Tactical Traffic Engineering with Segment Routing Midpoint Optimization

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

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

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

***Abstract*—Tactical Traffic Engineering (TE) plays a crucial role in the operation of modern backbone networks as it enables operators to quickly react to failures or unforeseen traffic changes. In this paper, we propose a new Segment Routing (SR)-based optimization algorithm called MOLS. It is the first algorithm that applies the recent concept of Midpoint Optimization (MO) for SR to the use case of fast, tactical TE. In an extensive evaluation, based on various real-world topologies, we show that our algorithm performs virtually on par or better than comparable state-of-the-art tactical TE approaches that rely on conventional SR. However, especially for larger networks, the number of SR policies required by our algorithm is substantially lower. This not only reduces the introduced overhead but also allows for faster deployment of the computed configurations since less changes have to be applied to the network. Furthermore, we show that MOLS is able to remove congestion in sub-second fashion for multiple TE use cases. Lastly, MOLS also is the first algorithm in literature to fully utilize the capabilities of MO without any artificial limitations. This enables it to find similar or even better solutions than the only other MO-capable algorithm in literature in just a fraction of the time.**

I. Introduction

Segment Routing (SR) [6] is a recent source-routing architecture that has been shown to offer great Traffic Engineering (TE) capabilities. Its deployment is, therefore, pushed by many operators and vendors and a lot of research is conducted towards exploring the capabilities of SR for different TE use cases (see [24] for an overview). One of these use cases is the field of *tactical* TE. Contrary to *strategic* TE, which strives for optimization in normal operation, tactical TE aims at offering reasonably good solutions within short amounts of time to allow operators to quickly react to failures or sudden traffic changes [17]. There are SR-based algorithms that allow to optimize a network within a couple minutes (e.g., [11]) or even in sub-second fashion [8]. All these approaches, however, rely on conventional SR that only utilizes SR policies as end-to-end "tunnels" for individual demands, which often results in unnecessarily high policy numbers (cf. [5]).

A recently studied innovation in the context of SR is the concept of Midpoint Optimization (MO) [5]. Instead of having to configure a dedicated SR policy for each demand that has to be rerouted, MO allows for multiple demands to be routed via a single policy. This can substantially reduce the number of policies required to implement TE solutions while still achieving optimization results that are on par with conventional SR approaches. The exceptionally low policy numbers that can be achieved when utilizing MO also render it a promising candidate for tactical TE. The lower the number of policies required for a solution, the easier and faster it can be deployed in a network. However, current MO-capable algorithms often require multiple hours to compute solutions on larger instances, which renders them completely unsuitable for tactical TE with its tight time constraints.

In this paper, we address this lack of suitable algorithms and propose Midpoint Optimization Local Search (MOLS), the first MO-capable SR algorithm that is able to optimize even large networks within seconds instead of hours. In an extensive evaluation featuring real-world network data, we show that the solutions found by our algorithm require substantially less policies than current, state-of-the-art tactical TE approaches relying on conventional SR, while being of similar or even better quality regarding the achievable Maximum Link Utilization (MLU). Furthermore, we show that MOLS reliably removes congestion in sub-second fashion in different use cases, including link failure scenarios in the backbone of a Tier-1 Internet Service Provider (ISP). Lastly, MOLS also is the first algorithm in literature that fully utilizes the capabilities of MO without any artificial limitations. This enables it to find better solutions than the current state-of-the-art MO algorithm for multiple instances in just a fraction of the time.

II. Background

This section provides background information on important aspects of this paper: Local Search (LS), Segment Routing (SR), and the concept of Midpoint Optimization (MO) for SR.

A. Local Search

Local Search (LS) [1] is a popular heuristic concept in the area of combinatorial optimization. It is based on the idea of exploring slight alterations of the current solution in the hope of finding an improvement from which the search will be continued. This way, the solution space (or at least parts of it) is iteratively explored until no improvements can be found anymore or until a certain stop-criterion is met. The allowed alterations of the current solution are called *moves*. The set of solutions that can be obtained by applying a certain move-type to a solution is called its respective *neighborhood* and individual solutions are its *neighbors*. For example, in the context of the well known *Knapsack-Problem* a solution could be represented by the set of selected items and the insertion-neighborhood could be defined as all those solutions that can be obtained by inserting another item into the current solution.

An important aspect when designing LS algorithms is the selection of an appropriate move-set and neighborhood. It should be ensured that the solution space is *connected* by the set of feasible moves. This means that, for each solution, there should exist a sequence of moves that leads to an optimal solution from there. However, the resulting neighborhood also should not be too large because it still has to be efficiently explorable. If it is not possible to find a sufficiently small neighborhood that can be fully explored in reasonable time, heuristic approaches can be applied. Those select a certain subset of possible moves that tend to have a higher chance of improving the solution. For example, in the context of TE, when trying to select a traffic flow to be rerouted, it can be helpful to put more emphasis on flows that put large amounts of traffic on the currently highest utilized edge (cf. [8]).

A big problem of standard LS, which always chooses the most improving move in each iteration, is that it often gets stuck at local optima that can be far worse than the global optimum. In order to escape these local optima and to continue the search in other regions of the solution space (*diversification*), a wide variety of approaches can be applied. One of those is called *Tabu Search* [10]. Here, the idea is to also accept non-improving moves under certain conditions to escape local optima. However, in order to prevent same or similar solutions to be explored multiple times, a list of prohibited moves or solutions (the *Tabu-List*) is maintained.

B. Segment Routing

Segment Routing (SR) [6], is a recent source-routing architecture that builds upon the idea of introducing certain waypoints (called *segments* or *labels*) to a packet. The packet is then routed to each of the interim destinations (one after the other) via the Interior Gateway Protocol (IGP) shortest path before being forwarded to its original destination. This allows for the realization of virtually arbitrary forwarding paths in a network, which renders SR a premier tool for TE.

SR labels are added to a packet via so called *SR policies*. Those can be understood as rules configured on individual nodes that determine the stack of labels to be applied to packets steered into the respective policy [7]. There are various types of labels/segments depending on the nature of the related waypoint (c.f., [6]). However, in this work, we focus on the use of only *node-segments* (referring to individual routers in a network). In theory, arbitrarily many segments can be applied to a packet, but in practice this number is often limited upwards by hardware constraints or to reduce the additional overhead resulting from long segment lists. This restricted form of SR is then referred to as k-SR with k denoting the maximum allowed number of segments per packet.

One of the great advantages of SR compared to other TE techniques like Multiprotocol Label Switching (MPLS) [3] with Resource Reservation Protocol (RSVP)-TE [2] is the significantly lower overhead that is introduced into the network. Contrary to RSVP-TE tunnels that have to be configured and maintained on every associated node, SR policies are basically stateless and only need to be configured at the head-end of the policy. All other required information is encoded in the respective label-stack that is applied to a packet. For a more in-depth description of SR and related research see e.g., [24].

C. Midpoint Optimization for Segment Routing

In nearly all of the SR literature, SR policies are used only in end-to-end fashion with a policy exclusively routing the traffic demand originating at its respective start- and destined to its respective endpoint. Other traffic transiting over the startpoint while already in the SR domain is not steered into it. In a carrier-grade ISP backbone, for example, this means that an SR policy between nodes A and B will only route the traffic that enters the network at node A (*ingress*) and that is set to leave it again at node B (*egress*).

An innovation in the context of SR[1] that breaks with this convention is the concept of Midpoint Optimization (MO), recently studied in [5]. It is based on the idea of allowing for multiple demands to be routed via a single policy. This can be realized in multiple different ways depending on the selected way of steering traffic onto a policy. The *IGP Shortcut* MO variation studied in [5] only steers a packet onto a policy if it comes across the policy startpoint and the endpoint lies on the IGP shortest path from the policy startpoint to the destination of the packet. Additionally, packets that already are "inside" of a policy will not be steered into other ones encountered along the way. The latter constraint efficiently prevents the accidental configuration of loops.

It is shown in [5], that MO can significantly reduce the number of policies required to implement TE solutions, while still achieving optimization results of similar quality as conventional SR approaches. If the number of segments is limited, MO even has the potential to achieve better solutions than conventional SR by "mimicking" higher segment-number paths via a concatenation of multiple MO policies.

III. Related Work

In the context of time-constrained TE with SR, there are two prevalent state-of-the-art approaches. The first one is the Declarative and Expressive Forwarding Optimizer (DEFO) [11]. 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) [21]. 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

[1]While applying this concept to SR is new, there are similar approaches for TE based on MPLS tunnels with RSVP-TE (c.f., [4], [19], [23]).

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) [8]. It aims at true sub-second optimization to allow for 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 MLU. This enables SRLS to remove congestion in a sub-second fashion for many networks, being significantly faster than DEFO and Linear Program (LP)-based approaches.

Both of the above approaches focus solely on the use of conventional end-to-end SR and, hence, often require a substantial number of policies to implement their solutions (cf. Section VI-A). MO offers the potential to significantly reduce the number of required policies. However, there is no scientific publication on the use of MO for tactical TE yet. In fact, since it is a very recent innovation in the context of SR, the only scientific publication dealing with MO for SR is [5] published in 2022. It formally describes the concept of MO for SR and discusses its potential advantages as well as disadvantages compared to conventional end-to-end SR. Apart from these theoretical foundations, the authors also propose an LP-based optimization algorithm called Shortcut 2SR (SC2SR) that utilizes MO for the objective of MLU minimization. It is shown that SC2SR is able to achieve optimization results that are on par with current end-to-end SR algorithms for many (real-world) topologies. Simultaneously, the number of required SR policies is substantially reduced, lowering the resulting overhead and configuration effort.

A downside of the SC2SR algorithm is the fact that it 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. [5, Sections V-B and V-C]). 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, our MOLS algorithm does not suffer from such limitations and is the first one in literature to fully utilize the capabilities of MO.

Another downside of the SC2SR algorithm are its high demands regarding memory and computation time. According to the authors, "it can take multiple hours and [...] a couple hundred gigabytes of RAM to find the optimal solution" [5] for larger networks. While this might be acceptable for strategic TE only carried out on a weekly or monthly basis, tactical TE requires significantly lower computations times. As a result, the SC2SR algorithm is basically unsuitable for this use case and, hence, there is a need for MO algorithms that can find good solutions within a significantly shorter amount of time.

IV. Optimization Algorithm

This section first introduces the operational requirements for an algorithm to be used in a tactical TE scenario, followed by a detailed description of our proposed MOLS algorithm.

A. Design Goals and Operational Requirements

The main use case of our algorithm is fast, tactical TE with a focus on MLU minimization. This means that its main objective is to find reasonably good solutions to the MLU minimization problem within limited amounts of time. SRLS [8] puts great emphasis on sub-second optimization with the reasoning of supporting a fast, fully automated network reconfiguration in case of failures or sudden traffic changes. However, our talks with network experts and operators from a Tier-1 ISP have shown that, while such a fully automated solution is considered as a potential ultimate goal in the future, there currently is a lot more interest in *recommender-systems*. In those systems, the computation of required reconfigurations can be automatically triggered as soon as failures or drastic traffic changes are detected, but the computed solutions are not automatically rolled out into the network. Instead, the new configuration should be presented to a human expert that validates and checks it for potential errors or unwanted side-effects (cf. e.g., [16, col. 9, ll. 38ff.]). The reason for this is that, while congestion on some links can have a negative impact on the overall network performance and user experience, an accidental misconfiguration carried out by an automated system can have much more detrimental effects. Therefore, our algorithm is targeted for the use in such a recommender-system. For this, it has to find reasonably good solutions in short time, but is not bound as strictly to the sub-second limit anymore. Instead, while faster computation is preferred, optimization times of up to two minutes are still acceptable according to operator experts.

B. Implementation

We base our algorithm on the *IGP Shortcut* MO variation (cf. Section II-C) because this is the implementation used in literature [5] where it was shown to perform really well regarding TE use cases. Furthermore, this is supposed to be the MO variation that is implemented by many routing manufacturers (cf. e.g., [14, pp. 641ff.]). However, information on this topic is rather sparse and sometimes a bit ambiguous.

Since LS has already proven to be able to provide exceptional results in the context of conventional SR (cf. [8] and [11]), we also base our algorithm on this concept. In the following, we further 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.

Neighborhood: Our neighborhood consists of two types of moves: *insertion* and *removal* of an MO-capable 2SR policy between two nodes s and t with intermediate segment m. These are sufficient to connect the solution space since every possible solution can be obtained from any starting solution by carrying out a sequence of insertion and removal moves.

Neighborhood Exploration: To explore the above neighborhood more efficiently, we decided to follow an approach similar to the one used in [8] 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 selection of a demand is done randomly with the probability $P(d)$ of selecting demand d being based on the traffic $load(d, e_{mlu})$ that this demand puts on e_{mlu}. The exact formula for computing this probability is:

$$P(d) = \left(\frac{load(d, e_{mlu})}{Load(e_{mlu})} \right)^{\alpha} \tag{1}$$

where $Load(e_{mlu})$ denotes the total load of edge e 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 actually 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 do not only select a single demand and evaluate its related moves, but instead select a fixed number of demands and evaluate their respective move sets. Out of all the evaluated moves, we then select the one that reduces the MLU in the network the most.

***Tabu List**:* There are scenarios for which there is no singular move that results in an immediate improvement of the MLU. An example for this would be a situation in which multiple edges actually have a load equal to the MLU. Especially if these edges lie in different parts of the network, chances are high that there is no single move that can reduce all their loads at once. Hence, the overall MLU will always stay the same, irrespective of the selected move. When only accepting moves that improve the current MLU, the algorithm will get stuck in such local optima. 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. A move that actually worsens the MLU is chosen, which then gets immediately reverted in the next iteration step since reverting it obviously results in an MLU improvement. After this, the same non-improving move as before might be selected which is then reverted again. This can continue ad infinitum causing the optimization to be stuck. To prevent this, our algorithm utilizes a *Tabu-List* which keeps 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.

***Diversification**:* In order to allow for even further diversification of the explored solution space, we implemented *reset* and *revert* mechanics inspired by those used in [8]. After applying a certain number of non-improving moves (n_{reset}) that do 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. The respective values for n_{reset} and n_{revert} can be adapted to fine-tune the algorithm to a users needs. Additionally, we also incorporated a *restart* functionality that completely restarts the optimization if there is no further improvement after a certain number of reset and revert operations. In this case, we consider the optimization irreparably stuck and start anew.

***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 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, this number can lie in $\mathcal{O}(N^2)$ with N denoting the number of nodes in the network.

To compute the MLU resulting from a move, the traffic of each impacted demand first needs to be removed from its respective "old" path and then added to the new one. Fortunately, it is not required to compute the full path from scratch but only the sub-path between the starting point of the inserted/removed policy and the respective demands destination, since a policy does not impact the routing decisions "in front" of it.

Furthermore, some of the additional complexity can be reduced due to the following observation: All traffic flows that visit the same node and share a common destination will follow the same path from there on, irrespective of the original demand source. This allows us to group these 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 respective individual demands. This procedure is able to reduce the number of potential path recomputations per move evaluation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$.

***Stop Criteria**:* 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.

V. Evaluation 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 [9]. It features many different real-world network topologies (many of them taken from the *Topology Zoo* [15]) with artificially generated[2]

[2]Matrices were generated based on the *gravity model* described in [22].

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

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 [8] 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 Shortest Path Routing (SPR) as this renders them uninteresting for our evaluation scenarios.

The second dataset used for our evaluation is based on data collected during the peak-hours 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 [5] 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 created another ten instances for our algorithm to be evaluated on. In the following, those are referred to as the *ISP backmapped* dataset.

VI. Evaluation Results

This section presents the results of our evaluations of the MOLS algorithm. All evaluations are carried out on a 64-core 3.3GHz machine with around 500GB of RAM using. For algorithms that rely on LPs (e.g., SC2SR), CPLEX [13] is used as LP-solver. If not stated otherwise, we use a timelimit of up to two minutes for all algorithms, since this is what ISP experts specified as an acceptable upper timelimit for tactical TE (cf. Section IV-A). Experiments with algorithms with non-deterministic components (e.g., SRLS, DEFO, or MOLS) are repeated five times and the following results show the averages across these five runs. All MOLS results are obtained with the following parameter settings (c.f., Section IV-B): $n_{reset} = 20$, $n_{revert} = 10$, and $\alpha = 1.5$. Those were selected based on preliminary experiments which we cannot cover here.

A. Comparison Against Tactical Traffic Engineering Approaches using Conventional Segment Routing

This part of our evaluation focuses on assessing the performance of our MOLS algorithm in terms of tactical TE under limited time constraints, especially in comparison to other state-of-the-art approaches that utilize conventional, end-to-end SR. For this, we compare it against DEFO and SRLS with respect to the achievable MLU and the number of required policies. The latter two algorithms both rely on conventional SR. The only MO-capable optimization algorithm in literature is SC2SR [5]. However, since it is LP-based, it scales very poorly with network size resulting in computation times of multiple hours for larger instances (cf. Section III). Hence, it is (by design) not suited for fast, time-constrained optimization and will, thus, not be taken into consideration here.

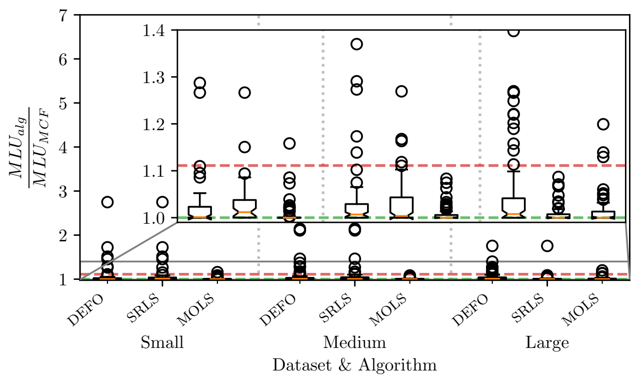
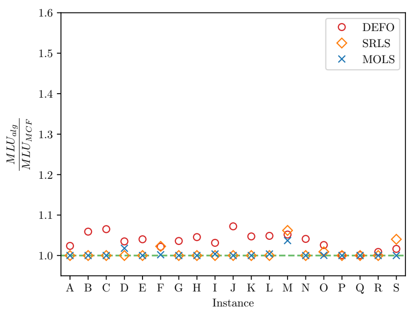


Figure 1: Distribution of MLUs achieved by different tactical TE algorithms on the Repetita dataset.

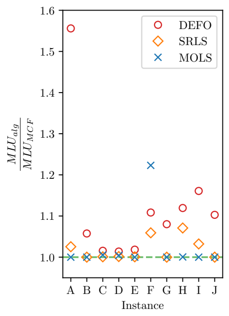
*1) **Maximum Link Utilization:*** Besides directly comparing the MLUs achieved by the different algorithms, we also compare them to Multicommodity Flow (MCF) [18, Ch. 4.4] for reference. MCF is used to obtain the theoretically best achievable MLU. However, MCF solutions are generally not deployable in practice due to many real-world constraints and restrictions being completely ignored in the standard MCF formulation (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.

The respective results for our Repetita dataset are depicted in Figure 1. The MLU values are given in relation to the theoretically optimal MCF MLU. In this context, a quotient-value of 1.0 denotes an optimal solution. The dashed red line denotes the overutilization threshold (MLU > 1.0). It can be seen that DEFO and SRLS achieve quite good results for the small and medium sized instances. However, there often is still room for improvement. MOLS, however, finds optimal solutions in nearly all cases. For the large instances, DEFO gets outperformed by both MOLS and SRLS, with the latter one also achieving slightly better MLUs than MOLS. However, the latter differences are marginally small (often $< 1\%$). In fact, such subtle differences are probably not even noticeable in a practical deployment. Traffic, while mostly being quite stable and predictable, is still subject to small ongoing variations which cover up such marginal MLU differences.

Results for the ISP instances (cf. Figure 2) are quite similar, with DEFO getting outperformed by both SRLS and MOLS in terms of MLU and the latter two again achieving



(a) Original ISP.



(b) Backmapped ISP.

Figure 2: MLUs achieved by different tactical TE algorithms on the instances from the ISP datasets.

rather similar solution quality with some slight advantages for MOLS, especially on the backmapped instances. Overall, in terms of MLU, MOLS is able to consistently outperform DEFO and is on par and sometimes even better than SRLS. It should also be noted that while MOLS is run with a maximum timelimit of two minutes, this limit is seldomly used to its full extent. Often, near-optimal results are already achieved within a couple of seconds or less, especially for the Tier-1 ISP backbone instances (cf. Figure 8 in the Appendix).

*2) **Number of Policies**:* The number of policies required by DEFO, SRLS, and MOLS to achieve the above results on the Repetita dataset are depicted in Figure 3. For the small instances, numbers are relatively similar across all three approaches, but already for the medium sized instances the number of policies required by MOLS is lower than those of DEFO and SRLS while also achieving better MLUs (cf. Figure 1). For the large instances, the benefits of MOLS become even more apparent. Here, MOLS requires just around 60 policies on average, rarely installing more than 100. In contrast, SRLS often needs a couple hundred policies even ranging up to multiple thousands while achieving similar or just marginally better MLUs. On average, MOLS configures around 76% less policies than SRLS on the large instances. DEFO generally requires fewer policies than SRLS but still substantially more than MOLS, while also achieving generally worse MLUs.

For reasons of space, we cannot include a figure depicting the policy numbers required in our ISP datasets. However, the results are very similar to those obtained on the Repetita data. MOLS finds virtually optimal solutions with only a few tens of policies. In contrast, DEFO and SRLS require hundreds or even thousands of policies to achieve solutions of similar or, in the case of DEFO, often even worse quality.

All in all, we can conclude that especially for larger instances, MOLS tends to substantially outperform DEFO as well as SRLS regarding the number of policies while achieving comparable or even better MLUs. This reduction of the number of policies is especially useful in the context of tactical TE. It not only reduces the overhead introduced into the network but also allows for a faster and less complex reconfiguration due to significantly less changes having to be

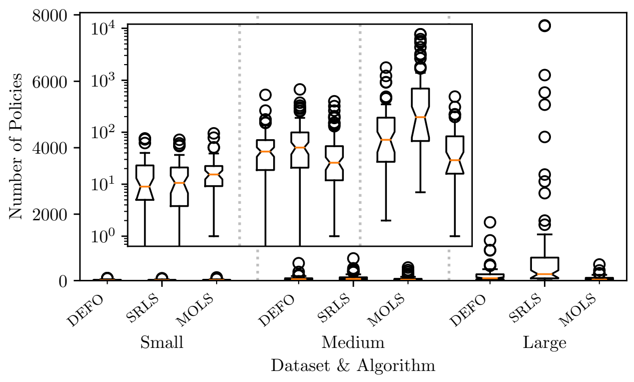


Figure 3: Distribution of the number of policies required per instance by the different algorithms for the Repetita dataset. The smaller inset plot shows the same data but with a logarithmic scale for better readability.

applied. In this context, having to configure multiple thousands of policies, as it is sometimes required by DEFO and SRLS, might even come close to being actually impractical to do.

B. Congestion Removal

While finding truly optimal solutions could theoretically be defined as the ultimate TE goal, a much more important objective in the context of tactical TE is to ensure proper operability of the network. In case of sudden failures or traffic changes that result in an overutilization of some parts of the network, performance can be significantly deteriorated. In such scenarios, it is more important to quickly find a new configuration that removes congestion in the network than actually finding a truly optimal solution. Therefore, we also evaluate MOLS on its ability to remove congestion in different scenarios and the time required to do so.

Traffic Changes: First, we take a look at unforeseen traffic changes. Sometimes the behavior or characteristics of the traffic change in such a substantial manner that previous TE solutions are not applicable anymore. In those scenarios, we quickly need to find a new TE configuration to ensure proper operability of the network. Since such a change in traffic characteristics is basically nothing else than having to optimize the network for a new traffic matrix, the performance of our MOLS algorithm in these scenarios can be evaluated simply by measuring the time it takes to bring the MLU for a given network and traffic matrix below the overutilization threshold. We do this for all those instances of our Repetita dataset that have an SPR MLU of more than 1.0. The distributions of the required optimization times are depicted on the left of Figure 4. The lower 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 remove congestion in a sub-second fashion for virtually all of the small and medium sized Repetita instances, and even though there are a few more outliers above the 1-second threshold, sub-second

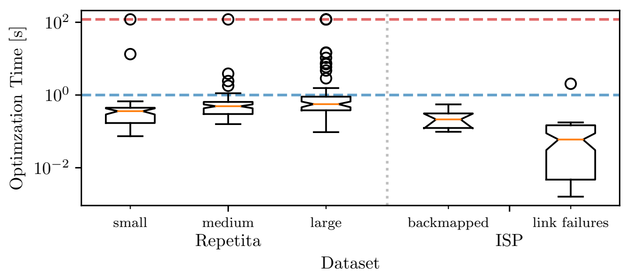
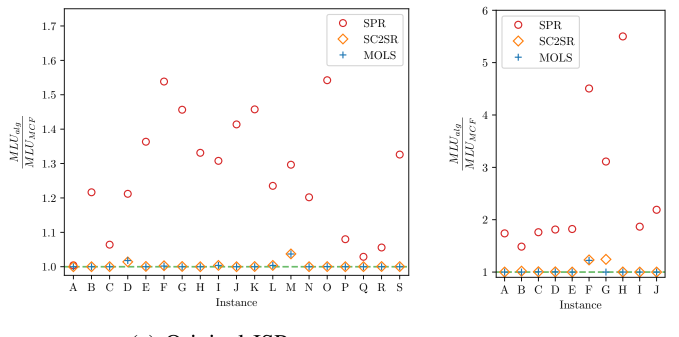


Figure 4: Time taken by MOLS to remove congestion in different networks and scenarios.



(a) Original ISP. (b) Backmapped ISP.

Figure 5: MLUs achieved by different optimization approaches on the instances from the ISP datasets.

congestion removal is also achieved for most of the larger instances. Overall, MOLS is able to remove congestion in a sub-second fashion for 169 out of the 190 considered Repetita instances, corresponding to around 89%. When given up to two minutes, this number increases to around 97%. This supersedes the results of DEFO and SRLS which are only able to remove congestion for around 86% and 93% of instances, respectively.

Traffic Surges: Our second evaluation scenario focuses on a special type of traffic change, namely so called *traffic surges*. Here, the amount of traffic that passes through the network increases drastically within a short time-frame. This can, for example, be caused by large sport events or the release of a highly anticipated game or TV series (cf. [20]) and can result in the need for a reconfiguration to deal with this additional burden. Evaluating the ability of our algorithm to deal with such traffic surges is a bit more complicated since obtaining any data on recorded real-world traffic surges proofs to be difficult. Instead, we decided to simulate it via the traffic backmapping approach described in Section V. By mapping newer traffic matrices onto older expansion states of the ISP network, we basically simulate an unforeseen increase in traffic, as we are forcing more traffic though the network than expected during its design. The required optimization times to remove congestion are also depicted in Figure 4. It can be seen that MOLS is able to reliably remove congestion within less than a second for 100% of the considered scenarios.

Link Failures: Another important use case for TE heuristics is fast congestion removal in failure scenarios. One of the most common types of failures in ISP networks are link failures in which a single link becomes unusable, for example because of hardware malfunction or misconfiguration. Those tend to occur on a daily basis [12] and, hence, need to be dealt with frequently during day-to-day operation. We examine the ability of MOLS to remove congestion for link failure scenarios in our Tier-1 ISP dataset. For each instance, we individually fail each link and then optimize the resulting scenario in order to remove (potentially occurring) overutilization. Again, MOLS is able to remove congestion for 100% of the considered scenarios and, for all but one, requires substantially less than a second to do so (see Figure 4). However, it has to be noted that, while we look at around 600 to 750 individual failure scenarios per instance, the number of scenarios in which the failure actually results in overutilization is substantially lower (around 0 to 12 scenarios per instance). The reason for this low number is that ISP networks are normally build with a lot of redundancy to deal with such failures. Hence, a single link failure rarely brings them to their limits. If, however, such a rare case eventually occurs, it is important to be able to quickly react and resolve it. As shown, this can be done with our MOLS algorithm.

Unfortunately, we are not able to carry out a similar, in-depth analysis of the respective optimization times of DEFO and SRLS. The publicly available implementations of these algorithms do not feature the functionality to specify a target MLU for which the algorithm terminates if it is reached. However, a similar analysis for DEFO and SRLS on the Repetita data was already carried out in the original SRLS paper [8] on a machine that is comparable to the one used by us. Hence, we expect the results presented there to be applicable here, at least in terms of a rough overall comparison. Given the results in [8, Fig. 10], it seems like the SRLS algorithm is slightly faster than MOLS. This is expectable as the candidate evaluation of our algorithm is significantly more complex due to the nature of the optimization problem (cf. Section IV-B). However, the overall ability of achieving sub-second congestion removal seems to be on par between those two algorithms.

All in all, this evaluation shows that MOLS fulfills and even supersedes the requirements for tactical TE. While it would be sufficient to remove congestion in as much as two minutes, MOLS achieves this goal in sub-second fashion for nearly all of our evaluations, including traffic changes and failure scenarios in the backbone of one the worlds largest ISPs.

C. Comparison Against other MO-capable Algorithms

Lastly, we want to compare the performance of our MOLS algorithm to SC2SR [5], even though both were developed for inherently different use cases (fast reoptimization vs. long-lasting optimization in normal operation). The reason for this is that SC2SR is the only other MO-capable algorithm in literature and it actually only solves a restricted version of the MO optimization problem (cf. Section III), which can negatively impact its solution quality. MOLS does not suffer from such limitations and the previous evaluations have shown that it tends to find (near) optimal solutions for most scenarios.

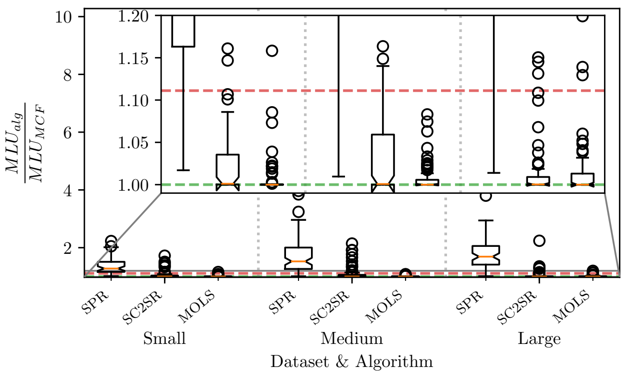


Figure 6: Distribution of the MLUs achieved by different optimization approaches on the Repetita dataset.

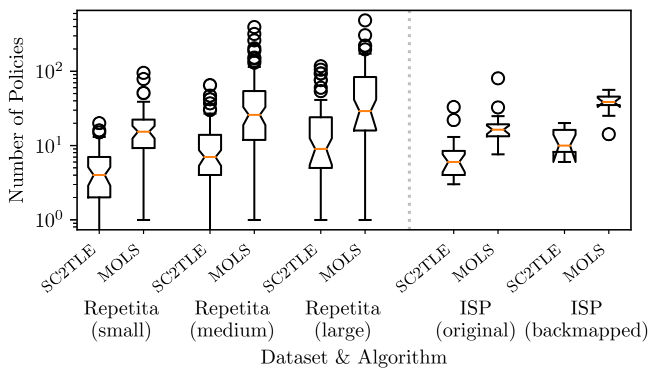


Figure 7: Log-scale distribution of the number of policies required by SC2TLE and MOLS on our different datasets.

Hence, it might be able to actually find better solutions than SC2SR while using just a fraction of the computation time. In the following, MOLS was again run with a timelimit of two minutes while SC2SR was given unlimited time. For larger instances, this 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.

*1) **Maximum Link Utilization:*** The MLU results achieved on the ISP datasets are depicted in Figure 5. For each instance, it shows the MLUs obtained by SC2SR and MOLS relative to the theoretically optimal MLU computed with MCF. This time, we also included the MLU achieved by SPR as an additional reference value that basically represents the current state of routing in many networks that we want to improve upon with our TE approaches. It can be seen that for the original as well as the backmapped ISP instances, MOLS performs on par with SC2SR finding virtually optimal solutions for nearly every instance. In the rare cases in which MOLS does not reach the MCF solution (e.g., instance M of the original dataset), it is still very close to this optimum and SC2SR does not find a better solution either. For one instance (G in Fig. 5b), MOLS is even able to undercut the MLU achieved by SC2SR, finding the optimal solution while the latter fails to do so. This shows that the fact that SC2SR does not utilize MO to its full potential (cf. Section III), can truly be a limiting factor, not only in theory but also in practice. Since MOLS does not suffer from these limitations but fully utilizes the capabilities of MO, it is able to find better solutions.

This becomes even more apparent when looking at the results of the Repetita instances (Figure 6). The number of instances is too large to display individual results. Hence, MLU distributions are shown instead. The dashed red line denotes the overutilization threshold (MLU > 1.0). On the small and medium sized instances, SC2SR achieves good results but still gets outperformed by MOLS. For the large instances, SC2SR finds slightly better solutions but the differences are marginally small and would (most likely) not be noticeable in a practical deployment for which traffic is never perfectly stable.

*2) **Number of Policies:*** The number of policies required by the two algorithms to obtain the previously shown results are depicted in Figure 7. It can be seen that MOLS generally computes solutions that use at least a few more policies than SC2SR when the latter is used in combination with its extension dedicated to policy minimization (SC2SR Tunnel Limit Extension (SC2TLE)). A partial explanation for this is that, as we have shown in the previous section, MOLS tends to find solutions with a lower MLU than SC2SR for quite a few of the Repetita instances. By the nature of the problem, better solutions virtually always require more policies. This assumption is supported by the fact that for the ISP instances, where the achieved MLUs were basically equal between the two algorithms, the differences in policies tend to be smaller. Furthermore, when interpreting these numbers, it has to be remembered that the SC2TLE extension explicitly minimizes the number of policies in a dedicated second optimization step. Hence, it always finds the guaranteed minimum number of policies required to obtain the respective MLU. However, this comes at the cost of additional computation time. For large instances this can lie in the magnitude of hours, further adding to the already high computation times of SC2SR. In contrast, our heuristic computes its solutions mostly within seconds. Even though it might not achieve as low policy numbers as SC2TLE, there is only a reasonably small difference between the two, especially when interpreted in relation to the policy numbers required by conventional, end-to-end SR. Those approaches often require hundreds if not thousands of policies (cf. [5]) while both of the MO algorithms compute solutions with far less than 100 policies for most instances.

All in all, this evaluation has shown that, in terms of MLU, MOLS is able to achieve optimization results of similar or even better quality than the state-of-the-art SC2SR algorithm while only requiring slightly more policies. These are quite impressive results in the context of SC2SR being designed as a strategic TE algorithm that often requires multiple hours of computation time in order to find near-optimal solutions, while MOLS finds solutions in just a small fraction of this time.

VII. Conclusion

MO is a recent innovation in the context of SR that can drastically reduce the number of SR policies required to implement

TE solutions. This renders it a promising candidate for tactical TE since a lower number of policies allows for a faster and less complex network (re-)configuration. In this paper, we studied the applicability of MO for this use case. For this, we proposed MOLS, the first MO-capable optimization algorithm that is able to compute near-optimal solutions within very small amounts of time. In an extensive evaluation featuring real network data from a Tier-1 ISP, we showed that MOLS is able to achieve optimization results that are on par or even better than current state-of-the-art tactical TE algorithms that rely on conventional SR. We further demonstrated that MOLS is able to consistently remove congestion in a sub-second fashion for most of our evaluation scenarios, including link failures in the backbone network of a large Tier-1 ISP. The most important benefit of MOLS, however, lies in the exceptionally low number of SR policies required to implement its solutions. While conventional SR approaches, such as DEFO or SRLS, often require hundreds if not thousands of policies for larger instances, MOLS achieves similar or even better solution quality with only a fraction of these policies, often requiring significantly less than 100. Furthermore, MOLS is also the first algorithm in literature to fully utilize the capabilities of MO without any artificial limitations. This enables it to find solutions of similar or better quality than current state-of-the-art MO algorithms, even though the computation times of the latter can reach multiple hours, while MOLS finds its solutions mostly within seconds. This reduction in computation time is an important step towards a practical use of MO in large ISP networks which can benefit the most from its ability to reduce policy numbers. In fact, an efficient exact algorithm for the MO optimization problem has not been found yet. This renders the observation that heuristic algorithms can be used to find near-optimal solutions within seconds even more important.

As future work, we plan to incorporate further service-related constraints (e.g., delay) into our algorithm. Furthermore, while we have shown that it can deal with link-failures in sub-second fashion, it would be interesting to see how MOLS performs for more complex failure scenarios like Shared Risk Link Group or node failures.

Appendix

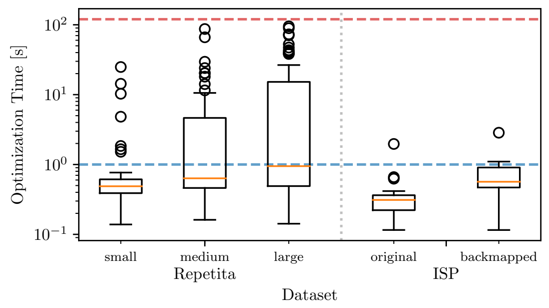


Figure 8: Log-scale distribution of the optimization time required by MOLS to get within one percentage-point of the best found MLUs depicted in Figures 1 and 2.

References

[1] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., 1997.

[2] 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.

[3] D. O. Awduche and B. Jabbari, "Internet Traffic Engineering Using Multi-Protocol Label Switching (MPLS)," *Computer Networks*, vol. 40, pp. 111–129, 2002.

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

[5] 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.

[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 8402, 2018.

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

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

[10] F. Glover and M. Laguna, *Tabu Search*. Springer US, 1998.

[11] 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. of the ACM Conf. on Special Interest Group on Data Communication (SIGCOMM)*, 2015, pp. 15–28.

[12] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of Link Failures in an IP Backbone," in *Proc. of the ACM SIGCOMM Workshop on Internet Measurment*, 2002, p. 237–242.

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

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

[15] 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.

[16] T. LaBerge, C. Filsfils, and P. Francois, "Tactical Traffic Engineering Based on Segment Routing Policies ," US Patent US10 742 556B2, 2020. [Online]. Available: https://patents.google.com/patent/US10742556B2

[17] T. Li, C. Barth, A. Smith, and B. Wen, "Tactical Traffic Engineering (TTE)," Internet Draft draft-li-rtgwg-tte-00, 2023.

[18] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., 2017.

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

[20] Paul Stobart. "2022 Internet roundup: World Cup, Call of Duty and Peaky Blinders spike online traffic". Zen Internet. Accessed on: 01-20-2023. [Online]. Available: https://www.zen.co.uk/blog/posts/zen-blog/2023/01/03/2022-internet-roundup/

[21] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., 2006.

[22] M. Roughan, "Simplifying the Synthesis of Internet Traffic Matrices," *SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 93–96, 2005.

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

[24] 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.