Failure Resiliency With Only a Few Tunnels – Enabling Segment Routing for Traffic Engineering

Timmy Schüller, Nils Aschenbruck, *Member, IEEE*, Markus Chimani, and Martin Horneffer

***Abstract*— Traffic engineering is an important concept that allows Internet Service Providers (ISPs) to utilize their existing routing hardware more efficiently. One technology that can be used is Segment Routing (SR). In this paper, we address the use of SR to increase the resilience against failure scenarios. In addition, we develop solutions that are manageable and, thus, deployable in a tier 1 ISP network. We propose a post-convergence aware SR based optimization model. With it, we can proactively find a single SR configuration that is beneficial in all predefined failure scenarios, including single link failures, shared risk link group failures, and node failures. In addition to this use-case, we also extend the optimization model to include other important practical requirements such as keeping the number of SR tunnels to a minimum, avoiding arbitrary traffic splitting, or meeting latency bounds. We evaluate our approaches with recently measured data from a tier 1 ISP and show that we can improve over state of the art routing approaches.**

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

I. Introduction

As WE are slowly but surely stepping into the age of 5G, higher mobile bandwidth and lower latency will likely motivate end users to stream and consume more content. Even without 5G, the increasing demand [5] is a non-trivial challenge for ISPs. The obvious solution is to physically expand a network, but this is very expensive. To use existing hardware efficiently, providers additionally deploy some form of traffic engineering [14]. The general goal is to distribute traffic demands over the networks resources so that certain requirements are met. Avoiding congestion is the most typical example for such a requirement.

A common way to perform traffic engineering in networks, where shortest-path-based Interior Gateway Protocols (IGPs) such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS), are used, is to tweak link metrics [3], [11]. This, however, is a very delicate process, which is shown to be NP-hard [10]. This is intuitive, as there typically are many more demands to find arbitrary paths for, than metrics to configure. Most often, it is not possible to achieve metric configurations that result in an optimal routing and resource utilization. Another way to perform traffic engineering is by signaling explicit tunnels with Resource Reservation Protocol Traffic Engineering (RSVP-TE) [2]. The downside here is that these tunnels account for a significant maintenance overhead. Thus, many providers currently are looking into using SR [7] as a traffic engineering strategy [6]. SR is a source routing paradigm, which defines selected checkpoints along a packet's intended path and consults the IGP in between. This results in a packet following multiple, efficiently concatenated shortest paths. By using either Multiprotocol Label Switching (MPLS) or an IPv6 extension, as well as an IGP extension, it introduces no additional protocol and no additional state information to the network. Instead, all decisions take place at the source and are attached to the packet itself. SR is sometimes also referred to as the de-facto architecture for Software-Defined Networking (SDN) [6], as it can easily be used to tailor a network using a central controller.

There already are propositions for traffic engineering optimization models using SR. But when trying to translate them from theory into a real deployment, many practical constraints arise [24]. One important and non-trivial requirement is to make the network resilient against failures. While this is a fact everyone can agree on, it is not directly obvious what this means in detail. There are multiple different types of failures that may or may not be important to an operator. This paper considers three different failure types: link failures, node failures, and Shared Risk Link Group (SRLG) failures. We furthermore assume at most one failure to occur at any one point in time. This is realistic, as about 70% of all unplanned failures can be traced back to single link failures [20]. The remaining cases are multi-link failures, however, most of them can be attributed to a common, single SRLG. In other words, the SRLG failures we use in this work can be interpreted as practically relevant multi-link faults, while node failures essentially are an extreme case of SRLG failures.

The timing with which to look at a failure produces completely different use-cases. In an ISP network, link failures happen on a daily basis, but most of them are resolved quickly, as presented in [15]. For these cases, there are technologies that aim to provide a short-term compromise as fast as possible. An example for this is the classic Fast Reroute (FRR) or the more widely applicable Topology Independent Loop

Manuscript received October 24, 2019; revised September 8, 2020; accepted October 7, 2020; approved by IEEE/ACM Transactions on Networking Editor S. Uhlig. Date of publication October 20, 2020; date of current version February 17, 2021. This is an extended version of an article previously published at the 44th IEEE Conference on Local Computer Networks (LCN) in October 2019. *(Corresponding author: Timmy Schüller.)*

Timmy Schüller is with the Institute of Computer Science, University of Osnabrück, 49090 Osnabrück, Germany, and also with Deutsche Telekom Technik GmbH, 48147 Münster, Germany (e-mail: schueller@uos.de).

Nils Aschenbruck and Markus Chimani are with the Institute of Computer Science, University of Osnabrück, 49090 Osnabrück, Germany (e-mail: aschenbruck@uos.de; chimani@uos.de).

Martin Horneffer is with Deutsche Telekom Technik GmbH, 48147 Münster, Germany (e-mail: martin.horneffer@telekom.de).

Digital Object Identifier 10.1109/TNET.2020.3030543

Free Alternate (TI-LFA) [7, Chapter 9]. But a small portion of failures do take a substantial amount of time to resolve. A prominent example for this are failures with undersea cables. This work can be categorized as a solution to these longer outages. Additionally, a traffic engineering technology may be classified by the action it takes when an outage occurs. A central controller, for example, could recompute upon failure and then reconfigure routers. Alternatively, backup paths for all possible failures could be precomputed. In this case routers only have to switch and nothing has to be computed. The model we propose here, however, achieves a resilient configuration without any routing changes and solely relies on the synergy between SR and the IGP.

The contributions of this work are three-fold.

1) We propose the proactive SR-based traffic engineering optimization model that is resilient against severe failures that are unlikely to be fixed in a few hours. We call it Post Convergence Aware 2SR (PCA2SR). The mathematical formulation is described and options on how to choose the configurable parameters are discussed.
2) We combine our approach for failure resiliency PCA2SR with an extension that limits the number of tunnels deployed: The Tunnel Limit Extension (TLE) (introduced in [24]). By doing so, we make sure our approaches can be deployed in real networks. We also include the compliance with specific latency constraints.
3) We evaluate both approaches on a recent set of traces measured in a tier 1 ISP backbone network. This includes topology information, traffic data, and real SRLGs. Our approaches show that they are successful and advance the state of the art towards failure resilient and manageable traffic engineering using SR.

The paper is structured as follows. In Section II, we cover the basics of SR. Related work is presented in Section III. With this knowledge, we are ready to present PCA2SR in Section IV and show the linear programming formulation in Section V. Before showing how this formulation performs in trace based simulations in Section VII, we shortly describe our evaluation setup in Section VI. Section VIII introduces a set of additional practical requirements along with an extended optimization model and evaluation. Finally we close with a summary and future work in Section IX.

II. Segment Routing

SR is an emerging source routing architecture. Its development and standardization is largely pushed by Cisco [6] in cooperation with multiple ISPs and the Source Packet Routing in Networking (SPRING) working group of the IETF. RFC 8402 [9] defines the SR architecture along with different segment types that can be thought of as different types of checkpoints that a packet has to visit. For this paper, only node segments, which uniquely identify a node within a network, will be considered. To steer a packet through a series of segments, the segment IDs are added onto the packet itself. This is done either by using Multiprotocol Label Switching (MPLS) labels or directly via an IPv6 Type-Length-Value (TLV) field. The IGP is consulted to reach a segment, overall creating a concatenation of multiple shortest paths. Traffic engineering is only one of the possible use cases of SR.

The number of segments that can be stacked onto one packet is a practical constraint. In [4], [24], it was shown that, on the scale of tier 1 ISPs, a near optimal routing is achievable with only 1 intermediate segment per path. The approach of limiting the segment stack to one intermediate and one destination segment is called 2SR and will be used in this work.

Multiple additions to a basic 2SR Linear Program (LP) formulation by Bhatia *et al.* [4] exist, adding different practical constraints [23], [24]. While at first we focus exclusively on failure resiliency, these additional requirements will be reincorporated in Section VIII. General use-cases of resiliency in conjunction with SR are documented in [8].

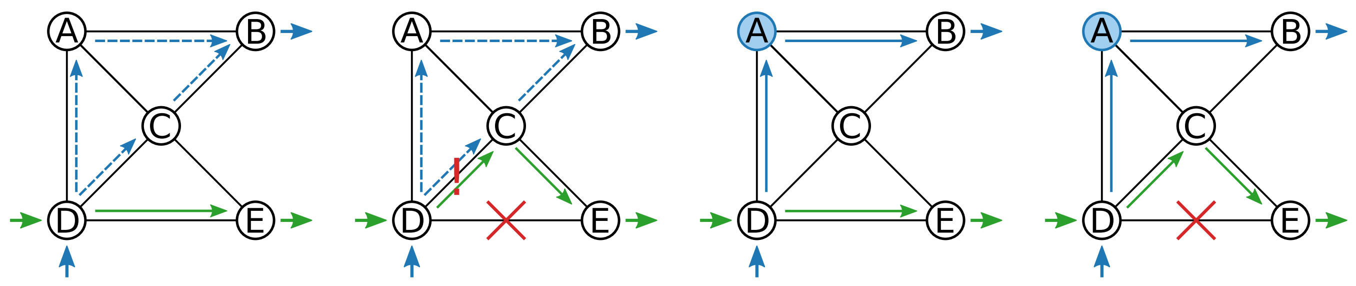
III. Related Work

TI-LFA [7, Chapter 9] is a strategy that can be used to precompute backup paths from the Point of Local Repair (PLR) to the destination for predefined failures. With the use of SR and strategically placed segments, it manages to avoid routing loops and steer the packet along the post-convergence path. It does, however, not incorporate any considerations regarding possible network congestion. Also, the backup path only represents the so-called post-convergence path starting at the PLR and not from the ingress node, where the demand enters the network. Thus, even though TI-LFA works well as a short term compromise, it is not well suited for longer outages.

Exact optimizations of SR are not fast enough to be used in an online fashion. To recompute and reconfigure a network upon failure, heuristic approaches are required. In [12] Gay *et al.* develop a local search that can be used to provide sub-second optimization with SR. Another framework that is capable of online computations is Declarative and Expressive Forwarding Optimizer (DEFO) [13]. It uses a constraint programming technique to solve the problem. Both strategies use up to 2 intermediate segments and attempt to minimize the Maximum Link Utilization (MLU). This is a widely used objective in network traffic engineering that will also be applied in this work. These and similar online approaches do, however, require the abilities of a central controller to recompute routing paths and reconfigure the network. This can be achieved, for example, by an SDN.

Kumar *et al.* [19] also make use of an SDN, but follow a very different approach. They propose a system that finds multiple paths for each end-to-end connection that are selected by predefined metrics, but independent of the traffic demands. A central controller then dynamically adapts sending rates for those paths, taking the current state of the network into account. This idea is based on the fact that redefining end-to-end paths is relatively slow, while only adapting sending rates can be done quickly. The authors show that this method yields competitive results and is robust against single link failures. For the latter, a simple recovery mechanism is implemented, which redistributes traffic onto all paths that are still intact.

Aubry *et al.* [1] define the notion of robustly disjoint paths. They subsequently propose an optimization problem to find these paths with the help of SR. Their general approach is similar to the one presented in this paper, as they try to find SR



(a) no failure and no SR (b) failure and no SR (c) no failure and SR (d) failure and SR

Fig. 1. Routing of two demands, blue and green, before and after a failed link, with and without proactive SR. All links have the same metric and can route exactly one demand. Dashed demands indicate load balancing. A colored node specifies that it has been configured as an intermediate segment for the matching traffic.

paths that remain disjoint after IGP convergence in predefined failure cases. As a secondary goal the latency is minimized. They test the approach using all single link and 2-link, as well as random 3-link failures. This work, while being similar to ours in idea, serves a different use-case. Both approaches could be run in parallel. The robustly disjoint paths optimization works best for enterprise traffic, where it is critical to have disjoint paths in place. PCA2SR results can in turn be applied to best-effort traffic to keep utilization to a minimum.

While we focus on SR based IGP traffic engineering in this work, significant impact can also be achieved by tweaking Border Gateway Protocol (BGP) to manipulate the entry and exit points of traffic within an Autonomous System (AS). One example is the tool Tweak-it [26] that additionally minimizes changes to BGP. In [3] another hybrid approach is proposed. The authors focus on optimizing metrics within an AS, but they add virtual nodes with information from BGP with the goal to include interdomain links in the weight optimization process.

IV. Post Convergence Aware 2SR

This section discusses the conceptual idea behind the optimization, which is explained in detail in the next section. Please note that the concept below is too complex to be implemented with a traditional metric tuning approach, as discussed in the second paragraph of the introduction.

The concept of PCA2SR is introduced in the form of an example in Figure 1. It shows a simple, constructed network containing 5 nodes and 7 undirected edges. All edges have the same capacity and link metric. We now attempt to route two demands. The blue demand enters the network on node D and exits through node B, while the green demand also originates at node D, but is destined towards node E. Both demands are equal in size and match the capacity of an edge. Figure 1a traces the shortest paths for each demand. Since there are two equal routes for the blue demand, it is distributed evenly between them. We now assume that the link between node D and E breaks down. Figure 1b depicts the post-convergence paths. The path of the blue demand remains unchanged, while the green path now uses C to bypass the failed link. This congests the link between node D and C, as 50% of the blue demand and 100% of the green demand now traverse it.

Because post-convergence paths can be computed in advance, this scenario could have been anticipated and avoided with the help of SR. Figure 1c shows the network before the failure occurs. This time, the blue demand follows an SR path with an intermediate segment on node A. It is no longer load shared along two paths and, thus, does no longer use the link between node D and C. Upon failure of edge D-E, as seen in Figure 1d, edge D-C is free to be used by the post-convergence path of the green demand and all demands can be handled without issue.

While this is a very contrived example, it showcases the key idea behind PCA2SR. First, topological data of the network and traffic data is measured and collected. Additionally, the operator defines a set of failure cases, which the network should be resilient against. Then, all post-convergence paths of these failure cases are computed in advance. Using this information, PCA2SR finds an SR configuration that minimizes utilization in all predefined failures. The linear programming formulation behind this is explained in the following section. The resulting SR configuration can be deployed. If any of the predefined failures do occur, a short term solution, such as FRR, takes over to provide a well defined and fast emergency routing to bridge the gap until the IGP reconverges. Then, the network will continue to run with little to no congestion without any configuration changes.

V. Optimization Model

The optimization model described in this section is based on the 2SR formulation by Bhatia *et al.* [4]. The original linear program minimizes the Maximum Link Utilization (MLU), which is a widely used traffic engineering objective. We add a set of failure constraints and adapt the objective function to match our new use-case.

Before discussing the formulation, a few notations have to be introduced and defined. We define a network as a multi-graph $G = (V, E)$ with a set of nodes V and a multi-set of directed edges E between pairs of nodes. Each edge $e \in E$ is associated with a capacity $c(e)$ and a link metric $m(e)$. With this information we can compute shortest paths between all node pairs. Using these, we can compute the share of each end-to-end traffic that is going to be routed along an edge e, which is denoted by $g_{ij}^{k}(G, e)$. Note that there are three node indices: i, j, and k. This is because each traffic is being routed on a 2SR path, with i being the source, j the destination, and k the intermediate segment. In practice, we do not need a 2SR path for each end-to-end connection, as in some cases the simple shortest path is sufficient. We use x_{ij}^{j} to define the amount of traffic that is being routed along this shortest path. If k happens to be unreachable, for example in case of a node failure, the function g returns $g_{ij}^{j}(G,e)$ instead of $g_{ij}^{k}(G,e)$. In this case, when node i knows that k is unreachable and recognizes that, the configured SR path is, thus, invalid. The node then proceeds to route traffic along the shortest path instead. The total amount of traffic that is to be routed throughout the network is measured and recorded in t_{ij}, where i is the ingress node and j is the egress node of a specific demand.

We now define a set of failure scenarios F. A failure scenario $f \in F$ is a list of one or more edges. This allows us to model various failure types in a modular fashion. We consider the following three failure types:

- **Link failure.** A link failure of a single edge e can be modeled by the failure scenario $f = [e]$. Note that in most ISP backbone networks, there is more than one physical link between two nodes. This scenario specifically focuses on *single* link failures, which does not necessarily mean a disconnect between two nodes. Rather, in most of the cases, a single link failure implies a reduction of the total bandwidth installed between two routers.
- **Node failure.** The failure of node n can be modeled by the failure scenario $f = [\{(i,j) \mid n \in \{i,j\}\}]$. It contains all ingoing and outgoing edges of the failing node.
- **Shared Risk Link Group failure.** An SRLG contains multiple resources that share a common risk. Usually many links are bundled and buried together. This not necessarily means that they have the same source and/or destination, but all of them will fail if the link bundle is cut. The failure of an SRLG s can be modeled by failure scenario $f = [\{e \mid e \in s\}]$.

Since the model does not differentiate between different failure types, any combination of them, or even completely new types can be used.

All symbols introduced until now are either constants or can be computed in advance. Problem 1 shows the optimization model including its variables, objective function, and configurable parameters.

$$\min \sum_{f} (\phi_f - \Phi)$$

$$\text{s.t.} \quad \sum_{k} x_{ij}^{k} \geq t_{ij} \qquad \forall ij \qquad (1)$$

$$\sum_{ij}\sum_{k} g_{ij}^{k}(G,e)\, x_{ij}^{k} \leq \Theta\, c(e) \qquad \forall e \qquad (2)$$

$$\sum_{ij}\sum_{k} g_{ij}^{k}(G \setminus f,e)\, x_{ij}^{k} \leq \phi_f\, c(e) \qquad \forall f,e \qquad (3)$$

$$\phi_f \geq \Phi \qquad \forall f \qquad (4)$$

$$x_{ij}^{k} \geq 0 \qquad \forall ijk \qquad (5)$$

Problem 1: Post-convergence aware 2 SR optimization

First, we define the set of 2SR path variables x_{ij}^{k}. Semantically speaking, x_{ij}^{k} counts the amount of traffic originating at node i that is destined towards node j and uses node k as an intermediate segment. Inequality (1) guarantees that the sum over all intermediate nodes matches the measured traffic t_{ij}.

There are two parameters by which the model can be configured. First, we define Θ as the normal case link utilization upper bound. Inequality (2) ensures that, while the network is completely intact and no failures occur, the link utilization of each edge is less or equal than Θ. Note that, rather than computing and minimizing the MLU as it is done in the original formulation, we use it as a constant to limit the solution space. This also means that Θ may not be lower than the optimal MLU, or the LP will be infeasible. From an operator's point of view, it is less important how exactly the utilization looks like or if the configuration is optimal. In fact, it is more relevant to avoid congestion. By using Θ as an upper bound for the normal case, this is guaranteed. We found that if the normal case constraint is left out, this guarantee does not hold, despite the fact that the topology gets more sparse with any failure and thus more difficult to route.

The second parameter is Φ. We define it as the failure case MLU threshold. If the MLU is below or equal to this value after a failure f occurred and the IGP converged, we ignore it. If it is above the threshold, we add it to the objective value, which is to be minimized. In other words, we minimize all failure MLUs higher than Φ. From an operator's point of view this can be motivated similarly as Θ. As long as the utilization stays below a certain threshold, in this case Φ, it does not matter if it is not the most efficient routing configuration. Instead, we want to focus on the more difficult failure scenarios and keep overutilization to a minimum.

In Problem 1, this behavior is ensured by the objective function, as well as constraint (3). We define a set of variables ϕ_f, which represents an upper bound to the MLU after the IGP reconvergence, triggered by failure scenario f, is finished. In the mathematical formulation, this is indicated by the use of $G \setminus f$, which is the network graph G without the edges contained in f. Apart from this, the computation of the MLU for each failure scenario is the same as in inequality (2).

The objective function essentially minimizes the sum of all ϕ_f. The minimum value of ϕ_f is set to Φ, to reflect the fact that only values above this threshold matter in the optimization. This does not imply that every failure case MLU has to be at least Φ—it only has to be less or equal than the respective ϕ_f. To make the objective value easier to interpret, the sum of all minimal values of ϕ_f is subtracted again. The final result is the total failure MLU overutilization. Since it is highly non-trivial to weight one failure against another, we decided to treat every failure scenario equally. If the importance or probability of each failure scenario could be quantified, a vector of weights could easily be incorporated into the objective function.

Another potential modification of the objective is the addition of counting variables. Instead of minimizing the total overutilization, the number of links or link failures leading to congestion could be minimized. The objective in this case is to minimize the cases where congestion occurs, regardless of how bad it is. This, however, would require the introduction of integer variables. Since this makes the problem harder to solve, we only briefly tested this modification. We found no benefit over the presented formulation.

VI. Evaluation Setup

In this section we briefly describe the algorithms we run PCA2SR against, as well as tested parameter configurations. We also discuss the computational performance and describe the traces the evaluation is based on.

A. Algorithms

We compare our approach against the following two routing strategies:

- **Shortest Path Routing (SPR)**. SPR is used as a simplified IGP simulation. It uses Equal-Cost Multi-Path (ECMP) to split traffic onto equally weighted paths and serves as a comparison against state of the art routing.
- **Theoretical Lower Bound (LB)**. To get a sense for the lower bound, we computed a separate 2SR configuration for *each* individual failure scenario. It uses the optimization problem introduced in [4]. Please note that this is a theoretical lower bound, as it would have to be calculated and configured for the specific failure, when the failure is detected. Unfortunately, such real-time calculations can not be performed fast enough today. Nevertheless, for each individual failure scenario this is optimal.

All computations were performed on a Dell PowerEdge R620 cluster node with a 12 core Intel Xeon E5-2620 CPU and $8\times$32GB DDR3 synchronous 1600 MHz RAM running 64-bit Ubuntu 12.04.5. LPs were solved using CPLEX 12.6.2 [16].

B. Computational Performance

A major challenge of trying to find one SR configuration for many failure scenarios is the high number of variables and constraints it entails. Constraint (3) is written down for each edge in every failure case. If we look at single link failures this results in $\mathcal{O}(n^4)$ inequalities. Within each constraint, we summarize over all possible source, destination, and intermediate nodes. We, therefore, have a complexity of $\mathcal{O}(n^7)$ by just writing down constraint (3). This leads to a runtime of approximately 7–8 hours for each instance, which already incorporates parallelization.

Because a majority of the time is spent writing down constraint (3), we tried modeling it via a separation routine. This means the LP is initially solved without failure constraints and the solution is checked for violations. If any occur, the corresponding constraints are added and the problem is optimized again. Following this strategy, we managed to significantly reduce the runtime for testing topologies with up to 50 nodes. Unfortunately this did not carry over to the full sized instances used in the evaluation.

C. Data

We test our optimization model on nine snapshots measured in a tier 1 ISP backbone network evenly spaced within a time frame between April and September 2018. Each snapshot contains topological data, which includes the network graph, as well as SRLG information, and a traffic matrix. Each traffic matrix is a capture of a 15 minute period during the peak hour [24]. One network topology snapshot consists of about 114 nodes with approximately 2800 individual edges. The edges are configured with link weights as optimized by the operator. This improves the quality of simple SPR, making it a good estimation for a state of the art traffic engineering approach. At the same time, we have shown in [21] that metric optimization neither hinders SR traffic engineering, nor is it necessary to get practical results.

It should be noted that, while the data is based on a real network, all results shown in this paper are attained by trace-based simulations. They may not necessarily reflect the actual state of the network if any of the algorithms were actually deployed. Unfortunately, we can not provide further details on the configuration of the network due to confidentiality beyond the details (including a visualization of the topology) described in [24].

VII. Evaluation Results

As discussed in Section V, PCA2SR is configurable by two parameters: the normal case utilization upper bound Θ and the failure case MLU threshold Φ. We evaluate different values of these parameters on the set of single link failures to show that the optimization works with almost any configuration. Furthermore, we evaluate the ability of our optimization model to optimize for single link failures and SRLG failures at the same time. Subsequently, we investigate node failures, before investigating an outlier trace that proves to be especially difficult to optimize. We close the evaluation with performance considerations.

A. Single Link Failures

We use a failure set of all possible single link failures to evaluate the parameters Θ and Φ. To reiterate, our optimization model seeks to find a singular, completely static configuration that is acceptable while the network is fully intact (i.e. there are no failures) and minimizes overutilization in case of any single link failure.

While we recommend to selected an Φ greater than Θ, it can theoretically be set to any value. A higher value decreases the complexity of the optimization, because it accepts a higher number of sub-optimal solutions that are still acceptable and, thus, terminates more quickly. But a higher threshold does, in turn, leave less breathing room for any unexpected traffic spikes or failures that were not taken into account. Constant Θ, on the other hand, may not be set to an arbitrary value. If it is set too low, the linear problem becomes infeasible. This happens, because constraint (2) limits the solution space to configurations that achieve an MLU of at least Θ when the network is completely intact. We, thus, recommend to run a basic 2SR optimization first, to get an estimate on the lower bound, and then select a value higher than the estimate for Θ.

For the parameter evaluation, we tested values of $\Theta \in \{0.7, 0.8, 0.9\}$ and $\Phi \in \{0.9, 0.95, 1\}$. We found that PCA2SR manages to remove any cases where congestion might have occurred, reducing the objective value to zero in all but one instance for all parameter combinations. The last instance will be excluded in the following figures and discussed later on, as it illustrates a special case. To look at the optimization

TABLE I

THE DISTRIBUTION OF MLUS IN ALL SINGLE LINK FAILURES AND IN ALL INSTANCES AS OPTIMIZED BY PCA2SR WITH DIFFERENT CONFIGURATIONS OF Θ AND Φ IS SHOWN HERE. THE RANGE OF RESULTING MLU IS CLUSTERED INTO BINS OF SIZE 0.025 ON THE Y-AXIS. ONE CELL ENTRY COUNTS THE NUMBER OF FAILURE CASES THAT RESULT IN AN MLU THAT FALLS WITHIN THE CORRESPONDING BIN. IT BECOMES CLEAR THAT MOST OF THE RESULTING VALUES POSITION THEMSELVES TOWARDS THE BIN Θ BELONGS TO, WHILE ANOTHER CLUSTER CAN BE OBSERVED AT THE TOP OF EACH COLUMN

Θ	0.7	0.7	0.7	0.8	0.8	0.8	0.9	0.9	0.9
Φ	0.9	0.95	1	0.9	0.95	1	0.9	0.95	1
$rlim$	# of failure cases with MLU in range $[rlim_{row}, rlim_{row+1})$								
$> \Phi$	0	0	0	0	0	0	0	0	0
1.000	0	0	385	0	0	461	0	0	447
0.975	0	0	61	0	0	58	0	0	85
0.950	0	448	45	0	537	66	0	561	50
0.925	0	140	79	0	87	37	0	121	65
0.900	590	59	53	600	63	45	5810	5128	5163
0.875	114	64	51	112	59	61	0	0	0
0.850	51	61	49	81	69	70	0	0	0
0.825	80	84	78	129	81	97	0	0	0
0.800	78	70	69	4888	4914	4915	0	0	0
0.775	78	91	73	0	0	0	0	0	0
0.750	108	92	81	0	0	0	0	0	0
0.725	153	146	123	0	0	0	0	0	0
0.700	4558	4555	4663	0	0	0	0	0	0

results in more detail, Table I shows the distribution of MLUs in all failures and all instances in different parameterizations. The range of possible MLUs is grouped into 13 bins in steps of 0.025. Each bin contains the number of failure cases leading to an MLU that fits into its range. The table uses $rlim$ to define the maximum value of each range. The bin corresponding to a row with $rlim = 0.875$, for example, collects all MLUs with a value v of $0.85 < v \leq 0.875$.

The two parameters are distinctly visible, as every bin above the normal case utilization upper bound Θ or below Φ is empty. All configurations show comparably large bins at the value of Θ. Note that the optimization model prohibits normal case utilizations higher than Θ, but does not distinguish them any further. Hence, in most cases, a normal case utilization of exactly Θ will be selected. Approximately 10% of single link failures appear to be more challenging and form a second cluster at the upper threshold Φ, but none surpass it. It should be noted that all values that are being grouped in the top-most bin, which theoretically can contain values greater than Φ, exactly match Φ. To make this clear, we added an extra line at the top, which counts all values above Φ, which is empty in all parametrizations. This means that the objective was fully met by all shown configurations.

To conclude, PCA2SR is able to work with different configurations and can eliminate congestion for all single link failures in all shown instances. The parameters can, thus, be almost freely chosen by the operator, following their individual requirements. For the remainder of this paper, we are going use a configuration of $\Theta = 80\%$ and $\Phi = 100\%$. This leaves a reasonable buffer for anomalies in the normal case and avoids congestion in failure cases.

Now that a configuration for PCA2SR is selected, we can look at the performance in relation to other traffic engineering strategies. Figure 2 additionally shows the results of SPR, an estimation of state of the art routing, and the lower bound for PCA2SR. Each subfigure visualizes the distribution of failure MLUs ϕ_f for a single instance. Each distribution is visualized by its Empiric Complementary Cumulative Distribution Function (ECCDF). The distribution corresponding to PCA2SR is colored in blue, SPR in orange, and the lower bound computed by individual 2SR solutions per failure in green. Φ is indicated with a red vertical line and Θ with a green dashed line. Generally, values on the left are better than values on the right. However, to satisfy the use-case described in Section IV, it is sufficient to minimize values to the right side of the red line. For a few failures handled by SPR, this goal is not met. Instance C or G, for example, show more than one link failure with a resulting MLU of more than one. If we consider the complete distribution, it may seem that SPR is better than PCA2SR, because in most other cases, its curve lies to the left of the blue optimized distribution. But, to re-iterate, for the objective of minimizing overutilization in failure cases, this is irrelevant. A utilization of 60% instead of 80% does not matter in our use-case. More importantly, PCA2SR avoids congesting links in all nine instances, whereas SPR can avoid congestion only for a single instance; in all other instances, at least some failures result in MLUs above one.

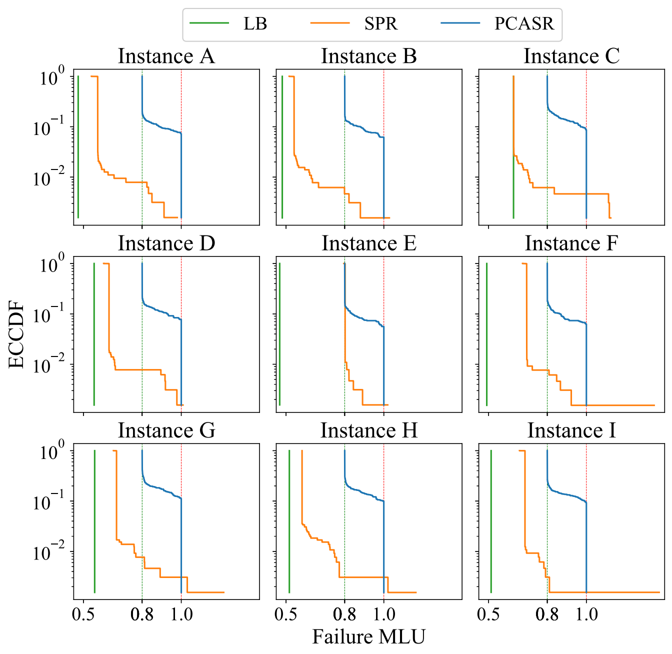


Fig. 2. Comparison of MLU distributions in all **single link failure cases** as determined by SPR, PCA2SR ($\Theta = 0.8$, $\Phi = 1.0$), and the lower bound.

While our use-case does not call for the lowest possible network utilization, our optimization model is also able to produce results close to the theoretical lower bound. We test this by setting Φ to zero, thereby minimizing the MLU in all failure scenarios. We also set $\Theta = 70\%$, a tighter value than chosen in our reference parametrization, to show the potential of PCA2SR. The results are visualized in Figure 3. There appear to be two categories of distributions across the instances. (1) PCA2SR is able to reach the optimal MLU for 98–99% of all failure scenarios in instances E–I with just a single, static SR configuration. (2) In instances A–D, the failure

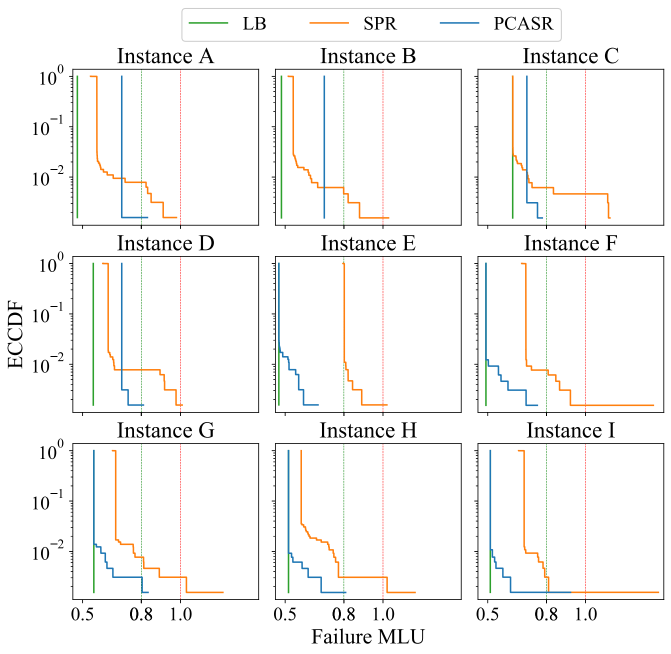


Fig. 3. Comparison of MLU distributions in all **single link failure cases** as determined by SPR, PCA2SR ($\Theta = 0.7$, $\Phi = 0.0$), and the lower bound.

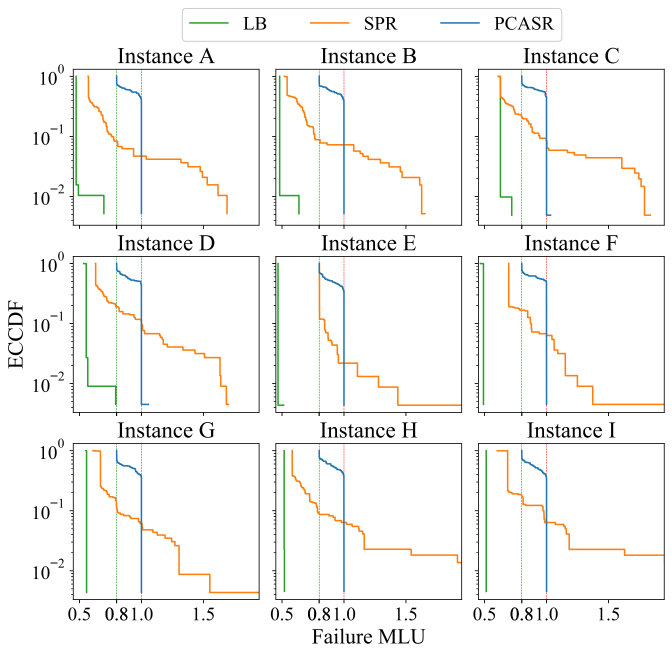


Fig. 4. Comparison of MLU distributions in all **SRLG failure cases** as determined by SPR, PCA2SR (simultaneously optimized for single link failures, $\Theta = 0.8$, $\Phi = 1.0$), and the lower bound.

MLU are only pushed as low as $\Theta = 70\%$. This is likely due to an implementation detail to save memory: assuming $\Theta \leq \Phi$, we omit ϕ_f variables and corresponding constraints (3) if their respective $g(G \setminus f, e)$ function is identical to $g(G, e)$. In the somewhat strange scenario $\Phi \ll \Theta$ here, this sometimes results in a solution that is already considered optimal when achieving Θ (similar to before). Nonetheless, we can see in instances E–I that PCA2SR is theoretically able to yield close to optimal results when optimizing single link failures.

B. Shared Risk Link Group Failures

The impact of PCA2SR seemed very low and most of the time unnecessary when only looking at single link failures, as most of these failure cases did not notably affect the network at all in the first place. This changes, when the complexity and number of failure cases increases. Now, we add SRLG failures into the set of failures to optimize. One SRLG commonly contains many links, which makes it more difficult to find configurations that avoid congestion. It should also be noted that we consider these SRLG failures *in addition* to single link failures. In other words, PCA2SR aims to find a single 2SR configuration that removes congestion in every possible single link failure *and* every SRLG failure case. Despite this increased difficulty, the distribution of MLUs for single link failures appears to be the same as without adding SRLG failures.

Figure 4 shows the MLU after each SRLG failure that was configured. We compare the same three traffic engineering strategies as before. The number of failure cases where the result of PCA2SR matches the normal case link utilization lower bound Θ is considerably lower than observed in Figure 2. This supports the assumption that SRLG failure cases are more challenging to compensate than single link failures. But most importantly, there are now much more failures where using SPR leads to congestion. These cases, additionally, are now much more severe, as we can observe values of up to two. While the lower bound manages to eliminate these completely, PCA2SR manages to avoid congestion in seven out of nine instances. In Instance C and D there remains a single link failure that results in an almost negligible overutilization. Overall it provides a significant improvement over SPR, for which up to 10% of all SRLG failures result in congested links.

C. Node Failures

Node failures are the most severe failure types considered in this paper. They rarely occur in practice, as there typically is some form of redundant backup hardware that takes over if the primary router fails. More commonly, single components, such as line cards, fail. These are already covered by single link and SRLG failures. Nonetheless, node failures should at least be considered. Due to the severity, we solely focus on node failures and do not simultaneously optimize other failures.

Figure 5 shows the results after computing SPR, PCA2SR, and the lower bound considering node failures. It quickly becomes apparent that these failures are indeed difficult to compensate, because even the lower bound is now unable to avoid congestion in five out of nine cases. Yet, in four instances PCA2SR does manage to remove any overutilization that SPR produces. In the remaining cases it matches the lower bound. This supports our argument that PCA2SR is able to yield results close to the theoretical lower bound (cf. Section VII-A), despite only using one, static SR configuration.

Interestingly, when looking at SPR, the overutilization that is present appears to be less severe than with SRLG failures. In particular instances B and E appear to have no cases of congestion for any strategy and failure case. This may be,

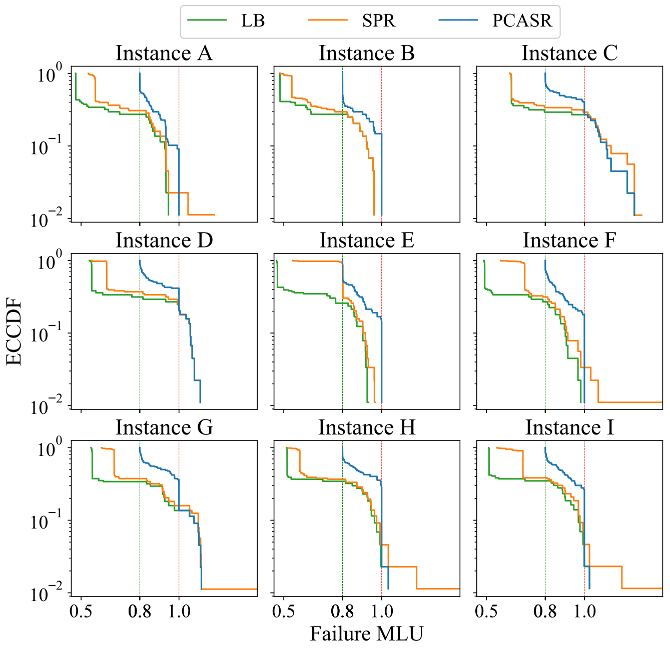


Fig. 5. Comparison of MLU distributions in all **node failure cases** as determined by SPR, PCA2SR ($\Theta = 0.8$, $\Phi = 1.0$), and the lower bound.

because the traffic that originates at or is destined towards a node that fails will be ignored, as discussed in Section IV. As a result, there are fewer traffic demands across the whole network.

D. Optimization of Challenging Topologies

In this section, we enlarge our view by considering another (tenth) topology. It is from the same network as the other nine instances (see Section VI-C). But it was measured at a point in time where there were two simultaneous intercontinental sea-cable faults already present in the topology. This is a special, challenging scenario, as the failure optimization starts in a state where failures are already present. Nevertheless, the optimization of this instance does yield some interesting results.

The optimization of link, SRLG, and node failures are displayed in Figure 6. As before, we optimized link and SRLG failures simultaneously, while node failures are kept separate. Looking at the performance of SPR, we can see that the network is in a state that is congested regardless of which failure is added. This proves that the outage of two major sea-cables is a severe scenario. Yet, PCA2SR manages to eliminate the overutilization in most failure cases. While it is not optimal for link and SRLG failures, it matches the lower bound for node failures.

VIII. Incorporating Real-World Constraints

In [24], we extended the basic 2SR formulation by Bhatia *et al.* [4] with various real-world constraints and requirements to make the results practically deployable. Failure resiliency, however, was not addressed in [24]. We now want to combine all those practical requirements, including the failure resiliency of PCA2SR. First, we give a brief overview

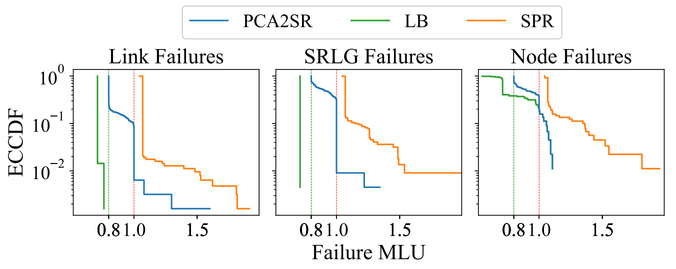


Fig. 6. Comparison of MLUs in all failure classes in a tenth instance as determined by SPR, PCA2SR ($\Theta = 0.8$, $\Phi = 1.0$), and the lower bound.

of the additional requirements. Second, the extended mathematical model is presented. Third, we conduct an evaluation and discuss the results.

A. Practical Requirements

We consider the following four additional real-world constraints (6)–(12), as shown at the bottom of the next page.

1) Number of SR Tunnels: The previously featured optimization results require upwards of 3000 2SR tunnels to be installed, because the PCA2SR model is completely agnostic in this regard. It does not incentivize choosing shortest paths over 2SR tunnels and, therefore, almost all demands are routed using 2SR. Additionally, a single demand is commonly divided onto multiple 2SR paths, which increases the total number of tunnels further. However, this poses a significant maintenance overhead in practice, which is avoidable, as we will see later in the evaluation.

2) Traffic Splitting: The basic 2SR formulation allows each end-to-end demand to be divided into arbitrary fractions on any number of different SR paths in addition to ECMP. Current router hardware is not able to replicate this. For example, a router running JUNOS [18] can split a single traffic demands into at most 32 or 64 equally sized parts. We found that restricting each demand to use at most one intermediate segment is a good solution and synergizes perfectly with requirement *1*. It should be noted that this does not interfer with ECMP, which can freely be used in between segments.

3) Router Blacklist: ISPs may have special nodes in their network architecture that must not be chosen as intermediate nodes. This can be solved using a blacklist. Nodes added to the blacklist can only be source or destination nodes of demands. An ISP could, for example, blacklist all Label Edge Routers (LERs), in order to avoid transit traffic on edge nodes.

4) Latency Policies: With SR, latency policies that were previously enforced through link metric design, can be easily bypassed. We implement a simple hierarchical node classification to counteract this. Each node is categorized in regard to its continent, country, and site. If source and destination are in the same country, for example, then the intermediate node may not be from a different country. This approximation method was first presented in [25].

B. Post Convergence Aware 2SR Tunnel Limit Extension

The mathematical model of the Post Convergence Aware 2SR Tunnel Limit Extension (PCA2TLE) differs slightly from the original design of the 2SR TLE in [24]. The main idea,

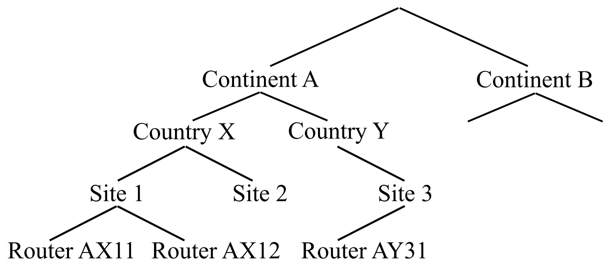


Fig. 7. Exemplary geographical classification of router locations for requirement *3*.

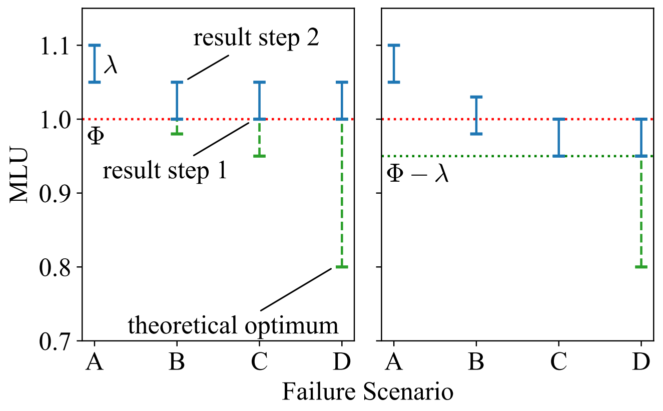


Fig. 8. Two design concepts for PCA2TLE. On the left, the tradeoff constant λ is used only in the second optimization step. On the right, it is incorporated in both steps. The plots show exemplary MLU values for four fictional failure scenarios A–D for both steps, as well as a theoretical optimum.

however, is similar. There are two objectives: Minimizing the MLU (in failure cases) as primary objective, and minimizing the number of SR paths used as secondary objective. At first, the primary objective is optimized. Afterwards, the result is injected into the model and the secondary objective is optimized. Latency is another objective that could have been investigated here. We chose not to, because it is implicitly covered in the secondary objective.

In [24], the first step serves as quick approximation of 2SR. Additional constraints are introduced in the second step only. Due to this design, the first problem can be much simpler than the second one. This can lead to the second optimization being infeasible and especially becomes problematic if the constraints become more impactful, like the failure constraints do. It also makes the model more difficult to configure. Thus, in this extension, we chose to incorporate all constraints to *both* optimization steps.

The first optimization step is presented in Problem 2. In order to satisfy requirement *(1)*, the absolute, fractional traffic flow variables x_{ij}^k are changed to relative, binary traffic flow variables α_{ij}^k. This implicitly entails that if an α_{ij}^k is set to 1, then k (and no other k) is used for the traffic from i to j. This implicitly fulfills requirement *(2)*. Apart from this change, the inequalities 6–8 are as in Problem 1. Equation 9 models requirement *(3)*. We define B as a set of nodes that the network operator would like to blacklist. A blacklisted node may not be selected as intermediate node k, except if the shortest path ($k = j$) is used:

Consider a 2SR path that originates at node i, visits the intermediate segment k, and ends at node j. A corresponding demand is then treated as being routed along the direct shortest path iff $\alpha_{ij}^k = 1$ with $k = j$.

Equation 10 handles requirement *(4)*. Each node is associated with geographical information, which can be sorted into three hierarchy tiers, and thus, a geographical tree according to its continent, country, and site. We consider the exemplary hierarchy depicted in Figure 7 for demonstration purposes. The Lowest Common Ancestor (LCA) of Router AX11 and Router AX12 is Site 1. Every leaf node of Site 1 may be used as intermediate node for traffic between these two routers. Every other node, such as Router AY31, must not be used and the corresponding traffic flow variable is set to 0 by equation 10. To give another example, $T_{LCA(\text{AX11,AY31})}$ returns the subtree with Continent A as the root. Router AX12 is a leaf node of Continent A, and may, therefore, be used as an intermediate node for traffic between AX11 and AY31. All nodes in Continent B, however, are forbidden.

In the second optimization step, we change the objective to

$$\min \sum_{k \neq j} \alpha_{ij}^k$$

$$\min \sum_f (\phi_f - \Phi - \lambda)$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_k \alpha_{ij}^k = 1 && \forall ij && (6)\\
& \sum_{ij} \sum_k g_{ij}^k(G,e)\alpha_{ij}^k t_{ij} \leq \Theta\, c(e) && \forall e && (7)\\
& \sum_{ij} \sum_k g_{ij}^k(G \setminus f, e)\alpha_{ij}^k t_{ij} \leq \phi_f\, c(e) && \forall f, e && (8)\\
& \alpha_{ij}^k = 0 && \forall ijk \mid k \in B \wedge k \neq j && (9)\\
& \alpha_{ij}^k = 0 && \forall ijk \mid k \notin T_{LCA(i,j)} && (10)\\
& \phi_f \geq \Phi - \lambda && \forall f && (11)\\
& \alpha_{ij}^k \in \{0,1\} && \forall ijk && (12)
\end{aligned}$$

Problem 2: Post-convergence aware 2 SR optimization with real-world constraints.

TABLE II
APPROACHES CONSIDERED IN THE EVALUATIONS

Approach	Details in Section
Shortest Path Routing (SPR)	VI
Theoretical Lower Bound (LB)	VI
Post Convergence Aware 2SR (PCA2SR)	IV
PCA2SR Tunnel Limit Extension (PCA2TLE)	VIII-B

This minimizes the total number of SR tunnels chosen (requirement *1*). Note that we do not count traffic variables that are associated with the shortest paths, as the shortest path will be used automatically if no SR tunnel is specified.

We also adjust the upper and lower bounds of ϕ_f to

$$0 \leq \phi_f \leq \phi_f^* + \lambda \quad \forall f$$

where ϕ_f^* are the now constant results from Problem 2. The *tradeoff constant* λ is added onto the upper bound to give the secondary objective some more room to work with. It should be noted that λ is not only used here, but also lowers the failure MLU threshold Φ in the first step. In [24], λ only came into play in the second optimization step. This, however, does not work in combination with Φ. To illustrate this issue, we use four fictional failure scenarios A–D and plot hypothetical MLUs after both optimization steps and a theoretical optimum (possibly achieved by setting Φ to 0) in Figure 8. We show two different designs in terms of the usage of λ. On the left-hand side, the first step runs without the use of the tradeoff constant. We use a λ of 5% in the second step, which is added onto the results of the first step, just like in 2TLE. But, because the model does not distinguish results that score lower than Φ, most scenarios will return Φ in step one. Adding the additional 5%, therefore, leads to unnecessarily high utlization, especially in scenario D. On the right-hand side, we additionally reduce the failure MLU threshold in step one by λ, just like in Problem 2. As a consequence, we can add λ to the intermediate results and solve step two without producing unnecessary high utilization. Meanwhile, difficult failure scenarios, like scenario A, still get the same amount of flexibility as in the design on the left. Note that λ, in contrast to the original formulation of 2TLE, is used as an additive constant instead of a coefficient. This makes the model much easier to understand and configure, since we are handling multiple utilizations instead of just one.

C. Evaluation

To test the combination of PCA2SR and TLE, we use the same scenarios, configurations, and hardware as in Section VI. We furthermore use a tradeoff constant of $\lambda = 5\%$, as suggested in [24]. Table II summarizes all approaches considered in the evaluations.

In Figure 9 we can see the results for optimizing link failures. Additionally to the format of the previous figures, each subfigure now features a counter in the top right corner that shows how many SR tunnels were deployed to reach the results of PCA2TLE, visualized by the purple distribution. The number of 2SR paths required to implement the PCA2SR results are not displayed, because PCA2SR optimizes oblivious of this secondary objective. The resulting numbers are all well above 3000 in all instances and, as noted earlier, this is not a manageable number in practice. There are two main points to observe in Figure 9. First, the distribution for the optimized failure MLUs of PCA2TLE is much more akin to the SPR curve than PCA2SR. This can be explained using the second observation: The number of deployed SR tunnels is quite low in all instances. In Instance A, there is no SR needed at all, because the SPR solution already satisfies all our thresholds and constraints. In the other cases, as stated in the first part of the evaluation, SPR only leads to congestion in very few cases, which can be repaired using as few as one to at most four SR tunnels in Instance C. While we are now using a significantly smaller number of SR tunnels, at the same time this eliminates the impression that the optimized results appear to be worse for the majority of failure scenarios, as discussed in Section VII-A. Here, the results of PCA2TLE consistently look on par with SPR for most failures and better for the tail end of the distributions.

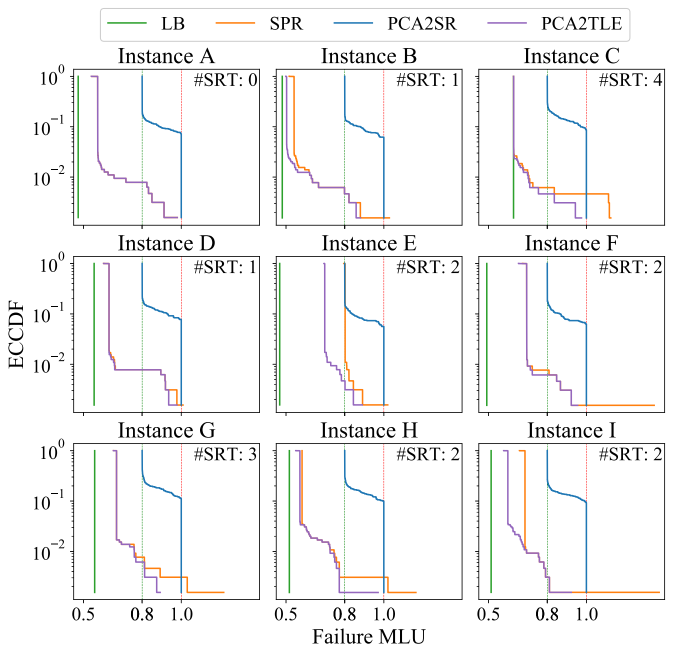


Fig. 9. Comparison of MLU distributions in all single **link failure cases** as determined by SPR, PCA2TLE ($\Theta = 0.8$, $\Phi = 1.0$), and the lower bound. For PCA2TLE, the corresponding number of mandatory SR tunnels (#SRT) is displayed additionally.

As explained in Section VII-B, the introduction of SRLGs makes the problem more difficult. This is underlined by the noticeably higher number of SR tunnels per instance, as visualized in Figure 10. Instances C and E stand out especially, as they require more than 100 SR tunnels. Meanwhile, the other instances can be optimized using between 10 and 20 paths. Just as observed with single link failures in the previous paragraph, the optimized distribution of SRLG failure MLUs is not as close to the two configurable thresholds as with PCA2SR. Rather, it is closer to the distribution of SPR results. This, however, is only true for utilizations below Φ. In most instances the distribution of PCA2TLE remains below this threshold, while the usage of SPR exceeds it significantly. For most failure cases the results even look better than as optimized by PCA2SR, because of the fact that the latter does

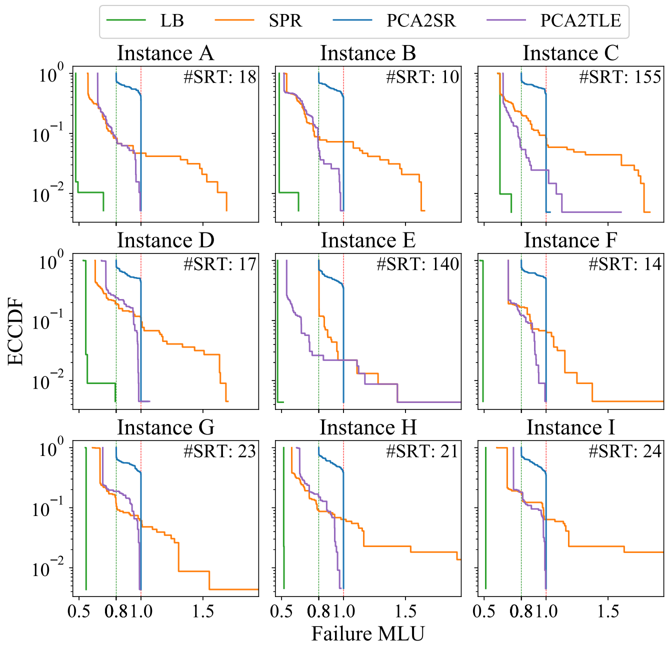


Fig. 10. Comparison of MLU distributions in all **SRLG failure cases** as determined by SPR, PCA2TLE (simultaneously optimized for single link failures, $\Theta = 0.8$, $\Phi = 1.0$), and the lower bound. For PCA2TLE, the corresponding number of mandatory SR tunnels (#SRT) is displayed additionally.

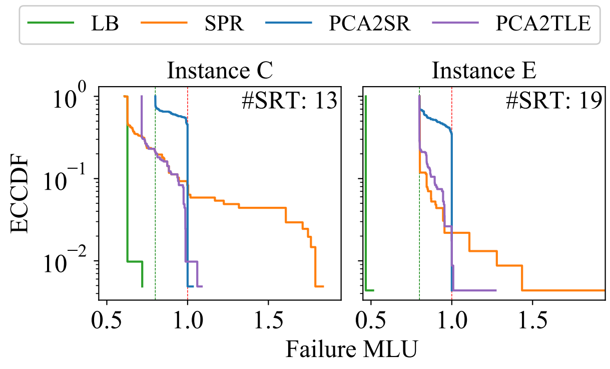


Fig. 11. Results of PCA2TLE with $\lambda = 0.1$ for two instances that were difficult to compute with $\lambda = 0.05$ in comparison to PCA2SR ($\Theta = 0.8$, $\Phi = 1.0$), SPR, and the lower bound. For PCA2TLE, the corresponding number of mandatory SR tunnels (#SRT) is displayed additionally.

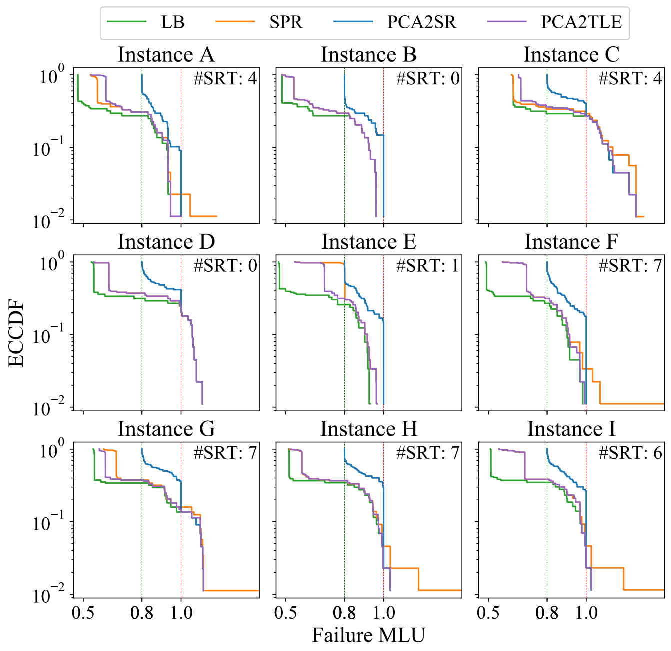


Fig. 12. Comparison of MLU distributions in all **node failure cases** as determined by SPR, PCA2TLE ($\Theta = 0.8$, $\Phi = 1.0$), and the lower bound. For PCA2TLE, the corresponding number of mandatory SR tunnels (#SRT) is displayed additionally.

not further differentiate any solutions that result in an MLU of below Φ. PCA2TLE does this by rewarding solutions with less 2SR tunnels.

It should be noted that the results for instances C and E are not fully optimal, as the optimization was interrupted when coming close to running out of memory. Even though the results have a relatively small relative gap upon termination, which means that the results are not far off, the fact that the results are not exactly optimal and the high number of 2SR tunnels is not satisfying. In Figure 11 we present an alternative configuration of PCA2TLE for the problematic instances. In order to reduce the number of 2SR tunnels, we increase the tradeoff constant λ from the previously used 5% to 10%. As the numbers in the respective subfigures prove, we are now able to reduce the number of tunnels to a similar level as in the other instances. In order to do so, the purple distribution does surpass the blue curve of PCA2SR for a handful of failure scenarios, but the overutilization is only marginal in comparison to the state of the art. To draw a preliminary conclusion, PCA2TLE works well with the simultaneous optimization of single link and SRLG failures. While some instances require an adaptation of the configuration, all requirements can be satisfied appropriately.

The optimization of node failures is a little easier to compute than the previous scenario in regards to additional requirements, because we have a much smaller set of failure cases. But, as explained in Section VII-C, the lower bound starts to be more relevant. The results are displayed in Figure 12. In some instances and failure scenarios, even the lower bound appears to have congested links. As it is close to the shortest path simulation, the optimization model can get away with a very low number of SR tunnels. The most extreme case can be observed in Instance D, where SPR is optimal above the 100% mark. As a result, no SR tunnels are needed, even though some failure cases lead to congestion. These cases, as indicated by the lower bound, are not possible to avoid with 2SR. Because of the tradeoff constant λ, the failure MLU can be slightly higher than with PCA2SR. This effect is visible *only* in Instance C, where the purple distribution of PCA2TLE sometimes lies slightly to the right of PCA2SR, but this is barely noticeable. Most importantly, we are able to compute a deployable set of SR tunnels that leads to no congestion, and, where impossible, close to the lowest possible overutilization.

D. Discussion

Before we move on to the conclusion, we would like to discuss the evaluation approach, as well as ideas that go beyond this publication.

First, it should be noted that the algorithms proposed here are not lightweight. It takes 7–8 hours to optimize one instance. These numbers are definitely not viable for online traffic engineering. Furthermore, the optimization models use traffic aggregates and are not based on individual flows. But this, as indicated in the introduction, is not the intended goal. Rather, we aim to provide a solution that is stable in the long term. We envision that an operator re-optimizes its network once a week or once a month based on latest traffic peak matrices. In [23], we have shown that SR in itself can provide stable configurations for longer periods of time. The approach essentially uses statistical analyses on multiple optimization runs to achieve stability. It could be combined with the optimization model presented here.

Additionally it should be noted that the evaluation was done based on a single tier-1 network. As shown in [11], the success of traffic engineering approaches can vary between different networks. Our main contribution falls into the field of failure management and one of the major strengths lies in the use of real SRLG data. To our knowledge, there is no publicly available data set we could compare our results with and SRLGs are complex and - if at all - difficult to synthesize. Thus, we unfortunately can only see it as future work to evaluate our approaches on other topologies.

IX. Conclusion

Being resilient against failures is one of the most important use-cases for network traffic engineering systems. In this paper we propose PCA2SR, an optimization model that minimizes overutilization after the IGP converges, given any failure scenario contained in a predefined set of failures, while reducing the MLU to acceptable levels (below 80% in our sample parametrization) when no failure is present. As a result it returns a set of 2SR paths that attain this goal. We additionally enhance the model with real-world requirements that have to be respected in order to receive a practically deployable SR configuration. These include the minimization of the number of 2SR paths used, the prevention of splitting up traffic demands in arbitrary fractions, the possibility to blacklist nodes, as well as securing latency policies. We call the final model PCA2TLE.

We evaluated our two optimization models using topology snapshots, traffic matrices, and SRLG information measured in a tier 1 ISP backbone network. Within this dataset, some failure cases can lead to congestion if standard shortest path routing with ECMP is applied. PCA2SR manages to avoid congestion in almost all cases for single link, single link and SRLG, or node failures. Furthermore, it does not require any recomputation or reconfiguration if one of the predefined failures does occur. This holds true even for challenging situations. PCA2TLE additionally complies with all added practical requirements, without noticeably deteriorating the quality of results.

While our main intent is to provide a resilient and deployable traffic engineering solution, the optimization models can also aid in planning the physical expansion of a network. The resulting MLU distribution gives insight into which failures are problematic and difficult to circumvent. This insight could be used to specifically expand links or create redundancies.

Unfortunately, our approach is costly in terms of runtime. Optimizing one instance using PCA2TLE can range from 3 to 20 hours depending on the set of failures and on the configuration of the model. For future work, the idea of using constraint separation could be investigated further, or other ideas of improving the runtime could be explored. Alternatively, using a completely different approach to more efficiently optimize SR paths, such as proposed in [17], could be looked into.

Also, it would be valuable to expand the evaluation to different topologies. Unfortunately, most public data sets only contain topology information, rarely to no real traffic matrices, and no data about SRLGs.

References

[1] F. Aubry, S. Vissicchio, O. Bonaventure, and Y. Deville, “Robustly disjoint paths with segment routing,” in *Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2018, pp. 204–216.

[2] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, *RSVP-TE: Extensions to SVP for LSP Tunnels*, document RFC 3209, 2001.

[3] S. Balon and G. Leduc, “Combined intra-and inter-domain traffic engineering using hot-potato aware link weights optimization,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 441–442, 2008.

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

[5] Cisco Systems. (2019). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[6] Cisco Systems. (2019). *Segment Routing*. [Online]. Available: https://www.segment-routing.net

[7] C. Filsfils, K. Michielsen, and K. Talaulikar, *Segment Routing Part I*. Scotts Valley, CA, USA: CreateSpace Independent Publishing Platform, 2017. [Online]. Available: https://books.google.de/books/about/Segment_Routing.html?id=1zMyMQAACAAJ

[8] C. Filsfils, S. Previdi, B. Decraene, and R. Shakir, *Resiliency Use Cases in Source Packet Routing in Networking (SPRING) Networks*, document RFC 8355, 2018.

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

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

[11] B. Fu and S. Uhlig, “On the relevance of on-line traffic engineering,” in *Proc. ITC Spec. Seminar Netw. Usage Traffic*, 2008, pp. 1–15.

[12] S. Gay, R. Hartert, and S. Vissicchio, “Expect the unexpected: Sub-second optimization for segment routing,” in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2017, pp. 1–9.

[13] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, “Solving segment routing problems with hybrid constraint programming techniques,” in *Proc. Int. Conf. Princ. Pract. Constraint Program. (CP)*, 2015, pp. 592–608.

[14] G. Hasslinger, S. Schnitter, and M. Franzke, “The efficiency of traffic engineering with regard to link failure resilience,” *Telecommun. Syst.*, vol. 29, no. 2, pp. 109–130, Jun. 2005.

[15] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, “Analysis of link failures in an IP backbone,” in *Proc. 2nd ACM SIGCOMM Workshop Internet Measurment (IMW)*, 2002, pp. 237–242.

[16] (2014). *IBM ILOG CPLEX Optimization Studio V12.6.2*. [Online]. Available: http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.2

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

[18] Juniper. (2013). *Junos OS Administration Library for Routing Devices: Maximum-ECMP*. [Online]. Available: http://www.juniper.net/documentation/en_US/junos15.1/topics/reference/configuration-statement/maximum-ecmp-edit-chassis.html

[19] P. Kumar *et al.*, "Semi-oblivious traffic engineering: The road not taken," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2018, pp. 157–170.

[20] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, Aug. 2008.

[21] T. Schuller, N. Aschenbruck, M. Chimani, and M. Horneffer, "On the practical irrelevance of metrics on segment routing traffic engineering optimization," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, Oct. 2018, pp. 640–647.

[22] T. Schuller, N. Aschenbruck, M. Chimani, and M. Horneffer, "Failure resilient traffic engineering using segment routing," in *Proc. IEEE 44th Conf. Local Comput. Netw. (LCN)*, Oct. 2019, pp. 625–632.

[23] T. Schuller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Predictive traffic engineering with 2-Segment routing considering requirements of a carrier IP network," in *Proc. IEEE 42nd Conf. Local Comput. Netw. (LCN)*, Oct. 2017, pp. 667–675.

[24] T. Schuller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic engineering using segment routing and considering requirements of a carrier IP network," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1851–1864, Aug. 2018.

[25] T. Schüller, *Real-world Pitfalls of Segment Routing Traffic Engineering*, RACI contribution at RIPE, document 78, 2019. [Online]. Available: https://ripe78.ripe.net/archives/video/39/

[26] S. Uhlig and B. Quoitin, "Tweak-it: BGP-based interdomain traffic engineering for transit ASs," in *Proc. Next Gener. Internet Netw.*, Apr. 2005, pp. 75–82.

Timmy Schüller received the master's degree in computer science in 2015 and the Ph.D. degree from the University of Osnabrück, Germany. From 2015 to 2019, he was working in a joint project with Detecon International GmbH and Deutsche Telekom Technik GmbH. Since then, he has been working as a Systems Engineer with Deutsche Telekom Technik GmbH. As such, he works towards developing and deploying next-gen traffic engineering strategies in a global IP backbone network.

Nils Aschenbruck (Member, IEEE) received the graduate diploma and Ph.D. degrees in computer science from the University of Bonn, Germany, in 2003 and 2008, respectively. He continued as a Senior Researcher and the Head of the research area tactical wireless multi-hop networks at the Communication Systems Group, University of Bonn. Since 2012, he has been a Full Professorship for distributed systems with the University of Osnabrück. His research interests include mobile and wireless networks, security, and scenario modeling.

Markus Chimani received the diploma degree in computer science from TU Vienna in 2004 and the Ph.D. degree from TU Dortmund in 2008. From 2010 to 2013, he was a Junior Professor of algorithm engineering with University of Jena. Since 2013, he has been a Professor and the Head of the Theoretical Computer Science group, University of Osnabrück. His research interests include exact solutions for NP-hard problems, graph drawing and graph theory, i.p., non-planarity measures as well as topological network optimization, combinatorial optimization, and algorithm engineering.

Martin Horneffer studied electrical engineering at RWTH Aachen University until 1995. He received the Ph.D. degree in computer science from the University of Cologne in 2000. He currently leads the squad for the development of IP Backbone Network Design and Architecture at Deutsche Telekom Technik GmbH. As such, he is also the Lead Architect and a Network Designer for a global IP/MPLS core network comprising five autonomous systems and formerly separate core networks. He works toward evolving the network architecture by switching to segment routing and thinking multilayer.