting a demand d that puts load on the currently most utilized edge e_{mlu} and evaluating all the possible moves that would result in (parts of) this demand being detoured away from e_{mlu}. The demand selection is done randomly with

Algorithm 1 GETNEXTMOVE(S, TL) function of the MOLS algorithm. (S := Current Solution, TL := Tabu-List)

```
move ← None
// First, check removal neighborhood
for policy p ∈ S.GETPOLICIES() do
    if (p, remove) ∉ TL and f(S \ p) < f(S + move) then
        move ← (p, remove)
// Prefer removal if it improves MLU
if f(S + move) < f(S) then
    return move
// Otherwise, check insertion neighborhood
edge_MLU, MLU ← GETMLUEDGE()
for (i = 0; i < n_demand; i++) do
    d ← SELECTDEMAND(edge_MLU)
    P ← GETCANDIDATEPOLICIES(d, edge_MLU)
    for p in P do
        if (p, insert) ∉ TL and f(S + p) < f(S + move) then
            move ← (p, insert)
return move
```

the probability $P(d)$ of selecting demand d being based on the amount of traffic $\text{load}(d, e_{\text{mlu}})$ that the respective demand puts on e_{mlu} using the following formula:

$$P(d) = \left(\frac{\text{load}(d, e_{\text{mlu}})}{\text{Load}(e_{\text{mlu}})} \right)^{\alpha} \tag{2}$$

where $\text{Load}(e_{\text{mlu}})$ denotes the total load of edge e_{mlu} and α is a parameter that can be used to adjust how much emphasis is put on selecting demands that induce a high load on e_{mlu}.

During experiments with this approach, we observed that it is still quite likely that the selected demand is actually relatively far from being the "optimal" one and that the best possible move was not in the evaluated set of moves. To increase the chances of finding the best (or at least a really good) next move, we select not only one but a fixed number (n_{demand}) of demands and evaluate all their respective move sets and select the most improving move. However, we always prefer improving *removal* over *insertion* moves in order to keep policy numbers low. For a better understanding, Algorithm 1 provides a pseudocode description of this neighborhood exploration procedure.

C. Diversification

Only considering improving moves (aka *hill climbing*) does not lead to an optimal solution in most scenarios. Instead, optimization will get stuck at local optima that can be considerably worse than the true global optimum. In the context of MLU minimization, such a scenario can arise if there are multiple edges featuring a utilization equal to the overall network MLU. Finding a single move that reduces the utilization of all those edges can be infeasible, especially if

their number is high and they are located in different parts of the network, leading to the optimization being stuck. To overcome such issues, our algorithm is allowed to accept non-improving and even MLU-increasing moves if no improving moves can be found. The only requirement for accepting such moves is that they reduce the link utilization of at least one of the edges with a utilization equal to the current MLU. This ensures that there is at least some sort of "progress", even when accepting a non-improving move. A problem arising from allowing to accept non-improving moves while also focusing on improving the MLU whenever possible, is the fact that the optimization can get stuck in an infinite loop of carrying out an MLU-increasing move and then immediately reverting it in the next step as this obviously results in an MLU improvement. To prevent this, our algorithm utilizes a *Tabu-List* [35] as short-term memory to keep track of the previously carried out non-improving moves and prohibits their respective inverse moves. The list will be cleared if a new overall best solution is found.

Due to the above approach, MOLS is able to escape local optima but there is still a high chance of optimization being focused on a rather local section of the solution space. In order to facilitate further diversification, we implement *reset* and *revert* mechanics inspired by those used in [14]. After applying a certain number of non-improving moves (n_{reset}) that have not lead to an improvement of our overall best solution, we perturbate the search by carrying out a *reset* move that removes a randomly selected policy from the current solution. If this does not lead to an improvement of the globally best found solution after a certain number of iterations (n_{revert}), the algorithm *reverts* back to the best solution found so far and continues from there. Additionally, we incorporated a *restart* functionality. If there is no further improvement after n_{restart} reset or revert operations, we consider the optimization irreparably stuck and start anew. The respective thresholds can be adapted to fine-tune the algorithm.

D. Move Evaluation

One of the most important aspects of an LS algorithm is an efficient move evaluation. In the context of conventional SR as it is used by SRLS, this is rather straightforward since the insertion or removal of a policy does only impact the forwarding path of a single demand. When using MO, however, a policy can, and most often does, route multiple demands. Hence, inserting or removing a single policy can alter the path of a multitude of demands in the network. As a result, the complexity of evaluating a move additionally depends on the number of demands that are influenced by the respective policy. In the worst case, it can be up to $\mathcal{O}(|V|^2)$ many if the policy impacts a large share of the total demands, resulting in a substantial increase in the complexity of the move evaluation when considering MO instead of E2E SR. Fortunately, this complexity can be considerably reduced by exploiting certain characteristics of the IGP Shortcut-based MO implementation considered in this paper.

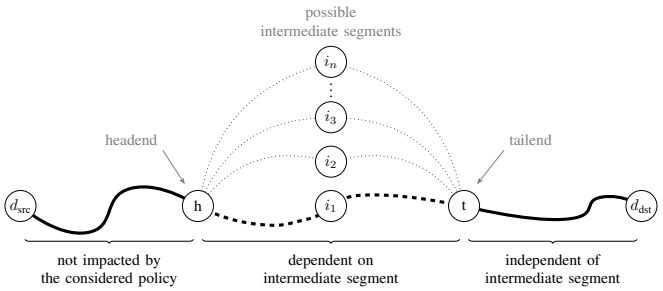


Figure 2: Illustration of which parts of the new forwarding path of a demand d are dependent or independent of the intermediate segment i when considering the insertion of a policy with headend h and tailend t as next move.

First of all, since IGP Shortcut is only *locally significant*[2], a policy can only impact demands that traverse its respective headend. Hence, the set of demands impacted by the insertion or removal of a policy can be efficiently derived by keeping track of the demands passing over each node in the current solution. Furthermore, due to policies being only locally significant, the forwarding path of a demand is only impacted from the respective policy headend onwards. Thus, it is sufficient to only recompute the forwarding path and resulting link utilizations starting from the respective policy headend. Everything "in front" of that stays the same. Most importantly, however, the latter observation also means that the new sub-paths to recompute are basically completely independent of a demand's source. As a result, all traffic flows that visit the same node and share a common destination will follow the same path from there on, irrespective of its source. This allows us to group such demands into a single "merged" one for which we then only have to compute the new forwarding path once, instead of having to carry out computations for each of the individual demands. This procedure is able to reduce the number of potential path recomputations per move evaluation from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|V|)$. Even though this does not yet reach the $\mathcal{O}(1)$ recomputations required for E2E SR, it is a notable reduction in complexity and enables very low computation times as we will see later in the evaluation.

The above considerations mainly focus on making the individual evaluation of a single move more efficient, but it is also possible to combine parts multiple move evaluations to further reduce the complexity of the whole neighborhood exploration process. This is based on the observation that the decision to steer a demand onto a policy is independent of its respective intermediate segment(s). As a result, removing impacted demands from their "old" path is the same for all insertion moves that share a common policy head- and tailend.

[2]Meaning that the existence of the policy is only known to its respective headend and *not* propagated to other (neighboring) nodes.

Furthermore, the new subpath from the policy tailend to the respective destination is also independent of the intermediate segment, leaving only the traffic forwarding "inside" of the policy being impacted when varying the intermediate segment (as schematically illustrated in Fig. 2). These observations allow us to evaluate insertion moves in badges, grouped by the head- and tailend nodes of the associated policy. For each move in a badge, the removal of the "old" traffic and the new paths "behind" the respective policy is the same and, thus, has to be computed only once, basically reducing the number of computations for those from $\mathcal{O}(|V|)$ (i.e. for each intermediate segment) to just a single one. Only the link utilizations resulting from traffic inside the respective policy have to be computed and added individually for each move in the badge. This, however, is rather straightforward as it only requires to adapt the utilizations on the links of two concatenated shortest paths whose associated edges and respective ECMP split-values can be efficiently precomputed.

E. Stop Criteria & Postprocessing

We implemented two possible stop criteria for our algorithm. The first is a simple *timelimit* that aborts optimization after the specified time and returns the so far best solution. The second allows for the specification of a *target MLU* for which the optimization will end if it is reached. Furthermore, during optimization, policies might be selected that do not contribute anything to the final solution (i.e. by a non-improving move or due to the policy "losing" its initial benefit in later steps). Thus, a postprocessing procedure is invoked after the main optimization loop, which checks for and removes such "leftover" policies from the solution.

VI. Evaluation Setup

To examine the performance of MOLS and, thereby, the applicability of the MO concept to the use case of tactical TE, we carry out extensive evaluations regarding different objectives, like achievable MLU, policy numbers, and computation times, and also compare it against other state-of-the-art SR TE algorithms. In the following, when considering MLU values, those are always given relative to the theoretically optimal solution to also provide a reference on how good a solution actually is. In this context, a relative value of 1.0 denotes an optimal solution. The theoretically optimal TE solutions are computed by solving the respective Multicommodity Flow (MCF) problem [36, Ch. 4.4]. It should be noted that, while MCF provides optimal MLU values, its solutions are generally not deployable in practice due to many real-world constraints and restrictions being completely ignored (e.g., the infeasibility of splitting traffic flows in arbitrary fractions). As a result, MCF should be interpreted as a theoretical lower bound for a realistically achievable MLU.

Table I: Repetita data distinguished by number of nodes.

Category	# of Nodes N	# of Instances
Small	$N < 20$	49
Medium	$20 \leq N < 40$	88
Large	$40 \leq N$	72

A. Data

The evaluation of our MOLS algorithm is conducted on three different sets of data. The first one consists of data from the publicly available *Repetita* dataset [37]. It features various different real-world network topologies (many of them taken from the *Topology Zoo* [38]) with artificially generated[3] traffic matrices for each of the topologies. Since the topologies in this dataset vary heavily in size (ranging from 4 to 197 nodes), we subdivided the instances into three categories (`small`, `medium` and `large`) based on their number of nodes. The same was done in [14] and we follow the same categorization for better comparability. The respective node limits as well as the number of instances in each category are listed in Table I. We exclude instances for which the optimal MLU is already achieved by SPR as this renders them uninteresting for TE.

The second dataset is based on real network data collected during the peak-hour in the backbone network of a globally operating Tier-1 ISP. It consists of 19 topology snapshots resembling different expansion states of the network between 2017 and 2021 and the corresponding measured traffic matrices. Depending on the expansion state, the network features around 100 to 200 nodes and 600 to 1100 edges.

For our third dataset, we follow an approach proposed in [18] to create (arguably) harder semi-artificial instances from topology and traffic data that spans multiple expansion states of the same network across a longer time period. The idea is based on the observation that network operators continuously expand their networks to deal with growing traffic. If we now map more recent traffic data onto the topologies from previous expansion states of the network, more traffic is forced through a network with lower capacity, which should result in harder instances for TE. By using this method, we additionally created ten new and arguably harder TE instances from the previously described ISP dataset. In the following, those are referred to as the `ISP Backmapped` dataset.

VII. Evaluation Results

This section presents the results of our extensive evaluations of the MOLS algorithm. All MOLS results are obtained with the following parameter settings which were determined based on preceding experiments which we cannot cover here: $\alpha = 1.5$, and $n_{\text{demand}} = 10$, $n_{\text{reset}} = 20$, $n_{\text{revert}} = 10$, and $n_{\text{restart}} = 10$. If not stated otherwise, a timelimit of two minutes[4] is used for all tactical TE algorithms. Evaluation of

[3]Matrices were generated based on the *gravity model* described in [39].

[4]Generally, the faster a solution is computed the better, but this was identified as an acceptable threshold in consultations with ISP experts.

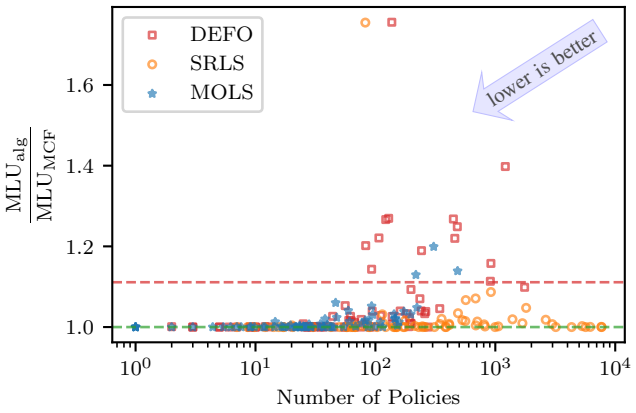


Figure 3: Comparison of the MLUs achieved by DEFO, SRLS, and MOLS for the `Repetita (Large)` dataset and the number of policies required to implement the respective solution. The dashed green and red lines mark the optimal MCF MLU and the overutilization threshold.

non-deterministic algorithms (i.e. SRLS, DEFO, or MOLS) are repeated five times and the following results show the averages across these five runs. All computations are carried out on a 64-core 3.3GHz machine with 500GB of RAM.

A. How does MOLS perform compared to state-of-the-art tactical TE approaches relying on conventional E2E SR?

First, we focus on assessing the performance of our MOLS algorithm compared to DEFO and SRLS, the state-of-the-art approaches utilizing conventional, end-to-end SR.[5] For this, we compare the optimization results obtained by these three algorithms within a timelimit of two minutes regarding the achievable MLU and the number of polices required to implement the respective solutions. The results for the `Repetita (Large)` dataset are depicted in Fig. 3. For each instance, a point marks the MLU achieved by the respective algorithm and the number of policies required to implement this solution. The closer a point is to the bottom left corner, the better a solution as this resembles achieving (near) optimal MLUs with a low number of policies.

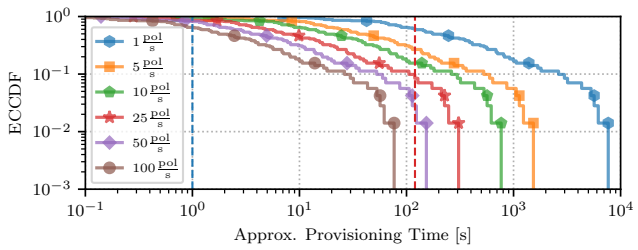
Achievable MLU: It can be seen that SRLS and MOLS are able to prevent overutilization for basically all instances and even achieve virtually optimal MLUs for most of them. While DEFO also achieves close to optimal MLUs for many instances, the share of instances with a noticeable difference to the optimum or even overutilization is considerably larger than for the other two. When looking at which instances are solved sub-optimally by DEFO, it becomes apparent that most of them are among the largest networks in the dataset, featuring close to 100 nodes or more. This indicates that, while the heuristic constraint programming approach of DEFO is considerably faster than conventional LP-based algorithms, it still does *not* scale well enough to reliably deliver good solutions for large networks. This might not appear too detrimental when looking at Fig. 3 as the number of such really large instances in the Repetita dataset is rather small (<15%) as it mostly features older networks. However, this is a bit misleading. Most modern Wide Area Networks (WANs) or ISP backbones are considerably larger, encompassing hundreds of nodes, which should considerably limit the applicability of DEFO for those networks, at least not within the tight timing constraints considered in this paper.

Policy Numbers: Fig. 3 also shows that, while SRLS and MOLS are virtually on-par regarding the achievable MLUs, there are substantial differences in the number of policies required to implement the respective TE solutions. Many SRLS solutions lie in the right half of the plot, corresponding to policy numbers in the range of hundreds or even thousands (note the logarithmic scale!). In contrast, apart from a few outliers, virtually all MOLS solutions are located on the left side, requiring only a double digit number of policies. The root cause of these differences lies in the fact that SRLS relies on E2E SR which requires to configure an individual policy for each demand that has to be detoured. MO, however, allows to detour multiple demands with a single policy. Compared to SRLS, this facilitates an average policy number reduction of around 76% across all instances in the `Repetita (Large)` dataset. When only considering the largest topologies (i.e. $|V| \geq 100$), which are of most practical relevance as detailed before, this even increases to around 97.5%.

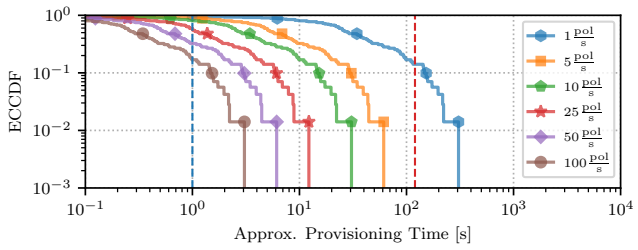
Provisioning Times: Such a substantial reduction in the number of policies required to implement TE solutions constitutes a notable advantage in the context of tactical TE as it allows for considerably improved provisioning times, which we aim to show in the following. Since provisioning times are generally heavily network dependent (i.e. regarding the used hard- and software or communication delays), accurately assessing them without such information is virtually infeasible. However, they can be (roughly) estimated based on the observation that they scale more or less linearly with the number of policies to configure [40], with the exact factor depending on the number of parallel operations that can be carried out by the respective TE controller. As the latter can vary substantially between controllers and there is basically no publicly available information on this topic, we consider different *policies per second* (pps) rates, starting at just 1pps[6] going all the way up to a (probably rather optimistic) rate of 100 pps. The respective approximated provisioning times for the TE configurations computed with SRLS and MOLS on the `Repetita (Large)` dataset are depicted in Fig. 4. Again, the tail of the distribution is the

[5]For this evaluation, we use the SRLS and DEFO implementations publicly available at https://github.com/svissicchio/Repetita.

[6]Based on results reported in [41] regarding the configuration of a single TE tunnel taking up a second or more in a large globally distributed network.



(a) SRLS.



(b) MOLS.

Figure 4: ECCDF of the approximated provisioning times required to roll out the TE configurations computed for the instances in the `Repetita (Large)` dataset with SRLS and MOLS for different *policies per second* values. The vertical dashed blue and red lines mark 1s and 120s, respectively.

most relevant part as it resembles the provisioning times for the largest networks, which come closest to the size of today's WANs and ISP backbones. It can be seen that the substantial reduction in policy numbers that can be achieved when utilizing MO instead of E2E SR also facilitate a considerable improvement in provisioning times. While MOLS solutions can be configured in less than 60s, even for rather low pps values, SRLS solutions can still take over two minutes to configure, even for rather high pps values (50pps). For lower values, this can go up to five or even more than ten minutes. In the context of tactical TE, where the main objective is to restore proper network functionality and service as quickly as possible, such high provisioning times impose a considerable limitation. Furthermore, what has not been taken into account yet is the time required by human experts to check and verify a selected solution before rolling it out. This is even harder to quantify but it should be easily comprehensible that it is considerably more cumbersome and, thus, time consuming to check hundreds or even thousands of changes instead of just a couple tens.

For reasons of space, we cannot provide figures for the two ISP datasets, but those results are qualitatively similar if not better than what we have already seen for the *Repetita* data. MOLS achieves virtually optimal MLUs in sub-second fashion for all instances, again being on-par with SRLS while outperforming DEFO. However, while the latter require hundreds and often even thousands of policies, MOLS achieves these results with only a couple tens of policies, corresponding to an average reduction of around 99% and facilitating a substantially faster verification and deployment.

B. How fast can MOLS resolve overutilization scenarios?

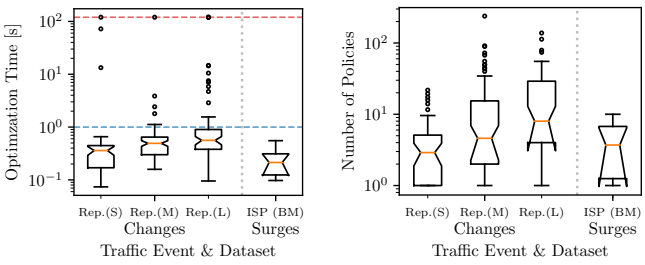
While, in most TE contexts, solution quality/optimality is considered one of, if not the most important objective, this changes to a certain degree when looking at the use case of tactical TE. Here, *timeliness* is generally more important while solutions only have to exhibit a certain level of quality, the latter mainly referring to it being able to restore proper network service. Thus, we also examine MOLS regarding the time taken to resolve overutilization resulting from traffic changes or failures and the number of polices (or policy changes) required to do so.[7] In the following, we again use a maximum time limit of two minutes. Furthermore, scenarios that do not result in overutilization or are generally not solvable with any kind of TE (indicated by having an MCF MLU > 1.0) are *not* included in the results.

*1) **Traffic Changes**:* The first class of scenarios that often cause overutilization on certain links are unforeseen changes in traffic. This can be general changes in traffic characteristics or so called called *traffic surges*, in which the amount of traffic that passes through a network increases drastically within a short time-frame (e.g., caused by large sport events or the release of a highly anticipated game or TV series). Obtaining real-world data related to such traffic change events proves to be difficult as operators are generally hesitant to share network information, especially of such critical events. However, a major change in traffic characteristics is basically synonymous with having to optimize a network for a completely new traffic matrix. Hence, the performance of MOLS for such a use case can be assessed by examining its ability to resolve overutilization resulting from a new traffic matrix presented to it. This is done for all three of the *Repetita* datasets. To imitate the behavior of traffic surges, we utilize our `ISP Backmapped` dataset which basically forces more traffic though the network than expected during its design.[8]

The respective optimization times required by MOLS to resolve overutilization in these scenarios are depicted in Figure 5a. The dashed blue line denotes the one-second threshold, while the upper red line indicates our maximum timelimit of two minutes. Every instance that could not be brought below the overutilization threshold within this time is placed on the red line. It can be seen that MOLS is able to resolve overutilization in sub-second fashion for all traffic surge events and virtually all of the traffic change events on the small and medium sized *Repetita* instances. While, for the larger instances, there are a few more outliers

[7]The DEFO and SRLS implementations do *not* support the specification of a *target MLU*. Hence, a similar, in-depth analysis is not feasible for them.

[8]The `ISP Original` dataset is not considered here, since it does not contain any instances that exhibit overutilization of any link.



(a) Optimization time. (b) Policy numbers.

Figure 5: Distributions of the optimization time and number of policies required by MOLS to resolve overutilization resulting from different traffic change scenarios.

that require a couple of seconds to compute, the majority can still be resolved in sub-second fashion, as well. Overall, MOLS is able to resolve overutilization in less than a second for 169 out of the 190 considered Repetita instances (i.e. 89%). When given up to two minutes, this number increases to around 97%, superseding the results of both DEFO and SRLS, which are only able to do so for around 86% and 93% of instances, respectively. Fig. 5b also shows that, apart from a few outliers, the respective TE configurations can be mostly implemented with only a low double digit number of policies. This means that MOLS is not only able to quickly find suitable configurations but those can also be rolled out within reasonable time (i.e. a couple of seconds).

*2) **Failures**:* The second class of scenarios that tactical TE algorithms frequently have to deal with are hardware failures. In this paper, we focus on the three failure types which are also mainly considered in literature: Link, node, and Shared Risk Link Group (SRLG) failures. The first two correspond to an outage of the respective link or router, while the latter resemble the simultaneous failure of multiple hardware components that share a common risk of breaking (i.e. links running through the same conduit or connected to the same linecard). We evaluate the ability of MOLS to deal with overutilization resulting from such failures by failing the respective components and optimizing the resulting scenario with MOLS in order to remove (potentially occurring) overutilization. For the examination of SRLG failures, detailed information on the respective SRLGs of the considered network are required. We obtained real SRLG data for our `ISP Original` and `ISP Backmapped` datasets but, for the *Repetita* dataset, such information is not available. As it also cannot be reliably inferred from just the network topology, we limit our evaluations on the latter dataset to just link and node failures.

To judge the ability of MOLS to deal with the above failure scenarios, we first look at the percentage share of scenarios for which it is able to find an SR configuration that removes any overutilization within the two minute timelimit. An overview of this is provided in Table II together with the respective

Table II: Percentage of overutilization scenarios resulting from different failure types that can be resolved by MOLS within at most two minutes of computation time. Failure scenarios that do not result in overutilization or that are generally unsolvable via TE are not taken into account. The respective distributions of the percentage of resolved scenarios for each instance are depicted in Fig. 12 in Appendix C.

		# Overutil.-Scenarios			Fixed by MOLS [%]			
Dataset	Failure	min	max	median	min	max	median	avg
ISP (Orig.)	Link	1	12	3	100.0	100.0	100.0	100.0
	SRLG	6	88	37.5	79.1	100.0	100.0	98.7
	Node	2	24	11	92.9	100.0	100.0	99.6
ISP (Back.)	Link	594	748	731	99.2	100.0	100.0	99.9
	SRLG	183	385	298	97.4	100.0	100.0	99.6
	Node	66	92	83	90.1	100.0	98.7	97.8
Repetita (L)	Link	1	449	126	1.7	100.0	100.0	96.1
	Node	2	174	48.5	11.7	100.0	100.0	96.9

number of total scenarios considered. Across all datasets and failure types, MOLS achieves very high average *solution rates* close to 100%. For the `ISP Original` dataset which best reflects realistic scenarios from the real world, MOLS is able to resolve *all* link failure scenarios and also close to or even over 99% of the more severe SRLG and node failures. Regarding the latter two failure classes, it should be noted that this average value is even skewed downwards by two/one outlier(s) for which MOLS performed considerably worse (see the respective *min* value in Table II). When ignoring those, MOLS is able to resolve 100% of the respective failure scenarios in the remaining instances, as well. This shows that, for the most realistic scenario sets, MOLS achieves basically perfect performance, apart from a minute number of outliers.

For the `ISP Backmapped` dataset, MOLS also resolves basically all link and SRLG failures scenarios. Only for node failures, its performance drops slightly to "just" 97.8%, on average. However, it should be taken into account that we are dealing with topology and traffic matrix combinations that were created to be intentionally challenging for TE (cf. Section VI-A) even without any failures. Combining this with node failures, which generally are one of the most complex and severe but also rarest failures encountered in practice [9], creates extremely challenging scenarios (basically a node failure *combined* with a simultaneous unexpected increase in traffic). Furthermore, our approach of using MCF to filter out unsolvable scenarios does *not* guarantee that all remaining instances are indeed solvable when adhering to real-world requirements (i.e. regarding traffic splitting), like MOLS does. Considering this, MOLS resolving overutilization in around 98% of these scenarios is a notable achievement.

Lastly, for the `Repetita (Large)` dataset, MOLS achieves the overall lowest average percentage of solved scenarios, with around 96% and 97% for link and node failures, respectively. However, these lower values are again caused by three substantial outliers that skew the overall average downwards (see Fig. 12). For both scenario sets, they

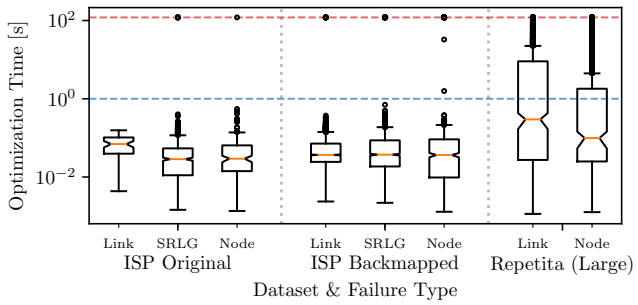


Figure 6: Optimization times required by MOLS to resolve overutilization resulting from different failure scenarios.

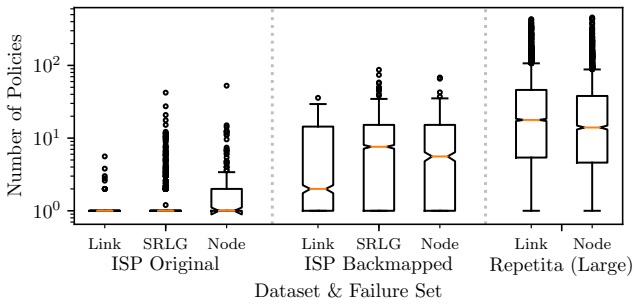


Figure 7: Number of policies required for MOLS to resolve overutilization resulting from different failure scenarios.

result from the same three instances (`Chinanet`, `Dfn`, and `Tw`) which MOLS performs rather poorly on, in the worst case only solving 1.7% of the respective link failure scenarios. We looked into this and found that these three instances, even when not considering any failures, seem to be intrinsically difficult to solve with SR, in general. For neither of them, the SC2SR algorithm [18] is able to match the optimal MCF MLU of 0.9, and even the guaranteed optimal E2E 2SR algorithm [3] is not able to do so for the `Chinanet` and `Tw` instances. Thus, we strongly believe that, for those three instances and their related failure scenarios, overutilization might be preventable with MCF but not with TE approaches that have to adhere to real-world constraints and limitations. When excluding these three outliers, the average percentage of resolved failure scenarios rises to over 99% again, for both link and node failures.

While MOLS is given up to two minutes, it is able to already find suitable solutions within considerably less than a second for virtually all instances in the two ISP datasets (see Fig. 6). While, for the `Repetita (Large)` dataset, optimization times are slightly higher, MOLS still finds suitable solutions in less than 10s for most instance, with the majority still being solved in sub-second fashion. When additionally looking at the number of policies required to implement the respective solutions (see Fig. 7), it becomes apparent that MOLS is not only able to quickly compute solutions but the vast majority of those can be implemented with just a low double digit number of policies. In fact, most of the scenarios in the ISP datasets can be resolved with just ten policies or less, making them easily verifiable by human experts as well as facilitating a rapid deployment.

C. MOLS even keeps up with state-of-the-art strategic TE approaches for SR MO.

As seen in the previous evaluations, despite its heuristic nature, MOLS is able to achieve MLUs that often come close to or even are virtually on-par with the optimal solution. This brings it close to the solution quality of SC2SR [18], the current state-of-the-art strategic TE approach for MO, while requiring substantially lower computation times than the latter. In fact, as already discussed in Section III, MOLS might even be able to surpass SC2SR in terms of solution quality, due to the fact that the latter features some artificial limitations regarding the considered TE solutions. Those do *not* apply to MOLS, which enables it to access regions of the solution space that are not attainable by SC2SR. To examine whether this constitutes an advantage for MOLS, we compare the MLUs achieved with both algorithms, in the following. MOLS is, again, run with a timelimit of just two minutes (often requiring substantially less time to come up with the respective solution), whereas SC2SR is given unlimited computation time. For larger instances, the latter resulted in running times in the magnitude of multiple hours. This difference in computation time should be kept in mind when interpreting the following results.

The relative percentage differences of the MOLS MLUs compared to those obtained with SC2SR for all our datasets are depicted in Fig. 8. A positive value resembles a *higher* (or *worse*) MOLS MLU, while negative values represent instances for which MOLS was actually able to find *better* solutions than SC2SR. It can be seen that, for the majority of instances, the solution quality of MOLS and SC2SR is quite similar, which is already a notable observation as MOLS, especially for the larger instances, requires substantially less computation time. For the *Repetita* datasets, there is a small share of instances for which MOLS is not quite able to match the SC2SR solution. However, the differences are only marginal (generally $<$2%). In fact, many of those would most likely be virtually negligible in a practical deployment since traffic, even when generally being quite stable on the larger scale, is always subject to smaller ongoing variations, which cover up such marginal MLU differences. The more interesting finding, however, is the fact that MOLS is indeed able to outperform SC2SR in a considerable number of instances due to the fact that, contrary to the latter, it does *not* impose any limitations on the explored solution space (cf. Section III). While, generally, differences are only of moderate scale (i.e. $<$10%), for some instances, MOLS is able to reduce the MLU by more than 40%.

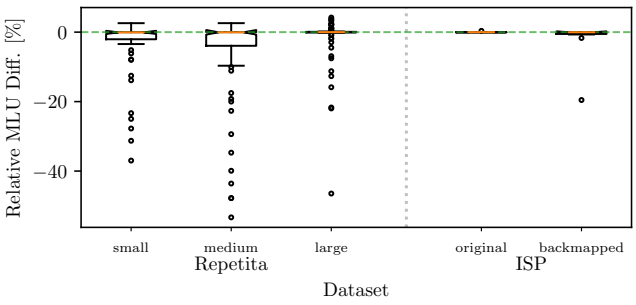
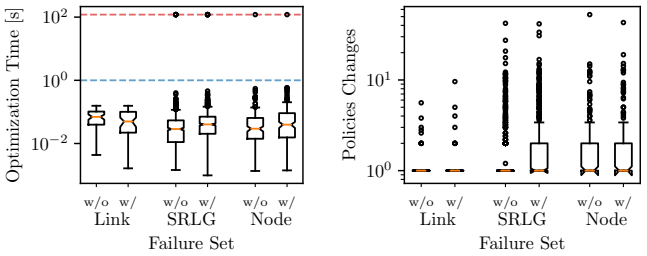


Figure 8: Distributions of the relative percentage difference of the MOLS MLUs compared to those obtained with the SC2SR algorithm on different datasets. The dashed green line marks the SC2SR solution level. Negative values indicate MOLS achieving *better* MLUs than SC2SR.



(a) Optimization time. (b) Policy numbers.

Figure 9: Distribution of the optimization times and policy changes required by MOLS to remove congestion resulting from different failure types in the `ISP Original` dataset *with* (`w/`) and *without* (`w/o`) an initial strategic TE configuration computed with the SC2TLE algorithm [18].

Overall, these results show that MOLS achieves exceptionally good results, which can compete or even surpass those of state-of-the-art optimization approaches that require considerably more computation time and memory. In fact, MOLS finds virtually optimal solutions (i.e. within $<$1% of the theoretical optimal MCF MLU) for the vast majority of instances across all considered datasets. This indicates that MOLS is not only well suited for its originally intended use case of tactical TE, but might also be applicable to other scenarios in which greater emphasis is put on solution quality and optimality, as well. One of those could for example be a continuous/periodic re-optimization of the network to utilize available resources in the most efficient way [41], [42]. We plan to further look into this in the future.

VIII. Extending MOLS to support further real-world requirements

So far we have seen that MOLS performs exceptionally well regarding the achievable MLU, computation times, and policy numbers (as well as the provisioning times related to the latter), which are basically the three most important quality metrics for tactical TE. However, when it comes to a practical deployment, there can be further requirements, individually depending on the respective network. During discussions with our ISP partner, we identified two such requirements which, while not strictly necessary, would ideally be taken into consideration during optimization, as well. In the following, we provide more details on the latter requirements and how they can be implemented into the MOLS algorithm.

A. Modification of existing (strategic) TE configurations

In the previous sections, we used MOLS to compute a suitable TE configuration completely from scratch, replacing any potentially already existing configuration. Due to the low policy numbers and respective provisioning times, this is perfectly doable but, in many networks, TE is not used solely for tactical purposes and reactive re-optimization but also during normal operation (i.e. to enhance general network performance). In that case, completely replacing an existing (strategic) TE configuration in the presence of a critical event is less desirable, as it would need to be restored after the event. Instead, simply adapting the already existing configuration with as few changes as possible is the preferred option. This not only makes it easier/faster to revert back to normal operation but also maintains (most of) the operational benefits of the original configuration, as well.

Implementation: We extend our MOLS algorithm to support the specification of an *initial policy configuration* that the optimization should start from. As MOLS basically only removes a policy if this results in an MLU improvement, initially present policies that do not negatively impact the latter remain untouched during optimization.

Evaluation: In the following, we show that this new feature can be implemented with negligible impact on the quality of the MOLS solutions reported in the previous sections. For this, we repeat the failure scenario evaluation from Section VII-B for the `ISP Original` dataset while supplying a strategic TE configuration computed with the SC2TLE algorithm [18] as initial TE configuration. The first thing to note is that, while the use of an initial TE configuration can influence whether a failure scenario results in overutilization or not, this impact is rather low as the overall number of considered failure scenarios is basically the same as before (cf. Table II). When it comes to the overall percentage of resolved scenarios, this number also stays basically the same compared to computing solutions from scratch with 100%, 99%, and 99.6% for link, SRLG, and node failures, respectively. The same holds true for the respective optimization time and the number of changes (i.e. addition or removal of a policy) that are applied to the initial configuration as can be seen in Fig. 9. At first glance, it might look like the number of policy changes for SRLG failures are considerably higher when starting from an initial configuration as the box of the boxplot

is visibly larger. This observation, however, is misleading and results from the logarithmic scale of the y-axis. The larger box only illustrates some scenarios now requiring *two* instead of just *a single* policy change, which is negligible in practice.

Overall, this shows that MOLS also allows to deal with failures without requiring a complete overhaul of the network configuration, but achieves virtually the same performance while only adapting an already existing TE configuration with just a small (mostly single digit) number of changes.

B. Latency Bounds

Another important constraint, especially in carrier networks like ISP backbones, are *latency bounds* which define the maximum acceptable delay for a traffic demand (i.e. resulting from service level agreements). Especially strategic TE approaches often assume that, initially, these bounds are fulfilled and thus only ensure that they do not introduce any new violations. This, however, is not sufficient in the context of tactical TE. Here, especially failures can result in latency bounds already being violated at the start of the optimization. Thus, fixing initially violated bounds has to be considered as well. However, as it is not guaranteed that all violations can be fixed and fast MLU minimization still remains the primary objective, resolving latency bound violations has to be done in a best-effort fashion to not negatively interfere with the latter. To address these requirements, we extend the MOLS algorithm with the following two functionalities:

- **No New Violations (NNV):** Prevents new latency bound violations to be introduced during optimization.
- **Violation Fix (VF):** Prioritizes resolving existing latency bound violations over MLU minimization, if the MLU is below a user-defined threshold MLU_{pref}.

Implementation: Preventing the introduction of new violations is rather straight forward. During the evaluation of a candidate, we compute the resulting delays for all impacted demands and compare them against there respective latency bound. If the latter is exceeded, the candidate move is rejected. Computing delay changes can be combined with the MLU recomputations that are carried out anyway during the candidate evaluation (cf. Section V) and, thus, do not result in a relevant increase in complexity or computation time. Adapting MOLS to also *fix* existing violations is more complicated as it basically requires to dynamically switch the optimization objective. For this, we implement an alternative neighborhood exploration that is only used if the MLU is below the specified threshold MLU_{pref} for which resolving bound violations should be preferred. This new neighborhood exploration follows a similar approach as the one related to MLU minimization described in Section V. First, it selects the demand that exhibits the highest (relative) bound violation and checks all possible moves that have the potential of reducing the delay of the respective demand. During this, it also makes sure that no move is selected which is ruled out by the tabu-list or results in the MLU surpassing the respective threshold MLU_{pref}. Out of all candidates, the one that reduces the demand's delay the most is selected, with the exception being the encounter of a move that completely fixes the respective bound violation. This is immediately accepted without considering the remaining candidates. For a better understanding, a pseudocode description of this procedure is also provided in Algorithm 2 in the Appendix. Since there is no guarantee that every latency bound violation can actually be fixed, scenarios can occur in which there is no suitable next move that reduces the delay of the respective demand. In order to prevent the optimization from getting stuck in such cases, violations for which no suitable move can be found will be excluded from consideration until a new globally optimal solution is found. Additionally, the solution postprocessing is extended to also feature one last round of violation fix attempts for each violated bound.

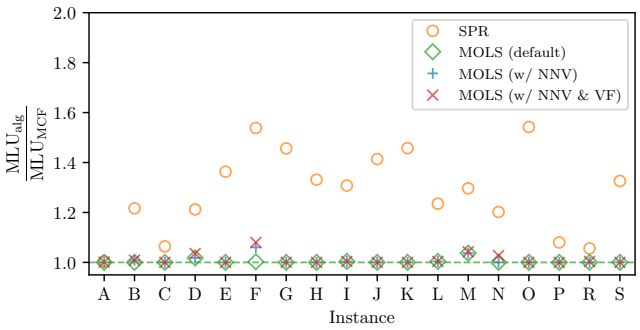


Figure 10: MLUs achieved by MOLS for the instances in the `ISP Original` dataset when using the *no new violations* (NNV) and/or *violation fix* (VF) functionalities. The dashed green line marks the optimal MCF solution.

Evaluation: Latency bound information is generally not publicly available (i.e. not featured in the *Repetita* dataset) as it is considered confidential by most operators. However, we were able to obtain real latency bounds for our `ISP Original` dataset to use in the following evaluation.[9] As before, MOLS is given up to two minutes to find a solution, and the MLU_{pref} value to distinguish between the prioritization of MLU minimization or fixing violated latency bounds is set to 0.8.[10] The corresponding results regarding the achievable MLU and the number of post-optimization bound violations are shown in Fig. 10 and Table III, respectively. It can be seen that the new features do not only effectively prevent the introduction of new bound violations but also enable MOLS to fix the majority of initially existing violations, while having

[9]Latency bound information for instance `Q` was not available. Thus, it is excluded from consideration in the following evaluation.

[10]The latter value is selected w.r.t. operational principles applied in the specific ISP backbone network considered here and, thus, might vary for other networks. However, similar values (i.e. in the 70-90% range) for an acceptable/desirable MLU threshold can be found in multiple different contexts in the literature as well (e.g., in [43], [44], [45], [46]).

Table III: Overview over the number of latency bound violations after optimizing a problem instance with i) the *default* MOLS version, ii) the *no new violations* (NNV) functionality, and iii) the NNV as well as the *violation fix* (VF) functionality.

	Number of latency bound violations per instance																	
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S
default	57	93	70	38	75	99	22	32	29	44	94	1	77	46	31	53	200	111
NNV	2	2	1	2	2	74	2	6	12	12	1	1	1	0	3	2	107	0
NNV & VF	0	0	0	0	0	4	0	1	2	11	0	0	0	0	0	0	7	0

basically no negative impact on the solution quality (i.e. the MLU). Furthermore, utilizing the NNV and VF features does *neither* results in a considerably increase in policy numbers *nor* computation time. The above findings also translate to the optimization of failures. Here, the number of violations is reduced to a single digit number for 94% of the considered scenarios while resolving all violations for around 62%.

When interpreting the latter numbers, it has to be kept in mind that MOLS is designed for the use case of tactical TE in which the primary objective is to quickly obtain a *reasonably good* solution that restores basic network functionality. While it is desirable for such solutions to also adhere to operational soft-constraints like latency bounds, some (slight) violations of the latter are often acceptable [47]. For some failure scenarios, it can even be impossible to adhere to all latency bounds due to important (high speed) network components being unavailable/broken. Overall, we have seen that, MOLS is able to provide (virtually) optimal solutions for the backbone network of a Tier-1 ISP within seconds, while not only preventing the introduction of new bound violations but also fixing the majority of initially existing ones.

IX. Conclusion

In this paper, we successfully applied the concept of SR MO to the context of tactical TE, showing that it can provide considerable benefits for the latter. While it allows to obtain solutions that are on-par or better than those of state-of-the-art approaches relying on conventional E2E SR, it facilitates an up to 99% reduction in the number of SR policies required to implement them. This does not only translate to substantially improved provisioning times but also greatly simplifies the verification of the required changes by human experts. Furthermore, an extensive evaluation on various real-world topologies, including the backbone of a Tier-1 ISP has shown that our proposed MOLS algorithm resolves over 99% of different failure scenarios, mostly within sub-second fashion and just a small (often single digit) number of configuration changes. Finally, we also presented an algorithmic extension that not only prevents MOLS from introducing latency bound violations during optimization but even allows to fix the majority of initially existing violations, as well. Overall, our findings and contributions constitute a notable improvement over the current state-of-the-art, further advancing the field of SR-based TE. Regarding future work, this paper, and other results in literature [18], [19], demonstrate that the concept of MO can provide considerable benefits for SR-based TE. Hence, we plan to further study MO and its possible applications for other TE use cases in the future.

Appendix A
Pseudo-Code Snippets

Algorithm 2 GetNextMoveLatency(S, TL, $\mathcal{B}$) function of the extended MOLS algorithm. ($\mathcal{B}$:= Latency Bounds)

```
1: move ← None
2: d, curDelay_d ← GetMostViolatedBound()
3: C ← GetCandidateMoves(d)
4: for candidate c ∈ C do
5:     newDelay_d ← ComputeNewDelay(S, c)
6:     if c ∉ TL and curDelay_d < oldDelay_d and f(S+c) ≤
       MLU_pref then
7:         move ← c
8:         curDelay_d ← newDelay_d
9:         // If violation is fully fixed, return immediately
10:        if curDelay_d ≤ B_d then
11:            return move
12: return move
```

Appendix B
The MO-variant of the SR TE Problem is NP-hard

Theorem 1. *The general (integral) MO SRTEP is NP-hard.*

Proof. Given any instance of the NP-hard *partition problem* that, for a given set $\mathcal{A}$ of n numbers $a_i \in \mathbb{N}$, aims to answer whether $\mathcal{A}$ can be partitioned into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ with so that the sum of numbers in both sets is equal:

$$\sum_{x \in \mathcal{A}_1} x = \sum_{y \in \mathcal{A}_2} y$$

This can be reduced to an instance of the MO SRTEP as depicted in Fig. 11. For each number a_i in $\mathcal{A}$, a pair of nodes (s_i and d_i) is created with a traffic demand of size a_i to be routed from s_i to d_i. All edges have the same capacity ($c = \sum_{i=1}^{n} \frac{a_i}{2}$), but thick-drawn edges feature a higher metric value than thin-drawn ones. If (and only if) there is a SR policy configuration that achieves an MLU of exactly 1.0 (or 100%), then there also is a solution for the respective partition problem. The reasoning behind this is the fact that the total volume of traffic that has to be routed

from the left side of the network to the right side corresponds exactly to the capacity of the two "bottleneck" paths between those sides ($S \rightarrow A_1 \rightarrow T$ and $S \rightarrow A_2 \rightarrow T$). Thus, the optimally achievable MLU (even with arbitrary traffic splitting) in this network is 1.0. Additionally, since both paths have equal capacity, traffic also needs to be distributed evenly across them for an optimal solution. However, splitting up a demand across multiple forwarding paths (not including ECMP splitting) is prohibited and ECMP itself will never split up a demand across both bottleneck paths as they are not of equal IGP weight. Thus, the only option to achieve the optimal MLU of 1.0 is finding an SR configuration that *partitions* all demands into two sets of equal volume, sending one over the top and one over the bottom bottleneck path. The respective partition sets can, thus, be derived by looking at which demands pass over node A_1 and A_2. *(This proof is derived from the E2E SRTEP NP-hardness proof in [33].)* □

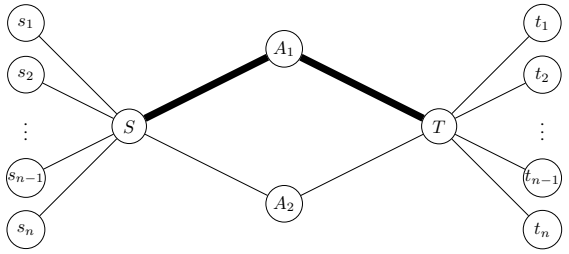


Figure 11: Example on how to encode the partition problem as an MO SRTEP instance.

Appendix C
Supplementary Figures

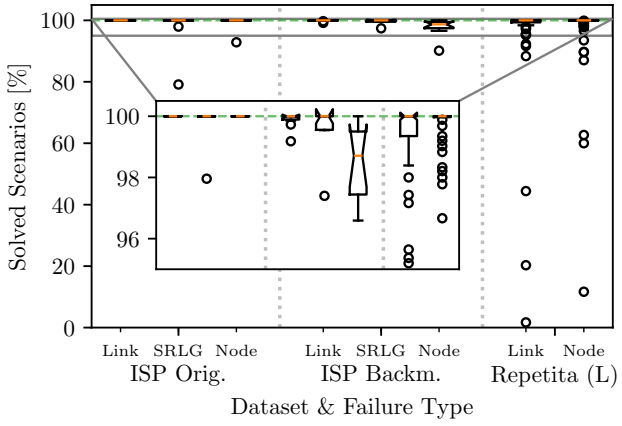


Figure 12: Distribution of the percentage of overutilization scenarios resulting from the respective failure type that can be resolved by MOLS within the two minute timelimit. The inset plot provides a zoomed view for better readability.

Acknowledgment

We thank the anonymous JSAC reviewers for their insightful comments improving the overall quality of this paper.

References

[1] A. Brundiers, T. Schüller, and N. Aschenbruck, "Tactical Traffic Engineering with Segment Routing Midpoint Optimization," in *Proc. IFIP Netw. Conf. (NETWORKING)*, pp. 1–9, 2023.

[2] Uptime Institute, "Annual outage analysis 2023," tech. rep., 2023.

[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)*, pp. 657–665, 2015.

[4] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic Engineering with Forward Fault Correction," in *Proc. ACM SIGCOMM*, pp. 527–538, 2014.

[5] U. Usubütün, M. Kodialam, T. V. Lakshman, and S. Panwar, "Oblivious Routing Using Learning Methods," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, pp. 5226–5231, 2023.

[6] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic-Oblivious Routing for Guaranteed Bandwidth Performance," *IEEE Commun. Mag.*, vol. 45, pp. 46–51, 2007.

[7] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic-Oblivious Routing in the Hose Model," *IEEE/ACM Trans. Netw.*, vol. 19, pp. 774–787, 2011.

[8] F. Hao, M. Kodialam, and T. V. Lakshman, "Optimizing Restoration with Segment Routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1–9, 2016.

[9] 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, 2021.

[10] A. Brundiers, T. Schüller, and N. Aschenbruck, "Live Long and Prosper – On the Potential of Segment Routing Midpoint Optimization to Improve Network Robustness," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, pp. 1–9, 2024.

[11] T. Li, C. Barth, A. Smith, and B. Wen, "Tactical Traffic Engineering (TTE)," Internet Draft draft-li-rtgwg-tte-01, 2023.

[12] 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, 2002.

[13] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks," in *Proc. ACM SIGCOMM*, pp. 15–28, 2015.

[14] S. Gay, R. Hartert, and S. Vissicchio, "Expect the Unexpected: Sub-Second Optimization for Segment Routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1–9, 2017.

[15] T. LaBerge, C. Filsfils, and P. Francois, "Tactical Traffic Engineering Based on Segment Routing Policies ," 2020. U.S. Patent 10742556B2. [Online]. Available: https://patents.google.com/patent/US10742556B2.

[16] R. Mota, "Segment Routing Survey," white paper, ACG Research, 2022.

[17] 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 Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 182–221, 2021.

[18] A. Brundiers, T. Schüller, and N. Aschenbruck, "Midpoint Optimization for Segment Routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1579–1588, 2022.

[19] A. Brundiers, T. Schüller, and N. Aschenbruck, "An Extended Look at Midpoint Optimization for Segment Routing," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 1447–1468, 2024.

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

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

[22] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, and P. Mattes, "Segment Routing Policy Architecture," RFC 9256, RFC Editor, 2022.
[23] 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, 2018.
[24] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, 2001.
[25] 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, 2019.
[26] Cisco Systems, "Cisco WAE Design 7.6.x User Guide," 2022.
[27] Juniper Networks, "Junos OS IS-IS User Guide," tech. rep., 2021.
[28] J. Zheng, H. Xu, X. Zhu, G. Chen, and Y. Geng, "Sentinel: Failure Recovery in Centralized Traffic Engineering," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1859–1872, 2019.
[29] A. Bashandy, S. Litkowski, C. Filsfils, P. Francois, B. Decraene, and D. Voyer, "Topology Independent Fast Reroute using Segment Routing," Internet Draft draft-ietf-rtgwg-segment-routing-ti-lfa-13, 2024.
[30] Z. Wang and J. Crowcroft, "Shortest Path First with Emergency Exits," in *Proc. ACM SIGCOMM*, pp. 166–176, 1990.
[31] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., 2006.
[32] A. Brundiers, T. Schüller, and N. Aschenbruck, "Preprocess your Paths – Speeding up Linear Programming-based Optimization for Segment Routing Traffic Engineering," in *Proc. IFIP Netw. Conf. (NETWORKING)*, pp. 303–312, 2024.
[33] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving Segment Routing Problems with Hybrid Constraint Programming Techniques," in *Proc. Int. Conf. Princ. Pract. Constr. Program. (CP)*, pp. 592–608, 2015.
[34] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., 1997.
[35] F. Glover and M. Laguna, *Tabu Search*. Springer US, 1998.
[36] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., 2017.
[37] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms," *ArXiv e-prints*, 2017.
[38] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011.
[39] M. Roughan, "Simplifying the Synthesis of Internet Traffic Matrices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 93–96, 2005.
[40] E. Husni and A. Bramantyo, "Design and Implementation of MPLS SDN Controller Application based on OpenDaylight," in *Proc. IEEE Int. Symp. Netw., Comput. and Commun. (ISNCC)*, pp. 1–5, 2018.
[41] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-Deployed Software Defined Wan," in *Proc. ACM SIGCOMM*, pp. 3–14, 2013.
[42] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-Driven WAN," in *Proc. ACM SIGCOMM*, pp. 15–26, 2013.
[43] R. D. Doverspike, K. K. Ramakrishnan, and C. Chase, "Structural Overview of ISP Networks," in *Guide to Reliable Internet Services and Applications*, pp. 19–93, Springer London, 2010.
[44] F. Francois, N. Wang, K. Moessner, and S. Georgoulas, "Optimizing Link Sleeping Reconfigurations in ISP Networks with Off-Peak Time Failure Protection," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 2, pp. 176–188, 2013.
[45] A. Nucci, N. Taft, P. Thiran, H. Zang, and C. Diot, "Increasing the Link Utilization in IP over WDM Networks Using Availability as QoS," *Photonic Netw. Commun.*, vol. 9, pp. 55–75, 2005.
[46] D. Otten, M. Ilsen, M. Chimani, and N. Aschenbruck, "Green Traffic Engineering by Line Card Minimization," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, pp. 1–8, 2023.
[47] S. Steffen, T. Gehr, P. Tsankov, L. Vanbever, and M. Vechev, "Probabilistic Verification of Network Configurations," in *Proc. ACM SIGCOMM*, pp. 750–764, 2020.

ALEXANDER BRUNDIERS (Student Member, IEEE) received his master's degree in computer science from Osnabrück University, Germany, in 2020. He is currently pursuing his Ph.D. degree with the Distributed Systems Group of the Institute of Computer Science at Osnabrück University. His research mainly focuses 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 his master's degree and Ph.D. in computer science from Osnabrück University, Germany, in 2015 and 2020, respectively. From 2015–2019 he was working in a joint project with Detecon International GmbH and Deutsche Telekom Technik GmbH. Since then he is working as a devops engineer at Deutsche Telekom Technik GmbH. As such, he works towards 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. degree in computer science from the 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 at Bonn University. Since 2012, he has been holding a Tenured Professorship for distributed systems at the Osnabrück University. His research focus is on dependable and robust networked systems including scenario modeling, traffic engineering, and network security.