IP Fast ReRoute: Loop Free Alternates Revisited

Gábor Rétvári and János Tapolcai
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Email: {retvari, tapolcai}@tmit.bme.hu

Gábor Enyedi and András Császár
TrafficLab,
Ericsson Research
Email: {gabor.sandor.enyedi,andras.csaszar}@ericsson.com

***Abstract*—IP Fast ReRoute (IPFRR) is the IETF standard for providing fast failure protection in IP and MPLS/LDP networks and Loop Free Alternates (LFA) is a basic specification for implementing it. Even though LFA is simple and unobtrusive, it has a significant drawback: it does not guarantee protection for all possible failure cases. Consequently, many IPFRR proposals have appeared lately, promising full failure coverage at the price of added complexity and non-trivial modifications to IP hardware and software. Meanwhile, LFA remains the only commercially available, and therefore, the only deployable IPFRR solution. Deployment, however, crucially depends on the extent to which LFA can protect failures in operational networks. In this paper, therefore, we revisit LFA in order to give theoretical insights and practical hints to LFA failure coverage analysis. First, we identify the topological properties a network must possess to profit from good failure coverage. Then, we study how coverage varies as new links are added to a network, we show how to do this optimally and, through extensive simulations, we arrive to the conclusion that cleverly adding just a couple of new links can improve the quality of LFA protection drastically.**

***Index Terms*—IP protection, IP Fast ReRoute, Loop Free Alternates**

I. Introduction

Transporting delay and loss sensitive traffic in the Internet has become an important requirement in the last few years. At the moment, the IP protocol suite is not yet amenable to fully support multimedia streams due to various reasons, one of which is slow response to failures. Recovery with current Interior Gateway Protocols (IGPs) is in the order of hundreds of milliseconds [1], typically beyond what is tolerable to a multimedia stream. Similar is the case of MPLS networks that rely on LDP for label exchange, as LDP is dependent on the IGP for routing. Therefore, the IETF defined a framework, called IP Fast ReRoute (IPFRR [2]), for native IP protection in order to reduce failure reaction time to tens of milliseconds in an intra-domain, unicast setting.

IPFRR is based on two principles: *local rerouting* and *precomputed detours*. Local rerouting means that only routers directly adjacent to a failure are notified of it, which eliminates one of the most time-consuming steps of IGP-based restoration: global flooding of failure information. Additionally, IPFRR mechanisms are proactive in that detours are computed, and installed in the forwarding engine, long before any failure occurs. Thus, when a failure eventually shows up, routers are able to switch to an alternate path instantly. Once alternate next-hops are active, traffic flows undisrupted bypassing the failed component, letting the IGP to converge in the background.

A basic specification for IPFRR is Loop-Free Alternates (LFA, [3]). When connectivity to some next-hop is lost, all traffic that would have used the unreachable next-hop is passed on to an alternate next-hop, called a Loop Free Alternate, that still has a path to the destination that is unaffected by the failure. LFA is simple, it can be realized with straightforward modifications to current IGPs, and deployment is easy thanks to the fact that it does not require support from other routers. On the other hand, LFAs not always protect both link and node failures at the same time and may also lead to temporary loops when multiple simultaneous failures show up. But the major problem is that often not all routers have LFAs to all other routers, which means that certain failure scenarios are impossible repair rapidly.

Unfortunately, complete IP-level local protection is difficult due to IP's destination-based forwarding paradigm. As only adjacent routers are aware of a failure, remote routers do not know whether an arriving packet is traveling on its shortest path or it is already on a detour and so exceptional forwarding should be applied. Without being able to differentiate between these two cases, local IP protection can never attain 100% failure coverage[1]. Most IPFRR proposals, therefore, either change IP's destination-based forwarding [4]–[6] or introduce some forms of signaling to indicate that a packet is on a detour. Some call for out-of-band failure signaling [7], others use invaluable extra bits in the IP header [8], [9] or add special information to it for in-band signaling [10], and still others propose to mark detours by tunneling [11]–[13]. While modern routers can handle tunneled packets at wire speed, tunneling needs additional address management [13], [14] and, if the additional IP header does not fit into the MTU, can cause packet fragmentation and time-consuming reassembly at the tunnel endpoint. It seems, therefore, that the price for IP-level local protection, capable to handle all possible failure cases, is considerable added complexity and management burden, modifications to the essential IP infrastructure, and the breaking of the incremental deployment path.

It is, therefore, no wonder that today LFA is the only standardized and readily available IPFRR technology. At least two major router vendors are offering LFA-based IPFRR

G.R. was supported by the János Bolyai Fellowship of the Hungarian Academy of Sciences. J.T. was supported by the Magyary Zoltán program.

[1]One can easily prove this claim formally, taking the example of a ring.

support out of the box [15], [16], and other vendors are expected to follow suit. Consequently, operators in need for improving network resilience are now facing the question whether to deploy LFA, and this decision depends crucially on the extent to which LFA can protect failures in the particular topology. This paper aims to assist making this decision.

Even though thorough, simulation-based reports are available [17]–[21], a deep understanding of how certain network characteristics affect LFA failure coverage is still missing. Thus, in the first part of the paper we study the graph topological ingredients needed for good LFA protection. In this regard, this paper is a sequel to [22], where the authors study to what extent IP's destination-based forwarding permits protection routing, and [23] presenting a similar study for the O2 scheme. As we find that LFA failure coverage strongly depends on the topology as well as on the link costs, we study the effects of both separately.

Existing proposals modify standard IP forwarding in some way to achieve full protection. Why not choose the other way around? That is, instead of bending IP to provide full protection in all networks, paying huge price in complexity and deployability, why not bend the network topology itself so that even LFA can guarantee full protection? We study this question in the second part of the paper. We show real networks where by adding just two or three new links full LFA protection can be attained. This might be an acceptable price for an operator to take the easy deployment path. In some cases, however, our analysis reveals that full LFA protection can only be achieved at the cost of a substantial topology redesign, which is a clear indication to choose alternative protection schemes [24]. At the least, such an analysis can be instructive in the next regular network upgrade cycle.

The rest of the paper is organized as follows. Section II gives an overview of LFA and provides a useful mathematical model. Section III is devoted to graph theoretical LFA failure coverage analysis, and Section IV discusses the LFA graph extension problem. Then, numerical results are described in Section V and finally, Section VI concludes the paper.

II. Loop Free Alternates

Perhaps the easiest way to demonstrate LFA is through an example. Consider the network depicted in Fig. 1 and suppose that router d wishes to send a packet to router f. The next-hop of d along the shortest path towards f is c. If, however, link (d, c) fails, then d needs to find an alternative neighbor to pass on the packet to. It cannot send the packet to, say, b, as b's shortest path to f goes through itself, so b would send the packet back causing a loop (remember that in IPFRR, routers not immediately adjacent to a failure do not get notified of it). Instead, it needs to find a neighbor that is closer to the destination than the length of the route from the neighbor through itself. This relation can be expressed as follows:

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

where s is the actual source node, d is the node the packet is destined to, n is a neighbor of s other than the failed next-hop

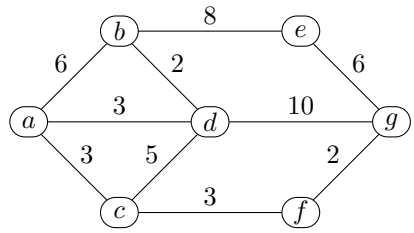


Figure 1: A sample weighted network topology.

and $\mathrm{dist}(x, y)$ denotes the length of the shortest $x \to y$ path. A neighbor fulfilling (1) is called a *link-protecting Loop Free Alternate* (LFA). For instance, a is a link-protecting LFA for node d towards node f as $\mathrm{dist}(a, f) < \mathrm{dist}(a, d) + \mathrm{dist}(d, f)$.

Many alternative LFA types exist. For instance, g is also an LFA for d towards f, but it also protects against the failure of node c, so it is a *node-protecting LFA*. It is also a *downstream neighbor*, as it is closer to f than d, as well as a *per-link LFA* for the link (d, c) as it protects all the nodes reachable by d through (d, c). For a full taxonomy, see [3], [20], [21].

A. Model

We model the network topology by a *simple, undirected weighted graph* $G(V, E)$ with V being the set of nodes and E the set of edges. Let $n = |V|$ and $m = |E|$, and denote the complement of the edge set with $\overline{E}$. Let the cost of edge (i, j) be $c(i, j)$. For simplicity, we assume that *(i)* edges are bidirectional and point-to-point; *(ii)* costs are symmetric; and *(iii)* each node has a well-defined next-hop towards each destination. This means that if multiple shortest paths are available to a destination, then one is chosen arbitrarily.

Being by far the most common in operational networks (accounting for about 70% of unplanned outages [25]), we shall limit our attention to *single link failures* exclusively. As simple *link-protecting LFAs* safeguard against just this type of failures and they contain all the other LFA types as special cases, we do not treat those henceforth. Consequently, we shall usually assume that the graph describing the network is *2-edge-connected*, which is the minimum topological requirement for being able to repair every possible link failure.

B. Theoretical framework

Definition 1: Consider an undirected, weighted graph $G(V, E)$. For each $d \in V$, define a relation $(\prec_d)$ on V as follows: let $u \prec_d v$ if at least one shortest path from v to d goes through u. Let $u \preceq_d v$ if either $u \prec_d v$ or $u = v$. Finally, put $u \succsim_d v$ if u is not ordered with respect to v by $(\preceq_d)$.

In Fig. 1, for instance, $d \preceq_f b$ but $d \succeq_b b$, and $a \succsim_f d$.

The relation $(\preceq_d)$ defines a partial order on V, since it is reflexive, transitive and antisymmetric. The partially ordered set $(V, \preceq_d)$ is called the d-poset. Each d-poset has exactly one lower bound: d. We say that some $u \in V$ is an ancestor (descendant) of some $v \in V$ in the d-poset if $u \prec_d v$ ($u \succ_d v$, respectively). Additionally, a parent (child) is a neighboring ancestor (descendant). By assumption, if a node has multiple parents, then one is assigned as next-hop arbitrarily.

Using this model, we redefine (1) as follows:

Definition 2: For some $s \in V$ and $d \in V$, a neighbor $n \in V$ of s that is not the next-hop is a link-protecting LFA (simply LFA, henceforth) if $s \not\preceq_d n$.

Simply put, n is an LFA if no shortest path from n to d passes through s. Hence, no loop will arise if a packet destined to d is handed by s to n instead of its primary next-hop. Note that the condition $s \not\preceq_d n$ means that either $s \succeq_d n$ or $s \succ_d n$.

From the above discussion, it is clear that in general networks not all nodes have LFA protection to every other node [17]–[21]. To measure the *LFA failure coverage* $\eta(G)$ in a weighted graph G, we adopt the simple metric from [3]:

$$\eta(G) = \frac{\text{\#LFA protected } (s,d) \text{ pairs}}{\text{\#all } (s,d) \text{ pairs}}$$

III. LFA FAILURE COVERAGE: A THEORETICAL ANALYSIS

Next, we give a graph-theoretical characterization of LFA failure coverage, as measured by $\eta(G)$. First, we identify worst-case graphs G with minimal $\eta(G)$. Then we seek the opposite extreme: graphs with perfect LFA coverage (i.e., $\eta(G) = 1$). Since both the topological properties of the underlying network and the actual edge costs have profound and intricate effect on the efficiency of LFA, it is worth examining their impact separately. Thus, we first study full LFA failure coverage under the assumption that costs are uniform, and then we endue our graphs with costs and see how our results generalize to the weighted case (if at all).

A. Worst case graphs

It has been observed previously that the quintessential worst-case graphs for IPFRR are rings, i.e., cycle graphs in which all nodes are of degree 2 [6], [26]. The reason is the bad interplay between destination-based forwarding and the small path diversity in rings. It is not surprising, therefore, that we find the even ring to have the smallest LFA coverage out of all 2-connected graphs with the same number of nodes.

Theorem 1: The LFA failure coverage of a 2-connected graph G on n nodes is bounded by $\frac{1}{n-1} \leq \eta(G) \leq 1$ and the lower bound is tight for rings with even number of nodes and uniform edge costs.

Proof: Consider a ring $G(V,E)$ with $n = |V| > 2$ and even, let costs be uniform and choose some $d \in V$. Now, there are $n-2$ nodes having exactly one parent and one child in the d-poset. Since a node cannot get LFA from its children or its next-hop, these $n-2$ nodes do not have LFA. The remaining node (at the opposite side of d in the ring) has two parents, one of them is the next-hop and the other provides an LFA. Hence, for each $d \in V$ there is only one node with LFA towards d, which yields $\eta(G) = \frac{n}{n(n-1)} = \frac{1}{n-1}$.

To prove that this is a lower bound, we use the fact that in a 2-connected graph each $d \in V$ is contained in at least one cycle. Take the smallest cycle containing d. Note that this cycle has no chords. One can use the above reasoning to show that, over arbitrary edge costs, at least one node has LFA towards d in this cycle, and from this the result follows. ■

Rings are important in telecommunications, and the above theorem suggests that they are very badly suited for LFA. Besides, the theorem also suggests that graphs without short cycles are problematic for LFA. For instance, fault-tolerant networks are often connected in hypercube topologies, 2D and 3D meshes (each consisting of 4-cycles), or 2D and 3D torus topologies (consisting of longer cycles), and these cannot have perfect LFA coverage either.

B. Perfect LFA coverage: uniform edge costs

Next, we turn to the characterization of networks with perfect LFA coverage. Herein, we concentrate on the uniform cost case, when shortest path routing boils down to min-hop routing.

Observation 1: Consider an undirected graph G with uniform edge costs. Now, $\eta(G) = 1$, if and only if each node has an LFA towards each of its neighbors.

Easily, if all neighbors are protected, then all nodes in the graph are protected as well since these are reached through the neighbors. The other way around: if $\eta(G) = 1$ then, evidently, all neighbors must be protected.

Previously, we argued that graphs with long cycles are problematic for LFA. The next result makes this claim explicit.

Theorem 2: Consider an undirected, simple graph $G(V,E)$ with uniform costs. Now, $\eta(G) = 1$, if and only if each edge is contained in at least one triangle (cycle of length 3).

Proof: First, we show that if all $(u,v) \in E$ are contained in a triangle, then $\eta(G) = 1$. Let some triangle containing (u,v) be $u - v - w$. One easily sees that $u \not\preceq_v w$, as it is the direct path through edge (w,v) that is the shortest (min-hop) path from w to v, and $w \to u \to v$ is strictly longer than that. Thus, w is an LFA for u towards v protecting edge (u,v), and the claim then follows from Observation 1.

To see the reverse direction, we prove that if $\eta(G) = 1$, then every edge is contained in a triangle. If $\eta(G) = 1$, then for each $(u,v) \in E$ node u has an LFA towards v. Let this be w. Easily, $(u,w) \in E$. We only need to show that $(w,v) \in E$ as well to have a triangle. Indirectly, if $(w,v) \notin E$, then $u \preceq_v w$, which contradicts the assumption that w is an LFA. ■

Theorem 2 implies that complete graphs, chordal graphs and maximal planar graphs have full LFA coverage in the uniform cost case.

C. Perfect LFA coverage: weighted graphs

Next, we extend our analysis to weighted graphs. Call an edge (u,v) a *forwarding edge*, if the next-hop from u to v is exactly v. In other words, a forwarding edge is an edge that connects a node to a neighbor that is a next-hop towards some destinations. Below is a generalization of Observation 1.

Observation 2: Consider an undirected, weighted graph G. Now, $\eta(G) = 1$, if and only if for each forwarding edge (u,v), u has an LFA to v.

This observation basically says that we have full LFA protection if and only if all next-hops remain reachable after a link failure. The difference from Observation 1 is that we need to protect forwarding edges only, as in a weighted graph not

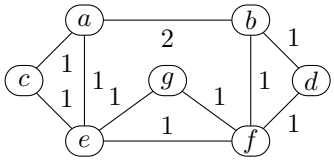


Figure 2: A weighted graph with full LFA coverage.

all edges are actually used to forward traffic. Unfortunately, Theorem 2 does not extend so naturally to the weighted case.

Theorem 3: Consider an undirected, weighted simple graph $G(V, E)$. Now, $\eta(G) = 1$, if each forwarding edge (i, j) is contained in a triangle $i-j-k$ for which the triangle inequality holds with strict inequality:

$$\text{dist}(i, j) < \text{dist}(i, k) + \text{dist}(k, j) \tag{2}$$

$$\text{dist}(i, k) < \text{dist}(i, j) + \text{dist}(j, k) \tag{3}$$

$$\text{dist}(k, j) < \text{dist}(k, i) + \text{dist}(i, j) \tag{4}$$

Proof: Eq. (4) basically states that $i \not\preceq_j k$, so k is an LFA from i to j. Then, the result follows from Observation 2. ■

The theorem suggests that densely connected networks that use geographical distances as costs are particularly well-suited for LFA. Many fixed wireless networks fall into this category.

Note that the condition is only sufficient, but not necessary. As one easily checks, the graph in Fig. 2 has full LFA coverage, even though the forwarding edge (a, b) is contained in no triangle at all.

IV. LFA GRAPH EXTENSION

The conditions for full LFA coverage turn out to be somewhat restrictive, suggesting that only special topologies admit full protection. Indeed, practical studies show that in common networks LFA coverage is usually in the order of 50-90% [17]–[21]. In this section, we seek ways to improve this situation.

There are essentially three approaches to increasing LFA coverage: changing link costs, changing the topology, or doing both. We study the first option in [27], while this paper is dedicated to the second approach. In particular, we ask how to extend the network with new links (e.g., by leasing additional capacity, provisioning new virtual links, or deploying new fibers) to improve LFA coverage, and we shall recur to manipulating costs only if improvement cannot be achieved otherwise. The reason is that in many operational networks edge costs are deliberately optimized, for the purposes of load-balancing, reducing delay and improving resiliency [28], or to obviate equal-cost paths in order to eliminate packet re-ordering, unwanted packet fragmentation [29], etc. Modifying costs would destroy carefully engineered shortest paths and this would make deploying LFA less attractive to operators.

As before, we again study the impacts of the graph topology and edge costs separately.

A. LFA graph extension: uniform link costs

In this section, we ask how to add edges to a graph to achieve full LFA coverage, provided that both the edges originally existing in the graph and the edges we add have the same cost. Obviously, we want to do this with the fewest new edges possible. Consider the problem statement:

Definition 3: LFA graph extension problem in the uniform cost case (minLFAu): Given a simple, undirected graph $G(V, E)$ with uniform edge costs c on all edges and an integer l, is there a set $F \subseteq \overline{E}$ with $|F| \leq l$ and $\forall (u, v) \in F : c(u, v) = c$ so that $\eta(G(V, E \cup F)) = 1$?

Note that adding uniform cost edges to a uniform cost graph necessarily changes the shortest paths between some of the nodes. This is because at least the nodes connected by the new edge will use it to reach each other, instead of whatever shortest path they had before. Hence, the requirement that shortest paths be invariant to LFA graph extension cannot be met when solving minLFAu.

Theorem 4: The LFA graph extension problem in the uniform cost case (minLFAu) is NP-complete.

For a complete proof, the reader is referred to the Appendix. The transformation is from the minimal set cover problem (SP5, [30]), which is known to be NP-complete.

Next, we turn to discuss the algorithmic aspects of LFA graph extension. First, we give an Integer Linear Program (ILP) with $O(n^3)$ binary variables for obtaining an exact solution. Due to its complexity, the ILP is expected to work in small networks only. Therefore, we also give a greedy approximation suitable for larger topologies.

Consider a graph $G(V, E)$ with uniform costs. Then, the task is to compute the minimal set of edges $F \subseteq \overline{E}$ so that $\eta(G(V, E \cup F)) = 1$. By Theorem 2, this can be achieved by ensuring that each edge is contained in a triangle. We introduce $\binom{n}{2}$ binary variables $x_{ij} : (i, j) \in E \cup \overline{E}$ to indicate whether the edge (i, j) is to be added to the graph. We set $x_{ij} = 1$ for each edge already in G. Additionally, we introduce another $(n-2)\binom{n}{2}$ binary variables y_{ikj}, whose role will be clear immediately. Consider the ILP:

$$\min \sum_{(i,j) \in E \cup \overline{E}} x_{ij} \tag{5}$$

$$y_{ikj} \leq \tfrac{1}{2}(x_{ik} + x_{kj}) \quad \forall (i, j) \in E \cup \overline{E}, \forall k \in V \setminus \{i, j\} \tag{6}$$

$$x_{ij} \leq \sum_{k \in V \setminus \{i,j\}} y_{ikj} \quad \forall (i, j) \in E \cup \overline{E} \tag{7}$$

$$x_{ij} = 1 \quad \forall (i, j) \in E \tag{8}$$

$$x_{ij}, y_{ikj} \in \{0, 1\} \quad \forall (i, j) \in E \cup \overline{E}, \forall k \in V \setminus \{i, j\} \tag{9}$$

According to the constraint (6), y_{ikj} can only take the value 1, if both edges (i, k) and (k, j) are to be contained in the extended graph. Then, (7) expresses that we want each edge to be contained in at least one triangle. This is because (7) requires that for each $(i, j) \in E \cup \overline{E}$ with $x_{ij} = 1$ there be at least one $k \in V$ with $y_{ikj} = 1$, i.e., both edges (i, k) and (k, j) be in the graph making up a triangle with (i, j). Finally, the objective says that we want this to be achieved with the minimum number of edges. After obtaining an optimal solution x^* to (5)–(9), we add the edges $(i, j) \in \overline{E} : x^*_{ij} = 1$ to G to attain perfect LFA coverage.

The above ILP has $O(n^3)$ binary variables, so solving it to optimality might not always be an option. Therefore, next we

present a greedy heuristics (inspired by [31]) which, in contrast to the ILP, runs in polynomial time. The greedy algorithm is as simple as it can get: in every iteration we add the edge that increases LFA coverage the most.

Algorithm 1 Greedy LFA graph extension for graph $G(V, E)$

1: **while** $\eta(G(V, E)) < 1$
2: $\quad (u, v) \leftarrow \underset{(i,j)\in\overline{E}}{\operatorname{argmax}}\ \eta(G(V, E \cup \{(i, j)\}))$
3: $\quad E \leftarrow E \cup \{(i, j)\}$
4: **end while**

Theorem 5: Given a graph $G(V, E)$ with uniform costs as input, Alg. 1 terminates with full LFA coverage.

Proof: Alg. 1 certainly terminates when all complement edges are added to the graph. Since uniform cost complete graphs have perfect LFA coverage, the statement follows. ■

The algorithm needs a procedure to compute $\eta(G)$. This can be done in $O(n^3)$ using the Floyd-Warshall algorithm to compute the $\mathrm{dist}(\cdot)$ function and another $O(n^3)$ for checking (1) for each node tuple (s, d, n). The procedure is called $|\overline{E}|$ times in every iteration and at most $|\overline{E}|$ iterations are run, which puts the complexity of Alg. 1 to $O(n^3(n^2 - m)^2)$.

B. LFA graph extension: weighted graphs

In contrast to uniform cost graphs, where we could not solve the LFA graph extension problem without changing some shortest paths, in weighted graphs we can. If an edge with sufficiently large cost is added to the graph, then shortest paths remain intact while LFA coverage may improve. Here, and in the rest of this paper, "sufficiently large" will generally mean "larger than the length of the longest shortest path".

Definition 4: LFA graph extension problem in the weighted case (minLFAw): Given a simple, undirected, weighted graph $G(V, E)$ and an integer l, is there a set $F \subseteq \overline{E}$ with $|F| \leq l$ and properly chosen costs, so that $\eta(G(V, E \cup F)) = 1$ and the shortest paths in $G(V, E)$ coincide with the shortest paths in $G(V, E \cup F)$?

In minLFAw, the task is to add edges as well as to choose their cost to attain full LFA coverage, with touching no shortest paths at all. Unfortunately, this latter requirement cannot always be met. Intuitively speaking, if under the actual choice of the edge costs some node d has the property that all traffic destined to d enters via a single edge, say, (n, d), then that edge can never be protected by an LFA: to whatever alternate node n tried to send traffic in case of the failure of (n, d), that traffic would eventually arrive back to n causing a loop. The following theorem makes this idea explicit:

Theorem 6: Let $G(V, E)$ be a simple, weighted graph. Now, there is some integer l so that the LFA graph extension problem in the weighted case (minLFAw) is solvable in G for l, if and only if each $d \in V$ has at least two children in the d-poset that are not ordered with respect to each other.

Proof: First, we show that if each $d \in V$ has at least two children, say, n_1, n_2: $n_1 \not\succeq_d n_2$, then minLFAw is solvable. We give a trivial LFA graph extension. By the assumption: $\nexists u$ so that for each $v \in V \setminus \{d\} : u \preceq_d v$. Thus, for each $u \in V \setminus \{d\}$, there is another node $v \neq d$ so that either $u \not\succeq_d v$ or $u \succ_d v$. In the first case, if $(u, v) \in E$ then v is already an LFA from u to d. If not, add (u, v) to E with sufficiently large cost. The latter case means that u has an ancestor v. Now, there are three cases. Either *(i)* $(u, d) \notin E$ in which case add (u, d) with sufficiently large cost; *(ii)* $(u, d) \in E$ but u is not a child of d, then u already has an LFA to d through edge (u, d); or *(iii)* $(u, d) \in E$ and u is a child of d. In this case, u has at least two parents in the d-poset: d and the parent on the shortest path to the ancestor v which is guaranteed to exist by the assumption $u \succ_d v$. The parent that is not the next-hop then provides an LFA. Since adding high cost edges does not alter shortest paths, we can repeat the above process for each $d \in V$ independently to eventually obtain full LFA coverage.

Second, we show that if the condition does not hold then at least one node cannot have an LFA. Suppose that for some $d \in V$ all children of d in the d-poset are ordered with respect to each other. Then, there is a "minimal" child $n : \forall v \in V \setminus \{d, n\} : n \preceq_d v$. This precisely means that no node fulfills the LFA criterion, so neither n has an LFA in G nor G can be extended with new edges so that is has. ■

Next, we characterize the complexity of minLFAw. In what follows, we suppose that the network satisfies the requirements of Theorem 6, so the existence of a solution is guaranteed.

Theorem 7: The LFA graph extension problem for the weighted case (minLFAw) is NP-complete.

Consult the Appendix for the full proof.

It seems that LFA graph extension is difficult, both in the uniform cost case and the weighted case. Thus, we again present two algorithms, an exact solution for small networks and an approximation for larger topologies. However, before turning to the algorithms themselves, we need to make sure that the problem is solvable in the first place.

Theorem 6 suggests that some networks cannot be extended for perfect LFA coverage without altering the costs. However, this opens the door for a wide selection of strategies, based on whether the operator prefers the invariance of shortest paths or the invariance of the topology. Some strategies would change edge costs but would not add new edges, other strategies would do both in order to minimize the number of shortest paths altered. Discussing all these strategies goes well beyond the scope of this paper. Below, we present a simplistic solution, which changes at most one shortest path and adds at most one edge per each node that violates Theorem 6.

Let $D \subseteq V$ be the set of nodes not satisfying Theorem 6. In addition, let $\mathrm{leaf}(d)$ denote the leaf nodes on the shortest-path tree rooted at d: $\mathrm{leaf}(d) = \{v : \nexists u \in V \text{ so that } u \succ_d v\}$. Finally, choose some $\epsilon < \min_{(i,j)\in E} c(i, j)$.

The idea is to choose some non-transit node v for each destination node d violating Theorem 6 and make it a child of d. This amounts to adding an edge (v, d) if no such edge existed before and setting its cost to ensure that v is not ordered with respect to any other child of d in the d-poset. Obviously, this will bring d to terms with Theorem 6.

Theorem 8: Alg. 2 adds at most $|D|$ edges to the graph and

Algorithm 2 Pre-process graph $G(V, E)$ for minLFAw

```
for d ∈ D
    choose some v ∈ leaf(d)
    if (v, d) ∉ E then E ← E ∪ {(v, d)}
    c(v, d) ← dist(v, d) − ε
    recompute D
end for
```

changes at most $|D|$ shortest paths.

Proof: Let G' denote the graph obtained by executing Alg. 2 on some graph G. The first claim is straight forward. To prove the second one, we show that only the shortest path from the new child v to d changes for each $d \in D$. Suppose some other shortest path, say, from u to w, changed as well. Obviously, this path must contain (v, d). Since $v \in \mathrm{leaf}(d)$, $w \neq d$. Let n be the next-hop from d to w in G'. Now, (v, d) must be contained in the shortest path from v to n as well. Note that n is a child of d. Because $v \succ_d n$ (otherwise, d would not be in D), $\mathrm{dist}(v, d) = \mathrm{dist}(v, n) + \mathrm{dist}(n, d)$. From this, we write $\mathrm{dist}(v, d) + \mathrm{dist}(d, n) > \mathrm{dist}(v, n)$. Due to the way we selected ϵ, this remains true in G' as well: $\mathrm{dist}'(v, d) + \mathrm{dist}'(d, n) > \mathrm{dist}'(v, n)$, which contradicts the assumption that the shortest path from v to n in G' goes through d. ■

Next, we discuss algorithms to solve minLFAw. First, we give an ILP to obtain an exact solution. Let $(s_k, d_k) : k = 1, \ldots, K$ be the set of source-destination pairs (s_k, d_k) with the property that s_k does not have an LFA to d_k. Additionally, let $(u_i, v_i) : i = 1, \ldots, L$ be the set of edges in the complement edge set $\overline{E}$ and let δ_{ik} be an indicator whose value is 1 if edge (u_i, v_i), when added with sufficiently large cost to the graph, would provide an LFA for (s_k, d_k), and zero otherwise. Note that δ_{ik} is well-defined and it is invariant to the operation of adding high cost edges to the graph. Introduce a binary variable x_i for each $i = 1, \ldots, L$ indicating whether edge (u_i, v_i) is to be added to the graph. Consider the ILP:

$$\min \sum_{i=1}^{L} x_i \tag{10}$$

$$\sum_{i=1}^{L} \delta_{ik} x_i \geq 1 \qquad k = 1, \ldots, K \tag{11}$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, L \tag{12}$$

Constraint (11) requires that the edges added to the graph provide LFA for each unprotected source-destination pair, and the objective (10) expresses that we want to achieve this with the fewest edges possible. Readers proficient in combinatorial optimization will recognize the minimal set cover problem in the ILP (10)–(12). Indeed, the ILP requires to find a set of edges that "covers" all the source-destination pairs in that they provide an LFA. As we used the exact same problem to prove the NP-completeness of minLFAw (see the Appendix), we arrive to the interesting conclusion that weighted LFA graph extension is precisely equivalent to the minimal set cover problem.

The ILP has $O(n^2 - m)$ binary variables, which makes it intractable in larger topologies, calling for an approximation. We observe that Alg. 1 readily generalizes to the weighted case. The only modification is that before examining whether a particular edge, when added to the graph, would improve LFA coverage, we must take care of setting its cost sufficiently large. Observing that in every graph conforming to Theorem 6 an edge improving LFA coverage can always be found (see the trivial LFA graph extension in the proof) leads us to a simple generalization of Theorem 5 to the weighted case.

Corollary 1: Given a weighted graph satisfying Theorem 6, the approximate minLFAw algorithm terminates with full LFA coverage.

V. Numerical studies

First, we examine how many edges one must add to a network to achieve full LFA coverage. For conducting the numerical evaluations, we chose topologies that represent a broad selection of today's transport networks. We used inferred ISP data maps from the Rocketfuel dataset [32] (AS1221, AS1239, AS1755, AS3257, AS3967 and AS6461). We obtained approximate POP-level maps by collapsing the topologies so that nodes correspond to cities and we eliminated leaf-nodes. These networks come with inferred link costs. We also chose some network topologies from [33], 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. Note that solving minLFAu and minLFAw yields different results even for uniform costs graphs, as shortest paths are allowed to change in the first case but must remain intact in the latter.

The details are in Table I. Our first conclusion is that, in line with what is reported in the literature [17]–[21], LFA failure coverage in real networks is usually far from being complete. Most results are in the range 75-85%, rarely reaching 95% and never attaining 100%. Curiously, however, we found many cases when full protection could be attained by adding only a few new links. For smaller topologies, only some 1-6 additional links are needed, while in larger and sparser networks we need significantly more links. For instance, we would have to add about one fourth of the number of links originally existing in the German backbone (Germ_50). Additionally, initial LFA coverage tends to be smaller in uniform cost graphs, while more links are needed for full coverage in weighted graphs. Finally, we observe that the greedy algorithm performs quite close to the optimum. This result is expected: LFA graph extension is nothing more than a minimal set cover problem under the hood, and the greedy algorithm has been reported earlier to perform well for this particular problem [31].

Next, we study how robust these results are against the changing of costs. For this, we generated 100 graph instances for each topology, where costs were taken randomly from the range $[1, 100]$ according to a uniform distribution, and we executed the greedy algorithm on the resultant networks. The

Table I: LFA graph extension results: topology name, number of nodes (n) and edges (m); initial coverage (η_0), ILP size, number of added edges ("ext") by the optimal and the greedy ("Gr.") algorithms, and number of link costs changed and edges added in the preprocessing phase ("Pre. c/e") for the uniform cost and the weighted case with real and random costs.

			Uniform cost				Weighted					Weighted random	
Topology				ILP		Gr.		Pre.	ILP		Gr.	Pre.	Gr.
Name	n	m	η_0	ILP size	ext	ext	η_0	c/e	ILP size	ext	ext	c/e	ext
AS1221	7	9	0.833	135 x 126	1	1	0.833	1/1	7 x 11	2	2	1.13 ± 0.18/0.72 ± 0.15	2.85 ± 0.2
AS1239	30	69	0.898	12684 x 12615	6	6	0.877	0/0	107 x 366	6	7	2.68 ± 0.26/1.84 ± 0.22	10.6 ± 0.43
AS1755	18	33	0.889	2634 x 2601	4	4	0.886	0/0	35 x 120	8	8	1.32 ± 0.2/0.92 ± 0.16	7.55 ± 0.3
AS3257	27	64	0.946	9190 x 9126	2	3	0.903	7/7	68 x 280	10	11	4.62 ± 0.42/3.24 ± 0.39	7.69 ± 0.46
AS3967	21	36	0.864	4236 x 4200	7	7	0.743	0/0	108 x 174	9	11	1.53 ± 0.25/1.03 ± 0.19	10.3 ± 0.33
AS6461	17	37	0.919	2213 x 2176	2	2	0.882	3/2	32 x 97	4	4	1.98 ± 0.25/1.18 ± 0.18	4.6 ± 0.3
Abilene	12	15	0.56	767 x 726	6	6	0.56	1/1	44 x 50	7	8	1.35 ± 0.1/1.26 ± 0.18	8.19 ± 0.17
Italy	33	56	0.784	17116 x 16896	12	13	0.784	0/0	228 x 472	17	20	1.82 ± 0.27/1.14 ± 0.2	17.4 ± 0.42
Germany	17	25	0.695	2257 x 2176	5	5	0.695	0/0	83 x 111	9	12	0.6 ± 0.16/0.36 ± 0.11	10.7 ± 0.28
NSF	26	43	0.86	8275 x 8125	9	10	0.86	0/0	91 x 282	11	12	1.24 ± 0.2/0.88 ± 0.17	12.6 ± 0.34
AT&T	22	38	0.823	5023 x 4851	5	6	0.823	0/0	82 x 193	10	13	2.21 ± 0.25/1.57 ± 0.21	10.2 ± 0.35
Germ_50	50	88	0.801	60362 x 60025	21	22	0.92	0/0	194 x 1137	18	21	1.52 ± 0.2/1.28 ± 0.2	26.1 ± 0.49

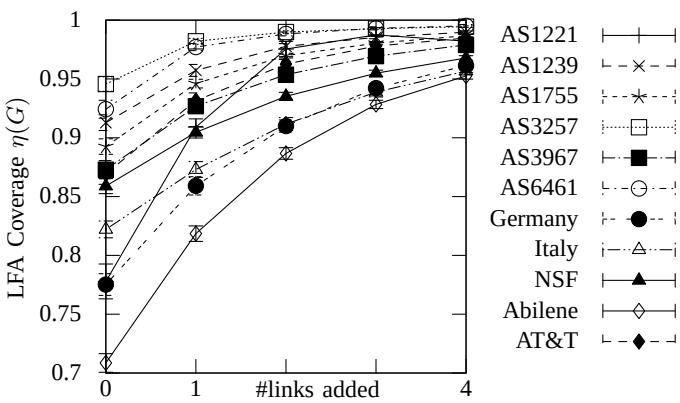


Figure 3: LFA coverage in subsequent iterations of the greedy algorithm in graphs with random costs.

results are given in the last columns of Table I. In addition, the procession of the greedy algorithm is highlighted in Fig. 3. Results displayed are the mean values and the confidence intervals near 95% significance. The results indicate that *(i)* the preprocessing phase is necessary, but in most of the cases it only touches at most 1-3 shortest paths and adds about 1-2 links; *(ii)* some topologies readily lend themselves to LFA extension (in particular, AS1221 and AS6461); *(iii)* the first iteration attains a significant 5-12% improvement in the LFA coverage, while subsequent iterations gradually attain less; and *(iv)* the greedy algorithm usually realizes about 95% LFA failure coverage by adding no more than 2-4 new links.

VI. Conclusions

At the moment, Loop Free Alternates seems the best choice for providing fast protection in pure IP and MPLS/LDP networks. Enabling LFA is a matter of issuing just a handful of configuration commands on modern routers, and this prompted many operators to seriously consider deployment. In this paper, we attempt to help making this decision.

As far as we know, this is the first time that a thorough graph theoretical analysis for LFA failure coverage is given. We showed worst case graphs for LFA and we gave conditions for full coverage, characterizing numerous important network topologies. As many real-world networks do not have 100% LFA coverage, we formulated the LFA graph extension problem to augment graphs with new links for higher coverage. This problem proved NP-hard, but efficiently approximable in practice. Our numerical results suggest that there are some cases when only a minor topology upgrade is enough for 100% LFA coverage, but in most cases significantly more new links are needed. On the other hand, we found that only 2-4 links can bring most networks close to full coverage, and this might be an acceptable price to many operators for being able to benefit from cheap IP-level protection by deploying LFA. For the rest, alternative protection schemes might be more attractive [24].

References

[1] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, 2005.

[2] M. Shand and S. Bryant, "IP Fast Reroute framework," RFC 5714, Jan 2010.

[3] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-Free Alternates," RFC 5286, 2008.

[4] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing," in *IEEE INFOCOM*, 2004.

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

[6] G. Enyedi, G. Rétvári, and T. Cinkler, "A novel loop-free IP fast reroute algorithm," in *EUNICE*, 2007.

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

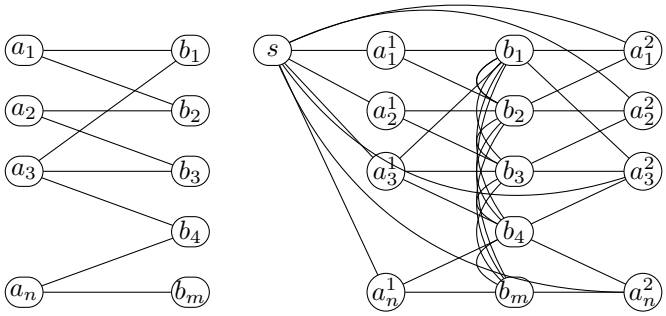
[8] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Multiple routing configurations for fast IP network recovery," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 473–486, 2009.

[9] T. Čičic, A. F. Hansen, and O. K. Apeland, "Redundant trees for fast IP recovery," in *Broadnets*, 2007, pp. 152–159.

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

[11] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, "IP Fast Reroute using tunnels," Internet Draft, Nov 2007.

[12] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using Not-via addresses," Internet Draft, March 2010.



(a) G' (b) G

Figure 4: A minSC instance (a) and the converted graph (b).

[13] G. Enyedi, P. Szilágyi, G. Rétvári, and A. Császár, "IP Fast ReRoute: lightweight Not-Via without additional addresses," in *IEEE INFOCOM Mini-conf*, 2009.

[14] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of NotVia," in *ACM CoNEXT*, 2007.

[15] "Cisco IOS XR Routing Configuration Guide, Release 3.7," Cisco Press, 2008.

[16] "JUNOS 9.6 Routing protocols configuration guide," Juniper Networks, 2009.

[17] P. Francois and 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, and X. Yang, "Evaluation of IP fast reroute proposals," in *IEEE Comsware*, 2007.

[20] M. Menth, M. Hartmann, R. Martin, T. Čičić, and A. Kvalbein, "Loop-free alternates and not-via addresses: A proper combination for IP fast reroute?" *Comput. Netw.*, vol. 54, no. 8, pp. 1300–1315, 2010.

[21] C. Filsfils *et al.*, "LFA applicability in SP networks," Internet Draft, March 2010.

[22] K.-W. Kwong, L. Gao, R. Guerin, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," in *IEEE INFOCOM*, 2010.

[23] C. Reichert and T. Magedanz, "Topology requirements for resilient IP networks," in *MMB*, 2004, pp. 379–388.

[24] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," RFC 4090, 2005.

[25] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. 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, 2008.

[26] T. Čičić, "An upper bound on the state requirements of link-fault tolerant multi-topology routing," in *IEEE ICC*, vol. 3, 2006, pp. 1026–1031.

[27] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, and A. Császár, "Optimizing IGP link costs for improving IP-level resilience," submitted to IFIP Networking 2011.

[28] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Comm. Mag.*, vol. 40, no. 10, pp. 118–124, Oct 2002.

[29] G. Swallow, S. Bryant, and L. Andersson, "Avoiding equal cost multipath treatment in MPLS networks," RFC 4928, June 2007.

[30] M. Garey, , and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[31] L. Lovász, "On the ratio of optimal integral and fractional covers," *Discrete Mathematics*, vol. 13, no. 4, pp. 383–390, 1975.

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

[33] "Survivable fixed telecommunication Network Design library (SNDlib)," http://sndlib.zib.de.

APPENDIX

Definition 5 (SP5, [30]): Minimal set cover problem (minSC): Given a bipartite graph $G'(A \cup B, C)$ and a positive integer k, is there a set of nodes $B^c \subseteq B$ with $|B^c| \leq k$, such that every node in A has a neighbor in B^c?

We make the following trivial assumptions: *(i)* each node $a \in A$ is connected to at least two nodes in B (otherwise covering a is trivial); *(ii)* $|B| \geq 3$; and *(iii)* $|A| \geq 2$.

Proof of Theorem 4: Instead of solving minLFAu directly, we use Theorem 2. Consider the definition:

Definition 6: Minimal triangular problem (minTR): Given a graph $G(V, E)$ and a positive integer l, is it possible to add at most l edges to G so that every edge is contained in a triangle?

Easily, minTR is solvable for l if and only if minLFAu is also solvable for l. Hence, proving that minTR is NP-complete will yield the required result. MinTR is in NP, since a nondeterministic algorithm needs to guess the set of edges F with $|F| \leq l$ and a polynomial time algorithm can verify if every edge is part of a triangle. To prove that minTR is indeed NP-hard, we (Karp-)reduce it to minSC: given a minSC instance with a bipartite graph $G'(A \cup B, C)$ and an integer k, our task is to define an input graph $G(V, E)$ for minTR that is solvable with at most k edges, if and only if the minSC instance is solvable with at most k nodes.

Construct $G(V, E)$ as follows: let $V = A_1 \cup A_2 \cup B \cup \{s\}$, $|V| = 2|A| + |B| + 1$. Denote the nodes in V by $a_i^1 \in A_1$, $a_i^2 \in A_2$, $b_j \in B$ and s, respectively, where $i = 1, \ldots, |A|$ and $j = 1, \ldots, |B|$. Additionally, let $E = E_1 \cup E_2 \cup E_3$ where:
E_1: (a_i^1, b_j) and (a_i^2, b_j) if $(a_i, b_j) \in C$,
E_2: (s, a_i^1) and (s, a_i^2) for $i = 1, \ldots, |A|$,
E_3: (b_j, b_l) for all $j = 1, \ldots, |B|$, $l = 1, \ldots, |B|$ and $j \neq l$.

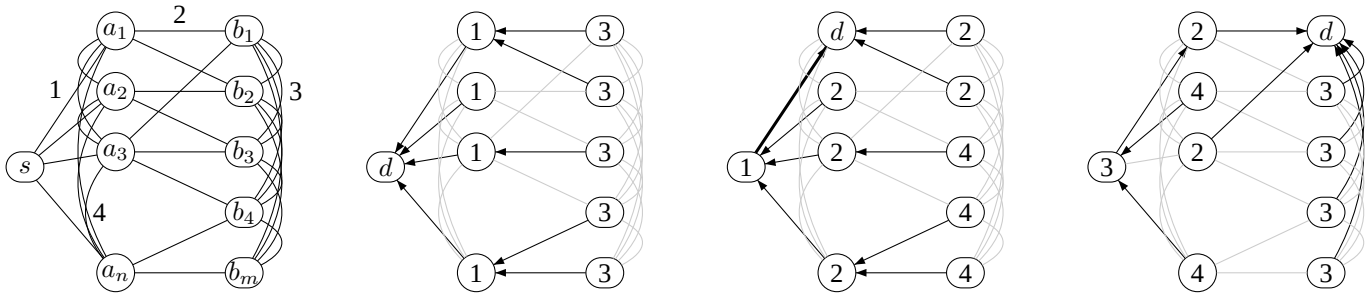
A minSC instance and its transformation are given in Fig. 4.

We say that an edge is protected if it is part of a triangle, unprotected otherwise. We make the following observations: edges in E_1 are protected because of assumption *(i)*, similarly edges in E_3 are also protected because of *(ii)*, while edges in E_2 are all unprotected. The idea is that in order to protect all edges in E_2 we need to add $(s, b_j) : b_j \in B$ edges, called *cover edges* henceforth. Each such (s, b_j) cover edge, when added, protects the $(s, a_i^1), (s, a_i^2) \in E_2$ edges for all $a_i \in A$ nodes adjacent to b_j in G'. Thus, we get a minimal cover of G' exactly when all edges in E_2 become protected.

To conclude the proof we need to show that *(a)* if G' can be covered with k nodes from B, then we can identify k edges to be added to G so that every edge becomes protected; and *(b)* if k edges are added to G so that every edge becomes protected, then we can identify a k node subset of B that covers every node in A.

(a) Let $B^c \subseteq B$ be a cover in G' with $|B^c| \leq k$. Add edges $(s, b_l) : \forall b_l \in B^c$ to E. Then, since B^c is adjacent to every node in A_1 and A_2 (due to it being a cover), every edge in E_2 becomes protected. This is because, the edges $(s, a_i^x) \in E_2$, $(a_i^x, b_l) \in E_1$ and (s, b_l) make up a triangle; where a_i^x is adjacent to b_l and $x = 1, 2$.

(b) Suppose that there is a set of cover edges F' such that $F' = (s, b_j) : b_j \in B' \subseteq B$ and B' is adjacent to every node in A_1 and A_2. Then, by the above reasoning, F' can be converted to a set cover. Let $F \in \overline{E}$ be a set of edges which, when added to G, protect all edges. First, we add all cover edges of F to F'. Now, suppose that some edges in F are not cover edges.



(a) $G(V, E)$ (b) $d = s$ (c) $d \in A$ (d) $d \in B$

Figure 5: The converted graph topology and link costs. Fig. (a) depicts the graph and Fig. (b), (c) and (d) give the shortest paths and distances with different choices of the destination d.

F does not contain edges of the form $(b_i, b_j) : b_i, b_j \in B$ and $(s, a_i^x) : a_i^x \in A_1 \cup A_2$, since these all exist in E. Edges $(a_i^x, b_j) : a_i^x \in A_1 \cup A_2, b_j \in B$ can be dropped from F, as these do not protect any unprotected edge. What remains are edges of the type $(a_i^x, a_j^z) : a_i^x, a_j^z \in A_1 \cup A_2$. Call these *cross edges*. Furthermore, call a node $a_i^x \in A_1 \cup A_2$ protected if the edge (s, a_i^x) is protected. We shall convert cross edges to cover edges before adding to F'. Clearly, we need to consider only those cross edges where both $a_i^1 \in A_1$ and $a_i^2 \in A_2$ are protected by a cross edge, otherwise a cover edge in F' already covers both a_i^1 and a_i^2. Let $A_1' \subseteq A_1$ and the "opposite nodes" $A_2' \subseteq A_2$ be the set of these nodes. Since one cross edge can protect at most two nodes, there are at least $|A_1'|$ cross edges protecting A_1' and A_2', and these are trivially substituted by exactly $|A_1'|$ cover edges: for each $a_i^1 \in A_1'$ choose a neighbor $b_j \in B$ and add the cover edge (s, b_j) to F'. This will protect $a_i^2 \in A_2'$ too. Finally, all nodes B' are adjacent to every node in A_1 and A_2, which completes the proof. ■

Proof of Theorem 7: minLFAw is in NP, since it was formulated as an ILP in Section IV-B. To prove that it is NP-hard, we again reduce it to minSC (see Definition 5): given a minSC instance with a bipartite graph $G'(A \cup B, C)$ and an integer k, our task is to define an input graph $G(V, E)$ for minLFAw, so that minLFAw is solvable by adding at most k edges if and only if minSC is solvable with at most k nodes.

Let us construct $G(V, E)$ as follows: $V = A \cup B \cup \{s\}$ and $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where

E_1: (s, a_i) with cost 1 for each $i = 1, \ldots, |A|$,
E_2: (a_i, b_j) with cost 2 for each $(a_i, b_j) \in C$,
E_3: (b_j, b_l) with cost 3 for all $j = 1, \ldots, |B|$, $l = 1, \ldots, |B|$ and $j \neq l$,
E_4: (a_i, a_j) with cost 4 for all $i = 1, \ldots, |A|$, $j = 1, \ldots, |A|$ and $i \neq j$.

Fig. 5a shows the converted graph for the same minSC instance we used in the previous proof (see Fig. 4a).

The idea here is that we embed G' into G and, by carefully choosing the edge costs, we ensure that achieving perfect LFA coverage in G precisely solves the minSC instance on G'. More formally, we show that some $B^c \subseteq B$ is a cover, if and only if $\eta(G, V \cup F) = 1$ for some set of edges $F : \{(s, b) : b \in B^c\}$ of sufficiently large cost. We discuss the LFA coverage for different destinations separately.

1. $d = s$: As one easily checks in Fig. 5b, each node has an LFA towards destination s. Nodes in A provide LFA for each other as they are not ordered in the d-poset and $|A| \geq 2$ by assumption *(iii)*. Using *(ii)*, one easily shows that nodes in B also provide LFA for each other.

2. $d \in A$: All nodes in $A \setminus \{d\}$ have an LFA towards d, since they are not ordered in the d-poset and $|A| \geq 2$ due to *(iii)* (see Fig. 5c). Similarly, each $b \in B$ has an LFA from some other node in B. This leaves us with the single node s that does not have an LFA to d. This is because all of its neighbors are in A, but d is its next-hop so it can not be used as an LFA and all other nodes in A are its children. Nodes not connected to d can not provide an LFA to s either, since these are all its descendants. What remains are the neighbors of d. So, we have to add an edge (s, b_j) for some $(d, b_j) \in C$ with sufficiently large cost to protect edge (s, d).

3. $d \in B$: First, each $b \in B$ is protected due to *(ii)*. Second, each $a \in A$ that is a neighbor of d is protected by some neighbor $b \in B \setminus \{d\}$. Such neighbor exists due to *(i)*. Similarly, each $a \in A$ that is not a neighbor of d is protected by some neighbor $b \in B \setminus \{d\}$, which again exists due to *(i)*. Again, node s remains without an LFA to d. However, at this point there already is at least one edge $(s, b_j) : b_j \in B$ that we added previously to have an LFA from s to some $a \in A$. This will serve as an LFA for s to d in this case too. Note that it is guaranteed by *(iii)* that at least one such (s, b_j) edge must have been added, and it is enough to have just a single edge from s to B to have an LFA for each $d \in B$.

In summary, in order to have an LFA from s to all nodes in A, we need to add edges from s to some nodes $B^c \subseteq B$. Observe that B^c is adjacent to every node in A, otherwise, we do not have LFA available from s to some of the nodes in A. So B^c is a cover exactly when we have perfect LFA coverage, which completes the proof. ■