Optimizing IGP Link Costs for Improving IP-level Resilience with Loop-Free Alternates

Levente Csikor*, János Tapolcai, Gábor Rétvári

HSNLab, Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics

Abstract

The IP Fast ReRoute–Loop-Free Alternates (LFA) standard is a simple and easily deployable technique to provide fast failure protection right in the IP layer. To our days, most major IP device vendors have products on the market that support LFA out of the box. Unfortunately, LFA usually cannot protect all possible failure scenarios in a general network topology. Therefore, it is crucial to develop LFA-based network optimization tools in order to assist operators in deciding whether deploying LFA in their network will supply sufficient resiliency. In this paper, we give a new graph theoretical framework for analyzing LFA failure case coverage, and then we investigate how to optimize the Interior Gateway Protocol (IGP) link costs in order to maximize the number of protected failure scenarios. We show that this problem is NP-complete even in a very restricted formulation, and we give an exact algorithm as well as a complete family of heuristics to solve it. Our simulation studies indicate that a deliberate tuning of the approximation strategy can significantly improve the quality of the IGP link costs, and we conclude that LFA cost optimization has the potential for boosting LFA-based resilience in most operational networks significantly.

Keywords: IP Fast ReRoute, Loop-Free Alternates, resilience, network optimization

*Corresponding author

Email addresses: csikor@tmit.bme.hu (Levente Csikor), tapolcai@tmit.bme.hu (János Tapolcai), retvari@tmit.bme.hu (Gábor Rétvári)

1. Introduction

To our days, the Internet Protocol (IP) suite has become the de-facto standard for large-scale inter-networking throughout the world. The protocol suite, with its accompanying control plane protocols, has come a long way to become a viable bearing platform for commercial telecom services. Unfortunately, there still exists missing functionality in IP that make it difficult to sustain the transmission quality required by multimedia applications, like VoIP, IPTV, online gaming, etc., in an IP environment. One of the most prominent shortcomings in existence today is the slow reaction to device and link failures. Interior Gateway Protocols (IGPs), like the Open Shortest Path First (OSPF,[1]) or the Integrated IS-IS (IS-IS, [2]) routing protocol, adopt a restoration-based resilience approach, based on a global flooding of failure information and a lengthy network-wide re-convergence process. This slow reaction to failures, inherent to the traditional IP control plane, does not only hinder operators providing telecom services over pure IP, but a growing number of service providers that switched to MultiProtocol Label Switching–Label Distribution Protocol (MPLS/LDP) also suffer, because MPLS-LDP also relies on the IP control plane for routing information.

The key to the slow convergence of IGPs is the *global, reactive response* philosophy they adopt: failure information is distributed to all routers in the administrative scope, which in turn react by recomputing their routing tables and refreshing their forwarding information bases in accordance with the changed network topology. This often leads to convergence time in the range of couple of hundreds of milliseconds to several seconds, and even a very careful adjustment of the IGP parameters [3] is insufficient to decrease this to less than 50 milliseconds, usually used as a rough estimate on the longest outage a modern multimedia application can tolerate.

In order to achieve a sub-50 ms convergence time, one needs to go beyond conventional IGP-based *restoration* and invoke a *proactive, local protection* method, called IP Fast ReRoute (IPFRR, [4]). In IPFRR, routers precompute alternate next-hops proactively, and traffic is instantly switched to these secondary next-hops should the primary next-hop become unavailable. This ensures that traffic flows without interruption until the IGP converges in the background. Note that in IPFRR only the routers in the immediate vicinity of the failed component participate in the failure recovery process, and routers several hops away do not even get notified about the outage. This saves the time needed for global failure notification, one of the most

time-consuming steps in IGP-based restoration.

It turned out, however, that combining local protection with IP's intrinsic destination-based forwarding scheme is notoriously difficult. This is because a router not immediately adjacent to the failure, not knowing that a failure in fact has occurred, has no way to decide whether a received packet is traveling on its default shortest path to the destination, or it is actually being routed around a failure and so out-of-order forwarding rules should be applied to it. Any IPFRR mechanism, therefore, that does not adopt special remedies to this problem, is prone to either producing micro-loops or being unable to handle certain failure cases [5]. To avoid this, IPFRR proposals either apply explicit or implicit failure signaling [6, 7, 8], or alter IP's destination-based forwarding [9], or introduce tunnels to route around the failed component [10, 11, 12]. Deploying these IPFRR mechanisms, however, would either demand non-trivial modifications to the essential IP infrastructure or impose considerable management burden on network operations [13] (or both), making network device vendors reluctant to implement them and discouraging operators from deploying IPFRR all together.

To our days, only a single IPFRR specification has found its way into commercial IP routers: Loop-Free Alternates (LFA, [14]). This can be attributed to the fact that LFA is a clever trade-off between simplicity and protection-capability, in that LFA has never been intended to provide 100% protection for all possible failure cases because, as we argued above, this would require widespread modifications to the IP infrastructure and so would hinder deployment. Instead, LFA is as simple as it can get: traffic impacted by a failure is passed on to an alternate next-hop (called a Loop Free Alternate) that still has an intact path to the destination. When the aim is merely to protect against link outages then it is enough to ensure that the detour bypasses the link to the next-hop, while for node-protecting LFAs it is a requirement to avoid both the link to the next-hop *and* the next-hop itself. LFA can be implemented with a straightforward upgrade to IGPs, without special staff-training and extensive pilot deployments, and so it can be introduced incrementally. On the other hand, as the price of this simplicity, depending on the network topology and IGP link costs very often not all routers have LFAs to all destinations, making it impossible to repair certain failure scenarios rapidly with LFA.

Consequently, many operators are hesitating to enable LFA, trying to measure the expected benefits against the additional costs. In this paper, we seek ways to assist in making this important decision. In the first part,

we give new graph theoretical tools for analyzing LFA failure case coverage in operational networks. Similar protectability analyses are already available for some non-standardized IPFRR mechanisms: [15] considers the O2 method and [16] discusses a centralized destination-based routing scheme. For LFA, only simulation-based reports have been available this far [17, 18, 19, 20], and mathematical analysis has been confined to the link-protection case [21, 22]. Below, we extend previous work on mathematical LFA-coverage analysis with new tools for studying both the link- and node-protection cases as well.

Initial deployments as well as numerical analyses confirmed that in many operational networks LFA indeed does not guarantee protection for all failure scenarios [19]. This calls for developing network optimization tools to tune the network topology in a way as to increase the number of failure cases protectable by LFA. There are various approaches to reach this end. One way is *LFA network design*, which aims to design LFA-friendly network topologies right from the outset [20]. Another approach is *LFA graph extension*, where the task is to alter the network topology to boost LFA coverage [21]). Third, *LFA cost optimization* asks to construct IGP link costs in a way as to maximize the number of possible failure cases protectable by LFA [23, 24, 22]. This LFA cost optimization problem is in the main focus in the second part of this paper. While improving IP resilience is a recurring theme in the literature (see [25] for deflection routing, [15] for O2, or [16] for a review), for the specific case of LFA only the joint optimization of network performance and resilience has been investigated previously [23, 24]. Thus, at the moment very little understanding is available as to how much LFA-based IP Fast ReRoute is suitable to protect an IP network and to what extent this can be improved by optimizing link costs.

The main contributions in this paper are as follows.

- We develop a comprehensive graph theoretical LFA analysis framework, for the first time considering both the link-protection and node-protection cases.

- We study the LFA cost optimization problem in huge detail. We show that this problem is NP-complete, and we give an exact algorithm of exponential complexity as well as a family of heuristics with tunable performance and running time. Our selection of heuristics facilitate for picking the right approximation algorithm for the particular problem under consideration.

- We provide a comprehensive numerical evaluation of LFA cost optimization methods to compare their performance on a wide range of artificial and realistic graph topologies.

The rest of this paper is organized as follows. After reviewing the related literature in Section 2 and introducing the notations and the model in Section 3, we first discuss LFA failure coverage analysis (see Section 4). Then, in Section 5 we turn to discuss the LFA cost optimization problem. In Section 6, we evaluate the proposed algorithms numerically and finally we conclude our work with Section 7.

2. Related works

The IP Fast ReRoute framework was initiated by the Internet Engineering Task Force in [4], and the Loop-Free Alternates standard, as the basic specification for IPFRR, was subsequently documented in [14]. It was from the very beginning made clear by the IETF that LFA does not guarantee fast protection for all possible failure scenarios in all network topologies. This was later confirmed by extensive simulation studies, which indicated that, depending on the topology and link cost settings, LFA can usually protect only about 50-80% of the possible link failure scenarios, and the level of node protection is even worse [17, 18, 19, 26]. These LFA coverage analyses are all *quantitative* studies, based on calculating the LFA coverage for various real-life network topologies. Perhaps the most detailed amongst these is [20], which inspects the applicability of LFA in common access network topologies. So far, no *qualitative* analyses have been available in the literature, which would help uncover the graph theoretical ingredients needed for good LFA coverage. We initiated the work in that direction in [21] and [22], and in this paper we refine our earlier results and generalize them to the node-protection case as well. Possibly the closest to ours is the study in [16], where the authors perform a qualitative protectability analysis for a fast resilience scheme they call IP protection routing. Protection routing is appealing for such an analysis in that it is much easier to approach theoretically than LFA, however, in practice it is somewhat less attractive as implementing it requires centralized control over the routing tables.

Since the appearance of the original LFA draft, countless IPFRR proposals have surfaced. Implicit in these proposals is the recognition that in order to protect all failure scenarios one either needs to go beyond standard

IP forwarding and/or apply some forms of failure notification. The reason for this is that a router must give special treatment to packets traveling on a detour around a failure, or otherwise forwarding loops will arise in certain failure scenarios.

Most IPFRR proposals choose the former option and intervene at the level of IP packet forwarding. Failure Insensitive Routing [9, 27, 5] differentiates packets based on the incoming interface they arrive through, letting the router to guess the failure's location from the direction of the received packets and exploit this information in the course of packet forwarding. Multiple Routing Configurations [8] call to achieve the same goal with explicit packet marking, while other proposals, like Not-via Addresses, use tunnels to this end [10, 11, 12, 13]. Unfortunately, the former solution would allocate invaluable bits in the IP header, while the latter might cause painful packet fragmentation and time-consuming reassembly at the tunnel endpoint if the additional IP header did not fit into the MTU. Deflection routing for fast rerouting purposes is proposed in [25], while O2 routing, a resilient multipath data forwarding method, is specified in [28]. Both require non-standard IP forwarding functionality, unavailable in commercial routers at the moment.

A different approach is to use explicit signaling to notify routers about failures [6, 29]. This avoids having to modify standard IP forwarding at the price of a establishing a separate signaling mechanism just for IPFRR. Proposals also exist to combine different IPFRR mechanisms to achieve full protection [26]. Good overviews on IPFRR are [17] and [19].

So far, only one IPFRR method has found its way into commercial routers, and hence into operational IP networks: LFA. At least two major vendors are already providing LFA out of the box [30, 31], and other vendors are expected to follow suit.

Finding methods to design or optimize networks in an attempt to improve fast resiliency has been an actively researched topic lately. In the recent literature, [25] seems to be the first reference that, besides motivating the need for fast IP resilience with detailed failure case analysis in an operational backbone, proposes a method to improve the robustness of the network against such failures. Theory and algorithms for topology optimization for O2 are presented in [15], and a generic approach for protection routing is given in [16]. Apart from our studies in [21] and [22], the only attempts at LFA-oriented network optimization seem to be [23] and (partly) [24].

A common theme shared by most approaches is that (with the exception

of [15]) each one addresses the optimization of network resilience and routing performance simultaneously. The former aims at better protection against failures, while the latter is called to minimize congestion and distribute load evenly in the network with respect to some known, measured or predicted, traffic matrix [32]. A good example of this approach is [23], where the authors formulate the joint LFA cost optimization and traffic engineering problem as a constraint-programming task and feed it into a generic solver.

In this paper, we study LFA cost optimization, that is, the task of improving LFA coverage by tuning IGP link costs, separately from load balancing. Our work, in this regard, is complementary to the above joint optimization frameworks and, as shall be shown, provides interesting further insights. For instance, we find that LFA cost optimization alone, even in a very minimalistic setting, is already NP-complete. This far, only NP-completeness for the joint optimization problem was known [23], but this could have easily been attributed to the well-known NP-completeness of OSPF traffic engineering [33]. Our approach also allows to investigate the inherent limitations of LFA-based IP Fast ReRoute, without the distortion of load balancing concerns, and the extent to which optimizing costs just for the purpose of IPFRR can improve the resilience in IP networks.

We acknowledge, however, that applicability of our LFA cost optimization methods might be limited due to the lack of built-in load balancing criteria. We note, though, that we are aware of many operators that weigh certain operational concerns more important than load balancing, and even if load balancing is a must, the heuristics we present in the paper are easy to augment to consider such issues.

3. Model and problem formulation

We model the network with a connected, undirected graph $G(V, E)$, the set of nodes is denoted by V ($|V| = n$) and set of edges by E ($|E| = m$). Let N_i denote the set of neighbors of some node $i \in V$. IGP link costs are represented by an edge cost function $c : E \mapsto \mathbb{N}$. The cost of an edge (i, j) is denoted by $c(i, j)$. We presume that the network topology $G(V, E)$ and the cost function c are readily available to the network nodes through the IGP, using which all routers can compute the shortest path distance between any two routers in the network. Denote the distance from node i to node j with $\mathrm{dist}(i, j)$.

For a list of notations, consult Table 1.

Table 1: List of notations

$G(V, E)$	connected undirected graph, with node set V and edge set E
N_i	set of neighbors of some node i
c	edge cost function, $c : E \mapsto \mathbb{N}$
$c(i, j)$	cost of an edge between node i and node j
$\mathrm{dist}(i, j)$	shortest path distance from node i to node j
$\eta(G, c)$	LFA failure case coverage for graph G over link costs c
$\eta^{\mathrm{LP}}(G, c)$	link-protecting LFA coverage
$\eta^{\mathrm{NP}}(G, c)$	node-protecting LFA coverage
$\mathcal{S}$	set of node-pairs, $\mathcal{S} \subseteq V \times V$
Δ	average node degree
$\Delta_{\max}$	maximum node degree

In this paper, we make the following key assumptions about the network:

- the network consists of point-to-point links only and there are no broadcast LANs;
- link costs are symmetric;
- if multiple shortest paths towards a destination exist, each node fixes one default path arbitrarily (no Equal-Cost MultiPath); and
- failure events are independent and singular, so no Shared Risk Link Groups (SRLGs) and multiple failures are taken into consideration.

Note that these assumptions are easy to eliminate, but the development and the notation become substantially more complex.

3.1. Loop-Free Alternates

Fig. 1 shows a sample network, with costs indicated near the edges and shortest paths towards node f marked by arrows. For instance, node b's next-hop along the shortest path to node f is node e. Should the link from b to its next-hop e become unavailable, b can safely switch to an alternate next-hop, in this case node d, even without explicitly notifying it about the failure, as d will never send packets destined to f through b so no loop can

arise. In such cases, we say that d is a *link-protecting LFA* for node b towards the destination node f [14].

Definition 1. *For some source s and destination d, let e be the default next-hop of s towards d. Then, some neighbor n of s is a* link-protecting LFA *for s to d if*

(i) $n \neq e$, *and*

(ii) *the loop-free condition applies:*

$$\text{dist}(n, d) < \text{dist}(n, s) + \text{dist}(s, d) \ . \tag{1}$$

In other words, any neighbor that is not an upstream in the shortest path tree is a link-protecting LFA. Besides node b, e also has an LFA to f (the same d as that of b), and so has d and c (e and d, respectively). What is more, the LFAs of b and c are node-protecting as well, as the shortest path from the LFA to d does not traverse the default next-hop and so it protects against both the failure of the link to the next-hop and the next-hop itself. The below definition formalizes this requirement.

Definition 2. *For some source s and destination d, let e be the default next-hop of s towards d. Then, some neighbor n of s is a* node-protecting LFA *for s to d if, in addition to* (i) *and* (ii) *in Definition 1, the node-protection condition also applies:*

$$\text{dist}(n, d) < \text{dist}(n, e) + \text{dist}(e, d) \ . \tag{2}$$

Continuing with our enumeration of LFA-types, we find that d is also a so called per-link LFA for b, as it protects all nodes reachable from b through the link (b, e). For a full taxonomy, see [14, 20].

We observe that, in the present network topology with the given link costs, node a does not have an LFA to f. This is because it has only two neighbors, one is the next-hop d towards f whose failure we want to protect, and the other is an upstream node, which cannot provide an LFA by (1). Given a graph $G(V, E)$ and a cost function c, let $I^{\text{LP}}_{s,d}(G, c)$ be an indicator variable whose value is 1 if node s has a link-protecting LFA to node d, and zero otherwise. Then, given a set of source-destination pairs $\mathcal{S} = \{(s_k, d_k) :$

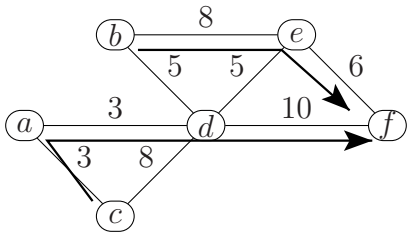


Figure 1: Sample network, edge costs and shortest paths to node f.

$k \in 1, \ldots, K, s_k \neq d_k\}$ the *link-protecting LFA coverage with respect to* $\mathcal{S}$ is defined as (inspired by [14]):

$$\eta_{\mathcal{S}}^{\mathrm{LP}}(G, c) = \frac{1}{|\mathcal{S}|} \sum_{(s,d)\in\mathcal{S}} I_{s,d}^{\mathrm{LP}}(G, c) \ . \tag{3}$$

Similarly, let $I_{s,d}^{\mathrm{NP}}(G, c)$ be an indicator variable for node-protecting LFA coverage. Note, however, that special care must be taken to handle the so called *last-hop problem*, which arises when d is an immediate neighbor of s and the default shortest path between them is exactly the (s, d) link (see for instance the case of the (e, f) pair in Fig. 1). In such cases, the node failure we want to protect is exactly the failure of destination d itself, a failure case hardly protectable by LFA. Therefore, for such (s, d) pairs we only require that the link (s, d) be protected by a link-protecting LFA, and we ignore the stronger node-protection requirement (2). Consequently, $I_{s,d}^{\mathrm{NP}}(G, c)$ takes the value 1 if and only if

(i) d is not the immediate next-hop of s to d and s has a node-protecting LFA to d, or

(ii) d is the immediate next-hop of s to d and s has a link-protecting LFA to d.

Then, the node-protecting LFA coverage is defined as

$$\eta_{\mathcal{S}}^{\mathrm{NP}}(G, c) = \frac{1}{|\mathcal{S}|} \sum_{(s,d)\in\mathcal{S}} I_{s,d}^{\mathrm{NP}}(G, c) \ . \tag{4}$$

In some cases, it will be convenient to refer to both link-protecting and node-protecting LFAs under a common term. In such cases, we shall only

say "an LFA exists" and the corresponding coverage metric will be written as $\eta_{\mathcal{S}}(G, c)$. Moreover, we shall often confine ourselves to the special cases when $\mathcal{S}$ is the set of all node pairs whose destination is a given terminal node d: $\mathcal{S}_d = \{(s, d) : s \in V \setminus \{d\}\}$, or when $\mathcal{S}$ contains all distinct node pairs in $V \times V$. In the latter case, we shall neglect to indicate $\mathcal{S}$ in the LFA coverage metric and simply write $\eta^{\mathrm{LP}}(G, c)$, $\eta^{\mathrm{NP}}(G, c)$, and we shall use the shorthand notation $\eta(G, c)$ when LFA type does not matter.

3.2. LFA cost optimization: problem formulation

As our example shows, usually not all nodes have LFA to all destinations. There are basically two ways to remedy this: by adding new edges to the graph or by altering the edge costs. Taken the example of Fig. 1, adding the new edge (a, b) to E and setting its cost to, say, 10, will let b to become an LFA of a (and vice versa). The *LFA graph extension* problem asks to achieve maximal LFA protection by adding the minimum number of new edges. We address this problem in a separate paper [21]. The other way is to change edge costs: if we, for instance, reduce the cost of edge (c, d) from 8 to 5, then c's shortest path to f will bypass a and so a and c will become (link-protecting) LFAs to each other. This paper is devoted to investigate this very problem. Formally, we define the *LFA cost optimization problem for the link-protecting case* as follows:

Definition 3. *LFACostOptLP(G, $\mathcal{S}$): Given a graph $G(V, E)$ and a set of source-destination pairs $\mathcal{S}$, is there a cost function c so that $\eta_{\mathcal{S}}^{LP}(G, c) = 1$?*

Easily, a similar problem formulation LFACostOptNP exists for the node-protecting case as well. When no ambiguity arises, we shall refer to both problems simply as LFACostOpt. In addition, we shall in many cases treat the optimization version of these problems, that is, we shall seek the costs that maximize network-wide LFA coverage instead of merely asking whether or not a cost setting for full protection exists.

4. LFA failure coverage analysis

Before turning to discuss how to solve the LFA cost optimization problem, first we show some simple theoretical limits on LFA coverage. In particular, we give tight graph theoretical lower and upper bounds on the LFA coverage achievable in a given graph under *any* selection of link costs. We shall discuss both the link-protecting and the node-protecting cases.

Our analysis is intended to serve for operators to quickly assess the benefits LFA-based fast protection can bring in their network as well as the inherent limitations thereof. In addition, the lower and upper bounds provide rule-of-thumb guidance on how "good" the actual selection of IGP link costs is from the aspect of LFA: a cost setting that realizes the theoretical upper bound is considered favorable, while a cost setting that yields LFA coverage close to the lower bound is considered poor for LFA. In the latter case, LFA cost optimization might be especially beneficial. Last but not least, the theoretical analysis reveals an interesting relation between LFA coverage and an essential characteristic of the underlying graph topology, the average node degree. As shall be seen, a sparse graph (one with average node degree less than about 2.5) usually does not admit good LFA coverage, no matter which link costs are chosen. If an operator finds his network falls under this characterization, then this may be a good indicator that LFA cost optimization will most probably not bring significant improvements in LFA coverage. In such cases, the operator should resort to alternative LFA network optimization methodologies [21] or alternative fast protection mechanisms [34].

Some preliminaries. In what follows, we assume that $\mathcal{S} = (V \times V) \backslash \{(v, v) : v \in V\}$. Let Δ denote the average node degree in G and let $\Delta_{\max}$ be the maximum degree. Easily, $\Delta \geq \frac{2(n-1)}{n}$ for any connected graph, since the sparsest connected graphs are trees for which $\Delta = \frac{2(n-1)}{n}$. A Δ-regular graph is a graph in which all nodes are of constant degree Δ. An even (odd) ring is a cycle graph with an even (odd) number of nodes. Rings are the smallest-degree 2-connected regular graphs (in particular, $\Delta = 2$).

First, we extend the analysis on even and odd rings given in [21] to the node-protecting case.

Lemma 1. *For an even ring with $n > 2$ and uniform costs: $\eta^{LP}(G, c) = \eta^{NP}(G, c) = \frac{1}{n-1}$. For an odd ring with $n > 2$ and uniform costs: $\eta^{LP}(G, c) = \eta^{NP}(G, c) = \frac{2}{n-1}$*

Proof. Consider a ring C_n of $n > 2$ nodes and n even, let costs be uniform, and let d be some node in C_n. Then, any node $s \neq d$ has exactly one shortest path to d, except the node on the opposite of d, which has two. One easily sees that this is the only node that has LFA to d in C_n (which is both link- and node-protecting), because for any other node the only possible alternative is upstream and so cannot be LFA by (1). Due to symmetry, the derivation holds for each node and so exactly n nodes have LFA, which gives

$\eta(G,c) = \frac{1}{n-1}$. The development is similar for odd rings, but now we have two nodes per destination that have LFA instead of one. □

Next, we present simple lower and upper bounds on LFA coverage. Our bounds are based on the following idea. The shortest path tree to some destination d can contain only $n-1$ edges, and all the remaining edges can be used for providing LFAs to their endpoints. In particular, an out-of-tree edge provides at most 2 LFAs (either node-protecting or link-protecting or both), and at least 1 link-protecting LFA towards d.

Consider the following lemma.

Lemma 2. *For any connected simple graph G with $n > 2$:*

$$\eta^{NP}(G,c) \leq \eta^{LP}(G,c) \leq \frac{n}{n-1}(\Delta - 2) + \frac{2}{n-1} \ .$$

PROOF. First, $\eta^{\mathrm{NP}}(G,c) \leq \eta^{\mathrm{LP}}(G,c)$ is trivial[1] from Definition 1 and 2. To prove the second part of the claim, we observe that an edge not contained in the shortest path tree rooted at some d provides at most 2 link-protecting LFAs towards d (when the edge lies between two branches of the tree), while on-tree edges do not create any LFA. Since the number of out-of-tree edges is exactly $m-(n-1)$, at most $2(m-n+1) = n\Delta - 2n + 2 = n(\Delta-2)+2$ nodes can have LFA to d. Taken the sum over all nodes and dividing by the number of source-destination pairs gives $\eta(G,c) \leq \frac{n(n(\Delta-2)+2)}{n(n-1)} = \frac{n}{n-1}(\Delta-2) + \frac{2}{n-1}$. □

The Lemma is non-trivial for $\frac{2(n-1)}{n} \leq \Delta < 3$. For trees, in particular, we obtain $\eta(G,c) \leq 0$, which implies that the Lemma is tight for trees over arbitrary link costs. It is tight for uniform cost odd rings as well, for which we obtain $\eta(G,c) \leq \frac{2}{n-1}$ (c.f., Lemma 1).

What the above Lemma in essence says is that in large sparse graphs LFA-coverage is upper bounded by the average node degree: $\eta(G,c) \leq \Delta - 2$. In the course of our numerical evaluations, we found that this relation is present in most real-world networks as well (see later).

The next Lemma gives a lower bound on the LFA coverage. Note, however, that the result concerns link-protecting LFAs exclusively.

[1]Note that this is only true when all links in the network are point-to-point. When the network contains broadcast LANs, the relation becomes dimmer [14].

Lemma 3. *For any connected simple graph G with $n > 2$:*

$$\eta^{LP}(G,c) \geq \frac{n}{n-1}\frac{\frac{\Delta}{2}-1}{\Delta_{\max}-1} + \frac{1}{(n-1)(\Delta_{\max}-1)} \ .$$

PROOF. Again, exactly $n-1$ nodes are contained in the shortest path tree of d, and an out-of-tree edge (of which we have $m-n+1$) can provide at least one LFA towards d: *(i)* if the edge is inside a single branch of the shortest path tree, then it provides LFA from the upstream to the downstream; *(ii)* if the edge lies between two branches, it still creates at least one link-protecting LFA (in fact, it creates two), but it might not create any node-protecting LFA at all. In consequence, there are $m-n+1$ out-of-tree edges that are incident to at least $\frac{m-n+1}{\Delta_{\max}-1} = \frac{n(\frac{\Delta}{2}-1)+1}{\Delta_{\max}-1}$ nodes providing a link-protecting LFA towards d (the term $\Delta_{\max}-1$ is because every node has at least one in-tree edge, so only the rest count as out-of-tree edges). Taking the sum over all nodes and dividing by $n(n-1)$ gives the required result. □

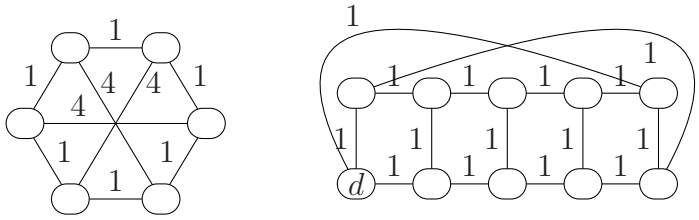
The most important message here is that LFA coverage increases with Δ, that is, the denser the network the higher the link-protecting LFA coverage.

Corollary 1. *For a Δ-regular graph R_Δ on n nodes:*

$$\eta^{LP}(R_\Delta,c) \geq \frac{1}{2} - \frac{1}{2}\frac{n-\Delta-1}{(n-1)(\Delta-1)} \ .$$

This gives $\eta^{\mathrm{LP}}(R_2,c) \geq \frac{1}{n-1}$ and $\eta^{\mathrm{LP}}(R_3,c) \geq \frac{1}{4} + \frac{3}{4}\frac{1}{n-1} > \frac{1}{4}$. From this, we conclude that the lower bound of Lemma 3 is tight for even rings (again, by Lemma 1). One easily sees that it is tight for trees as well.

We have seen that LFA coverage fundamentally depends on the average node degree. This raises the question whether we can find graphs of low degree with 100% LFA coverage. We found that the 2-connected graph with the smallest possible average degree that can still be fully protected using LFA is the 3-ring C_3. Every other 2-connected graph with complete LFA coverage has average degree higher than 2. By Lemma 1, $\eta(C_3,c) = 1$ which is attained when c is uniform, and one easily sees that $\eta(C_3,c)$ is the only 2-connected graph of average degree $\Delta = 2$ with this property. Graphs with $\Delta < 2$ cannot have full protection because such graphs contain at least one node with degree 1 whose single outgoing link can never be protected. On the other hand, larger 2-connected graphs with $\Delta = 2$ are all ring topologies, and rings can only have full LFA coverage if $n = 3$ (again, by Lemma 1).



(a) $\eta(G, c) = 1$ (b) $\eta(G, c) < \frac{1}{2}$

Figure 2: Möbius ladder topologies.

5. LFA cost optimization

Next, we turn to the LFA cost optimization problem. This problem asks for an IGP link cost setting that maximizes the LFA coverage, given the inherent limitations of the network topology under consideration. First, we characterize the extent to which such an optimization can improve LFA coverage, then we discuss the complexity and the algorithmic aspects of the problem. Most of the observations apply to LFAs generally, without regard to link-protection or node-protection, so, unless otherwise stated, the term LFA will refer to link-protecting LFAs in the sequel. We shall indicate clearly in the text when LFA types indeed matter.

5.1. The potential of LFA cost optimization

The question immediately arises as to whether it is worth optimizing costs for LFA at all. Easily, readjusting costs in most of the cases alters, possibly in a negative way, default shortest paths, which might have been previously tweaked with great accuracy to match the needs of the network in terms of load balancing, traffic engineering, etc. [32, 35, 36]. On the other hand, as shall be shown through an example below, the wins achievable with optimizing link costs for LFA can be substantial (more than 50%), and such a huge improvement in fast resiliency might compensate for the losses in forwarding efficiency in certain cases.

Consider the so called "Möbius ladder" topologies depicted in Fig. 2. These graphs consist of an even ring with all the main diagonals added. In Fig. 2a, the cost of diagonals is chosen so that the path between any two nodes is shorter around the ring than through it via a diagonal. This way, as one easily checks, the graph has complete LFA coverage, both in terms of link-protection and node-protection. The graph construction can

be generalized to arbitrary even n, and one can always choose the above cost setting strategy to achieve complete LFA protection. Fig. 2b also depicts a Möbius ladder (for $n = 10$), just with setting costs uniformly at all edges and drawn in a slightly awkward layout. The layout was chosen so that one can easily check the validity of the following claim for any Möbius ladder with $\frac{n}{2}$ odd, $n > 2$ and c uniform: for every $d \in V$, exactly $\frac{n}{2} - 1$ nodes have link-protecting and node-protecting LFA to d. Considering the node d we marked in Fig. 2b, there is exactly one node in each "column" that has an LFA to d, except for the column of d in which there is no protected node. This gives $\eta(G, c) = \frac{1}{2} - \frac{1}{2}\frac{1}{n-1} < \frac{1}{2}$, again, in terms of both link-protection and node-protection. For instance, in our example $\eta(G, c) = \frac{4}{9}$.

This example shows that different selections of edge costs can produce dramatical differences in LFA failure case coverage. Simulation studies presented later also seem to support this claim. The other lesson is that resilience and forwarding efficiency are usually contradicting requirements in routing: in our example in the latter case all traffic flows along min-hop paths but resilience is poor, while in the former case we have full protection but long forwarding paths going around the ring instead of taking the shortcuts through it. Such "joker" links that do not carry traffic seem a general requirement for protectability [15].

5.2. Complexity

Next, we turn to discuss how to solve the LFA cost optimization problem as of Definition 3. First, we characterize the computational complexity of the problem.

Theorem 1. *The LFA cost optimization problems LFACostOptLP(G, $\mathcal{S}$) and LFACostOptNP(G, $\mathcal{S}$) are NP-complete.*

This result is not particularly unexpected, as we found basically all other LFA-related network optimization problems NP-complete as well [21]. Taking a closer look, we find that there are two reasons due to which the problem is difficult. First, there is an inherent coupling between the LFAs to different destinations through the link costs, which makes it difficult to take independent decisions. In particular, assigning a neighbor as an LFA towards some destination necessitates adjusting edge costs accordingly, but this may destroy LFAs to other destinations. Second, even assigning LFAs to *just a single destination* seems difficult enough. Consider the following theorem characterizing the difficulty of the node-protecting case.

Theorem 2. *Given a graph $G(V, E)$ and a node $d \in V$, LFACostOptNP(G, $\mathcal{S}_d$) with $\mathcal{S}_d = \{(s, d) : s \in V \setminus \{d\}\}$ is NP-complete.*

PROOF. Easily, LFACostOptNP(G, $\mathcal{S}_d$) is in NP. To prove NP-hardness, we show that it is essentially equivalent to the protection routing problem, proved to be NP-complete in [16].

Definition 4. *PR(G, d): given a graph $G(V, E)$ and some $d \in V$, is there a directed spanning DAG $R_d(V, E_d) : E_d \subseteq E$ rooted at d, so that for any single node or link failure f every node $s \in V \setminus \{d\}$ has a neighbor $k : (s, k) \notin E_d$ for which it holds that* (i) *k is not upstream of s in R_d^f, and* (ii) *there is a $k \to d$ path in R_d^f, where R_d^f is obtained from R_d by removing the failed component f.*

The basic differences are that *(a)* LFACostOpt(G, $\mathcal{S}_d$) is defined in terms of costs, while PR(G, d) in terms of a routing DAG R_d, and *(b)* item *(ii)* in the above definition. To show equivalence, we need to handle these differences.

First, we show that a cost function c uniquely determines R_d and vice versa, in that we can show a mapping from c to R_d so that a path is shortest path over c if and only if it is contained in R_d (this will handle *(a)*). Easily, the shortest paths over c are always in a DAG. The reverse direction, that is, taking R_d and creating a cost c of it, is equally easy: take a topological ordering $o(v) : v \in V$ of R_d (this always exists) and for each $(i, j) \in E$ set $c(i, j) = o(j) - o(i)$ if $(i, j) \in E_d$ and $c(i, j) = n$ otherwise.

Second, *(b)* means that in PR(G, d) we only take a node for protected, if after a failure f all the paths of the the secondary next-hop in R_d avoid f. However, this is guaranteed by the node-protecting condition (2). □

A similar result can be shown for the link-protecting case as well. Below, we only state the result but we do not give a detailed proof. We only note that the proof involves observing that the NP-completeness argumentation in [16] remains valid if we treat link failures only and disregard node failures, from which the derivation is straight forward.

Theorem 3. *Given a graph $G(V, E)$ and a node $d \in V$, LFACostOptLP(G, $\mathcal{S}_d$) is NP-complete.*

Obviously, Theorem 2 and Theorem 3 prove Theorem 1 stated for the general case, of which LFACostOpt(G, $\mathcal{S}_d$) is a special case. Additionally, we also observe that the optimization version, which asks for a cost maximizing LFA coverage, is also intractable.

5.3. An exact algorithm

LFA cost optimization is difficult, yet solving it would be extremely useful for improving the resilience in operational IP networks. Next, we give an Integer Linear Program (ILP) suitable for obtaining optimal solutions only in small networks. For simplicity, we assume that $\mathcal{S}$ contains all distinct node-pairs, noting that the algorithms are easy to generalize to arbitrary $\mathcal{S}$.

The ILP is formulated in the dual space: to every node i we assign a node potential π_i^d that signifies the shortest distance from i to some d over the costs c, and then we require that the potentials and the costs together fulfill the Shortest Path Optimality Criteria [37] while also maximizing LFA coverage.

Consider the below ILP for the link-protecting version of the LFA cost optimization problem:

$$\max \sum_{(s,d)\in S} \alpha_s^d \tag{5}$$

$$\pi_j^d + s_{ij}^d = \pi_i^d + c_{ij}, \quad 0 \le s_{ij}^d \le C y_{ij}^d \qquad \forall (s,d) \in \mathcal{S}, \forall (i,j) \in E \tag{6}$$

$$\sum_{v\in N_s} y_{sv}^d \le |N_s| - 1 \qquad \forall (s,d) \in \mathcal{S} \tag{7}$$

$$y_{sv}^d \in \{0,1\} \qquad \forall (s,d) \in \mathcal{S}, \forall v \in N_s \tag{8}$$

$$\pi_v^s - \pi_s^s + \pi_s^d - \pi_v^d + z_{sv}^d \le 0, \quad 0 \le z_{sv}^d \le 1 \qquad \forall (s,d) \in \mathcal{S}, \forall v \in N_s \tag{9}$$

$$\sum_{v\in N_s} z_{sv}^d \ge \alpha_s^d, \quad 0 \le \alpha_s^d \le 1 \qquad \forall (s,d) \in \mathcal{S} \tag{10}$$

$$c_{ij} = c_{ji}, \quad c_{ij} \in \{1, \ldots, C_{\max}\} \qquad \forall (i,j) \in E \tag{11}$$

In the ILP, (6)–(8) enforce the Shortest Path Optimality Criteria. In particular, for each destination d and each node s, the node potential π_s^d is set so that the potential-difference $\pi_d^d - \pi_s^d$ encodes the shortest path distance $\mathrm{dist}(s,d)$ from s to d. For each edge (i,j), the constraint $\pi_j^d \le \pi_i^d + c_{ij}$ relates the node potentials to the actual cost setting c, and the binary variable y_{sv}^d is used to guarantee that the inequality is satisfied with strict equality for at least one neighbor v of each s [37]. Namely, y_{sv}^d takes the value 0 if and only if v is a shortest path next-hop from s to d and 1 otherwise, and (7) guarantees that for at least one neighbor v of s variable y_{sv}^d will be set to 0.

Furthermore, (9)–(10) represent the link-protecting LFA condition as of (1). In particular, writing (9) in a more verbose form we get:

$$(\pi_d^d - \pi_v^d) + z_{sv}^d \leq (\pi_s^s - \pi_v^s) + (\pi_d^d - \pi_s^d) \ ,$$

which basically coincides with (1) when $z_{sv}^d > 0$ by substituting $\pi_j^j - \pi_i^j = \text{dist}(i,j)$. Correspondingly, z_{sv}^d in fact serves as an indicator variable whose value is positive if and only if v is a link-protecting LFA from s to d. Moreover, (10) ensures that α_s^d only becomes positive if at least one neighbor of s provides LFA towards d. The requirements (11) guarantee that costs are symmetric and are selected from the interval $\{1, \ldots, C_{\max}\}$. Finally, the objective function (5) maximizes the number of LFA protected node pairs.

There are two problem parameters to the ILP: $C_{\max}$ is the maximum permitted cost, while $C \geq nC_{\max}$ is the maximum allowed potential difference between two neighboring nodes. Then, solving the ILP to optimality yields the link cost setting c that maximizes $\eta^{\text{LP}}(G,c)$ over the input topology G. This can be done by any standard branch-and-bound ILP solver, at least as long as the size of the network is not particularly large (see later). Tightening the ILP, like strengthening the formulation by cutting planes [38], is beyond the scope of this paper.

The ILP is easy to modify to handle the node-protecting version of the LFA cost optimization problem. For this, we need to augment (9)–(10) with the following constraints:

$$\pi_v^e - \pi_e^e + \pi_e^d - \pi_v^d + z_{sv}^d \leq C y_{se}^d \qquad \forall (s,d) \in \mathcal{S}, \forall v \in N_s, \forall e \in N_s \setminus \{d, v\} \quad (12)$$

Here, the new constraints (12) will only let z_{sv}^d to take a positive value, indicating that v is a node-protecting $s-d$ LFA, if (2) holds in addition to (1).

5.4. Approximate algorithms

The above ILP has $O(n^3)$ integer variables, which makes it intractable in anything but the smallest topologies. Therefore, below we provide a set of approximation algorithms, facilitating to obtain a reasonable link cost setting in larger networks as well. In fact, we present a complete family of heuristics, each member of the family having distinct efficiency, running-time, and memory-requirements. This facilitates for picking the best heuristics for the particular requirements.

We chose the simulated annealing probabilistic metaheuristic as the main framework to fund our approximation algorithms onto [39], and within this framework we obtained different heuristics by fine-tuning certain aspects and parameters of the framework. The basic version of the simulated annealing metaheuristic operates as follows: starting from a randomly chosen initial cost c, choose randomly a neighbor c' "nearby" c. Here, two cost settings are nearby if they differ at exactly one link by exactly 1. If the new cost c' provides larger LFA coverage, then it is unconditionally accepted. On the other hand, if c' is worse then it is still accepted with a certain probability, depending on a system parameter called the temperature. The temperature is set so that from an initial, relatively high value it gradually decreases as the algorithm proceeds, ensuring that the system easily escapes from local optima in the beginning, while it will increasingly tend to get stuck in a good quality optimum eventually. The iteration terminates if the temperature reaches a certain threshold or the algorithm could not improve the LFA coverage after a certain number of steps.

The pseudo-code for the approximation framework is given in Alg. 1. Note that the pseudo-code works the same for the link-protecting and the node-protecting case, therefore we below give a generic treatment suitable to handle both cases. The input to the heuristic is the graph $G(V, E)$, initial temperature T_0 and maximum allowed cost $C_{\max}$, and the output is the final cost c.

Algorithm 1 Heuristic LFA cost optimization framework.

$c \leftarrow \text{random\_cost}(C_{\max})$, $T \leftarrow T_0$
while $T > 0$ and $\eta(G, c) < 1$
  $c' \leftarrow \text{choose\_cost}(c)$
  $\Delta\eta \leftarrow \eta(G, c) - \eta(G, c')$
  if $\text{accept\_cost}(\Delta\eta, T)$ **then**
    $c \leftarrow c'$
  end if
  $T \leftarrow T - 1$
end while

The above framework uses a couple of procedures, yet to be specified.

- random_cost(C): this procedure returns a random initial cost in the range $\{1, \ldots, C_{\max}\}$ for each link. Throughout our numerical studies, we used uniformly distributed initial costs.

- choose_cost(c): this procedure selects a cost "nearby" c. Let neigh(c) denote the set of costs that differ from c at one link by 1. We used two different cost selection policies: we either chose a nearby cost randomly (choose_random_cost(c)), or we chose greedily the cost setting that improved LFA coverage the most (choose_greedy_cost(c)), i.e, sets c' according to $\text{argmax}_{q \in \text{neigh}(c)} \eta(G, q)$.

- accept_cost($\Delta\eta, T$): this procedure guides the way the new cost setting c' is accepted. Again, we used two different policies. Both policies share the property that a cost that improves LFA coverage is accepted unconditionally. In addition, the policy proportional_test($\Delta\eta, T$) accepts the new cost even if worse with probability proportional to the temperature. Correspondingly, proportional_test($\Delta\eta, T$) returns true if $\Delta\eta < 0$ or $T >$ random(T_0). Here, the procedure random(x) returns a uniformly distributed random sample from $[0, x]$. The so called *Metropolis-test* (metropolis_test($\Delta\eta, T$)), on the other hand, returns true if $\Delta\eta < 0$ or random(1) $< \exp(-\Delta\eta/T)$.

A useful consequence of defining our approximation framework in the above general form is that different choices for the input parameters as well as the selection of the procedures choose_cost(c) and accept_cost($\Delta\eta, T$) yield different heuristics, with drastically varying performance and running time. For instance, setting the initial temperature low causes faster termination but reduces the probability of finding the global optimum, since this allows the algorithm to explore only a limited domain of the problem space. Furthermore, picking the choose_greedy_cost(c) procedure for selecting the best candidate nearby cost has the potential to rapidly improve the costs initially, but might cause overly long running time due to having to check LFA coverage for each neighboring cost in each iteration. The choice for the accept_cost($\Delta\eta, T$) procedure, on the other hand, influences the proneness of the heuristics to get stuck in local optima.

We also experimented with several small modifications of the above basic approximation framework, in an attempt to obtain good solutions [39].

- *Tabu lists* are useful to preclude the iteration from oscillating between two ore more cost settings, through prohibiting the algorithm to revisit a certain number of previously visited solutions.

- *Restarting* allows the iteration to be restarted from an earlier good

solution in the case when the algorithm gets stuck in a local optimum. Note that no temperature reduction is made during the restart.

- *Quantum tunneling* is similar to restarting, but instead of jumping back to a previous solution we rather jump to another random solution. Again, the intention is to avoid sticking in local optima. In our implementation, if after a configurable number of iterations LFA coverage could not be improved then the heuristics set each link cost randomly up or down by the average link cost. This will change at least one shortest path in the network, and so the iteration transitions into a different domain of the problem space.

The running time of the heuristics principally depends on the choice of the choose_cost(c) procedure. With greedy selection, the complexity is $O(T_0 mn^3)$, dominated by the need to evaluate $\eta(G, q)$ (needing $O(n^3)$ steps) in each iteration for each $2m$ neighbor q of the current cost c. With selecting the choose_random_cost(c) procedure, on the other hand, complexity decreases to $O(T_0 n^3)$, as LFA coverage needs to be evaluated in each iteration only once. This, however, comes with a substantial drop in efficiency, as evidenced by the numerical results presented next.

Finally, we call the attention to an appealing aspect of our heuristic framework. In particular, the framework is easy to adapt for different operational requirements not explicitly addressed in the paper. For instance, the heuristics are completely neutral to whether the underlying graph representation is undirected or directed (a case more relevant to IP networks), or whether link costs are symmetric or asymmetric. The framework does not even require link costs to be integral. Further operational issues, like traffic engineering concerns, suppressing equal cost shortest paths [36], or considering broadcast LANs, etc., are also easy to incorporate into the optimization algorithms through tuning the objective function. The same applies to more elaborate failure models, like multiple failures, Shared Risk Link Groups, etc. Discovering the breadth of these options is beyond the scope of this paper.

6. Numerical evaluations

In the course of our numerical studies, first we were curious as to how close the approximate LFA cost optimization algorithms can get to the optimum. Therefore, we implemented the ILP (6)–(11) and the approximation framework described in Section 5. Below, only results for the greedy cost selection

rule (choose_greedy_cost(c)) and the temperature-proportional acceptance rule (proportional_test($\Delta\eta, T$)) are given, with a tabu list of size 20, no restarting and no quantum tunneling, as this proved most efficient in comparison studies to be discussed later. The simulated annealing procedure was executed 500 times consecutively ($T_0 = 150$, $C_{\max} = 20$) and the cost c^* that attained the highest LFA coverage was selected. Below, we only give the results for the link-protecting case.

We found that about the largest non-trivial graphs for which the ILP can be solved to optimality are of 8 nodes. Unfortunately, very few real topologies of this small size are available in the literature. Thus, the first round of the evaluations were run on Erdős-Rényi random graphs ($n = 8$, expected node degree 3). Out of the 20 random graphs generated, 17 was 2-connected, and results are only given for these instances. Table 2 gives some characteristics of the graphs (number of nodes n, and number of links m); the theoretical lower and upper bounds on LFA coverage (as of Lemma 2 and Lemma 3); and the actual LFA coverage $\eta(G, c_{\text{opt}})$ for the costs c_{opt} obtained by the ILP and the above customized greedy version of the approximation algorithm (η^*_{gr}). We observe that from the 17 experiments only in 2 cases the approximation did not find the optimum (these experiments are marked by an asterisk in Table 2), and the difference is at most 2-3% in LFA coverage. This indicates that in small networks the simulated-annealing-based heuristics perform quite efficiently. Additionally, we found that the theoretical bounds provide a solid estimate on the LFA coverage.

In the second round, we examined the performance of the different approximate LFA cost optimization algorithms we proposed in the previous section in larger real network topologies, where the ILP could not be solved to optimality. First, we deal with the link-protecting case, while node protection will be discussed subsequently.

We used inferred ISP data maps from the Rocketfuel dataset [40] (AS1221, AS1239, AS1755, AS3257, AS3967 and AS6461). We obtained POP-level maps by collapsing the topologies so that nodes correspond to cities and we eliminated leaf-nodes (this preprocessing method was suggested in [41]). These networks come with inferred link costs (these costs are needed to compute the "default" LFA coverage $\eta(G, c)$ of the network). We also chose some network topologies from [42], namely, the Abilene, Italy, Germany, NSF and AT&T networks and the 50 node extended German backbone (Germ_50). Unfortunately, except for the last network no valid link costs were available, so we set each cost to 1. We also chose some representative ISP topolo-

Table 2: LFA cost optimization in random topologies.

Num	n	m	Lower/Upper	$\eta(G, c_{\text{opt}})$	η^*_{gr}
1*	7	11	0.278/1	1	0.976
2	8	9	0.095/0.571	0.536	0.536
3*	8	13	0.214/1	1	0.982
4	7	11	0.278/1	1	1
6	8	9	0.143/0.571	0.571	0.571
9	7	11	0.208/1	0.952	0.952
10	8	11	0.114/1	0.857	0.857
11	8	10	0.143/0.857	0.75	0.75
12	8	9	0.095/0.571	0.429	0.429
13	8	11	0.143/1	0.911	0.911
14	8	11	0.19/1	0.821	0.821
15	8	11	0.19/1	0.946	0.946
16	7	8	0.111/0.667	0.5	0.5
17	8	14	0.2/1	1	1
18	8	11	0.114/1	0.714	0.714
19	8	9	0.143/0.571	0.482	0.482
20	8	10	0.143/0.857	0.679	0.679

gies from [43], in particular, the Arnes, Deltacom, Geant, and the InternetMCI topologies. Link costs were set inversely proportional to the link capacities (this setting is recommended by Cisco, see documentation on `ospf auto-cost` in [44]). Additionally, we also ran the evaluations on some artificial topologies with uniform costs. In particular, M_n are the Möbius ladder graphs of n nodes as discussed in Section 5.

The particular approximation algorithms we used were as follows. First, we chose the "textbook" version of the simulated annealing algorithm, with random cost selection (choose_random_cost(c)) and the Metropolis-test for cost acceptance (metropolis_test($\Delta\eta, T$)), with a tabu list of size 20 and different customized settings for the restarting and quantum tunneling thresholds. The notation η^* will be used to denote the LFA coverage as provided by this textbook version of the heuristics. In particular, $\eta^*_{Q=q}$ will signify the version with the quantum tunneling threshold set to q (i.e., after q unsuccessful trials the algorithm jumps to a new cost set). We used the setting $q = 1$, $q = 10$ and $q = 20$. In addition, $\eta^*_{R=r}$ denotes the LFA coverage by setting the restarting threshold in a similar vein. Evaluations were run for $r = 2$, $r = 10$, and $r = 20$. Finally, we also executed the greedy version as used in the first simulation round (choose_greedy_cost(c), proportional_test($\Delta\eta, T$), tabu list of size 20, no restarting and no quantum tunneling). Results are again

Table 3: Link-protecting results for the LFA cost optimization heuristics in real and artificial topologies.

Name	n	m	Δ	$\eta(G,c)$	η^*	$\eta^*_{Q=1}$	$\eta^*_{Q=10}$	$\eta^*_{Q=20}$	$\eta^*_{R=2}$	$\eta^*_{R=10}$	$\eta^*_{R=20}$	η^*_{gr}
AS1221	7	9	2.57	0.809	0.833	0.833	0.833	0.833	0.833	0.833	0.833	0.833
AS1239	30	69	4.60	0.873	0.958	0.958	0.959	0.959	0.96	0.959	0.958	0.963
AS1755	18	33	3.66	0.872	0.983	0.983	0.983	0.983	0.983	0.98	0.98	0.993
AS3257	27	64	4.74	0.923	0.997	0.995	0.998	0.997	0.997	0.997	0.997	1
AS3967	21	36	3.42	0.785	0.971	0.973	0.973	0.966	0.976	0.966	0.971	0.983
AS6461	17	37	4.35	0.933	1	0.996	0.996	0.996	1	1	1	1
Abilene	12	15	2.5	0.56	0.674	0.674	0.674	0.674	0.674	0.674	0.674	0.674
Arnes	41	57	2.78	0.623	0.702	0.704	0.707	0.704	0.706	0.703	0.7	0.709
AT&T	22	38	3.45	0.822	0.982	0.984	0.982	0.98	0.98	0.978	0.984	0.987
Deltacom	113	161	2.85	0.577	0.654	0.658	0.652	0.659	0.652	0.661	0.651	0.662
Geant	37	55	2.97	0.69	0.74	0.742	0.745	0.743	0.74	0.741	0.737	0.76
Germ_50	50	88	3.52	0.9	0.929	0.931	0.935	0.93	0.932	0.93	0.939	0.966
Germany	17	25	2.94	0.695	0.9	0.893	0.893	0.904	0.904	0.904	0.9	0.911
InternetMCI	19	33	3.47	0.904	0.932	0.932	0.932	0.932	0.932	0.932	0.932	0.932
Italy	33	56	3.39	0.784	0.926	0.928	0.922	0.932	0.927	0.93	0.922	0.944
NSF	26	43	3.3	0.86	0.949	0.955	0.964	0.95	0.958	0.956	0.955	0.977
M_6	6	9	3	0.4	1	1	1	1	1	1	1	1
M_{10}	10	15	3	0.444	0.922	0.922	0.933	0.933	0.922	0.933	0.922	1
M_{18}	18	27	3	0.470	0.882	0.875	0.885	0.885	0.882	0.872	0.888	0.905
M_{30}	30	45	3	0.482	0.889	0.883	0.886	0.886	0.889	0.889	0.888	0.904

marked by η^*_{gr}. For each topology, the algorithms were executed 1000 times ($T_0 = 1000$, $C_{\max} = 20$) and the best cost c^* was selected. There was only one topology on which we could solve the ILP to optimality: AS1221. For this particular network, each of the approximation algorithms could attain the ILP optimum ($\eta(G, c_{\mathrm{opt}}) = 0.833$).

Detailed results of link-protecting mode are presented in Table 3. The columns mean (in the order of appearance): the characteristics of the topologies (name, number of nodes n and edges m, and the average node degree Δ); the LFA coverage obtained by the original link cost setting for the graphs $\eta(G, c)$; and the LFA coverage obtained by the different approximation algorithms. We also highlight the results for some select topologies in Fig. 3.

Our observations are as follows. First, we found that the LFA coverage produced by the approximate algorithms is usually significantly higher (about 90% or more in the link-protecting case) than the LFA coverage produced by the network's original cost setting (around 70% on average). The improvement almost always exceeds 7%, but in many cases it amounts to

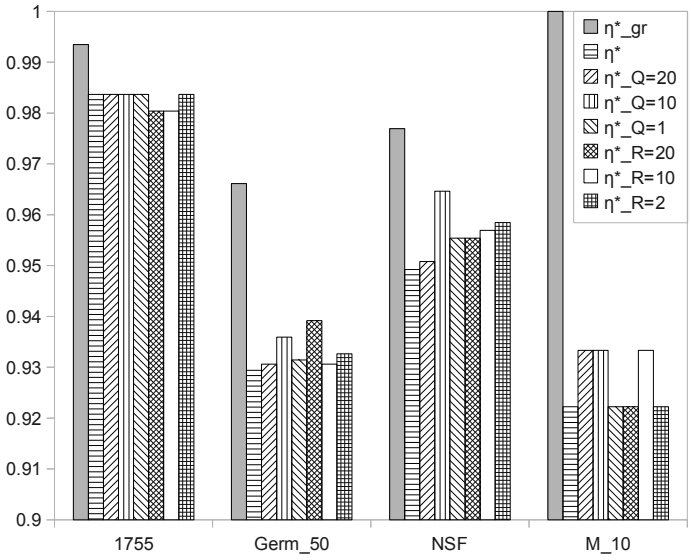


Figure 3: Final LFA coverage for some select topologies.

more than 20-25% (e.g., AS3967, Abilene, or the German backbone). This suggests that optimizing costs specifically for LFA usually attains significant improvement in network resilience. The improvement is especially significant for the artificial networks. We also found that the denser the network, the higher the LFA coverage. It seems that networks with an average node degree exceeding about 3.5 lend themselves especially well to LFA cost optimization (AS1239, AS1755, AS3257, AS6461, AT&T, Germ_50): in these networks even the default cost settings yield a higher than 80% LFA coverage and our cost optimization tool can bring these networks well beyond 95% and close to 100% in many cases. Networks of average node degree of 3 are still amenable to LFA, but when the degree falls below 3 the chances of getting a high LFA coverage rapidly vanish. For sparser networks (like the Abilene topology), the final LFA coverage $\eta(G, c^*)$ hardly reached 70%. These observations are in line with our theoretical analysis in Section 4. Note, however, that node degree alone is not sufficient to assess the extent to which LFA can protect a network, as there are topologies (the Möbius ladder graphs) that have small average degree of 3 but complete link-protecting LFA coverage over some appropriately chosen costs. It seems that LFA cost optimization is most difficult when the degree is about 3.

Second, we observe that for large Möbius ladder graphs the approximation could not get closer than 10% to the optimum (which we know is $\eta(G, c_{\text{opt}}) = 1$ in this case). This indicates that in larger topologies the efficiency of the heuristics we identified in small networks might not be present.

Third, we find that the approximation algorithms work roughly similarly. In particular, Fig. 4 compares the LFA coverage of the different heuristics, when taking the textbook simulated annealing heuristics as the basis for the comparison. We observe that the best performance is attained by the greedy version consistently. This is not surprising, considering the more elaborate (and more time-consuming) cost selection rule. In addition, quantum tunneling seems to work better for larger thresholds, while restarting produces consistently better results for smaller thresholds. The differences in the eventual LFA coverage between the different heuristics, however, are in the order of mere percents. We note, however, that minuscule differences in the LFA coverage can mean dozens of more protected source-destinations in reality. This is because the normalizing factor $n(n-1)$ in (3) can become very large in big networks. For instance, in the Deltacom topology the greedy heuristics covered 2367 source-destination pairs while the others reached less than 2300.

Last but not least, the execution time for 1000 rounds of the different heuristics is depicted in Fig. 5 in a logarithmic scale. As expected, for the wins in performance with the greedy version of the heuristics we pay by significantly increasing execution time. This is because, as mentioned previously, choosing the best neighbor in every step requires $2m$ times more evaluations of the LFA coverage metric than selecting one randomly. Note that these execution times are representative to the offline phase of LFA cost optimization, which is usually performed only once for the lifetime of the network before the final deployment, and in no way affect the *real-time* requirements of finding loop-free alternates in the online phase (which remains below 50ms as required). Therefore, we do not consider execution times to be a particularly pressing issue in LFA cost optimization. Finally, we mention that, curiously, running 1000 rounds of the heuristics is usually unnecessary, because the best solutions were realized after 200 rounds in each case.

We repeated the simulations for the node-protecting case as well (see Table 4). The observations are essentially the same as in the link-protecting case, just the numeric values seem somewhat smaller. We found that the initial node-protecting LFA coverage was 57% on average which the algorithms could improve by about 10-20%, so that eventual the LFA coverage was in

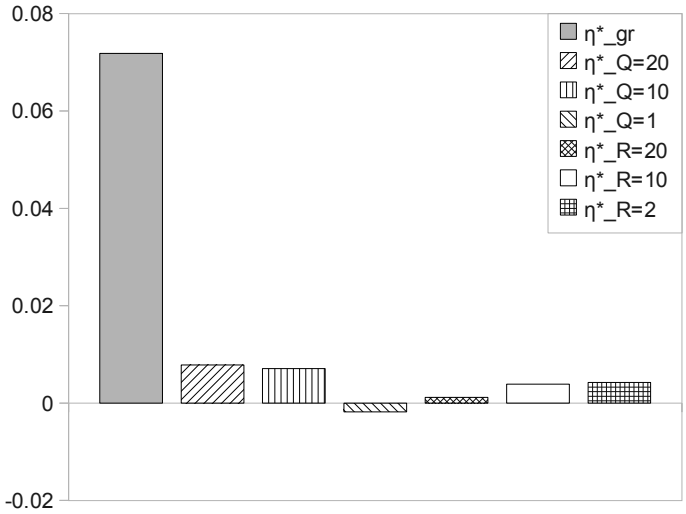


Figure 4: Final LFA coverage obtained by different heuristics compared to the textbook version.

the range of 75-80%. In some cases, however, the heuristics could reach more than 25% of improvement (for instance, this is the case for the AS3257 topology). Again, the greedy heuristic is consistently the most effective, and restarting and quantum tunneling seem worthwhile extensions to textbook simulated annealing.

In summary, our results suggest that most real network topologies, which are usually richly connected and highly redundant, lend themselves readily to LFA cost optimization, to the point that almost perfect link-protecting LFA coverage can be achieved in many cases. For fast execution time the textbook simulated annealing version with a reasonable quantum tunneling and restarting threshold seems most appealing, whereas the greedy version is the best choice when the aim is to find the highest quality link costs.

7. Conclusions

In this paper, we have assessed the possibilities of improving fast resilience in operational IP networks using the Loop-Free Alternates IPFRR technique. The motivation for choosing LFA over its alternatives is its simplicity, easy deployability, and availability in IP routers. We presented new

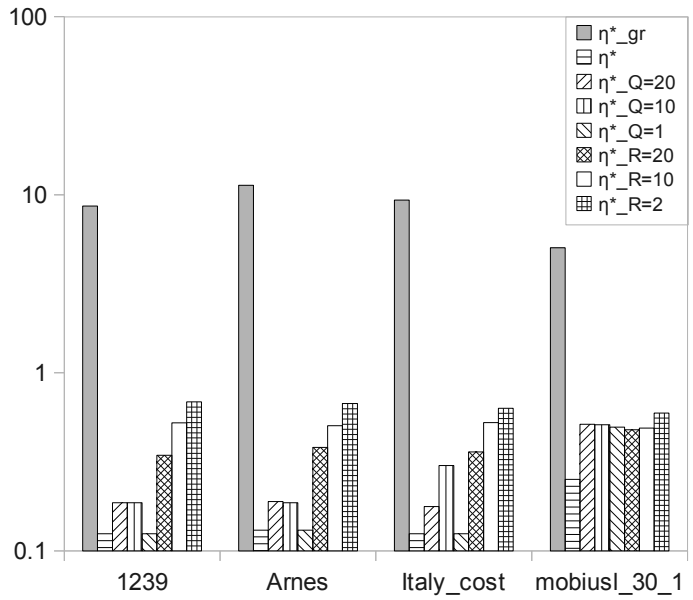


Figure 5: Execution time for different heuristics in hours.

tools to quickly estimate LFA failure case coverage both in the link-protecting and the node-protecting cases, and we sought ways to improve it by carefully adjusting IGP link costs. We showed that this problem is NP-complete and we gave an Integer Linear Program to obtain an exact solution. As our exact algorithm only works in small networks, we proposed a family of simulated-annealing-based approximations with different tunable performance and execution time parameters. Our heuristics could achieve significant boost in LFA coverage in many real-world network topologies, to the point that in some cases close to perfect protection could be guaranteed by LFA. Considering that LFA is just a router-configuration command away in many modern IP networks, we believe that these results have huge practical relevance. Nevertheless, we also found that some topologies are less amenable to LFA cost optimization. Future work involves combining the LFA network optimization tools we gave in [21] and the algorithms presented herein to improve IP-level fast resilience in such notorious network topologies.

Table 4: Node-protecting results for the LFA cost optimization heuristics in real and artificial topologies.

Name	n	m	Δ	$\eta(G,c)$	η^*	$\eta^*_{Q=1}$	$\eta^*_{Q=10}$	$\eta^*_{Q=20}$	$\eta^*_{R=2}$	$\eta^*_{R=10}$	$\eta^*_{R=20}$	η^*_{gr}
AS1221	7	9	2.57	0.452	0.523	0.523	0.523	0.523	0.523	0.523	0.523	0.523
AS1239	30	69	4.60	0.757	0.886	0.896	0.889	0.886	0.894	0.893	0.882	0.937
AS1755	18	33	3.66	0.764	0.895	0.885	0.888	0.885	0.882	0.882	0.911	0.941
AS3257	27	64	4.74	0.726	0.863	0.877	0.881	0.871	0.871	0.873	0.87	0.938
AS3967	21	36	3.42	0.642	0.84	0.838	0.838	0.847	0.847	0.857	0.835	0.897
AS6461	17	37	4.35	0.738	0.845	0.838	0.83	0.841	0.852	0.841	0.838	0.886
Abilene	12	15	2.5	0.515	0.606	0.598	0.606	0.606	0.606	0.606	0.606	0.606
Arnes	41	57	2.78	0.359	0.469	0.482	0.484	0.478	0.471	0.472	0.486	0.49
AT&T	22	38	3.45	0.58	0.783	0.8	0.796	0.783	0.779	0.781	0.781	0.82
Deltacom	113	161	2.85	0.488	0.559	0.571	0.566	0.568	0.567	0.564	0.558	0.581
Geant	37	55	2.97	0.41	0.572	0.590	0.585	0.581	0.577	0.573	0.57	0.622
Germ_50	50	88	3.52	0.827	0.829	0.816	0.807	0.815	0.829	0.822	0.82	0.86
Germany	17	25	2.94	0.562	0.731	0.72	0.709	0.713	0.713	0.705	0.705	0.727
InternetMCI	19	33	3.47	0.704	0.783	0.766	0.766	0.777	0.78	0.777	0.769	0.809
Italy	33	56	3.39	0.57	0.768	0.754	0.748	0.75	0.764	0.754	0.754	0.803
NSF	26	43	3.3	0.633	0.83	0.796	0.806	0.815	0.829	0.807	0.836	0.866
M_6	6	9	3	0.444	0.922	0.888	0.9	0.9	0.9	0.911	0.9	0.966
M_{10}	10	15	3	0.47	0.81	0.8	0.807	0.81	0.816	0.813	0.81	0.849
M_{18}	18	27	3	0.482	0.812	0.809	0.809	0.81	0.809	0.817	0.812	0.833
M_{30}	30	45	3	0.4	1	1	1	1	1	1	1	1

8. Acknowledgements

G.R. was supported by the János Bolyai Fellowship of the Hungarian Academy of Sciences. J.T. was supported by the Magyary Zoltán program. The project was supported by TÁMOP 4.2.2.B-10/1-2010-0009 grant.

References

[1] J. Moy, OSPF Version 2, RFC 2328 (April 1998).

[2] R. Callon, Use of OSI IS-IS for routing in TCP/IP and dual environments, RFC 1195 (December 1990).

[3] P. Francois, C. Filsfils, J. Evans, O. Bonaventure, Achieving sub-second IGP convergence in large IP networks, SIGCOMM Comput. Commun. Rev. 35 (3) (2005) 35–44.

[4] M. Shand, S. Bryant, IP Fast Reroute framework, RFC 5714 (Jan 2010).

[5] G. Enyedi, G. Rétvári, T. Cinkler, A novel loop-free IP fast reroute algorithm, in: EUNICE, 2007, pp. 111–119.

[6] I. Hokelek, M. Fecko, P. Gurung, S. Samtani, S. Cevher, J. Sucec, Loop-free IP Fast Reroute using local and remote LFAPs, Internet Draft (Feb 2008).

[7] A. Li, X. Yang, D. Wetherall, SafeGuard: safe forwarding during route changes, in: ACM CoNEXT, 2009, pp. 301–312.

[8] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, O. Lysne, Multiple routing configurations for fast IP network recovery, IEEE/ACM Trans. Netw. 17 (2) (2009) 473–486. doi:http://dx.doi.org/10.1109/TNET.2008.926507.

[9] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, C.-N. Chuah, Proactive vs reactive approaches to failure resilient routing, in: INFOCOM, 2004.

[10] S. Bryant, C. Filsfils, S. Previdi, M. Shand, IP Fast Reroute using tunnels, Internet Draft (Nov 2007).

[11] S. Bryant, M. Shand, S. Previdi, IP fast reroute using Not-via addresses, Internet Draft (March 2010).

[12] G. Enyedi, P. Szilágyi, G. Rétvári, A. Császár, IP Fast ReRoute: lightweight Not-Via without additional addresses, in: INFOCOM Miniconf, 2009, pp. 2771–2775.

[13] A. Li, P. Francois, X. Yang, On improving the efficiency and manageability of NotVia, in: ACM CoNEXT, 2007, pp. 1–12.

[14] A. Atlas, A. Zinin, Basic specification for IP fast reroute: Loop-Free Alternates, RFC 5286 (2008).

[15] C. Reichert, T. Magedanz, Topology requirements for resilient IP networks, in: MMB, 2004, pp. 379–388.

[16] K.-W. Kwong, L. Gao, R. Guerin, Z.-L. Zhang, On the feasibility and efficacy of protection routing in IP networks, in: INFOCOM 2010, long version appears as Tech. Rep. 2009, University of Pennsylvania, 2010, pp. 1–9.

[17] P. Francois, O. Bonaventure, An evaluation of IP-based fast reroute techniques, in: ACM CoNEXT, 2005, pp. 244–245.

[18] S. Previdi, IP Fast ReRoute technologies, APRICOT (2006).

[19] M. Gjoka, V. Ram, X. Yang, Evaluation of IP fast reroute proposals, in: IEEE Comsware, 2007, pp. 710–718.

[20] C. Filsfils, et al., LFA applicability in SP networks, Internet Draft (March 2010).

[21] G. Rétvári, J. Tapolcai, G. Enyedi, A. Császár, IP Fast ReRoute: Loop Free Alternates revisited, in: INFOCOM, 2011, pp. 2948–2956.

[22] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, A. Császár, Optimizing IGP link costs for improving IP-level resilience, in: Proc. International Workshop on Design Of Reliable Communication Networks (DRCN), 2011, pp. 62–69.

[23] H. T. Viet, P. Francois, Y. Deville, O. Bonaventure, Implementation of a traffic engineering technique that preserves IP Fast Reroute in COMET, in: Rencontres Francophones sur les Aspects Algorithmiques des Telecommunications, Algotel (2009), 2009.

[24] M. Menth, M. Hartmann, D. Hock, Routing optimization with IP Fast Reroute, Internet Draft (July 2010).

[25] S. Iyer, S. Bhattacharyya, N. Taft, C. Diot, An approach to alleviate link overload as observed on an IP backbone, in: INFOCOM, 2003.

[26] M. Menth, M. Hartmann, R. Martin, T. Čičić, A. Kvalbein, Loop-free alternates and not-via addresses: A proper combination for IP fast reroute?, Comput. Netw. 54 (8) (2010) 1300–1315.

[27] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, C.-N. Chuah, Failure inferencing based fast rerouting for handling transient link and node failures, in: INFOCOM, 2005.

[28] G. Schollmeier, J. Charzinski, A. Kirstadter, C. Reichert, K. Schrodi, Y. Glickman, C. Winkler, Improving the resilience in IP networks, in: High Performance Switching and Routing, 2003, HPSR. Workshop on, 2003, pp. 91–96.

[29] A. Csaszar, G. Enyedi, S. Kini, Ip fast re-route with fast notification, Internet Draft (March 2011).

[30] C. Systems, Cisco IOS XR Routing Configuration Guide, Release 3.7 (2008).

[31] J. Networks, Junos 9.6 routing protocols configuration guide (2009).

[32] B. Fortz, J. Rexford, M. Thorup, Traffic engineering with traditional IP routing protocols, IEEE Comm. Mag. 40 (10) (2002) 118–124.

[33] G. Rétvári, R. Szabó, J. J. Bíró, On the representability of arbitrary path sets as shortest paths: Theory, algorithms, and complexity, in: Lecture Notes in Computer Science: Proceedings of the Third International IFIP-TC6 Networking Conference, Athens, Greece, 2004, pp. 1180–1191.

[34] P. Pan, G. Swallow, A. Atlas, Fast reroute extensions to RSVP-TE for LSP tunnels, RFC 4090 (2005).

[35] G. Swallow, S. Bryant, L. Andersson, Avoiding equal cost multipath treatment in MPLS networks, RFC 4928 (June 2007).

[36] M. Thorup, M. Roughan, Avoiding ties in shortest path first routing, aT&T, Shannon Laboratory, Florham Park, NJ, Technical Report, `http://www.research.att.com/~mthorup/PAPERS/ties_ospf.ps` (2001).

[37] R. Ahuja, T. Magnanti, J. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice-Hall, New Jersey, 1993.

[38] G. Dahl, M. Stoer, A cutting plane algorithm for multicommodity survivable network design problems, INFORMS Journal on Computing 10 (1996) 1–11.

[39] M. Pióro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[40] R. Mahajan, N. Spring, D. Wetherall, T. Anderson, Inferring link weights using end-to-end measurements, in: ACM IMC, 2002, pp. 231–236.

[41] D. Applegate, E. Cohen, Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs, in: ACM SIGCOMM, 2003, pp. 313–324.

[42] SNDlib, Survivable fixed telecommunication network design library, http://sndlib.zib.de.

[43] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet Topology Zoo, http://www.topology-zoo.org.

[44] Cisco Systems, Cisco IOS Release 12.0, Network Protocols Configuration Guide (2011).