Mind the Paths You Choose: Speeding up Segment Routing-based Traffic Engineering with Path Preprocessing

Alexander Brundiers*[a], Timmy Schüller[b] and Nils Aschenbruck[a]

[a]Institute of Computer Science, Osnabrück University, Friedrich-Janssen-Straße 1, 49069 Osnabrück, Germany
[b]Deutsche Telekom Technik GmbH, Wollbecker Straße 268, 48155 Münster, Germany

ARTICLE INFO

Keywords:
Traffic Engineering
Segment Routing
Optimization
Performance

ABSTRACT

Many state-of-the-art Segment Routing (SR) Traffic Engineering (TE) algorithms rely on Linear Program (LP)-based optimization. However, the poor scalability of the latter and the resulting high computation times impose severe restrictions on the practical usability of such approaches for many use cases. A promising way to address these issues is to preemptively limit the number of SR paths considered during optimization by employing certain preprocessing strategies. In the first part of this paper, we conduct an extensive literature review of such preprocessing approaches together with a large-scale comparative performance study on a plethora of real-world topologies, including recent data from a Tier-1 Internet Service Provider (ISP). In the second part, we then use the insights gained from the former study to develop a novel combined preprocessing approach which also guarantees to not interfere with the satisfiability of practically important latency bound constraints. Our approach is able to reduce the number of SR paths to consider during optimization by as much as 97-99%, while still allowing to achieve close to optimal solutions. This facilitates an around 10× speedup for different LP-based SR TE algorithms, which is more than twice as good as what is achievable with any of the previously existing methods. Finally, we also study the applicability of the path preprocessing paradigm to the use case of tactical TE, showing that it facilitates an around 37% speedup in this context as well. All in all, this constitutes a major improvement over the current state-of-the-art and further facilitates the reliable use of LP-based TE optimization for large segment-routed networks.

1. Introduction

SR has become a premier choice for TE purposes in large networks. It offers great traffic steering capabilities while simultaneously offering good scalability. However, in order to use SR to its full potential, optimization algorithms are needed to compute the best possible TE configurations. In many state-of-the-art approaches (e.g., [2], [3], or [4]), this is done using LP-based optimization because it can provide guaranteed optimal solutions. Its major drawback, however, is its limited scalability and the resulting high computation times for larger networks. Depending on the algorithm and the size of the network, those can reach up to multiple hours or even days. For certain use cases, this is acceptable but, for many scenarios, such high computation times severely limit the practical usability of LP-based SR TE algorithms.

Over the recent years, a variety of preprocessing approaches have been proposed that aim to reduce the problem complexity and, thus, the resulting computation time by preemptively limiting the number of SR paths to consider during optimization. The corresponding research landscape, however, appears to not be well-formed yet, featuring mostly disjoint research efforts (see Section 3). Furthermore, while the individually reported results for those approaches look promising, evaluations are often carried out on a rather limited set of data and varying hardware. This raises questions regarding the generalizability of the results and makes it virtually impossible to compare approaches against each other to select the best fitting one.

In this paper, we aim to address these problems by thoroughly discussing, examining, and evaluating the existing works in the area of SR path preprocessing. Based on this, we then propose a combination of multiple different preprocessing concepts and show that the latter can be used to considerably improve the performance and reliability compared to existing approaches. Thereby, we make the following three major contributions:

This is an extended version of a paper [1] previously presented at the 23rd IFIP Networking Conference in June 2024.

E-mail addresses: brundiers@uos.de (A. Brundiers), timmy.schueller@telekom.de (T. Schüller), aschenbruck@uos.de (N. Aschenbruck).
*Corresponding author.
ORCID(s): 0000-0002-3166-0170 (A. Brundiers*); 0000-0003-2466-3231 (T. Schüller); 0000-0002-5861-8896 (N. Aschenbruck)

- We provide the first extensive literature review of existing preprocessing concepts, discussing their individual merits and potential shortcomings. This is further complemented by a large-scale comparative performance study of these approaches, using not only various publicly available topologies from the Repetita dataset [5] but also recent network data from the backbone of a globally operating Tier-1 ISP.
- Based on the insights gained from these examinations, we then propose a novel combined preprocessing approach. It facilitates an up to 10× speedup for different SR TE algorithms with only minor deteriorations in solution quality, thereby being more than twice as good as previously existing techniques. Additionally, it is also the first preprocessing concept that guarantees to not negatively interfere with the satisfiability of latency bound constraints, which is a crucial property for a practical deployment. Overall, this constitutes a major improvement over the current state-of-the-art and an important step towards the reliable usability of LP-based TE optimization for large segment-routed networks.
- Finally, we also are the first to study the applicability of SR path preprocessing concepts to heuristic optimization algorithms for the use case of tactical TE. In this context, we show that, while the achievable performance gains are considerably lower than for LP-based optimization, path preprocessing can still speed up a state-of-the-art tactical TE heuristic by around 37.5% on average without having a negative impact on the solution quality.

The remainder of this paper is structured as follows. First, Section 2 introduces fundamental concepts and background knowledge relevant for the understanding of this paper. After this, we motivate the need for an extensive literature review and performance study of existing SR preprocessing concepts in Section 3 by identifying and discussing issues in the corresponding research landscape. The respective literature review and performance study is then conducted in Section 4. Based on the latter, Section 5 proposes and examines the possibility to combine multiple preprocessing approaches to further improve performance. This is followed by a study on how to incorporate practically important latency bound constraints into the combined preprocessing approach (Section 6) and an examination of its applicability to the use case of tactical TE (Section 7). Finally, the paper is wrapped up by a discussion of possible limitations of our study (Section 8) before recapitulating on its most important contributions and findings, as well as providing an outlook on possible future research directions in Section 9.

2. Background & Related Work

This section briefly introduces the most important concepts and background information relevant for the understanding of this paper.

2.1. A Primer on Segment Routing and its Applications for Traffic Engineering

SR [6] is a network tunneling technique that implements the source routing paradigm. Its key feature is the possibility to add specific labels (also called *segments*) to a packet, which function as waypoints that the packet has to visit in a given order before heading to its original destination. In practice, these detours are implemented on the level of individual demands using so-called *SR policies* that specify the *SR path* the respective demand should be routed over by defining a list of segments (i.e. waypoints) to add to the packet. Depending on the nature of the related waypoint, different segment-types are used. For example, *node segments* refer to routers, while *adjacency segments* identify individual links. The forwarding paths between the waypoints are determined by the Interior Gateway Protocol (IGP) of the respective network. Overall, SR enables the definition of virtually arbitrary forwarding paths and allows for a precise, per-flow traffic control. For this reason, SR has become one of the premier choices for TE and there is a large body of work regarding SR in general and its applications for TE in particular (cf. e.g., [7]).

One of the fundamental works in the SR TE landscape is [2]. Here, the authors propose an LP-based optimization model that builds the foundation for many subsequent works. With it, they show that even SR with just two node segments (2SR) already enables virtually optimal TE solutions in many scenarios. A slightly adapted version of the respective 2SR LP formulation is shown in Problem 1. The objective is to minimize the Maximum Link Utilization (MLU) denoted by θ. The variables x_{ij}^k indicate the percentage share of the demand t_{ij} between nodes i and j, that is routed over the intermediate segment k. Equation (2) ensures that each demand is satisfied. Equation (3), is the so-called *capacity constraint*. For every edge e, $g_{ij}^k(e)$ indicates the load that is put on e if a uniform demand is routed from i to j over the intermediate segment k. These values are constants and can be efficiently precomputed. All in all, the left side of the constraint denotes the traffic that is put on e by the SR configuration represented by the x_{ij}^k.

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

$$\text{s.t.} \quad \sum_k x_{ij}^k = 1 \qquad \forall ij \tag{2}$$

$$\sum_{ij} t_{ij} \sum_k g_{ij}^k(e)\, x_{ij}^k \leq \theta\, c(e) \qquad \forall e \tag{3}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall ijk \tag{4}$$

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

This is then limited to the edge's capacity $c(e)$ scaled by θ. By minimizing this scaling factor, a SR configuration with minimal MLU is computed. The only difference to the original LP of [2] is that there the x_{ij}^k variables were continuous, allowing for demands to be split arbitrarily across various SR paths. However, such an arbitrary splitting is not feasible in practice [4]. Therefore, newer variations of the 2SR LP generally prohibit splitting demands over multiple SR paths by making the x_{ij}^k binary variables (cf. e.g., [8] or [4]).

A recent innovation in the SR landscape is the *Midpoint Optimization (MO)* concept [3, 9]. Demands no longer have to be optimized individually by deploying dedicated end-to-end SR policies. Instead, a single SR policy can be used to detour a whole set of demands. MO allows for a substantial reduction of the number of SR policies that need to be deployed to implement TE solutions, lowering the configuration effort and overhead in the network. However, the optimization problem becomes inherently more complex, resulting in considerably higher computation times.

2.2. Speeding up SR TE Optimization

A major challenge in the context of SR TE is the scalability of the used optimization algorithms. While LP-based approaches offer the major advantage of providing provable optimal solutions, they scale rather poorly with network size. For small to medium-sized networks, this is no issue since solutions can still be computed within seconds or at most minutes. However, for large networks (e.g., WANs or ISP backbones), computing TE solutions with LPs can take multiple hours or more (cf. [4]), while, in practice, solutions might be needed on a timescale of just a few minutes (cf. [10, 11]). There are different ways to approach these issues. Some focus on the use of advanced mathematical concepts like *column generation* [12] or *constraint programming* [10], while others try to deploy meta-heuristics to compute reasonable good solutions within really short time spans (e.g., [13] or [14]).

A completely different approach to bring down the complexity and, hence, computation time of SR LPs focuses on *preprocessing* the set of SR paths to consider during optimization. Each SR path basically consists of the source and destination node of the packet as well as a set of *middlepoints*[1] (the node segments) that it has to visit (cf. Section 2). In basically all SR LP formulations (e.g., in Problem 1), the model allows for every node segment to be used as a middlepoint for each traffic demand (aka. source-destination pair). While this guarantees optimality, it also is responsible for a large portion of the overall problem complexity. For every demand, the optimization has $|V|^{k-1}$ paths to choose from, resulting in a total number of $\mathcal{O}(|V|^{k+1})$ possible SR paths to evaluate, with $|V|$ being the number of nodes in the network and k the maximum number of segments per path. Here, the preprocessing (or also called *middlepoint selection* [15]) approaches come into play and try to reduce this complexity by limiting the set of available middlepoints per demand and, thus, the set of SR paths to consider prior to optimization. This results in smaller and generally faster to solve LPs.

3. Motivation: Issues of the SR Path Preprocessing Research Landscape

While there already is a plethora of different SR preprocessing concepts proposed in the literature [8, 15, 16, 17, 18, 19, 20, 21], the overall research landscape faces certain problems that negatively impact both the practical applicability of the respective approaches as well as further progress in this area.

Problem 1: The SR path preprocessing research landscape is rather disjointed. Despite there already being a plethora of different SR path preprocessing approaches, the respective research area is not really well-formed yet. This mainly becomes apparent by the fact that research efforts are mostly carried out completely independent of one another,

[1] The term "*middlepoint*" used in the remainder of this paper is **not** to be confused with the term "*midpoint*" from the MO concept (Section 2).

Table 1
Overview and comparison of existing work dealing with path/policy preprocessing concepts for SR TE optimization.

	Compares against ...					Eval. Setup	
	Centrality-based	Stretch-Bounding	Demand Pinning	Smart Node Selection	Policy Domination & Equivalency	Number of Topologies	(Partially) with Real Traffic
[16, 15]	○	✗	✗	✗	✗	2	✗ (0)
[17, 18]	✓	○	✗	✗	✗	6	✓ (1)
[19]	✓	✓	✗	○	✗	4	✓ (2)
[8]	✗	(✓)	✗	✗	○	260*	✗ (0)
This Work	✓	✓	✓	(✓)	✓	91	✓ (19)

*The majority of those topologies are from rather old and small networks (e.g., the Arpanet) and, thus, do not reflect the size and characteristics of modern large-scale networks anymore. Furthermore, due to their small size, most of those instances are generally solvable within milliseconds even when just using standard LP-based optimization, rendering an additional preprocessing step basically obsolete.

with authors being more or less fully unaware of other works in this area. Not only are newly proposed preprocessing concepts seldomly compared and evaluated against already existing ones (see Table 1), but the latter are often not even discussed as related work. As a result, there is no direct comparison between the different approaches, which makes it difficult for potential users to select the best suited one for their use case. Furthermore, such a disconnected research landscape not only makes it more difficult to learn and benefit from others findings, but even leads to duplicate work being done due to already existing concepts being independently "re-invented" (i.e. [8, Sec. 6.6] basically re-proposing a concept similar to the *stretch-bounding* approach of [17]).

Problem 2: Cross-comparisons are further limited by the heterogeneity of the evaluation setups. With there being virtually no direct comparative examinations between the different preprocessing concepts, one currently has to rely on (roughly) cross-comparing the evaluation results of different publications in order to judge the quality and performance of the respective approaches. This, however, is substantially complicated by the fact that basically all publications use different hardware and datasets, with the latter not only varying heavily in size but also in network characteristics.

Problem 3: Evaluations are mostly carried out in an exemplary and rather rudimentary fashion. While some works (e.g., [8]) feature an extensive evaluation on a large set of different networks, others (e.g., [15], [18], or [19]) only test their approaches on a very limited number of networks (see Table 1). Even though their results look promising, the sample size is probably far to low to draw meaningful conclusions regarding the generalizability of the results to other networks. Furthermore, basically all evaluations are carried out on (semi-)artificial data, like the *Repetita* dataset [5] which features topologies based on real-world networks but related traffic matrices are fully artificial (cf. Section 4.1.1). Even if full-on real-world data[2] is used (e.g., from the *Geant* network in [18]), it is rather old and mostly from research networks which do not feature the same characteristics as large ISP backbones. Hence, it is unclear whether the results obtained on such data are directly transferable to a practical application in large commercial networks. And lastly, while some approaches (e.g., Demand Pinning (DP) [20, 21]) sound very promising in theory, there are no evaluation results reported in the literature, at all.

Solution: Extensive Literature Review & Comparative Performance Study. All these aspects constitute limiting factors when it comes to a practical application of such SR preprocessing concepts, while also hampering further research progress in this area. Especially the latter requires a solid understanding and assessment of existing work, based on which those approaches can then be further refined or completely new concepts can be developed. Thus, in the following, we aim to address the above-mentioned problems by conducting the first extensive literature review of existing SR path preprocessing concepts, complemented by a large-scale comparative performance study of the latter on a plethora of different problem instances, including recent real-world data from a globally operating Tier-1 ISP (see Table 1). We believe that this can serve as a valuable reference point lying a solid foundation for future research in this area by not only clearly charting the existing research landscape but also examining and comparing existing work in terms of its performance. In fact, we even demonstrate this by utilizing the insights gained from this study to propose a novel combined preprocessing approach that outperforms all existing ones by at least 2× regarding the achievable speedup, while also taking practically highly relevant latency requirements into consideration (see Sections 5 and 6).

[2]Meaning topology *and* traffic data obtained from real operational networks.

Table 2
Graph properties of the topologies in the two datasets used for evaluation.

	Repetita (72 Topologies)				ISP (19 Topologies)			
	min	max	avg	stdDev	min	max	avg	stdDev
Nodes	40	197	68.69	31.81	108	186	143.11	29.90
Edges	86	486	171.94	77.31	660	1064	897.16	136.25
Density [%]	1.26	7.82	4.30	1.48	3.09	6.57	4.73	1.35
Diameter	4	35	11.79	7.57	6	8	7.32	0.58

4. SR Path Preprocessing – Extensive Overview & Comparative Performance Evaluation

In the following, we provide a detailed overview on (to the best of our knowledge) all existing preprocessing concepts in the SR TE literature, complemented by a comparative performance study of the respective approaches. In doing so, we focus on preprocessing concepts that primarily aim at reducing computation time and also are (more or less) universally applicable in most large-scale networks (i.e. WANs or ISP backbones). Other, domain-specific approaches that, while also implementing some form of path selection or preprocessing, are only applicable to quite niche use cases or applications (i.e. Deterministic Networking [22]) are out of scope and will not be considered here.

4.1. Evaluation Setup

Similar to the related work, we evaluate the effectiveness of the preprocessing approaches based on the 2SR LP (Problem 1). It is the de-facto standard LP for SR TE and builds the foundations for a wide body of derivative work (cf. Section 2) to which the findings should be transferable. Our evaluation focuses on the resulting MLU deterioration and the achievable speedup compared to the standard 2SR implementation. In this context, the *speedup factor* is used to characterize the performance improvements regarding computation time achievable with the different preprocessing approaches. As such, it is calculated by dividing the computation time $T_{default}$ required by the default algorithm (without any preprocessing) by the computation time of the same algorithm $\mathcal{A}$ when the respective preprocessing approach p is applied beforehand (*including* the computation time of the respective preprocessing):

$$SpeedupFactor(p, A) = \frac{T_{default}(\mathcal{A})}{T(p(\mathcal{A}))} \tag{5}$$

Lastly, all computations are carried out on the same 64-core 3.3GHz machine with around 500GB of RAM and using CPLEX 20.1.0 [26] as LP-solver.

4.1.1. Data

We carry out our evaluation on two sets of data. The first one consists of data from the publicly available *Repetita* dataset [5]. It features topologies of real-world networks (mostly WANs or ISP backbones) collected in the *Internet Topology Zoo* [27] and artificially generated traffic matrices (using a *random gravity model* [28]) for each topology. In addition to that, each topology also comes with two sets of IGP metrics (*unary* and *inverse capacity*). However, we limit our evaluations to only the *unary* metric set as previous results [29] have shown that the impact of different metric designs on SR performance is often negligible. Other results further indicate that SR middlepoint selection approaches also seem to be quite robust regarding the underlying metric (cf. e.g., [18]). We further discard all instances already solved optimally by Shortest Path Routing (SPR). Finally, since for smaller networks with just a couple tens of nodes, even rather complex LPs are generally solvable within seconds or less (cf. e.g., [30]), there is basically no practically relevant improvement to achieve for these networks. Therefore, we limit our evaluations to larger networks with at least 40 nodes, leaving us with a total of 72 networks featuring 40 to 197 nodes and around 85 to 500 edges (cf. Table 2).

Complementary to the Repetita data with artificial traffic, we also carry out evaluations on a second set of data collected from the backbone network of a globally operating Tier-1 ISP. It features 19 topology snapshots that resemble different expansion states of the network between 2017 and 2021 and a real traffic-matrix collected during the peak-hour of the respective day. Table 2 lists some further information on the most important graph properties across the respective topologies in each of the two dataset used in our evaluation. In this context, parallel links are counted as a single edge and the density denotes the ratio of (non-parallel) edges in the graph relative to a complete one.

4.1.2. A Primer Regarding CPLEX-Related Outliers

In rare occasions, there can be outliers with a speedup factor below one (e.g., in Figure 2c for $\alpha_{DP} = 0.3$ for the ISP data). This means using the respective preprocessing approach actually resulted in an increase in computation time. While this is rather surprising at first thought, there is a simple explanation for this phenomenon. LP-solvers like CPLEX stop optimization only if they find a "proof" that the currently best solution is truly optimal (or within a small margin to the optimum). This *optimality gap* is computed by comparing the currently best found solution against a lower bound for the best possible objective value which is continuously updated (increased) during optimization. If the gap between the lower bound and the best found solution is sufficiently small, the solution is considered to be optimal. By preemptively limiting the allowed set of available SR paths, the rare scenario can occur in which we prohibit a path that might not be required for an optimal solution but that facilitates a quick proof of optimality. There might be other options for such a proof but if these are explored in a much later stage of the *branch-and-cut search*, proving optimality and, thus, the whole optimization process might take substantially longer.

While the possibility of actually degrading performance when applying preprocessing approaches might be concerning, there is a straight-forward solution. Increasing the allowed *optimality gap* of CPLEX by only a small amount allows for an easier proof of "optimality" (even without the paths excluded by the preprocessing). In our experiments, increasing the optimality gap from the default 10^{-4} to around 10^{-3} proved promising to resolve these issues without having a practically relevant negative impact on the solution quality. Solutions are still within 0.1% (instead of 0.01%) of the optimum. In the context of practical deployments, such minute differences are basically negligible since traffic, while mostly being quite stable, is still subject to small ongoing variations. Those (most likely) cover up such marginal MLU differences.

4.2. Centrality-based Approaches

One of the first works that came up with the idea of preemptively limiting the number SR paths that are considered for optimization are [16, 15]. They only allow a certain subset of nodes as *middlepoints* (aka. intermediate segments) for SR paths and propose to use *graph centrality* metrics to select "important" or "central" nodes into this subset. This idea is evaluated in the context of datacenter networks and ISP backbones for various subset sizes and with different centrality measures. It is observed that, out of all considered centrality metrics, selecting the allowed middlepoints based on their *Group Shortest-Path (GSP) centrality* [23] performs best. For a group of nodes $\mathcal{G}$ the GSP centrality is defined as:

$$C_{gsp}(\mathcal{G}) = \sum_{s,t \in V | s,t \notin \mathcal{G}} \frac{\theta_{st}(\mathcal{G})}{\theta_{st}} \tag{6}$$

with θ_{st} denoting the total number of shortest paths from s to t and $\theta_{st}(\mathcal{G})$ being the number of shortest paths from s to t that include any node in $\mathcal{G}$ [15]. In other words, it characterizes how "central" a group of nodes is based on the number of shortest paths that run through this group.

Overall, it is shown that, when focusing on only a small number of nodes as available middlepoints, computation times can be substantially reduced. However, this comes at the prices of a considerable deterioration in solution quality. While the authors argue that this can be a sensible trade-off to make, in practice, an MLU deterioration is only acceptable up to a certain point. Furthermore, limiting the available middlepoints to the same small set of nodes for all demands can result in severe violations of certain operational latency constraints (i.e. from service level agreements). For example, if, for a globe-spanning network, the most "central" nodes are all located in Europe, intra-US traffic either needs to always follow its shortest path or be detoured all the way over a node in Europe, most likely exceeding latency bounds. In addition to that, if the number of SR paths grows larger and they are all forced over the same handful of middlepoints, this can put additional burden on the routing hardware of these nodes.

Implementation Details: We limit our evaluation of the centrality-based middlepoint selection approaches to the GSP centrality as it was identified as performing best in previous works (cf. Section 4.2). To get around the issues regarding the high algorithmic complexity of its computation [15] (and the resulting high computation times), we use an approximation algorithm provided by *NetworKit*[5] which approximates the node group with maximum centrality up to a given accuracy ϵ. Such an approximation would (most likely) also be used in a practical deployment due to the substantial performance gains. For our evaluations, we use $\epsilon = 0.005$ which allows for computing the respective maximum centrality group in a couple of seconds for most instances.

[5] https://networkit.github.io/

Evaluation Results

The evaluation results regarding the MLU deterioration and the achievable speedup for the centrality-based middlepoint selection are depicted in Figures 1a and 2a, respectively. The orange boxplots resemble the respective distributions for the Tier-1 ISP dataset and the blue boxplots for the Repetita dataset. For both of these datasets, it can be seen that allowing only a small set of nodes as available middlepoints can result in a substantial (i.e. 10–20×) speed-up in computation time. This, however, comes at the price of a significant deterioration of the overall solution quality. In the worst cases, MLUs increase by up to 45% for the ISP dataset and by more than 80% for the Repetita instances. By increasing the number of allowed middlepoints, these MLU deteriorations can be reduced but this also results in an increase in computation time. Ultimately, operators have to make an individual decision regarding the acceptable trade-off between speedup and resulting MLU deterioration. This might vary for different use cases, but from our experience, the highest acceptable MLU deterioration for many scenarios probably lies somewhere around 10-15%, at most. Based on this, we highlight in each plot the parameter configuration that produces the highest speedup while still allowing for, in our experience, practically usable MLUs. For the Repetita data this is at around 40% of the total nodes being allowed as middlepoints, respectively. For the ISP instances, the box and whiskers are already below the 10% deterioration threshold earlier (i.e. at just 3%). However, since the dataset only comprises 19 instances, the three "outliers" that (considerably) surpass this threshold still make up over 15% of the dataset. To improve the solution quality for these instances, a substantially higher percentage of allowed middlepoints is needed (i.e. around 45%). Hence, we argue that the number of middlepoints required to obtain practically usable solutions is (to some extent) instance-dependent but, in general, selecting at least 40-45% of nodes as available middlepoints seems to be required to still obtain good solutions. This translates to an average speedup factor of around 3-4 for both the Repetita dataset and the ISP backbone (cf. Figure 2a). It has to be noted, however, that for these parameterizations there is still a non-negligible number of instances with a significant MLU deterioration left.

Overall, our results generally confirm the findings of [15]. Selecting only a few central nodes as available middlepoints can substantially reduce computation times, but also results in significant deteriorations of the overall MLUs, especially when only allowing a small numbers of middlepoints to be used.

4.3. Stretch-Bounding

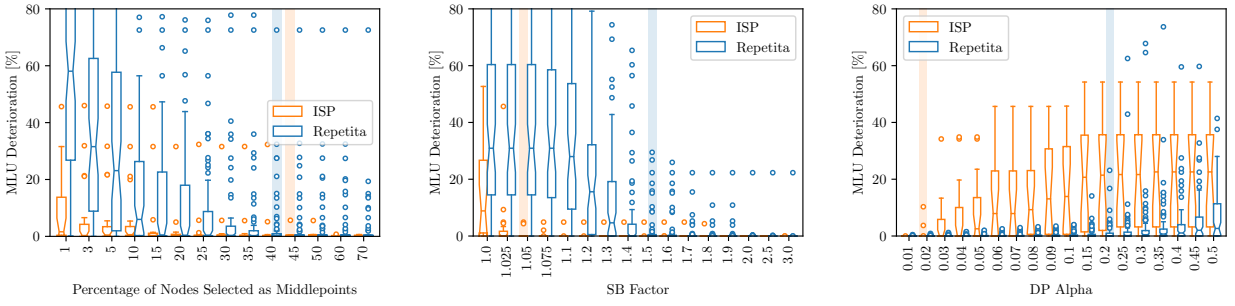
Another early middlepoint selection approach is the *Stretch-Bounding (SB)* concept proposed in [17, 18]. Its key idea is to rule out all nodes from being a potential middlepoint for an SR path if they are "too far away" from its source or destination regarding a given metric. This can be formalized as only considering middlepoint m for an SR path between src and dst if the following equation is satisfied:

$$\frac{DIST(src \rightarrow m) + DIST(m \rightarrow dst)}{DIST(src \rightarrow dst)} \leq \alpha_{SB} \tag{7}$$

with the $DIST()$ function denoting the shortest path distance between the respective two nodes and $\alpha_{SB} \in [1, \infty]$ being the so-called *SB factor*. This approach rules out all those SR paths that are more than α-times longer than the shortest path between the respective source and destination. It is shown in [18] that there is a trade-off between speedup and deterioration of solution quality depending on the chosen α-value. The authors define a factor of around $\alpha_{SB} = 1.4$ as the sweetspot of achieving close to optimal results while still considerably speeding up computations by a factor of 3 to 4. However, the SB implementation as described in Equation 7 inherits an issue that can negatively impact performance in certain scenarios (first pointed out in [8]). If the initial shortest path length is small (e.g., for paths with just one or two hops), small α-values can completely prohibit any kind of detour for the respective demand. The best example for this is a simple hop-count metric. If the shortest path of a demand has length 1 (one hop), this means that for all $\alpha_{SB} < 2$, there are no detours available for this demand as every detour would have at least length two. This can negatively impact the achievable MLU.

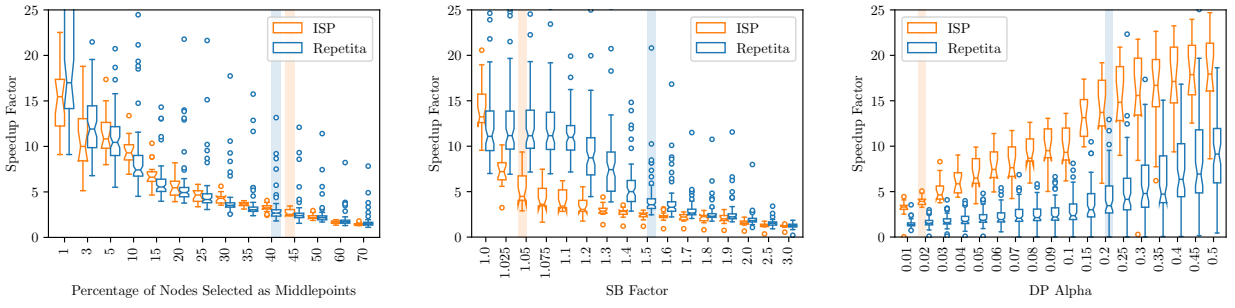
A rather similar concept to the SB approach of [17] was also proposed in [24], where nodes are assigned geographical tags on three different levels of granularity (`site`, `country`, and `continent`). SR paths between nodes that share a common tag value (e.g., *US* for the `country`-tag) are restricted to only use middlepoints with the same tag value (e.g., only nodes also located in the US). This also implements the idea of limiting the length of SR detours to a sensible maximum (e.g., by not routing traffic between Boston and New York over Europe).

Implementation Details: We implement the SB approach mentioned issues regarding demands with very low shortest path lengths. For this, we allow each demand with a shortest path length of just one hop to be rerouted over arbitrary paths with two hops (irrespective of the chosen α-value). This turns out to be sufficient to resolve most of the



(a) Centrality-based (b) Stretch Bounding (extended) (c) Demand Pinning

Figure 1: MLU deterioration for different preprocessing approaches. (A few very large outliers were cut off for better readability.)



(a) Centrality-based (b) Stretch Bounding (extended) (c) Demand Pinning

Figure 2: Achievable speedup for the 2SR optimization. (A few very large outliers were cut off for better readability.)

respective issues without significantly increasing the overall number of SR paths to consider during optimization. The same was also observed in [8].

Evaluation Results

In Figures 1b and 2b, it can be seen that, for the ISP dataset, near optimal results are achieved with a SB factor of just 1.05. For a factor of 1.1, there is basically no noticeable MLU deterioration anymore while still achieving a 4-5× speedup. For the Repetita dataset, the results differ noticeably. Here, such low SB factors result in a substantial MLU deterioration of around 40% on average and over 80% at max. Practically usable results can be achieved with a SB factor of 1.5 or higher and (virtual) optimal results require a factor of around 2.0. This translates to an around 4.5× and 2× speedup, respectively.

At first glance, it might seem like the SB approach performs substantially worse for the Repetita dataset. This observation, however, is (at least a bit) deceptive. While, for low SB factors, the MLU deterioration on the Repetita dataset is substantially higher, the speedup is also much better. The reason for this is that for the same SB factor, the overall number of prohibited SR paths is much higher for the Repetita dataset. For example, for a SB factor of 1.1, around 90% or more of all available SR paths are prohibited for many Repetita instances. Contrary, for the ISP data, only around 65-70% of paths are filtered out. As a result, the optimization for the Repetita instances is faster due to the lower number of options to evaluate, but this also results in a worse overall solution quality. If we instead compare results based on the percentage of excluded SR paths, they become much more similar. For example, for a SB factor of 1.4, the percentage of excluded SR paths is also in the range of 65-70% and the resulting speedup is comparable to the one of the ISP data with the respective "matching" SB factor of 1.1. We suspect that these variations are a product of topological differences between the instances in the Repetita dataset and the real ISP backbone network. However, investigating and identifying these differences is out of the scope of this work, but remains an interesting question for future work.

4.4. Demand Pinning

Another preprocessing technique briefly described in [20] and [21] is *DP*. It is based on the observation that in many networks (i.e. WANs and ISP backbones) traffic flows are not uniformly distributed in size. Instead, traffic consist of a few very large demands that make up a considerable amount of the total traffic volume and a rather large number of very small demands. DP fixes the forwarding paths of all these small demands to standard SPR and only runs a TE optimization for the larger ones. The idea is that the impact of the small demands on the overall solution quality is negligible compared to the larger traffic flows. Not optimizing them will have virtually no impact on the overall solution quality. To the best of our knowledge, there are no studies on the performance of the DP approach regarding its impact on solution quality and computation time. However, in [20], it is mentioned that around 68% of demands in the *Microsoft* WAN are of small size, making up a combined total of only 1.3% of the total traffic volume. If these results transfer to other networks, as well, DP seems like a promising candidate for an SR preprocessing since fixing the path of 68% of demands would translate to an equal reduction in the number of SR paths to consider during optimization.

Implementation Details: To implement DP, we first sort all traffic demands by size in ascending order. After this, we keep fixing the smallest demands to their SPR paths until the total sum of "fixed" traffic reaches a certain share of the total traffic volume given by the parameter $\alpha_{DP} \in [0, 1]$.

Evaluation Results

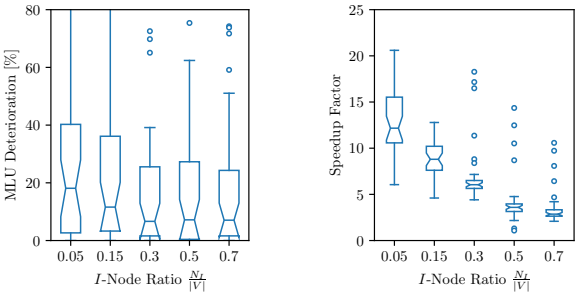
Results for the DP approach are depicted in Figures 1c and 2c. It can be seen that for the ISP data, an around 4.5× speedup can be achieved without substantially worsening the MLU. However, for larger α-values, the solution quality deteriorates quickly. Results for the Repetita data look rather different. Here, we are able to exclude up to 20% and more of the total traffic volume before a relevant MLU deterioration becomes observable. However, the speedup, while overall slightly better than for the ISP data, remains rather similar at around 5×. The reason for those differences lies in the distribution of the demand sizes in the traffic matrices of the two datasets. The real ISP traffic features a substantially higher number of really small demands (w.r.t. the total traffic) than the artificially generated matrices in the Repetita dataset. This is exemplarily depicted in Figure 5 for the largest instance of the ISP and Repetita dataset, respectively. As a result, the same α-value allows for the exclusion of substantially more demands for the ISP network. An α-value of 0.01, for example, excludes around 70-80% of all demands in the ISP matrices from optimization, while only excluding around 15-20% of demands from the Repetita matrices.

4.5. Smart Node Selection

A very recent addition to the SR preprocessing landscape is the *Smart Node Selection (SNS)* approach presented in [19]. It is the first to utilize AI/ML (i.e. deep reinforcement learning) for addressing the preprocessing (or *node selection*) problem, not only aiming for a reduction of computation times but also catering to other objectives like lowering control overhead and deployment costs. For this, SNS carries out two node selection procedures using a pre-trained model. In one step, it selects the set of nodes that are allowed to be used as intermediate segments (so-called *I-nodes*) and, in the other, it selects so-called *T-nodes* defining the set of nodes whose outgoing traffic is allowed to be detoured by SR. The latter can be seen as a different form of DP, as it forces all those demands that do not originate at one of the selected T-nodes to be routed via their standard SPR path.

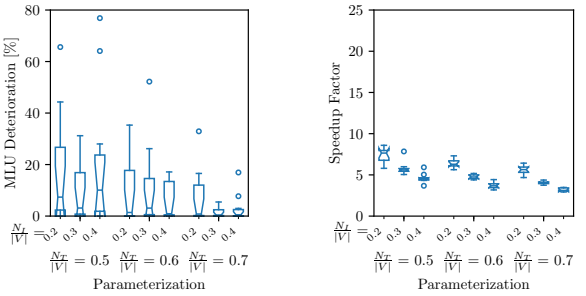
When using fitting sizes for the I- and T-node sets, this kind of preprocessing can facilitate a considerable reduction in computation time while only resulting in a small to moderate deterioration of solution quality [19]. The most notable benefit of this approach, however, is the exceptionally low time in which the node selection process can be carried out. Even for rather large networks with close to 150 nodes, the authors report preprocessing times in the single-digit millisecond range. In the context of LP-based optimization which often takes multiple seconds, minutes, or in the worst case even hours, these timings are basically negligible.

While the results reported in the paper generally look very promising, SNS also comes with a few drawbacks. First of all, the fast node selection is achieved at the price of having to carry out a rather time-consuming training process which, for larger networks, can take up to 10 hours or more. However, this is (arguably) acceptable since training can be done beforehand in an offline fashion. Furthermore, the results even indicate that the trained model is rather stable with regard to changes in traffic and even the network topology (i.e. failures) and, thus, retraining of the model does not seem to be required that frequently. Another (potential) weakpoint, however, is the fact the T-node selection approach used in SNS appears to be an inferior variant of the DP concept (cf. Section 4.4). Pinning *all* demands originating at certain nodes to their SPR path is far less flexible than the original DP concept of pinning *individual* demands based on their size (and, thereby, their presumed impact on TE performance). As a result, the respective "pinning" is considerably



(a) MLU deterioration. (b) Speedup.

Figure 3: Analysis of the performance of SNS for 36 instances from the Repetita dataset depending on the share of nodes chosen as I-nodes ($\frac{N_I}{|V|}$). (For all runs, 50% of nodes are chosen as T-nodes in accordance to the findings of [19].)



(a) MLU deterioration. (b) Speedup.

Figure 4: Analysis of the performance of SNS for 10 instances from the Repetita dataset depending on the share of nodes chosen as I-nodes ($\frac{N_I}{|V|}$) and T-nodes ($\frac{N_T}{|V|}$).

less fine-tunable and, thus, will (most likely) induce some form of "collateral damage" (i.e. large demands with non-negligible impact on TE performance being pinned or, vice versa, a large set of minute demands remaining unpinned in order to not prevent the detouring of only a few major demands originating at the same node). This can negatively impact both the achievable speedup and the TE performance. Finally, limiting *all* SR paths to use the same small set of allowed intermediate segments also introduces problems regarding latency constraints and router utilization, as we have already discussed in the context of centrality-based middlepoint selection in Section 4.2.

Implementation Details: To examine the SNS approach, we use the original source code made publicly available[6] by the authors to generate traffic matrices for training, train the model, and finally generate the respective I- and T-node sets. The (hyper-)parameters are left unchanged as described in [19].

Evaluation Results

Due to the high training times required by the SNS approach, carrying out a full "parameter-sweep" evaluation on all our evaluation instances is not feasible.[7] Thus, we had to limit our evaluation to only a subset of instances and parameter configurations.

Based on the recommendation in [19], our first evaluation fixes the share of nodes selected as T-nodes to 50% and only varies the number of allowed I-nodes. The results are depicted in Figure 3. It can be seen that SNS is able to facilitate a speedup of up to around 15×, but this comes at the price of a considerable deterioration in solution quality. Increasing the number of I-nodes reduces this deterioration but also the achievable speedup. Interestingly, the improvement in solution quality stagnates at an I-node ratio of around 30%, with a considerable MLU deterioration still observable (i.e. up to 60% or more). Even increasing it to as much as 70% brings virtually no improvement. For comparison, the centrality-based approach achieves substantially better solution quality while using the same percentage of nodes as intermediate segments (cf. Figure 1a).

We believe that this rather poor performance of SNS is due to the T-nodes being chosen too aggressively. Not being able to detour demands starting at 50% of the nodes is quite restrictive. Thus, we carry out a second examination with increased T-node ratios. The respective results are depicted in Figure 4. Here, it can be seen that, with higher T-node ratios, results get considerably better, allowing SNS to achieve an around 2.5× to 4× speedup with only a minor deterioration in solution quality. This puts SNS at the same level as most of the other approaches (i.e. SB and DP).

All in all, this evaluation basically confirms our initial considerations regarding potential shortcomings of the SNS approach (cf. Section 4.5). Prohibiting the detouring of demands on a per-node basis is far more inflexible and, thus, restrictive than the original DP approach. Using $\alpha_{DP} = 0.2$, the latter can often exclude around 50-70% of all demands on the Repetita instances with virtually no deterioration in solution quality (cf. Figure 1c). In contrast, the SNS approach already struggles considerably at just 50%. Nonetheless, we believe that applying AI/ML to the SR path preprocessing

[6] Available at `https://github.com/wlh320/SNS`.

[7] Even when optimistically assuming an average training time of two hours per instance, such an evaluation would take *hundreds* of days.

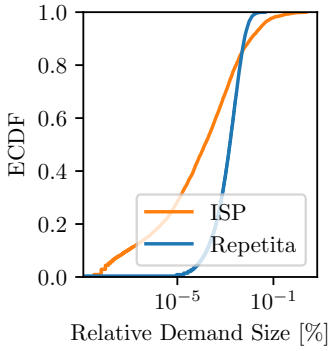


Figure 5: ECDF of the relative demand sizes (w.r.t to the total traffic volume) of the ISP 2021 and the Repetita `Cogentco` instances.

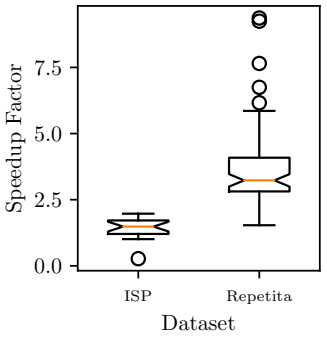


Figure 6: SR speedup achieved by the SR PDE preprocessing on the two evaluation datasets.

problem is a very promising approach as, even with the discussed shortcomings, SNS is already able to perform about on-par with other existing approaches. An adapted and extended version that fixes these "teething troubles" by carrying out "real" DP on the level of individual demands and also selects an individual set of allowed I-nodes for each demand might be able to considerably outperform the latter. Furthermore, SNS also shines by its fast node selection process that makes it especially interesting for applications within very tight time-constraints.

4.6. SR Path Domination & Equivalency

All the previous approaches carry the risk of excluding SR paths that are needed for an optimal solution. As a result, the solution quality can become arbitrarily worse when deploying these methods. To prevent this, one has to ensure to only exclude SR paths for which it can be proven that they are not needed for an optimal solution. A first step towards such an approach was presented in [25]. There, it is shown that a large portion of configurable SR paths actually contain loop-like structures and the authors suspect that many of these paths are not required to obtain optimal solutions. This assumption is further investigated and confirmed by Callebaut et al. [8]. They propose the concept of *dominated* and *equivalent* SR paths. An SR path p_1 is *dominated* by another path p_2 if three conditions are satisfied. First, both paths must have the same start- and endpoint. Second, assuming a uniform traffic flow on each path, for every link $l \in \mathcal{L}(p_2)$ used by p_2, the load put on l by p_2 must be lower or equal to the load put on l by p_1:

$$load(l, p_2) \leqslant load(l, p_1) \qquad \forall\, l \in \mathcal{L}(p_2) \tag{8}$$

Lastly, for at least one link in Equation 8 the strict inequality must hold. Analogously, two SR paths are *equivalent*, if their set of used links and the resulting link-loads are exactly equal. This is the case if the first two conditions for SR path domination hold but with exact equality for Equation 8.

Dominated SR paths are never needed for an optimal solution and for a set of equivalent paths, it is sufficient to consider just one of them, allowing to exclude all others. It is shown in [8] that, based on these two observations, a substantial number of SR paths can be ruled out prior to optimization, resulting in a significant reduction in computation time. Furthermore, just like SB and centrality-based preprocessing approaches, this just requires information on the network topology but not on traffic. Hence, it can be precomputed in advance which is quite useful since in [8], computation times of up to 30 minutes or more are reported for just the preprocessing of larger topologies.

Implementation Details: We re-implemented the Path Domination & Equivalency (PDE) preprocessing as described in [8].

Evaluation Results

Figure 6 depicts the speedup that is achievable with the SR PDE approach on our two datasets. Contrary to the previous approaches, there is no need to look at MLU deterioration since the main idea of the SR PDE concept is to retain provable optimality of the achievable MLUs. It can be seen that, for the Repetita data, computation times can be improved by around a factor of four on most instances, with a couple of outliers even reaching close to a 10× speedup. Those high outliers are a result of the special topology structures of certain instances. For example, the `Ulaknet`

topology consists of three star shaped networks whose centers are interconnected with each other. Basically all SR paths using one of the many stub-nodes as intermediate segment provide no TE benefit regarding the MLU and can be ignored. This results in over 98% of all SR paths being excluded from optimization. On more "realistically" shaped topologies (w.r.t. common network design principles), however, this number is much lower (mostly between 65-80%) and, hence, the achievable speedup is also more moderate.

While PDE works quite well for the Repetita data, this does not hold for the real-world ISP network. Here, the average achievable speedup is just around 1.5× and the maximum barely surpasses factor 2. The reason for this, again, lies in the number of SR paths that are ruled out for each respective dataset. For the ISP dataset, this number is substantially lower with just around 20% of the total number of SR paths compared to an average of around 80% for most Repetita instances. We do not have a definitive answer for what causes this behavior, but we suspect that it is a result of topological differences between the networks in the Repetita dataset and the real ISP network. For example, the ISP network has virtually no stub-nodes since a common design goal for modern networks is to achieve at least two-connectivity for all nodes. This facilitates reliability and robustness as it ensures that the network will not be partitioned by single-link failures. Contrary, the Repetita topologies feature a rather large number of stub-nodes. Since those are never needed as middlepoints to obtain an optimal solution, a larger number of stub nodes automatically results in a larger number of dominated SR paths. This becomes visible when removing all stub-nodes from the Repetita topologies, which reduces the number of excluded SR paths from around 80% on average to just 60%. For reasons of space, we cannot delve deeper into this topic here and leave it for future work.

4.7. Comparative Discussion of the Evaluation Results

All in all, we have seen that each preprocessing approach has its pros and cons, with some performing better on the Repetita data and some on the ISP data. Hence, there is no clear "winner" to be picked. However, we were able to identify a certain trend regarding preprocessing concepts that are based on limiting the available middlepoints to *the same* (small) set of nodes *for every demand*, namely SNS and the centrality-based approach. As already suspected in the initial discussions in Section 4.5, this inflexibility results in the necessity to allow for a rather large number of nodes as usable middlepoints in order to prevent the solution quality from deteriorating too much. This, in turn, causes the achievable speedup to be relatively low (i.e. around 2–2.5×). In contrast, approaches like DP and SB that filter paths *individually* for each demand are able to achieve higher speedup (i.e. around 4×) with less deterioration. Combined with the fact that limiting all SR detours to the same set of nodes is also likely to induce certain practical problems, i.e. regarding latency or hardware utilization (cf. Section 4.2), we come to the conclusion that filtering SR paths for each demand *individually* likely is the superior approach with respect to both performance as well as operational concerns.

Furthermore, what became more and more clear during our evaluation, is the fact that considerable differences in results are observable depending on the dataset used. This especially holds true for the SR PDE approach that works really well for Repetita networks but considerably worse for the real Tier-1 ISP backbone. We track this down to differences in topology and traffic characteristics between the two datasets. This reinforces our concerns regarding the direct transferability of results obtained on the Repetita data with artificial traffic to real networks, already expressed in Section 3. It also stresses the importance of also carrying out evaluations on real, recent network data. Of course, our ISP dataset is "just one datapoint" that does not allow for drawing definitive and universal conclusions but other recently reported information e.g., regarding the demand size distribution in the *Microsoft* network [20] is far closer to the ISP network characteristics than to the Repetita data. To further investigate this, it would be desirable to repeat our experiments on other recent data from real networks. However, to the best of our knowledge, there currently are no publicly available datasets that provide such information.

5. Combining Preprocessing Approaches to Improve Performance

As seen before, there is no definitive answer to what the generally best preprocessing approach is since performance varies between datasets. In this section we propose a concept for combining multiple preprocessing approaches to allow for a more consistent performance across all datasets and to further improve the achievable speedup.[8]

5.1. Concept & Implementation

Our approach is based on the observation that, until now, we always considered each preprocessing approach individually. However, they are not mutually exclusive. Therefore, it is possible to combine the different preprocessing

[8] A conceptually related but less extensive (and, thus, less effective) approach is also considered in [8], showing promising preliminary results.

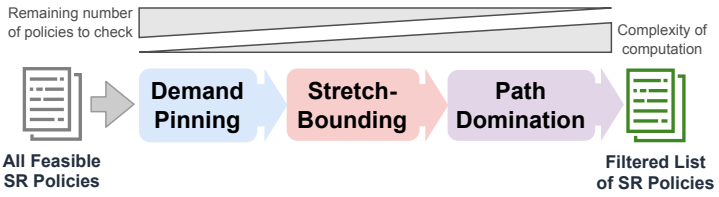


Figure 7: Computation pipeline of the proposed *combined preprocessing* approach. The most demanding preprocessing step (i.e. PDE) is carried out last with the lowest number of policies left to check.

approaches into a single one. This can yield multiple benefits. First and foremost, it holds the potential of further increasing the achievable speedup. Combining the individual sets of excluded SR paths allows to further increase the number of SR paths that can be ignored during optimization. However, it is unclear whether the combination of multiple exclusion sets that perform well individually will result in a well performing union set, as well. The combined set might also become too restrictive and, hence, might result in substantial MLU deterioration. Secondly, combining different preprocessing approaches might also lead to a more "stable" performance across our two datasets. In simple terms, by combining an approach that works better on the ISP data (i.e., SB) with one that is more suited for the Repetita data (i.e., PDE), we hope to leverage their individual benefits and get an algorithm that performs well on both datasets.

For our improved preprocessing algorithms, we combine the three approaches of SB, DP and SR PDE. The reason for not including the centrality-based approach is that it generally performs worse than the other approaches with regard to MLU deterioration and speedup. Furthermore, as already discussed in Section 4.2, it also features other weaknesses when it comes to practical use (e.g., regarding latency constraints). The computation pipeline of our new combined preprocessing approach (see Figure 7) starts with a DP operation that can be fine-tuned with the α_{DP} parameter. After this, a SB step is carried out using the α_{SB} parameter. The PDE filtering comes last as it is the computationally most demanding operation. Having already filtered out a large set of SR paths with the previous two operations which do not need to be checked by PDE anymore, facilitates lower computation times.

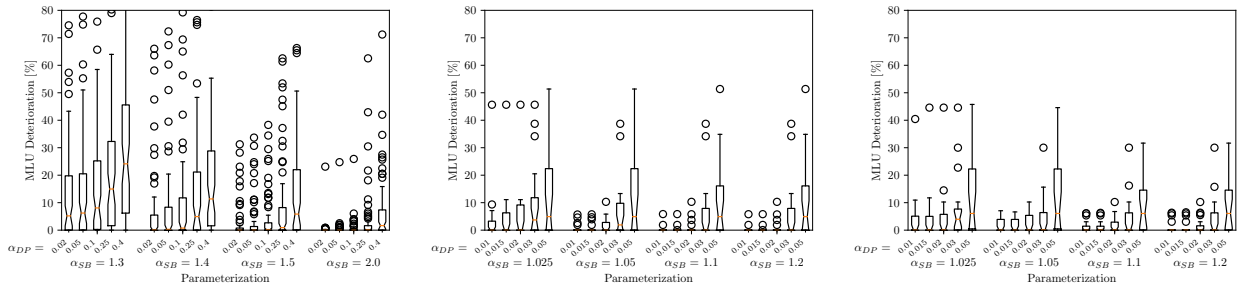
5.2. Evaluation

We examine the performance of our combined approach on our two evaluation datasets for the 2SR algorithm. Additionally, we also carry out a short exemplary evaluation for the MO-capable Shortcut 2SR (SC2SR) algorithm proposed in [3].[9] This is done to provide an insight into whether results are transferable to other SR TE algorithms, even if those utilize a rather different SR variation. The results regarding MLU deterioration and speedup for various parameter combinations are depicted in Figures 8 and 9, respectively. It can be seen that, for the Repetita dataset (Figures 8a and 9a), with the right parameter configuration, we are able to achieve a 2SR speedup of around factor 7 to 8 with virtually no significant MLU deterioration. This is more than twice as good as what is achievable with any of the existing approaches. Those all cap at a less than 4× median speedup before introducing a noticeable deterioration in solution quality. To better visualize this, Figure 10 provides a dedicated comparison of the different preprocessing approaches using the respective parameter setting that enables the highest speedup while resulting in close to no MLU deterioration for most instances (i.e. the parameter settings highlighted in Figures 1 and 2). It can be seen that our combined preprocessing considerably outperforms all existing approaches in terms of speedup while also inducing the lowest deterioration in solution quality.[10] Furthermore, when going back to Figures 8a and 9a, we see that if a moderate level of MLU deterioration is acceptable (i.e. up to 10%), the speedup achievable with our combined preprocessing approach increases even further to a factor of 10 or more.

The same findings also apply to the ISP dataset for both the 2SR (Figures 8b and 9b) and the MO-capable SC2SR algorithm (Figures 8c and 9c). This confirms that our new preprocessing approach not only performs (more or less) equally good across both datasets when the right parameters are chosen, but is also transferable to other SR optimization algorithms even if the underlying SR concept is inherently different. Due to the immense computational efforts related to the evaluation of SC2SR, we are not able to carry out a similar examination on the Repetita dataset. However, we

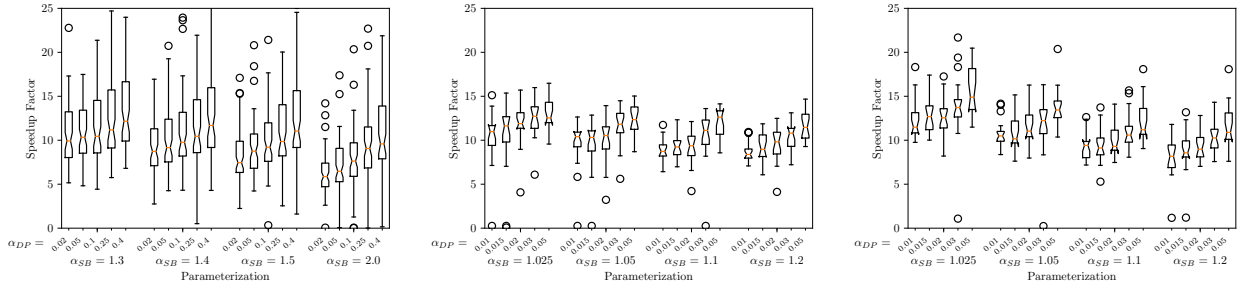
[9] For this, we need to adapt the DP procedure since, in the context of SR MO, policies are no longer associated with a dedicated demand, which makes excluding policies based on pinned demands less straightforward. Further details on this and how to adapt DP to still be usable with MO are provided in Appendix A.

[10] Apart from the PDE approach, of course, which is specifically designed to never induce any deterioration at all (cf. Section 4.6).



(a) 2SR on Repetita dataset (b) 2SR on ISP dataset (c) SC2SR on ISP dataset

Figure 8: MLU deterioration resulting from the combined preprocessing approach for different datasets and SR algorithms.



(a) 2SR on Repetita dataset (b) 2SR on ISP dataset (c) SC2SR on ISP dataset

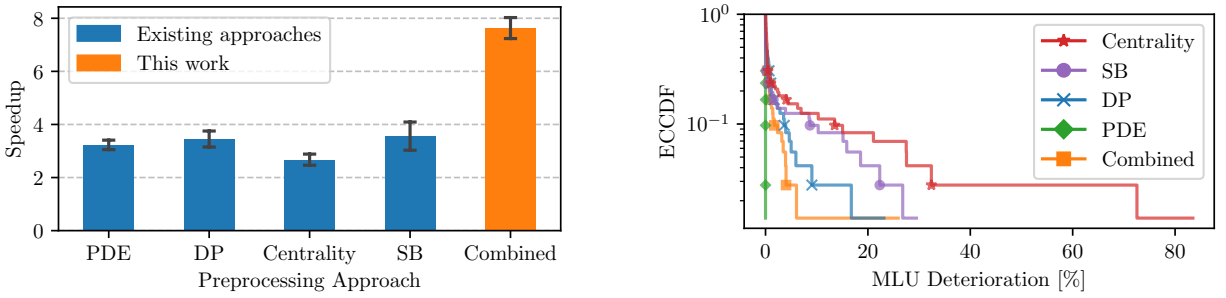
Figure 9: Speedup achieved by the combined preprocessing approach for different datasets and SR algorithms.

have seen that, for the ISP data, results are basically directly transferable from 2SR to SC2SR. Based on this, we expect the 2SR results depicted in Figures 8a and 9a to translate at least roughly to SC2SR, as well. This assumption is also further backed up by first preliminary experiments on smaller samples from the Repetita dataset.

To put into perspective what an around 8–10× speedup actually means, some information on the computation times of the ground-truth algorithms (i.e. without any preprocessing) are given in Table 3. It can be seen that, for example, the SC2SR algorithm takes around two hours to compute, on average, and over four hours at max. With our preprocessing, this can be reduced to just around 10 or 20 minutes, respectively. The benefits of our preprocessing become even more apparent when looking at the 2SR algorithm. Here, we are able to reduce the average computation time from 10min and more, to less than 2min for most of the ISP instances. This easily allows for the use of LP-based optimization for use cases where network configuration is continuously adapted on a timescale of just a few minutes (cf. e.g., [11]). Furthermore, we are now advancing into computation time regions in which it can be argued that the 2SR algorithm could even be used for tactical TE [31] that allows to quickly react to failures or traffic shifts [13]. Finally, we also believe that the performance achieved with our preprocessing approach is reasonably close to what can actually be achieved with preprocessing in general. The reason for this are its extremely high numbers of excluded SR paths. For the 2SR algorithm, our preprocessing already rules out 97-99% of all theoretically configurable 2SR paths. This probably does not leave much room for further improvement since a certain number of options to choose from is required before solution quality starts to degrade substantially.

6. Integrating Latency Bound Requirements into the Combined Preprocessing Approach

So far, we have mainly assessed the performance of preprocessing approaches based on the achievable speedup and the resulting MLU deterioration, which are the two most important performance metrics in this context. However, when it comes to utilizing such approaches for speeding up TE in actual production networks, there can be further aspects to consider. One that is of particular importance for operators of large carrier networks (i.e. ISPs) are so-called *latency*



(a) Achieved speedup (median value).

(b) Resulting MLU deterioration.

Figure 10: Comparison of the median 2SR speedup achievable with different preprocessing approaches and the resulting MLU deterioration for the Repetita dataset. (Parameter settings used for the respective approaches: DP: $\alpha_{DP} = 0.2$; Centrality: 40%; SB: $\alpha_{SB} = 1.5$; Combined: $\alpha_{DP} = 0.25$ and $\alpha_{SB} = 2.0$)

Table 3
Computation times (in seconds) of the respective ground truth algorithms without any preprocessing.

		Min	Max	Median	Average
2SR	Repetita	4	5456	25	248
	ISP	178	1407	691	611
SC2SR	ISP	2477	14568	5604	6571

bounds. Those define the maximum acceptable delay for a traffic demand, resulting from either rather general quality-of-experience related goals (e.g., perceived quality of telephone and video calls dropping significantly if latency is too high) or from more strict quality-of-service requirements specified in individual contracts with corporate customers (so-called Service-Level Agreements (SLAs)). Especially in the latter case, making sure that TE solutions fulfill these bounds is crucially important.

Unfortunately, excluding certain SR paths/policies from consideration during TE optimization, as it is the key idea of the preprocessing and middlepoint selection approaches considered in this paper, can have a negative impact on the satisfiability of such latency bound constraints. For example, by limiting *all* SR paths to use the same (small) number of allowed middlepoints (as done by the centrality-based and SNS approaches), some demands might either be forced over long detours that violate the respective latency bound or cannot be detoured at all, thereby negatively impacting the achievable solution quality. Furthermore, it is not necessarily guaranteed that all latency bounds are fulfilled in the initial network state (i.e. by SPR). Thus, it might not be sufficient to solely ensure that the respective TE approaches do not introduce any new violations, but TE can actually be required to "fix" such initial violations, as well. In this context, using preprocessing approaches like DP can induce problems[11], as it excludes certain demands from being optimized with TE, thereby completely removing the option to fix possible latency bound violations for the latter.

In the following, we address these concerns by not only demonstrating that our combined preprocessing approach can be adapted to prevent it from having a negative impact on the satisfiability of latency bound constraints, but that the whole objective of incorporating those into the optimization process can be done solely based on SR path preprocessing as well. In this context, we consider two scenarios of increasing complexity:

- **No-New-Violations (NNV):** The introduction of new latency bound violations has to be prevented but initially existing ones do *not* need to be fixed.
- **Violation-Fix (VF):** The introduction of new latency bound violations has to be prevented *and* all pre-existing latency bound violations have to be fixed as well (given that they are actually fixable[12] by TE).

[11] Especially when considering that "business customer" traffic flows for which there are specific SLA-related QoS constraints are often of rather small size compared to the more general residential customer traffic.

[12] In some scenarios (e.g., the failure of an important low-latency link), some latency bound violations can be impossible to fix by TE

Table 4
Number of latency bound violations in the 2SR solutions for the ISP instances computed with different variants of the combined preprocessing: 1) *default* (i.e. no latency considerations), 2) *no-new-violations* (NNV), and 3) *violation-fix* (VF).

	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	28	30	34	32	51	107	46	60	349	580	559	355	399	404	460	708	735	740
with NNV	2	2	1	2	2	74	2	6	12	12	1	1	2	1	3	2	108	0
with VF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1. Implementation

Extending our combined preprocessing for the NNV scenario is rather straight forward. We simply add an additional filtering step between the SB and the PDE filtering in our preprocessing pipeline (cf. Figure 7), that, for each demand, rules out all those SR paths that would result in a (new) violation of the respective latency bound. *Fixing* pre-existing latency bound violations, however, is a bit more complicated. For this, we first identify all those demands for which there is a pre-existing latency bound violation that is actually fixable via TE. The latter can, for example, be checked by running a shortest path computation based on link delays to get the minimal achievable delay for the respective demand. Each of these demands is then excluded from the PDE and SB filtering steps to prevent the removal of SR paths that could be used to fix the respective violation. Instead, we filter out all those SR paths (i.e. intermediate segments) that result in the respective latency bound being violated, also including the SPR path. As a result, only SR paths *not* required to fix latency bound violations are filtered out by the preprocessing and the only SR paths remaining for consideration in the TE optimization are those adhering to the specified latency bounds.

6.2. Evaluation

In the following, we show in an exemplary evaluation that these new features can be incorporated into our combined preprocessing approach without meaningfully deteriorating its previously seen performance. For this, we repeat the previous experiments for the 2SR algorithm using our extended preprocessing approach with the new NNV and VF features, respectively. For reasons of space and time, we do not conduct a full parameter sweep again, but only focus on the parameter setting that we identified as enabling good speedup with close to no deterioration in solution quality in the previous evaluations: $\alpha_{DP} = 0.015$ and $\alpha_{SB} = 1.1$ (cf. Figure 8b). Such an evaluation requires information on the link delays and latency bounds of the respective instances. Unfortunately, the required information on the link delays and latency bounds is notoriously hard to obtain and generally not publicly available as it is considered confidential by most operators. The Repetita dataset, for example, does *not* feature such information and, thus, we cannot include it in the following evaluation. Instead, we have to limit the latter to just the ISP dataset for which we were able to obtain *real* latency bound and delay information from our ISP partner.[13]

Number of bound violations: In a first step, we look at the number of latency bound violations in the computed 2SR solution with and without the new extensions to our combined preprocessing approach, which are depicted in Table 4. It can be seen that not considering latency bounds at all results in a substantial number of those being violated in the final solution (cf. `default` row). This illustrates the importance of integrating latency bound constraints into the optimization process in order to produce practically usable results. By using the new NNV preprocessing extension to prevent the 2SR optimization from introducing new violations, the number of violations can be considerably reduced. However, for virtually all instances, there are at least a few violations remaining which result from latency bounds that were already violated prior to optimization. While this number is generally quite low, in some cases (i.e. instances *F* and *R*), there is a considerable number of violations left. But even for those, our VF feature proves to be effective in also resolving such initial violations. With it, solutions can be computed that adhere to *all* specified latency bounds while still benefiting from the previously seen performance gains enabled by the combined preprocessing approach.

Impact on the achievable MLU: Having seen that the use of the combined preprocessing does not conflict with latency bound constraints but actually enables a neat way to easily enforce them using its VF extension, the question remains whether the latter has a negative impact on the achievable MLU. To analyze this, we compare the MLUs achievable by 2SR when used with the new latency bound extensions of the combined preprocessing to those obtained with its default version. The respective results are depicted in Figure 11). For reference, it also provides information

[13] For instance *Q*, this information was not available. Hence, it will be omitted in the following examinations.

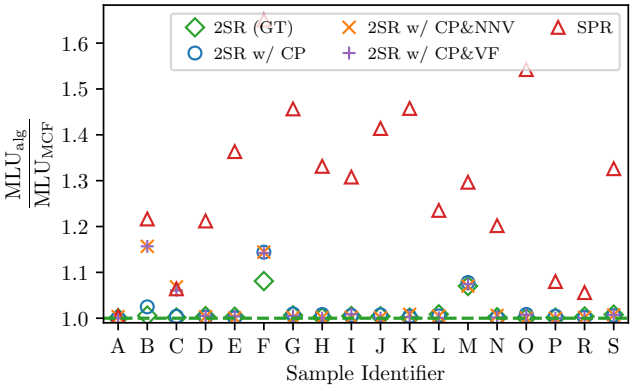


Figure 11: Comparison of the MLUs achievable by 2SR for the instances in the ISP dataset when also considering latency bound constraints in our combined preprocessing (CP) approach. (Used parameter settings: $\alpha_{DP} = 0.015$ and $\alpha_{SB} = 1.1$)

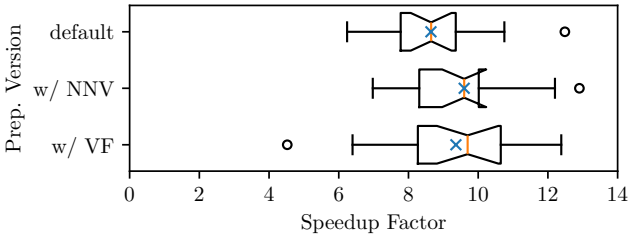


Figure 12: Distribution of the speedup factors achieved with the new latency bound related extensions of our combined preprocessing approach compared to its default version. (Used parameter settings: $\alpha_{DP} = 0.015$ and $\alpha_{SB} = 1.1$)

on the respective *ground-truth* (GT) 2SR (meaning without any preprocessing) and SPR MLUs. Furthermore, all MLU values are given relative to the theoretical optimal MLU computed with Multicommodity Flow (MCF).[14] In this context, a value of 1.0 denotes an optimal solution. It can be seen that, apart from two instances (i.e. ***B*** and ***C***), utilizing the new NNV and VF extensions does not result in any meaningful deterioration of the solution quality compared to the default version of our combined preprocessing (i.e. the blue circles), and still allows for (close to optimal) solutions for nearly all instances.

Achievable Speedup: Finally, we examine the impact of the new latency bound related extension on the achievable speedup (see Figure 12). In this context, the NNV and VF extensions do not only not result in a deterioration but actually even facilitate a small but noticeable improvement, making the respective 2SR optimization even faster. ´The explanation for this is rather straight forward. While the extensions (especially VF) may result in the DP and SB preprocessing stage being skipped for demands whose latency bound is initially violated, the number of such demands is rather small (cf. Table 4). Compared to the total number of demands which ranges between 4000 to 8000 depending on the respective instance, these numbers are basically negligible and, thus, only result in a marginal increase in problem complexity. However, the latency bound extensions simultaneously add a whole new filtering step which removes all those SR paths that do not fulfill the respective latency requirements. This number is considerably larger than the few "skipped" demands and, thus, the overall number of SR paths that are considered in the construction of the 2SR LP is further reduced. Intuitively, this also results in a further improved speedup.

Overall, these examinations show that it is possible to adapt our combined preprocessing approach to also consider important latency bound requirements without resulting in a meaningful deterioration of its previously demonstrated performance. Apart from very few exceptions, the obtained MLUs stay virtually the same while the achieved speedup even slightly improves.

[14] MCF [32, Ch. 4.4] provides a lower bound for the optimal MLU achievable with *any* kind of arbitrarily mighty TE.

7. Path Preprocessing for Heuristic Optimization in the Context of Tactical TE

So far, our study and related work only consider SR preprocessing in the context of improving performance of LP-based TE approaches, since those notoriously suffer from scalability issues, negatively impacting their usability for large networks. However, the observation that large portions of SR paths can be ignored during optimization without considerably worsening the achievable solution quality could potentially be used to speed up other algorithmic approaches as well. One area of TE in which computation time is particularly important is *tactical TE* [31] which aims to quickly address and resolve problematic networks states (i.e. resulting from failures or unexpected traffic changes). In this context, optimality of solutions is generally less important as long as a reasonably good one is provided fast. While the exact definition of "fast" depends on the respective use case, most tactical TE applications generally aim to provide solutions within a few seconds or less. Even when utilizing preprocessing approaches to boost performance, LP-based optimization often remains too slow to achieve this (cf. e.g., the SC2SR algorithm examined in Section 5.2). Therefore, many approaches catered towards tactical TE (e.g., [33, 14, 34]) instead rely on *heuristic* optimization. While such approaches are intrinsically fast, further improving them is of great practical interest as it allows to combat problematic network states even quicker, following the premise of *"The faster the better"*. In the following, we thus examine whether the concept of path preprocessing can also be used to further speed up heuristic optimization in the context of tactical TE.

7.1. Evaluation Setup

For this, we apply our combined preprocessing approach to *Midpoint Optimization Local Search (MOLS)* [13, 34], a state-of-the-art tactical TE heuristic for SR MO. This is done by altering the MOLS implementation to not consider policies that are ruled out by the preprocessing during the neighborhood exploration that determines the next local search move (i.e. insertion of an SR policy). Apart from that, MOLS is used and parameterized as described in [34].

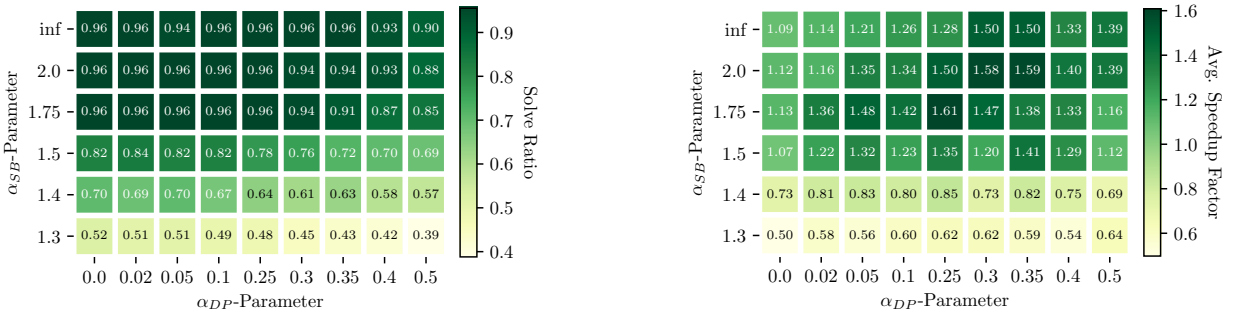
As already explained above, finding a truly optimal solution is of lesser interest in the context of tactical TE as long as the solution is sufficient for fulfilling the objective of resolving the respective critical network event (i.e. removing overutilization). Therefore, the following evaluation also focuses on the latter by no longer quantifying the deterioration in solution quality potentially resulting from the preprocessing based on the optimal achievable MLU but simply based on the binary decision of whether MOLS is still able to resolve the respective overutilization scenario or not. For this examination, we use all those instances from the Repetita dataset that exhibit SPR overutilization (i.e. an SPR MLU > 1.0), which corresponds to a total set of 67 instances.[15] For each of these instances, we measure the time taken by MOLS to find a solution that resolves the initial overutilization (i.e. has an MLU < 1.0), when used with different parameterizations of the combined preprocessing approach. Thereby, a maximum timelimit of two minutes is applied to account for instances that are not solvable anymore due to a potentially too restrictive preprocessing. Furthermore, since MOLS features non-deterministic components, all experiments are repeated five times and the following results show the average across these five runs.

7.2. Evaluation Results

The results of our evaluations regarding the achievable solution quality and speedup are shown in Figures 13a and 13b, respectively. It can be seen that, similar to the use case of strategic LP-based optimization (cf. Section 5.2), choosing too aggressive filtering parameters results in a considerable deterioration of the achievable solution quality, with the number of resolved overutilization scenarios dropping as low as just 39%. However, with a less restrictive preprocessing, a sufficiently good solution quality can still be achieved. For reference, it should be noted that even default MOLS is only able to prevent overutilization for 96% of the considered instances. Thus, a value of 0.96 can be interpreted as the preprocessing not inflicting any deterioration in terms of solution quality.

When it comes to the achievable speedup (see Figure 13b), the first thing to notice is the fact that a very aggressive preprocessing actually results in the computation time to *increase*. While this might seem odd at first glance, the explanation is rather simple. In our evaluation, MOLS stops as soon as it finds a solution that prevents overutilization. If, due to a too restrictive preprocessing, such a solution cannot be found, MOLS runs for the whole two minutes until it is stopped by the timelimit. As a result, preprocessing parameter configurations that are too restrictive and result in a large number of unsolved instances also cause higher computation times. However, being too conservative in choosing the preprocessing parameters also causes suboptimal results, as it will not be utilized to its full potential in that case. Taking into account both of these aspects, the optimal parameter setting according to our evaluation is $\alpha_{DP} = 0.25$ and

[15] Since none of the instances in our ISP dataset exhibit any overutilization, this dataset is not applicable for a use in the following evaluation.



(a) Share of solved instances. (A value of 1.0 denotes 100%)

(b) Average achieved speedup factor.

Figure 13: Analysis of the impact of different α_{DP} and α_{SB} values on the performance of the combined preprocessing approach when applied to the MOLS algorithm.

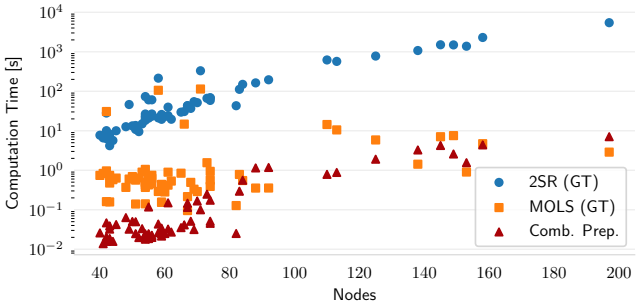


Figure 14: Comparison of the computation time of the default version (i.e. without any path preprocessing) of different SR optimization algorithms to the time required to carry out the combined preprocessing (with $\alpha_{DP} = 0.25$ and $\alpha_{SB} = 1.75$).

$\alpha_{SB} = 1.75$ which achieves an around 37.5% speedup on average while not resulting in any deterioration of MOLS' ability to deal with overutilization.

This speedup is considerably lower than those achieved for LP-based optimization (cf. Section 5.2), but this was to be expected. While LP-based algorithms generally consider basically all feasible SR paths/policies during optimization, heuristic approaches like MOLS put in great effort in order to only examine very small but promising areas of the whole solution space. As a result, a non-negligible share of the paths/policies excluded by the preprocessing would probably never be considered by the heuristic algorithm anyway. Some of the measures taken by MOLS even somewhat overlap with the concepts utilized in our combined preprocessing. For example, MOLS selects promising moves based on the volume of traffic they can detour away from the current bottleneck link (cf. [34]). To a certain extent, this resembles a form of demand pinning as it substantially reduces the chances of considering policies that only detour very small demands. Furthermore, due to the already very low computation times of tactical TE approaches, there is generally also less room for improvement, especially when considering that computing the respective preprocessing also takes up a fair share of computation time. As a result, the "return-of-investment" for spending extra time on preprocessing diminishes compared to LP-based optimization. This becomes particularly apparent when comparing the computation times of the respective base algorithm(s) to the time required to compute the combined preprocessing (see Figure 14). It can be seen that (especially for larger instances) the latter is quite close to the total computation time of default MOLS, whereas the LP-based 2SR algorithm takes orders of magnitude more time to compute, making a preprocessing far more effective and attractive here.

Considering all this, being able to still shave off around 37.5% of computation time from already rather performance optimized tactical TE algorithms like MOLS solely by preemptively pruning the set of considered policies is still a notable achievement. But further looking into possibilities to reduce the computation time of preprocessing approaches (e.g., by utilizing AI/ML) remains an interesting and promising direction for future work.

8. Discussion & Limitations

In our studies, we only consider SR using at most two node segments per path. While it has been shown that this is sufficient to obtain virtually optimal solutions in many practical use cases [2, 4], there also are scenarios in which higher order segment paths or the use of adjacency segments can be necessary. We argue that the performance gains achievable with preprocessing approaches for such algorithms are probably even higher than those observed by us regarding 2SR. The reasoning for this is as follows. While adjacency segments or a general higher number of segments can be required to facilitate the implementation of certain forwarding paths that cannot be built with 2SR, this number is relatively small in most scenarios. Most of the newly considered paths can already be implemented with 2SR or do not have any practical value (i.e. looping paths that visit a segment multiple times). Thus, increasing the number of segments generally also results in an increase in the ratio of "useless to useful" paths, which, in turn, improves the effectiveness of preprocessing approaches. This has also been reported by Callebaut et al. [8]. They show that the number of dominated SR paths grows from only 50% when using 2SR to around 90% and 97% for 3SR and 4SR, respectively. Thus, without having explicitly considered higher order segment paths or adjacency segments in our study, the performance improvements shown here should resemble a lower bound for what can be achieved for those, as well.

Furthermore, as briefly mentioned in the beginning of Section 3, there are other approaches to improve the computation time of LP-based TE or LP-solving in general. This includes *decomposition* methods like column generation [12], (Lagrangian) *relaxation* techniques [35], *distributed computing* [36], or even the application of ML-based approaches to speed up branch-and-cut procedures [37]. While examining this is out of the scope of this paper, looking further into the possibility of utilizing such concepts or even *combining* them with the path preprocessing approaches considered here remains an interesting direction for future work. Especially when considering that the performance gains achievable by "just" further refining and enhancing the path preprocessing concept are probably rather limited due to current approaches already ruling out up to 99% of SR paths (cf. Section 5.2). In this context, there is probably more to gain by examining the complementary usage of completely different speedup approaches and, thus, we plan to look into this in the future.

9. Conclusion

While LP-based optimization is a prominent way for solving SR TE problems, its poor scalability imposes a considerable limitation for its applicability in time-constrained use cases. To address this, a variety of *preprocessing* strategies have been proposed that aim to reduce problem complexity by preemptively limiting the number of SR paths considered during optimization. Unfortunately, the corresponding research landscape is not really well formed yet, resulting in basically all publications overlooking at least some or even all of the previously existing related work (cf. Table 1). This, together with other factors regarding the respective evaluation setups (e.g., the sole use of artificially generated network traffic and generally very small sample sizes) render a meaningful assessment and comparison of existing preprocessing strategies in terms of their effectiveness virtually impossible (cf. Section 3).

To address this, we conduct the first extensive literature review of existing preprocessing approaches, complemented by a large-scale comparative performance study featuring a plethora of different problem instances including recent real-world data from a globally operating Tier-1 ISP. This, hopefully, can function as a valuable point of reference for future works in this area, not only clearly outlining existing work but also providing deeper insights into their respective strength and weaknesses and the resulting implications on their performance. In fact, we already utilize the insights gained from this study to facilitate the second major contribution of this paper: The proposition of a combination of multiple preprocessing approaches to further improve performance. With this novel approach, the number of 2SR paths to consider for optimization can often be preemptively reduced by as much as 97-99%, while still obtaining close to optimal solutions. This lowers the computation times of different LP-based TE algorithms by a factor of 10 or more without a significant deterioration in solution quality, which is more than twice as good as what is achievable with any of the previously existing methods. Furthermore, our approach achieves this while not negatively interfering with the satisfiability of latency bound constraints, a crucial practical requirement in many networks, so far not addressed in previous works. In combination, this represents a major improvement over the current state-of-the-art and further facilitates the reliable use of LP-based TE in large segment-routed networks. Lastly, we also demonstrated that the concept of SR path preprocessing can also be applied in the context of tactical TE. While the achievable speedup is – as to be expected – considerably lower than for strategic LP-based optimization, the proposed combined

preprocessing approach still facilitates an around 37% performance improvement for MOLS, a state-of-the-art tactical TE heuristic.

We believe that the latter number can be even further improved by overcoming the limitations imposed by the – at least in the context of tactical TE – rather high computation times of our combined preprocessing for larger networks. For this, AI/ML-based approaches seem to be very promising as they allow for a lightning-fast node selection process [19]. However, our examinations have shown that existing approaches are still limited in their functionality and performance and also exhibit potential problems regarding the practical use in actual production networks. Thus, we plan to further investigate the topic of applying AI/ML-based methods to the SR path preprocessing problem by revising and extending existing approaches based on the insights gained in this paper.

CRediT authorship contribution statement

Alexander Brundiers*: Conceptualization, Data Curation, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Timmy Schüller:** Resources, Writing – review & editing. **Nils Aschenbruck:** Supervision, Project administration, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

A. Adapting Demand Pinning for the Use with Segment Routing Midpoint Optimization

The general idea of DP is to reduce the complexity of TE optimization by excluding a selected set of demands (i.e. negligibly small ones) from consideration during optimization and simply routing them via the standard shortest path. In the context of conventional end-to-end (E2E) SR where policies are directly associated with a certain demand (i.e. the demand between their head- and tailend nodes), translating this concept into a policy/path prefiltering approach is rather straight forward. For each pinned demand, all of its associated SR paths are excluded from consideration during the TE optimization (cf. Section 4.4). In the context of MO, however, this does not work anymore since SR policies are no longer directly associated to a certain demand (cf. Section 2). Simply excluding all policies that can detour a pinned demand does not work either since, due to MO policies being able to detour multiple demands, the respective policy might be needed to detour other unpinned demands as well, with its exclusion thereby unwantedly limiting the traffic steering options for the latter.

In order to still leverage the DP approach in the best possible way, we, therefore, adapt it for the use with SR MO. For this, we only exclude those policies from consideration for which we can be sure that they can *only* be used to detour pinned demands and are *never* needed to detour unpinned ones. To compute this set, we start off with an empty set of allowed policies and then iterate over all unpinned demands. For each of those, we generate the set of policies eligible to detour the respective demand according to the deployed MO concept and add them to our set of allowed policies. At the end of this iteration, the latter set contains all policies that we do *not* want to exclude. Hence, all policies not contained in it can be filtered out. For a better understanding, a pseudocode description of this procedure is given in Algorithm 1.

Finally, it should be noted that this adaption results in "pinned" demands not actually being guaranteed to be pinned to their SPR path since they might be detoured as "collateral damage" by MO policies configured for rerouting unpinned demands. However, this is no issue at all since the decision to pin a demand is not based on any kind of operational requirements or constraints, but simply done as a means to reduce the complexity of the computation. While the effectiveness of the latter might be slightly impacted due to fewer policies being excluded, it still performs really well in our combined preprocessing approach as shown by the results from Section 5.2.

Acknowledgment

We thank the anonymous COMCOM reviewers for their insightful comments which contributed to further improving the quality of this paper.

Algorithm 1 Adapted DEMANDPINNING(α_{dp}, TM) procedure for the use with MO. (TM := Traffic Matrix)

```
// At the start, all paths/policies are disabled
allowedPolicies ← ∅
sum_tmp ← 0
// Iterate over demands (sorted descending by size)
for demand d ∈ GETSORTEDDEMANDS(TM) do
    if sum_tmp / TOTALTRAFFIC(TM) ≥ (1 − α_dp) then
        return allowedPolicies
    else
        sum_tmp += SIZE(d)
        P ← GETELIGIBLEPOLICIESFORDEMAND(d)
        for policy p ∈ P do
            allowedPolicies.ADD(p)
return allowedPolicies
```

References

[1] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. Preprocess your Paths – Speeding up Linear Programming-based Optimization for Segment Routing Traffic Engineering. In *Proc. of the IFIP Networking Conference (NETWORKING)*, pages 303–312, 2024.

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

[3] A. Brundiers, T. Schüller, and N. Aschenbruck. Midpoint Optimization for Segment Routing. In *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1579–1588, 2022.

[4] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter. Traffic Engineering using Segment Routing and Considering Requirements of a Carrier IP Network. *IEEE/ACM Transactions on Networking*, 26(4):1851–1864, 2018.

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

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

[7] 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*, 23(1):182–221, 2021.

[8] Hugo Callebaut, Jérôme De Boeck, and Bernard Fortz. Preprocessing for segment routing optimization. *Networks*, pages 1–20, 2023.

[9] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. An Extended Look at Midpoint Optimization for Segment Routing. *IEEE Open Journal of the Communications Society*, 5:1447–1468, 2024.

[10] 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)*, pages 15–28, 2015.

[11] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving High Utilization with Software-Driven WAN. In *Proc. of the ACM Conf. on Special Interest Group on Data Communication (SIGCOMM)*, pages 15–26, 2013.

[12] Mathieu Jadin, Francois Aubry, Pierre Schaus, and Olivier Bonaventure. CG4SR: Near Optimal Traffic Engineering for Segment Routing with Column Generation. In *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1333–1341, 2019.

[13] A. Brundiers, T. Schüller, and N. Aschenbruck. Tactical Traffic Engineering with Segment Routing Midpoint Optimization. In *Proc. of the IFIP Netw. Conf. (NETWORKING)*, 2023.

[14] 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.

[15] George Trimponias, Yan Xiao, Hong Xu, Xiaorui Wu, and Yanhui Geng. Centrality-based Middlepoint Selection for Traffic Engineering with Segment Routing. *ArXiv e-prints*, 2019.

[16] George Trimponias, Yan Xiao, Xiaorui Wu, Hong Xu, and Yanhui Geng. Node-Constrained Traffic Engineering: Theory and Applications. *IEEE/ACM Transactions on Networking*, 27(4):1344–1358, 2019.

[17] Tossaphol Settawatcharawanit, Vorapong Suppakitpaisarn, Shigeki Yamada, and Yusheng Ji. Segment Routed Traffic Engineering with Bounded Stretch in Software-Defined Networks. In *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, pages 477–480, 2018.

[18] Tossaphol Settawatcharawanit, Yi-Han Chiang, Vorapong Suppakitpaisarn, and Yusheng Ji. A Computation-Efficient Approach for Segment Routing Traffic Engineering. *IEEE Access*, pages 160408–160417, 2019.

[19] Linghao Wang, Lu Lu, Miao Wang, Zhiqiang Li, Hongwei Yang, Shuyong Zhu, and Yujun Zhang. SNS: Smart Node Selection for Scalable Traffic Engineering in Segment Routing Networks. *IEEE Transactions on Network and Service Management*, 2024.

[20] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized Cloud Wide-Area Network Traffic Engineering with BlastShield. https://www.microsoft.com/en-us/research/uploads/prod/2021/11/blastshield-61a6c89e0ca05.pdf, 2021. Microsoft.

[21] Pooria Namyar, Behnaz Arzani, Ryan Beckett, Santiago Segarra, Himanshu Raj, and Srikanth Kandula. Minding the Gap between Fast Heuristics and Their Optimal Counterparts. In *Proc. of the ACM Workshop on Hot Topics in Networks (HotNets)*, pages 138–144, 2022.

[22] Jonatan Krolikowski, Sébastien Martin, Paolo Medagliani, Jérémie Leguay, Shuang Chen, Xiaodong Chang, and Xuesong Geng. Joint Routing and Scheduling for Large-scale Deterministic IP Networks. *Computer Communications*, 165:33–42, 2021.

[23] Martin G. Everett and Stephen P. Borgatti. The Centrality of Groups and Classes. *Journal of Mathematical Sociology*, pages 181–201, 1999.

[24] Timmy Schüller, Nils Aschenbruck, Markus Chimani, and Martin Horneffer. Failure Resiliency With Only a Few Tunnels – Enabling Segment Routing for Traffic Engineering. *IEEE/ACM Transactions on Networking*, 29(1):262–274, 2021.

[25] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. On the Benefits of Loops for Segment Routing Traffic Engineering. In *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, pages 32–40, 2021.

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

[27] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[28] Matthew Roughan. Simplifying the Synthesis of Internet Traffic Matrices. *SIGCOMM Comput. Commun. Rev.*, pages 93–96, 2005.

[29] Timmy Schüller, Nils Aschenbruck, Markus Chimani, and Martin Horneffer. On the Practical Irrelevance of Metrics on Segment Routing Traffic Engineering optimization. In *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, pages 640–647, 2018.

[30] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. Combining Midpoint Optimization and Conventional End-to-End Segment Routing for Traffic Engineering. In *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, pages 1–9, 2023.

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

[32] Deep Medhi and Karthik Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2017.

[33] Bernard Fortz and Mikkel Thorup. Optimizing OSPF/IS-IS Weights in a Changing World. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[34] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. Fast Reoptimization With Only a Few Changes: Enhancing Tactical Traffic Engineering With Segment Routing Midpoint Optimization. *IEEE Journal on Selected Areas in Communications*, 43(2):495–509, 2025.

[35] Hugh Everett III. Generalized Lagrange Multiplier Method for Solving Problems of Pptimum Allocation of Resources. *Operations Research*, 11(3):399–417, 1963.

[36] Mathias Bürger, Giuseppe Notarstefano, Francesco Bullo, and Frank Allgöwer. A Distributed Simplex Algorithm for Degenerate Linear Programs and Multi-Agent Assignments. *Automatica*, 48(9):2298–2304, 2012.

[37] Maxime Gasse, Didier Chetelat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2019.