Network Optimization Techniques for Improving Fast IP-level Resilience with Loop-Free Alternates

Levente Csikor, Máté Nagy, Gábor Rétvári

Abstract—**Recently, due to the growing need for multimedia communication over IP, IP Fast ReRoute, the IETF standard for fast IP-level failure protection, has become very important. The basic specification for IPFRR is Loop-Free Alternates (LFA). LFA is simple and unobtrusive but, unfortunately, it usually does not provide full protection for all possible failure cases. Hence, many IPFRR proposals have come into existence, providing 100% failure coverage at the cost of significant changes and additional complexity to the IP infrastructure. Not surprisingly, therefore, currently LFA is the only IPFRR solution available in commercial routers. In consequence, there is now a growing need for network optimization techniques for improving LFA coverage in operational networks. In this paper, we present three such techniques. First, we investigate how to augment a network with only a few new links so as to maximize the number of protected failure scenarios. Then, we ask how the same effect can be achieved by optimizing the link costs. Finally, for the first time in the literature we combine the two aforementioned techniques into a single LFA network optimization framework. We show that these problems are all NP-complete and we give exact and approximate algorithms to solve them. Our numerical evaluations show that the combined algorithm is by far the most efficient for LFA network optimization and our methods can bring many networks close to perfect LFA coverage with only minor change in the topology.**

Index terms: IP protection, IP Fast ReRoute, Loop Free Alternates, Graph theoretical analysis, network optimization, combined metrics

I. Introduction

Recently, need for multimedia communication over IP, e.g., VoIP, streaming media, online gaming, video conferencing, and IPTV, has been increasing in an ever faster pace. For the Internet to become a truly ubiquitous platform for delivering dependable multimedia experience, however, IP communication services must guarantee the high reliability and five-nines availability (99.999% uptime) we got used to expect from the PSTN (Public Switched Telephone Network). Although the interruption of connectivity is tolerable for some traditional IP services, like WWW or email, it is devastating for real-time applications.

In an operational network, failures occur frequently due to various reasons, such as the disruption of a link, a router crash, a faulty interface, etc. One of the main design objectives of the Internet has been the ability to recover from failures seamlessly [1]. Consequently, standard intra-domain routing protocols, like OSPF (Open Shortest Path First) [2] and IS-IS (Intermediate System To Intermediate System) [3], were from the outset designed with tolerance for failures in mind. After a failure, adjacent nodes recognize it and distribute this information to every node in the network, which in turn recalculate shortest paths with the failed component removed from the topology. This re-convergence, however, assumes full flooding of new link states, which is a time consuming process (it can take between 150ms and a couple of seconds depending on network size and routers' shortest path calculation capabilities). During this period packets are dropped due to invalid routes. Several studies analyzed the common failure patterns in operational networks and their effect on the convergence of intra-domain routing [4]-[6]. Most recently, Markopoulou et al. showed that more than 85% of unplanned failures affect only links, and 46% of these failures are transient [7]. Unfortunately, such transient failures are very difficult to handle with current intra-domain routing protocols effectively, as just a single flapping interface is enough to keep all routers in the domain busy, constantly flooding the network with link state signaling traffic and recomputing shortest paths [7].

To answer this challenge, the IETF (Internet Engineering Task Force) defined the IP Fast ReRoute Framework (IPFRR, [8]) for native IP-level protection. The main goal is to reduce failure reaction time to tens of milliseconds and improve the handling of transient failures. IPFRR techniques are based on two major principles: *local rerouting* after a failure and sending packets on a *precomputed detour* avoiding global re-convergence. Locality means that only nodes adjacent to a failure know about it and they do not inform others. Precomputed, on the other hand, implies that the protection mechanism is proactive, so detours are computed and installed long before any failure occurs. Thus, if a link or node fails, adjacent nodes are able to switch to an alternate path, this way bypassing the failed component and enabling the intra-domain routing protocol to converge in the background. Note that bypassing a failure can lead to congestion and packet loss in parts of a network [5].

In the past few years, many IPFRR proposals have appeared but only one solution made it into commercial IP routers [9], [10]. This method is called Loop-Free Alternates. In LFA, when connectivity to a next-hop is lost all the traffic is re-routed to an alternate next-hop, called a Loop-Free Alternate, that still has a path to the destination that is unaffected by the failure. The alternate next-hop is selected in a way as to guarantee that it does not pass the packet back, because that would lead to an IPFRR loop and, eventually, to grave congestion and packet loss. Availability of a suitable LFA, however, strongly depends on the actual topology and the link costs. Thus, in most network topologies not all next-hops can be protected with LFA, leaving the network vulnerable to certain failure scenarios. Nevertheless, only this method was able to make its way to commercial routers, so instead of addressing any modifications to existing IPFRR methods or proposing new ones, we rather dealt with topological possibilities to improve LFA coverage.

Levente Csikor is with the *Budapest University of Technology and Economics, Department of Telecommunication and Media Informatics, High Speed Networks Laboratory* (e-mail: csikor@tmit.bme.hu).

Máté Nagy is with the *Budapest University of Technology and Economics, Department of Telecommunication and Media Informatics, High Speed Networks Laboratory* (e-mail: mate.nagy@tmit.bme.hu).

Gábor Rétvári is with the *Budapest University of Technology and Economics, Department of Telecommunication and Media Informatics, High Speed Networks Laboratory* (e-mail: retvari@tmit.bme.hu).

In this paper, network optimization techniques are presented in an attempt to manipulate the topology and link costs to improve the level of protection attainable with LFA. In particular, we propose three methods that promise significant improvement in LFA coverage: the *LFA graph extension* problem is concerned with augmenting a network with as few links as possible to maximize the number of LFA-protected failure cases; the *LFA cost optimization* problem asks for setting link costs for achieving the same objective; and the *combined LFA network optimization* problem asks for both adding new links *and* optimizing link costs to the same end. For each problem, we analyze the computational complexity, we propose approximate algorithms, and we provide extensive numerical experiments demonstrating the viability of our approach.

The rest of the paper is organized as follows. In Section II, related works are discussed. Section III is devoted to introduce the model and notations for our LFA network optimization framework and providing some simple graph-theoretical bounds on LFA coverage. The LFA graph extension problem is discussed in Section IV, the LFA cost optimization problem is studied in Section V, and the combined LFA network optimization problem is considered in Section VI. Finally, we conclude our work in Section VII.

II. Related work

The Loop-Free Alternates method can usually protect only about 50-80% of the possible link failure scenarios, and the level of node protection is even worse [11-14]. Consequently, since LFA appeared many other proposals have surfaced to overcome the limitations inherent to it.

In [11] the authors presented a detailed measurement study of all the factors that on a router influence the convergence time attainable by an intra-domain routing protocol. This is characterized by "detection time + link state information origination time + distribution delay + flooding time + shortest path tree computation + update of routing table". They showed that a couple of hundreds of milliseconds can be achieved with IS-IS in case of link failures. Even though this study demonstrates that sub-second convergence can be provided even with current router technology, unfortunately this is still too large for applications with real-time demands.

The main idea of *Failure-carrying Packets* (FCP [15]) is as follows. A simple backup mechanism can only deal with the failure of single node/link. However, in order to provide guarantees for simultaneous failures of multiple arbitrary nodes/links, the number if precomputed paths needed is extremely high. In FCP, all routers have a consistent view of potential links (i. e. those that are supposed to be operational) called a Network Map. Because of its consistency, all that needs to be carried by packets is information about which of these links have failed. The authors also proposed a variant called Source-Routing FCP providing similar properties even if the network maps are inconsistent.

In *O2 (outdegree 2) routing* [16], each router holds multiple alternate paths through at least two distinct next-hops to each destination, in order to facilitate local failure reaction and loop-free destination based routing. By using two next hops, a node detecting an adjacent failure can re-distribute packets to the remaining next-hop instantly. Consequently, the network must meet a necessary condition: each node must form at least one triangle with its neighbors. The authors show that this necessitates using "joker" links, serving only for protecting certain failure cases instead of actively participating in default packet forwarding.

Another approach is using a small set of backup network configurations, called *Multiple Routing Configurations* (MRC, [17]). For each configuration, the standard routing algorithm calculates configuration specific shortest paths. Thus, for any single link or node failure a nearby router detects it and marks the packet with a backup configuration identifier designating an overlay topology that does not contain the failed component. Studies proved that the number of backup configurations needed is usually only three or four.

A qualitative protectability analysis for a fast resilience scheme called *protection routing* is presented in [18]. Protection routing is based on centralized control over the routing tables [18]. In general, centralized control brings numerous disadvantages, above all scalability and latency in reacting to failures. The latter arises from having to notify a central server and communicate the updated forwarding information back to all affected routers before the network can react to a failure. Protection routing solves this problem through a preventive mechanism, in which a central server pre-computes forwarding decisions for common failure scenarios and downloads these in the routers. Thus, after a failure the appropriate new forwarding state is already available locally. Thanks to the use of a central server, forwarding paths and detours do not necessarily depend on a common set of link weights, leaving broader range for improving the level of protection.

In the IPFRR method called *Not-via addresses* [19], when a failure occurs packets affected by it are encapsulated in a new IP header with an address that explicitly identifies the network component that the detour must avoid. In other words, Not-via uses the destination address in IP packets to mark whether the packet is being forwarded on the default path or in a tunnel along a detour. Unfortunately, at the moment there is no standardized protocol for advertising not-via addresses. Moreover, Not-via brings additional complexity into routing and, if the additional IP header does not fit into the MTU, it can cause packet fragmentation and time-consuming re-assembly at the tunnel endpoint. The *lightweight version of Not-via* [20] was proposed to bring down the management and computational complexity of Not-via. Lightweight Not-via is based on the concept of redundant trees [21]. Redundant trees are basically a pair of directed spanning trees having the appealing property that a single node or link failure destroys connectivity through only one of the trees, leaving the path along the other tree intact. This technique can significantly decrease the number of Not-via address (to a constant 3 addresses per node) and eliminates most of Not-via's computational complexity.

Another family of IPFRR techniques, called *Failure Insensitive Routing*, uses interface specific forwarding (FIR [22], FIFR [23]). FIR handles only link failures while FIFR takes care for node failures as well. If a node receives a packet through an unusual interface, it can infer implicitly the cause of the failure and a next-hop is chosen that bypasses the failed component. Unfortunately, interface specific forwarding is generally not available in IP routers today. In a nutshell, many proposals for IPFRR have appeared but all of them need

significant modifications to current routing protocols or changing IP's destination based forwarding (as in FIR); introduce some forms of signaling to indicate that a packet is on a detour; require extra bits in the IP header (like in MRC); or use tunnels imposing additional address management burden on the operator (like Not-via). Loop-Free Alternates, however, does not require any of these. Hence, it is straightforward to implement LFA in routers and deploy it in operational networks. As such, in our days LFA is the only IPFRR method available in off-the-shelf routers, despite of the fact that it usually cannot provide 100% failure protection in all networks. There is, therefore, an increasing demand for LFA-oriented network optimization methods that can improve the quality of protection provided by LFA in operational networks.

III. Loop-Free Alternates: Analysis

Throughout this paper, a network topology is modeled as 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 an edge as *(i, j)*, where i and j are nodes from V. Link costs are represented by a cost function $c: E \Rightarrow \mathbb{N}$. The cost of a link *(i, j)* is denoted with *c(i, j)*.

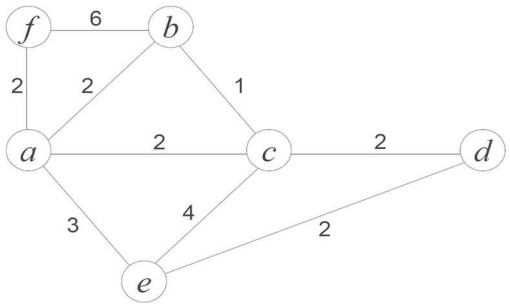


Figure 1. A simple weighted network topology

Perhaps the easiest way to understand how LFA works is through an example. Consider the network depicted in Figure 1 and suppose that node *a* wants to send a packet to the destination denoted by *d*. The next-hop of *a* along the shortest path towards *d* is *c*.

If link *(a, c)* fails, then *a* has to find an alternative neighbor to pass on the packets destined to *d*. It cannot send these packets to, say, *f*, as *f* 's shortest path to *d* goes through itself, so *f* would send the packet back causing a loop. This is because, as mentioned above, in IPFRR only adjacent routers are aware of a failure, so in this case *f* does not know that link *(a,c)* has disappeared and so it blindly uses its shortest path through the failed link. Instead, *a* needs to find a neighbor which is closer to the destination than the route from the neighbor through itself. Such neighbors are called Loop-Free Alternates (LFAs). In general, for some source node *s* and destination *d*, a neighbor *n* of *s* that is not the default-next-hop is a *link-protecting LFA* if

$$dist(n,d) < dist(n,s) + dist(s,d) \quad , \qquad (1)$$

where *dist(x,y)* denotes the length of the shortest path between *x* and *y*. (Note that, for the sake of simplicity, herein we disregard for Equal-Cost MultiPath and we shall concentrate on the link-protecting case exclusively. The development goes similarly when these assumptions do not hold, but the notation is significantly more complex [28].) For example, if link *(a, c)* fails then node *e* is a link-protecting LFA, because its shortest path to destination *d* avoids the failed component. For this source-destination pair, *e* is a *node-protecting LFA* for the failure of node *c*, because its shortest path doesn't go through *c*. Node *e* is also a *per-link LFA* for the link *(a, c)* as it protects all the nodes reachable by *a* through *(a, c)*. Now, consider node *d* as source and node *c* as destination. If link *(d, c)* fails, the only alternate neighbor, node *e,* is not an LFA, because it does not fulfill (1). Hence, this particular failure case cannot be protected by LFA.

To measure LFA failure coverage in a network G over the cost function *c*, we use the simple metric adopted from [24]:

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

One easily sees that $\eta(G,c)=1$ if and only if each node has an LFA towards each other node, and in general $\eta(G,c)$ varies between 0 and 1 depending on the actual network topology and link costs. We tightened this general characterization in [25] as follows.

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

This observation has two important implications. First, there is no network with exactly zero LFA coverage, but as n grows the proportion of LFA-protected failure cases to all failure cases can approach zero arbitrarily close. Second, the Theorem implies that it is the even ring topology that has the smallest LFA coverage out of all 2-connected graphs with the same number of nodes. This is not surprising in light of the fact that rings have been shown to be detrimental to other IPFRR mechanisms as well earlier [26],[27]. For complete proof, see [25]. In [28], we sharpened these bounds as follows:

Theorem 2: For any connected simple graph G with $n > 2$ and any cost function c:

$$\frac{n}{n-1}\,\frac{\Delta/2-1}{\Delta_{max}-1} + \frac{1}{n-1}(\Delta_{max}-1) \le \eta(G,c) \le \frac{n}{n-1}(\Delta-2) + \frac{2}{n-1} \quad ,$$

where Δ is the average node-degree in G and Δ_{max} is the maximum node-degree.

IV. LFA Graph Extension

In this section, we show how to increase the level of LFA protection by cleverly adding new links to the network. For instance, if one added the edge *(d,b)* of cost, say, 10, to the sample topology in Figure 1, then the so far unprotected source-destination pair *(d,c)* would gain an LFA. Motivation for increasing the LFA coverage this way is the recognition that in many cases augmenting the topology with new links, however costly, is still preferred over changing the link costs. This is because very often an operator pays special attention to properly engineer the link costs according to his or her own specific operational objectives, like minimizing delay, balancing load to eliminate congestion, etc., and optimizing

link costs just for LFA would interfere with these operational goals. In these cases, installing new links or leasing additional capacity is an effective way to improve LFA coverage. In order to guarantee that the cautiously engineered shortest paths remain intact, we set the cost of the added links sufficiently high, typically, to a value greater than the length of the longest shortest path. Obviously, we want to install the minimum number of auxiliary links to minimize expenditures and eventually we want to attain full LFA coverage. This problem is called the *LFA graph extension* problem, and it is formally defined as follows [25]:

Definition 1: Given a simple, undirected, weighted graph *G(V, E) and an integer k*, is there a set $F \subseteq \overline{E}$ with $|F| \leq k$ and properly chosen cost function c, so that $\eta(G(V, E \cup F), c) = 1$ and the shortest paths in *G(V, E)* coincide with those in $G(V, E \cup F)$?

Note that $\overline{E}$ denotes the complement of the edge set E. The above definition is deliberately formulated as a decision problem, so that it readily lends itself to the following complexity characterization.

Theorem 3: The LFA graph extension problem is NP-complete.

For a complete proof, see [25]. Usually, instead of the decision form, we want to solve the optimization version where we ask for adding the smallest number of new links to attain complete LFA coverage. Easily, this optimization version is also intractable by Theorem 3. To solve it, an optimal Integral Linear Program (ILP) is proposed in [25]. For large networks, however, the ILP might be impossible to solve to optimality. This raises the demand for efficient and computationally tractable heuristics, which promise a good approximation for the solution of the LFA graph extension proposition. Below, we trace back this question to the *Minimum set cover problem* problem, a well-known NP-compete problem, and we collect some heuristic algorithms from the literature to obtain an approximate solution to it.

Before turning to the algorithms, we note that in certain cases LFA coverage can not be improved to 100% just by adding complementary edges to the network. In [25], a polynomial time preprocessing method is presented that modifies the topology, making the problem solvable. Below, we silently assume that this preprocessing phase has already been applied to the input graph.

A. Model

First, we show that any instance of the LFA graph extension problem can be converted to an equivalent instance of the minimum set cover problem in polynomial time.

Consider the example topology in Figure 2. This graph does not have full link-protecting LFA coverage, as node *e,* for instance, does not have an LFA towards *a*. Let us examine what happens if new links are installed. Not all the complementary edges provide additional protection, e. g., establishing a new link between *a* and *d* would not increase the LFA coverage in the graph. In particular, a new link creates LFA for a well-defined subset of the unprotected node pairs, and this subset is easy to compute. Then, the idea is that if we match the set of unprotected source-destination pairs with the complementary edges that provide LFA to them, then we obtain a bipartite graph, and solving the LFA graph extension problem on the original graph is equivalent to solving the minimum set cover problem on the bipartite graph representation.

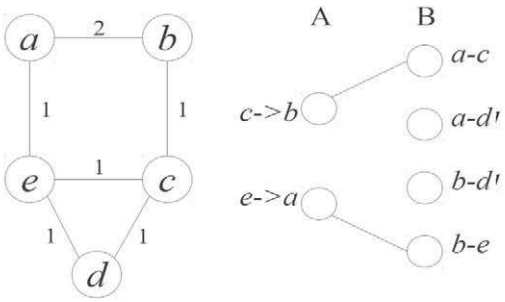


Figure 2: Bipartite graph model for LFA coverage improvement

Let $(s,d)_i : i \in 1,2,\ldots,k$ be the set of source-destination node pairs that have no LFA protection and let $e_j : j \in 1,2,\ldots,l$ be the set of complementary edges in *G(V, E)*. Let $G'(A, B, F)$ be a bipartite graph with node set $A \cup B$ and edge set *F*. Here, we add a node $a_i \in A$ corresponding to each $(s,d)_i$ pair and a node $b_j \in B$ to each e_j complementary edge. Connect some $a_i \in A$ to some $b_j \in B$ in G' if and only if e_j , when added to *G* with suitably large cost, would provide a link-protecting LFA to $(s,d)_i$. Then, it is easy to see that the LFA graph extension problem in *G* is equivalent to the minimum set cover problem in the corresponding $G'(A, B, F)$ bipartite graph formulated as follows: find the minimum set of nodes *C* in *B* (a cover), so that every node in *A* has at least one neighbor in *C*. Note that this is still NP-complete, however, there are plenty of well-tested heuristics available in the literature to approximate it. For the purposes of this paper, we chose four efficient heuristics, namely, the Lovász-Johnson-Chvatal (LJC) algorithm from [31], and the SBT, RSBT and MSBT algorithms from [30]. Note that these heuristics are in fact formulated for the *Minimum hypergraph transversal* problem, but this is equivalent to minimum set cover.

B. Approximate algorithms

Next, we go briefly through each of the selected heuristics. For more details and pseudo-codes, consult [29].

1. The Lovász-Johnson-Chvatal method (LJC): In every iteration the edge is inserted which improves LFA coverage the most. In particular, in every step the highest degree node $v \in B$ is chosen and added to the cover while all its neighbors from *A* are deleted. Unfortunately, the method does not always find a cover that is *minimal in the sense of inclusion*, which practically means that some subset of the resultant cover could also be an adequate cover.

2. SBT: Unlike LJC, the SBT algorithm always finds the set that is minimal in the sense of inclusion. In every iteration, it looks for the node $v \in B$ with the smallest degree and removes it from *B*. If any neighbor of *v* exists in *A* that was only covered by this *v*, then *v* is added to the cover. In this case all its neighbors are considered as covered and we remove them from *A,* and we go on with the next iteration.

3. RSBT: This algorithm works as the reverse of SBT, as in every step it seeks for the node $v \in B$ with the highest degree instead of the smallest degree. All other steps are the same.

4. MSBT: MSBT is a slightly modified version of SBT. Similarly to SBT, in the first phase it looks for a node $v \in B$ with the smallest degree and if there are nodes in A covered only by v, then v will be added to the cover. If the latter condition is not satisfied, then all the neighbors *neigh(v) of v* in A are checked, if there exists a node covered exactly by one node from B other than v. In other words, the algorithm looks for nodes $w \in neigh(v)$ with degree two. We add the nodes connected to these ws from B to the cover, and we remove them from B and all their neighbors from A. Then, we proceed to the next iteration.

Suppose that a heuristic algorithm has terminated with the cover $C \subseteq B$. Then, we obtain an approximation of the original LFA graph extension problem by adding the new links $e_j : b_j \in C$ to the network with sufficiently large cost.

C. Numerical Evaluation

We conducted extensive numerical evaluations with the heuristics on several real-world service provider topologies. The Abilene, Italy, Germany, NSF, AT&T, and the extended German backbone networks are taken from [32] and [33]. All other networks were used from [34]. Wherever available, we used the original link costs that came with the networks. In other cases, costs were set uniformly to one unit. The results are presented in Table 1 with the following notations: n represents the number of nodes and m the number of links in the network; $\eta(G, c)$ is the initial LFA coverage; the column ILP shows the optimal solution for the LFA graph extension obtained by the ILP in [25] (i.e., the minimum number of new links needed to attain full LFA coverage), and the LJC, SBT, RSBT and MSBT columns give the number of links as produced by the respective approximation method. We found that among the heuristics the MSBT algorithm offers the fewest links in general, with LJC as close second. In most cases, the approximation misses the optimum with only a few links. There are some exceptional cases, however, especially large networks, where the difference is more significant. Additionally, the results clearly state that the solution to the LFA graph extension problem is highly topology dependent: for smaller networks usually only a handful of new links is enough, while large networks can require dozens of links to achieve full protection. For such cases, it might be more beneficial to aim for merely increasing LFA coverage into a given safe range, say, above 90%, instead of shooting for complete protection. To see how our heuristics fit for this objective, we took a deeper look at the coverage during the processing of each heuristic in order to assess how LFA coverage improves with each new link added. The results for the Italian backbone are given in Figure 3. We observe that in the first steps it is the LJC algorithm that improves LFA coverage the most. In our case, LFA coverage is increased to 90% with only 3-4 new links, and it rapidly reaches more than 98%. Our conclusion is that solving the LFA graph extension problem requires fundamentally different approximation strategies depending on the actual optimization objective: if full coverage is the aim then the MSBT algorithm is the best choice, while it is the LJC algorithm that ensures the fastest increase in the LFA coverage initially.

1. TABLE: EXACT AND APPROXIMATE SOLUTIONS FOR THE LFA GRAPH EXTENSION PROBLEM IN REAL TOPOLOGIES.

Name	n	m	η(G,c)	ILP	LJC	SBT	RSBT	MSBT
AS1221	7	9	0.809	2	2	2	2	2
AS1239	30	69	0.873	6	6	7	11	6
AS1755	18	33	0.872	7	7	9	12	7
AS3257	27	64	0.923	10	11	10	12	10
AS3967	21	36	0.785	8	11	10	16	9
AS6461	17	37	0.933	3	3	3	4	3
Abilene	12	15	0.56	7	8	9	14	8
Italy	33	56	0.784	17	22	28	39	19
Germany	17	25	0.695	9	12	12	13	11
NSF	26	43	0.86	11	12	13	28	13
AT&T	22	38	0.822	10	12	12	12	11
Germ_50	50	88	0.9	18	21	29	44	25
Arnes	41	57	0.623	24	29	24	30	24
Deltacom	113	161	0.577	80	100	94	131	91
Geant	37	55	0.69	21	23	21	25	21
InternetMCI	19	33	0.904	5	6	5	5	5
Average:	**30.6**	**51.2**	**0.79**	**14.87**	**17.81**	**18**	**24.87**	**16.56**

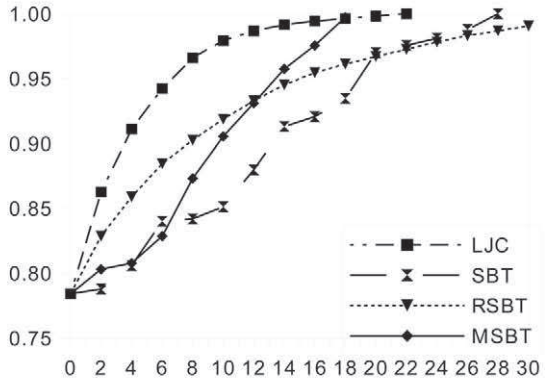


Figure 3. LFA coverage tendency in Italy networks

V. LFA COST OPTIMIZATION

In some networks, adding new physical links to the topology is very difficult or costly, therefore, LFA graph extension cannot be applied for improving failure case coverage. In such cases, it may be worth reconfiguring the link costs, even if this may alter the shortest paths that have been engineered to match the operational goals of the operator previously, because the gain in availability can easily compensate for the loss in forwarding efficiency. As shall be shown in numerical evaluations soon, optimizing costs for LFA can result in high improvements in failure case coverage, to the point that in many network topologies even close to perfect LFA coverage can be reached. The *LFA cost optimization* problem asks for a link cost setting that maximizes the LFA coverage, given inherent limitations of the network topology under consideration. It is formulated as follows:

Definition 2: Given a graph G, is there a cost function c so that $\eta(G, c)=1$?

Again, the LFA cost optimization is formulated above as a decision problem in order to be later subjected to complexity analysis. In practice, however, we are much more interested in the optimization version, which asks for the cost setting that maximizes the LFA coverage. In [28], we proved the following result.

Theorem 4: The LFA cost optimization problem is NP-complete.

Surprisingly, we found that even the task of assigning LFAs to *just a single* destination is already NP-complete. This suggests that this problem is very difficult to solve. And indeed, the ILP we gave in [28] is only applicable in very small networks (up to approx. less than 10 nodes), and thus for larger networks, we need to resort to heuristics.

A. Approximate Algorithms

Below, we present a Simulated annealing-based heuristic, which promises a high quality approximation of the optimal solution in polynomial time [28].

Algorithm 1. Heuristic LFA cost optimization algorithm. Input is graph G.

1: $c \leftarrow$ random_cost(C_{max}), $T \leftarrow T_0$
2: **while** $T > 0$ and $\eta(G, c) < 1$
3: $c' \leftarrow$ argmax $\eta(G, q)$, where $q \in$ neigh(c)
4: **if** $\eta(G, c') > \eta(G, c)$ or $T >$ random(T_0) **then**
5: $c \leftarrow c'$
6: **end if**
7: $T \leftarrow T - 1$
8: **end while**

Our heuristic LFA cost optimization algorithm given in Alg. 1 works as follows. Given the network as input, first the costs are randomized (random_cost function) between 1 and C_{max} (where C_{max} is a problem parameter, chosen as C_{max}=20 below) and the initial temperature T is set to T_0 (again a parameter, set to 150 in our numerical studies). While the temperature T is greater than 0 and $\eta(G,c)$ is less than 1, we perform the following iteration: take the actual cost function and increase or decrease (if possible) the cost by one at exactly one link. Line 3 searches for the best such neighbor. Do this for all links, and choose the cost function that produces the highest LFA coverage. If the new cost vector is better than the present phase (in terms of η), it is unconditionally accepted. Otherwise, accept it only if a random number generated between 0 and T_0 is less than the actual temperature T as described in Line 4. Decrease T and proceed to the next iteration. This way, the algorithm easily escapes from local minima at the initial steps, to eventually settle in a good local minimum by only letting greedy steps when T is low. We also apply a tabu list of size 20 to preclude the algorithm from oscillating. In order to let the algorithm discover the largest range of the problem space possible, we execute N rounds (where in our evaluations N=500) and we choose the cost function c^* that yields the highest LFA coverage in all rounds.

B. Numerical Evaluation

We conducted thorough numerical studies in order to assess the extent to which LFA coverage can be improved by optimizing link costs. The numerical evaluations were run on the same networks as presented in the previous section. Table 2. shows the results, in order of appearance: characteristics of the topologies (name, number of nodes n, number of edges m), LFA coverage $\eta(G, c)$ obtained by the original cost setting c, and the LFA coverage $\eta(G, c^*)$ for the best cost function c^* obtained by the heuristic algorithm presented previously.

2. TABLE: APPROXIMATE SOLUTIONS TO THE LFA COST OPTIMIZATION PROBLEM IN REAL TOPOLOGIES

Name	n	m	η(G, c)	η(G, c*)
AS1221	7	9	0.809	0.833
AS1239	30	69	0.873	0.957
AS1755	18	33	0.872	0.98
AS3257	27	64	0.923	0.997
AS3967	21	36	0.785	0.967
AS6461	17	37	0.933	0.996
Abilene	12	15	0.56	0.701
Italy	33	56	0.784	0.919
Germany	17	25	0.695	0.889
NSF	26	43	0.86	0.95
AT&T	22	38	0.822	0.984
Germ_50	50	88	0.9	0.934
Arnes	41	57	0.623	0.702
Deltacom	113	161	0.577	0.662
Geant	37	55	0.69	0.74
InternetMCI	19	33	0.904	9.932

The most important observation is that in many real network topologies close to perfect LFA coverage can be achieved with cost optimization. Nevertheless, there were some exceptional topologies where LFA cost optimization was less appealing. Our results indicate that LFA cost optimization greatly helps the operator to provide higher availability in the network, even when adding new connectivity to the topology is not feasible. In addition, we found that the running time of the approximation algorithm strongly depends of the topology, in particular, on the number of links and nodes. In some cases, all 500 rounds terminate in just a couple of minutes, but sometimes it takes a couple of hours. Note that running time is not that important in practice, because LFA cost optimization is performed offline only once, before the final deployment of a network.

VI. COMBINED LFA NETWORK OPTIMIZATION

So far, we have shown that both adding new links to a network and optimizing link costs are effective ways of improving the LFA coverage in operational networks. It is of question, however, to what extent the combination of these two methods can be effective for LFA-based network optimization. Hence, in this section we propose the *combined LFA network optimization* problem as the combination of the aforementioned two methods. This problem is formulated as follows:

Defintion 3: Given a simple, undirected, weighted graph $G(V,E)$ and a positive integer k, is there a set $F \subseteq E$ with $|F| \leq k$ and properly chosen cost function c, so that $\eta(G(V,E \cup F),c)=1$?

The difference from the LFA graph extension problem is that in the above formulation we allow for link costs and, consequently, the shortest paths to change. Note that for k=0 we obtain the LFA cost optimization problem, which immediately implies the following result.

Theorem 5: The combined LFA network optimization problem is NP-complete.

Easily, again an optimization version can be defined which asks for the minimum number of new links and an appropriate cost function, which, when applied to the network, result complete LFA coverage.

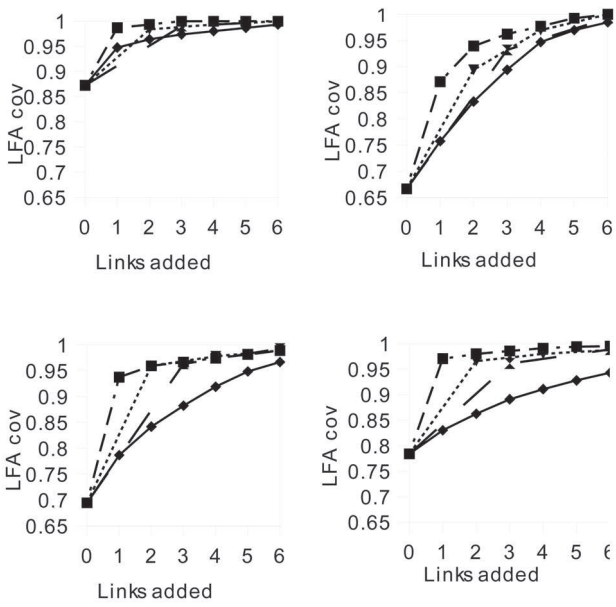
A. *Approximate algorithm*

From all the optimization problems treated so far, combined LFA network optimization is the most difficult. Therefore, instead of aiming for an optimal solution, in this paper we propose a heuristic algorithm based on the consecutive application of the heuristics presented for the individual subproblems in the previous sections. In particular, in every iteration we execute an LFA graph extension phase followed by an LFA cost optimization phase. In the LFA graph extension phase, we add l new links to the network (where l is a problem parameter) by using the LJC algorithm. This algorithm is chosen because it promises the largest increase in LFA coverage. In the LFA cost optimization phase, we compute a link cost setting that approximately maximizes the LFA coverage on the augmented graph obtained in he previous phase. The two phases are applied iteratively one after the other, until the LFA coverage reaches 1.

B. *Numerical evaluations*

In order to evaluate the performance of the combined algorithm as compared to the individual algorithms, we conducted several rounds of numerical experiments. Herein, we present the results for only a subset of the network topologies introduced in the previous sections. Similar results were obtained for the rest of the topologies.

First, we were curious as to what is the optimal setting for the parameter l. We experimented with the settings $l \in [1,2,3]$, as during our numerical studies we found that at most 3 new links is always enough to realize a reasonable improvement in LFA coverage. The results can be seen in Figure 4. For each network topology, the simple dashed line represents the LFA coverage realized by LFA graph extension alone (i.e., with no LFA cost optimization phase applied) as of the LJC algorithm, and the dotted dashed line, the fine-dotted line and the solid line give the LFA coverage for the combined algorithm for the case $l=1$, $l=2$, and $l=3$, respectively. Easily, for the case when $l=1$ we have a data point for each iteration, but for other cases some of these internal data points are missing, as improvement only appears after adding more than one link in such cases. The results show that the combined algorithm performs best when in every LFA graph extension phase we add only a single link, and this is immediately followed by a cost optimization phase. In consequence, we used the setting $l=1$ in the rest of the study.



4. Figure: Results of the several combined metrics (AS1755, Abilene, Germany, Italy)

After fixing l, we turned to the main question of our numerical studies: to what extent the combined algorithm outperforms previous algorithms? The results for some selected topologies are given in Table 3, which shows in order of the appearance: the characteristics of the topologies (name, number of nodes n and edges m); the initial LFA coverage $\eta(G,c)$; LFA coverage achieved with simple LFA graph extension in step 1, 2, 3, 4, 5, 6, and the number of links needed for full coverage (denoted by hashmark); and finally the same results obtained by the combined LFA network optimization algorithm. For the combined algorithm, the LFA coverage values in bold highlight the cases when the LFA cost optimization phase did not improve LFA coverage at all. Our results suggest that the combined algorithm reaches complete coverage with much fewer links than the LFA graph extension algorithm (the difference is highlighted in Figure 5). This is

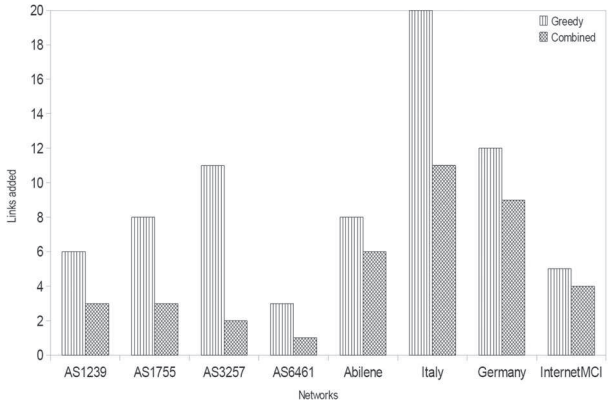


Figure 5.
Number of links needed to be added using the different algorithms

Topology				LFA Graph Extension							Combined LFA Optimization						
Name	n	m	η(G,c)	1	2	3	4	5	6	#	1	2	3	4	5	6	#
AS1239	30	69	0.873	0.932	0.971	0.985	0.992	0.997	1	**6**	0.994	0.998	1	-	-	-	**3**
AS1755	18	33	0.872	0.947	0.964	0.973	0.980	0.986	0.993	**8**	0.986	0.993	1	-	-	-	**3**
AS3257	27	64	0.923	0.971	0.983	0.987	0.990	0.991	0.993	**11**	0.998	1	-	-	-	-	**2**
AS6461	17	37	0.933	0.985	0.996	1	-	-	-	**3**	1	-	-	-	-	-	**1**
Abilene	12	15	0.56	0.757	0.833	0.894	0947	0.969	0.985	**8**	0.871	0.939	**0.962**	**0.977**	0.992	1	**6**
Italy	33	56	0.784	0.830	0.862	0.891	0.911	0.928	0.943	**20**	0.970	**0.980**	**0.985**	**0.991**	**0.994**	**0.995**	**11**
Germany	17	25	0.695	0.786	0.841	0.882	0.920	0.948	0.966	**12**	0.937	**0.959**	**0.966**	**0.974**	**0.981**	**0.988**	**9**
InternetMCI	19	33	0.904	0.973	0.985	0.994	0.997	1	-	**5**	0.991	**0.994**	**1**	-	-	-	**4**

TABLE 3. RESULTS FOR THE LFA GRAPH EXTENSION AND THE COMBINED LFA NETWORK OPTIMIZATION ALGORITHMS.

not surprising, as the combined algorithm is free to reconfigure link costs, which the LFA graph extension algorithm is not permitted to do.

We find that the combined algorithm significantly reduces (in average by more than 50%) the number of additional links necessary for reaching 100% LFA coverage. We also found that it is not worth running all the 500 rounds of the LFA cost optimization algorithm as we did previously, because in the combined algorithm the optimum was always realized in less than 200 rounds.

VII. CONCLUSIONS

In this paper, we proposed several network optimization algorithms for improving the level of fast IP-level resilience using Loop-Free Alternates, the only IPFRR technique available in commercial routers out-of-the-box today. First, we described the LFA graph extension problem, which asks for establishing new links to increase LFA coverage without altering the shortest paths in the network. Second, we presented an LFA cost optimization problem which, instead of adding new links, rather calls for modifying link costs to instantiate new LFAs. Finally, for the first time in the literature we presented a combined formulation. On the theoretical side, we found that each of the three network optimization problems are intractable. On the practical side, we proposed several approximation strategies for solving the problems on real instances and our numerical results suggest that each method is suitable to attain a 10-40% improvement in LFA coverage. We also found that the results strongly depend on the actual network topology. The combined algorithm was found to produce the best results, indicating that in many real networks complete LFA-based protection is attainable with adding only a few new links. We believe that the wide spectrum of LFA network optimization strategies presented in this paper provide a rich set of options for operators to choose from, according to their own preference on whether it is economically more feasible to add new links, change costs, or do both, to improve LFA-protection in their network.

Future works involves fine-tuning the parameters of the approximation algorithms, and a customization of the algorithm for the more realistic cases when, for instance, not all source-destination pairs need to be protected or some source-destination pairs have higher priority than others; only certain shortest paths must be retained but others can change arbitrarily; or the costs are optimized both for LFA coverage *and* to provide efficient utilization of the network with respect to the expected traffic matrix.

ACKNOWLEDGMENT

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

REFERENCES

[1] D. D. Clark, „The design philosophy of the DARPA internet protocols," SIGCOMM, Computer Communications Review, vol. 18, no. 4, pp. 106-114, Aug. 1988.

[2] J. Moy, „OSPF Version 2," RFC 2328, Apr. 1998.

[3] ISO, „Intermediate System-to-Intermediate System (IS-IS) Routing Protocol," ISO/IEC 10589, 2002.

[4] C. Labovitz, G. R. Malan, F, Jahanian, „Internet Routing Instability," IEEE/ACM Transactions on Networking, vol. 6, no. 5, pp. 515-528, 1998.

[5] S. Iyer, S. Bhattacharyya, N. Taft, C. Diot, „An approach to alleviate link overload as observed on an IP backbone," in Proc. Infocom, 2003.

[6] ETSI Document EN 300416 V1.2.1, „Network Aspects (NA): Availability Performance of Path Elements of International Digital Paths," Aug. 1998.

[7] A. Markopoulou, G. Iannacone, S. Bhattacharyya, C-N. Chuah, C. Diot, „Characterization of Failures in an IP backbone," in Proc. IEEE Infocom, Mar. 2004.

[8] M. Shand, S. Bryant, „IP Fast Reroute Framework," RFC 5714, Jan. 2010

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

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

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

[12] S. Previdi, „IP Fast ReRoute technologies," APRICOT, 2006.

[13] M. Gojka, V. Ram, X. Yang, „Evaluation of IP fast reroute proposals," in IEEE Comsware, 2007.

[14] 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., vol. 54, no. 8, pp. 1300-1315, 2010.

[15] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker and I. Stoica, „Achieving convergence-free routing using failure-carrying packets," in. Proc. SIGCOMM, 2007.

[16] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. Schrodi, Y. Glickman, C. Winkler, „Improving the resilience in IP Networks," in. Proc. HPSR, 2003.

[17] A. Kvalbeinm A. F. Hansen, T. Čičić, S. Gjessing, O. Lysne, „Fast IP network recovery using multiple routing configurations," in Proc. IEEE INFOCOM, 2006.

[18] 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 is available in Tech. Rep. 2009, University of Pennsylvania, 2010.

[19] S. Bryant, M. Shand, S. Previdi, „IP fast reroute using Not-via addresses," Internet Draft, available online: http://www.ietf.org/internet-drafts/draft-ietf-rtwg-ipfrr-notvia-addresses-00.txt, Feb. 2008.

[20] G. Enyedi, P. Szilágyi, G. Rétvári, and A. Császár, "IP Fast ReRoute: Lightweight Not-Via without Additional Addresses", in Proc. INFOCOM, pp.2771-2775, 2009.

[21] M. Médard, R. A. Barry, S. G. Finn, R. G. Galler, „Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," IEEE/ACM Transactions on Networking, vol. 7, no.5, pp. 641-652, Oct. 1999.

[22] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, C.-N. Chuah, „Proactive vs. Reactive Approaches to Failure Resilient Routing", in Proc. IEEE INFOCOM, Hong Kong, 2004.

[23] 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 IEEE INFOCOM, 2005.

[24] A. Atlas, A. Zinin, „Basic specifiaction for IP fast reroute: Loop-Free Alternates," RFC 5286, 2008.

[25] G. Rétvári J. Tapolcai, G. Enyedi, A. Császár, „IP Fast ReRoute: Loop Free Alternates Revisited", in Proc. IEEE INFOCOM, 2010.

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

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

[28] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, A. Császár, "Optimizing IGP Link Costs for Improving IP-level Resilience," to appear at DRCN, 2011.

[29] M. Nagy, G. Rétvári, "An Evaluation of Approximate Network Optimization Methods for Improving IP-level Fast Protection with Loop-Free Alternates", to appear at RNDM, 2011.

[30] B. Mazbic-Kulma and K. Sep, "Some approximation algorithms for minimum vertex cover in a hypergraph," in Computer Recognition Systems 2 (M. Kurzynski, E. Puchala, M. Wozniak, and A. Zolnierek, eds.), vol. 45 of Advances in Soft Computing, pp. 250–257, Springer Berlin / Heidelberg, 2007.

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

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

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

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

ABOUT AUTHORS

Levente CSIKOR was born in 1986 and graduated at Budapest University of Technology and Economics in technical informatics in 2010. Now he is a second-year PhD student at the High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, BME. He has experience in C/C++/LEMON, Java/JUNG, Linux, web development and databases. Currently he is involved in researching IP Fast ReRoute techniques, network optimization algorithms and heuristics to improve fast IP level resilience with Loop-Free Alternates.

Máté NAGY was graduated at Budapest University of Technology and Economics in electrical engineering in 2010. He spent a semester in Mikkeli University of Applied Sciences that raised up his interest in infocommunication. Now he is a software developer at Ericsson Magyarorszag Kft. and also takes part in researching IP FastReRoute solutions. He has experience in C/C++/LEMON, Java, Python and in Linux.

Gábor RÉTVÁRI received the M.Sc. and Ph.D. degrees in electrical engineering from the Budapest University of Technology and Economics (BME), Budapest, Hungary, in 1999 and 2007, respectively. He is now a Research Fellow at the High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, BME. His research interests include QoS routing, Traffic Engineering and the networking applications of computational geometry and the mathematical theory of network flows. He is a Perl expert, maintaining numerous open source scientific tools written in Perl, C and Haskell.