An Evaluation of Approximate Network Optimization Methods for Improving IP-level Fast Protection with Loop-Free Alternates

Máté Nagy, Gábor Rétvári
High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Email: {mate.nagy, retvari}@tmit.bme.hu

***Abstract*—Demand for fast failure recovery in modern IP-based networks has become compelling recently. Loop-Free Alternates (LFA) is a simple IP Fast ReRoute (IPFRR) specification proposed by the IETF that does not require profound changes to the network infrastructure before deployment. However this simplicity has a disadvantage, in that usually LFA does not provide complete protection for all possible failure cases in a general topology. The LFA graph extension problem asks for adding new links to the network in an attempt to improve the failure case coverage. Unfortunately, this problem is NP-complete. In this paper, we give a detailed graph model for this problem, for the first time formulating it both for the link and node protecting cases, and we propose several fast approximation algorithms to solve it. We compare the performance of the algorithms in extensive numerical studies and we conclude that the optimum can be approximated well in most cases relevant to practice.**

***Index Terms*—IP Fast ReRoute, Loop-Free Alternates, link and node protection, heuristics**

I. Introduction

Throughout the last few years, the amount of streaming media traffic transmitted over the Internet has increased significantly. This kind of traffic is more sensitive to delay than to packet loss, therefore the need to improve service reliability and availability has become more and more stressing to operators. Unfortunately, the state-of-the-art IP protocol suite is not adequate to meet these needs. The maximum affordable recovery time from a failure without severe degradation to video service quality is about 10-50 ms, yet recovery with today's Interior Gateway Protocols (IGPs) takes about ten times more than that [1].

Consequently, the IETF has initiated the IP Fast ReRoute (IPFRR [2]) framework to reduce the routing convergence time to the critical tens of milliseconds. IPFRR is based on two fundamental design principles. First, recovery in IPFRR is *proactive*, meaning that backup routes are calculated, and installed into the forwarding plane, well before a failure occurs. Second, IPFRR adopts a *local* rerouting scheme, that is, only routers directly adjacent to the failed component participate in the recovery process. This allows to eliminate one of the most time-consuming parts of the IGP recovery process: flooding the changed routing information throughout the network. Instead, when a network element becomes unavailable, its neighbors immediately switch to the backup routes and traffic flows without major disruptions while the IGP converges in the background.

Perhaps the simplest realization of the IPFRR framework is Loop-Free Alternates (LFA [3]). In LFA, when a router detects the loss of connectivity to one of its next-hops, it redirects the affected traffic to an alternate next-hop, called a Loop-Free Alternate, that still has an intact route to the destination. LFA can be implemented as an extension to IGPs, it is unobtrusive and therefore easily deployable. This simplicity, however, comes at a severe price: depending on the network topology and link costs, there might occur failure scenarios in the network for which no LFA exists and so no fast protection can be provided. Consequently, many alternative IPFRR proposals have come to existence lately, each providing 100% failure coverage at the price of increased management burden and deployment complexity [4], [5].

With its standardization and appearance in commercial off-the-shelf routers [6], [7], LFA has received renewed interest both from theoreticians and practitioners recently. Accordingly, there have been various efforts to improve the level of protection provided by LFA [8]–[10], one of the most recent of which is LFA graph extension [11]. The LFA graph extension problem asks for adding the smallest number of new links to the network so that, on the one hand, LFA failure case coverage becomes 100% and, on the other hand, shortest paths remain intact. The latter requirement is important, as shortest paths are usually engineered with great care to reflect crucial operational concerns of the operator [12]–[14]. A closely related problem is LFA graph improvement, where the task is to maximize LFA coverage when adding only a limited number of new links. As it turns out, these problems are all NP-complete. Apart from the NP-completeness proofs, [11] also presents an Integer Linear Program (ILP) and a greedy heuristic. Unfortunately, the model is only given for single link failures in a simplistic network model in which no equal cost paths are assumed, and no account is made as to whether

The second author was supported by the Janos Bolyai Fellowship of the Hungarian Academy of Sciences.

978-963-8111-77-7

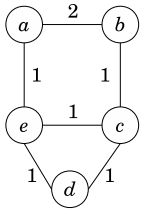


Figure 1: A simple network with link costs.

the greedy heuristics is suitable to efficiently approximate the optimal solution.

In this paper, we tackle the above challenges. The main contributions are as follows. First, building on [11], we provide a sophisticated model for LFA network optimization problems. We also give hints as to how to augment the model to treat Equal-Cost MultiPath (ECMP) and broadcast LANs. Second, we extend the model to handle node failures and the cases of both link and node failures. Third, we collect several optimal and heuristic algorithms from the literature to solve the LFA graph extension problem and we analyze their theoretical behavior. In particular, we find that the optimum can be approximated within a logarithmic factor. Finally, we compare the algorithms in extensive numerical evaluations and we observe that LFA graph extension and LFA graph improvement require fundamentally different approximation strategies.

The rest of the paper is organized as follows. Section II gives an overview of LFA and the related network optimization problems. Section III introduces the mathematical model and describes optimal and approximation algorithms. Numerical results are described in Section IV and finally, Section V concludes the paper.

II. Preliminaries

Throughout this paper, we model the network topology by a simple, undirected weighted graph $G(V, E)$ where V is the set of nodes and E is the set of edges. Let $n = |V|$ and $m = |E|$, let $\overline{E}$ denote the complement edge set, let $\deg(v)$ denote the node degree of $v \in V$ and let $\mathrm{neigh}(v)$ be the set of neighbors of v in G. Initially, for the sake of simplicity we assume that the network contains no broadcast LANs and there is no support for Equal-Cost MultiPath (ECMP). Accordingly, we presume that ties between equal cost shortest paths are broken arbitrarily. These assumptions will be relaxed later. We further assume that in every IPFRR cycle either one link or one router can fail, and when connectivity to some neighbor is lost, a router is not able to determine whether it is the link to the neighbor or the neighbor itself that has failed, so it uses the pessimistic assumption and presumes a node failure. This failure model is in line with current IP practice [15], [16].

Perhaps the easiest way to understand LFA is through an example. Consider the network in Fig. 1. If a packet is sent from node a to node d, the first hop along the shortest path is node e. If the link between a and e fails, a have to look for another neighbor to forward traffic to, which can then pass it on to d. Note, however, that not all neighbors suit, because if a neighbor's shortest path to d went through a, then it would immediately pass the packet back to a causing a forwarding loop (recall that the neighbor is not aware of the failure). The neighbors of a whose shortest path to d does not traverse a are called *link-protecting LFAs* from a to d. Formally, link-protecting LFAs fulfill the following loop-free condition [3]:

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

where s is the source of the packet, d is the destination, n is an LFA candidate neighbor of s and $\mathrm{dist}(x, y)$ denotes the shortest path distance between some x and y. In the above example, $\mathrm{dist}(b, d) = 2$, $\mathrm{dist}(a, d) = 2$ and $\mathrm{dist}(b, a) = 2$, and therefore b is a link-protecting LFA from a to d. Note, however, that in the sample network c does not have a link-protecting LFA to b as both its candidate neighbors d and e reach b through it. The same applies to (e, a). The level of LFA link protection $\eta_{\mathrm{LP}}(G)$ in a network G is measured as the proportion of the protected vs. all source-destination pairs [11]:

$$\eta_{\mathrm{LP}}(G) = \frac{\#\ (s,d)\ \text{pairs with link-protecting LFA}}{\#\text{all}\ (s,d)\ \text{pairs}} \ . \tag{2}$$

For our sample topology, $\eta_{\mathrm{LP}}(G) = 0.9$.

Not just that b protects against the failure of link (a, e) but it also protects against the failure of the next-hop e itself. This is because the shortest (b, d) path does not cross e. Formally, a node n is a *node-protecting LFA* from s to d if, besides (1), it also satisfies the condition

$$\mathrm{dist}(n, d) < \mathrm{dist}(n, e) + \mathrm{dist}(e, d) \ , \tag{3}$$

where e is the default next-hop from s to d. Special care must be taken, however, when $e = d$, that is, when d is the immediate next-hop of s. Since no LFA can protect against the failure of the destination node itself, in such cases s relaxes the pessimistic failure assumption and presumes that it is only the link (s, d) that failed and not d itself, and thus it can resort to a link-protecting LFA. This treatment of the *last-hop problem* is common in IPFRR [17].

In our example, $\mathrm{dist}(b, d) = 2$, $\mathrm{dist}(b, e) = 2$ and $\mathrm{dist}(e, d) = 1$ and so (3) holds. Some quick calculation yields that there are four source-destination pairs that are without node-protecting LFA in the above example: (c, b) and (e, a) are unprotected as the last-hop exception applies and no link-protecting LFA is available, and (d, a) and (d, b) does not have node-protection as their LFAs only fulfill (1) but not (3). LFA coverage $\eta_{\mathrm{NP}}(G)$ for the node-protection case is defined in similar vein to (2), with the slight modification that for (s, d) pairs for which d is the next-hop of s we only check condition (1) but not (3), while for all other source-destination pairs we check both. In our example, $\eta_{\mathrm{NP}}(G) = 0.8$.

There are several ways to improve the LFA protection in the network. A plausible choice would be to optimize the IGP link costs [8]–[10], but this would alter shortest paths. Instead, in this paper we adopt the approach from [11] and aim for increasing the LFA coverage by cleverly adding new links to the network without touching the shortest paths in any ways.

The *LFA graph extension* problem is defined as the task to augment a weighted graph with the minimum number of new edges with properly selected costs, so that the LFA coverage becomes 100% and the shortest paths remain in place:

Definition 1: LFA graph extension problem (minLFA): 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 $G(V, E \cup F)$?

The above definition is straightforward to adapt to the link-protecting as well as the node-protecting case by substituting the appropriate definition of $\eta_{\mathrm{LP}}(G)$ or $\eta_{\mathrm{NP}}(G)$. In [11], the following complexity characterization is given.

Proposition 1: The LFA graph extension problem for the link-protecting case is NP-complete.

Easily, the optimization version, which asks for the minimal l for which the above condition holds, is also intractable. The same applies to the *LFA graph improvement* problem, where, given some integer $l > 0$, the aim is to add at most l new links that improve the LFA coverage the most. Similar is the case for the node-protecting versions of the problems. Therefore, it is hard to obtain optimal solutions in large networks within acceptable running time, which calls for efficient heuristics. The rest of the paper is devoted to present such heuristic algorithms, to reason about their theoretical properties and to evaluate their performance in real-life networks.

III. Solving the LFA graph extension problem

In this section, we show algorithms to obtain optimal and approximate solutions to the LFA graph extension problem. The objective is to find the smallest number of new links that increase $\eta(G)$ to 1 both for the link-protecting and the node-protecting cases without altering the shortest paths. Note that this latter requirement is easy to satisfy, as it is enough to ensure that the links we add to the network are of sufficiently high cost, say, larger than the length of the longest shortest path. First, we give an elaborate graph model of the problem and then we turn to the algorithmic strategies.

Before delving into the details, an important note is pertinent here. It is shown in [11] that in certain cases LFA coverage cannot be improved to 100% just by adding new links of large cost to the network. Such is the case when there exist nodes into which all traffic enters via a single router. The authors propose a polynomial time preprocessing algorithm, which changes at most one shortest path and/or adds at most one link per problematic node, and yields a slightly modified graph on which the problem is guaranteed to be solvable. In the rest of the paper, we assume that the network has been properly preprocessed so that there always exists a solution to the optimization problems we treat.

A. Model

The first step to solving the LFA graph extension problem is to build a suitable model. First, we discuss the link-protecting case, then we extend the model to LFA node-protection and finally we cover some practical concerns.

Consider the sample topology in Fig. 1. Previously, we found that the graph does not have full link-protecting LFA coverage. For example, node e does not have an LFA to a. Our aim is to install a new link of high cost into the network so that e gains an LFA to a. One easily sees that link (b, e) is suitable, as with this link in place e could use a as an LFA. Similarly, adding link (a, c) would create an LFA for the other unprotected source-destination pair, (c, b). Observe, however, that not all the complement edges provide additional protection. For example, adding a link between a and d would not increase the link-protecting LFA coverage. In general, a new link can provide LFA to several source-destination pairs, and an unprotected source-destination pair could obtain an LFA from several complement edges. Our task is to find the smallest subset of the complement edge set $\overline{E}$ so that each unprotected source-destination pair gets and LFA. The general graph model for the LFA graph extension problem is based on the idea that this task is easy to represent as a minimum cover problem over a suitably defined bipartite graph.

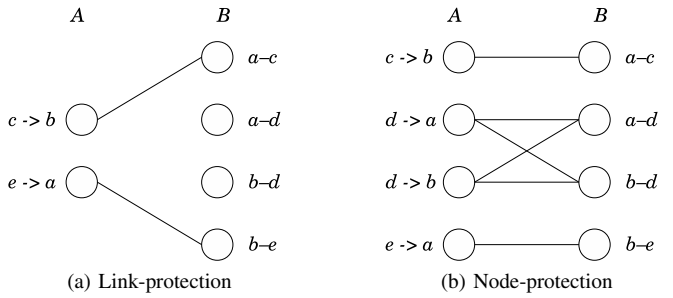
Let $(s_i, d_i) : i \in 1, \ldots, k$ be the set of unprotected source-destination pairs and let $e_j : j \in 1, \ldots, l$ be the set of complement edges. Let $G'(A, B, F)$ be a bipartite graph with node set $A \cup B$ and edge set F, where we add a node $a_i \in A$ corresponding to each $(s_i, d_i) : i \in 1, \ldots, k$ and a node $b_j \in B$ to each $e_j : j \in 1, \ldots, l$, and we connect some $a_i \in A$ to some $b_j \in B$ in G' if and only if edge e_j, when added with suitably large cost to G, would create a link-protecting LFA to (s_i, d_i). One easily sees that $G'(A, B, F)$ has $O(n^2)$ nodes and $O(n^4)$ edges, and it can be built in $O(n^2(n^2 \log n + nm))$ time as we need to perform an all-pairs-shortest path calculation for each of the $O(n^2)$ complement edges. Furthermore, the operation of adding a link e_j to G corresponds in G' to deleting the node b_j and all its neighbors from A. Since we take care of leaving the shortest paths in G intact, the resultant bipartite graph remains a valid representation.

The LFA graph extension problem in some G is then equivalently posed as a *Minimum set cover* problem over the corresponding bipartite graph $G'(A, B, F)$, a well-known NP-complete problem (SP5, [18]):

Definition 2: Minimal set cover problem (minSC): Given some positive integer p, is there a set of nodes $B^c \subseteq B$ with $|B^c| \leq p$, such that every node in A has a neighbor in B^c?

The bipartite graph representation for the link-protecting LFA extension problem over the sample network of Fig. 1 is depicted in Fig. 2a. Note that the problem is directly equivalent to the Minimum hypergraph transversal problem as well [19].

Next, we extend this model to the node-protecting case and we also improve the complexity of constructing it to $O(n^3)$. The node set of G' is built similarly. We add a node to A corresponding to each (s, d) if either *(i)* d is the next-hop of s and the loop-free condition (1) does not hold or *(ii)* d is not the next-hop of s and (3) does not hold for the next-hop n of s towards d. Additionally, we add a node to B for each complement edge in $\overline{E}$. Finally, we need to connect the right nodes in G'. Below, we show an optimized condition, based on the observation that, on the one hand, the dist function is



(a) Link-protection (b) Node-protection

Figure 2: Sample bipartite graph representations.

invariant to adding high cost edges to G and, on the other hand, an (u, v) edge, if added to G with high cost, can provide an LFA only for source-destination pairs whose source coincides with u or v. In particular, for some $a_i \in A$ and some $b_j \in B$ we add an (a_i, b_j) edge to F if for the corresponding source-destination pair (s_i, d_i), complement edge (u_j, v_j) and next-hop n of s_i towards d_i, one of the below conditions holds:

- $n = d$, $u = s$ and $\mathrm{dist}(v, d) < \mathrm{dist}(v, s) + \mathrm{dist}(s, d)$, or
- $n = d$, $v = s$ and $\mathrm{dist}(u, d) < \mathrm{dist}(u, s) + \mathrm{dist}(s, d)$, or
- $n \neq d$, $u = s$, $\mathrm{dist}(v, d) < \mathrm{dist}(v, s) + \mathrm{dist}(s, d)$ and $\mathrm{dist}(v, d) < \mathrm{dist}(v, n) + \mathrm{dist}(n, d)$, or
- $n \neq d$, $v = s$, $\mathrm{dist}(u, d) < \mathrm{dist}(u, s) + \mathrm{dist}(s, d)$ and $\mathrm{dist}(u, d) < \mathrm{dist}(u, n) + \mathrm{dist}(n, d)$.

This can be done in $O(n^3)$ time, checking the above conditions for each of the $O(n)$ neighbors for each $O(n^2)$ node in B.

Apart from its simplicity, one of the most appealing properties of the bipartite graph model is that it is easy to extend. For instance, we have deliberately ignored Equal Cost MultiPath so far. In ECMP, a router might have several next-hops towards a prefix along equal cost shortest paths and the task is to find an LFA for each of them. The problem is that a particular alternate might be a node-protecting LFA for one next-hop but only link-protecting for another. Fortunately, ECMP can be seamlessly incorporated into the above model: we add a node to A in G' for each source-destination-next-hop tuple and connect this node to a node in B if the corresponding complement edge would create LFA for this tuple. Thanks to this generality of the model, support for broadcast LANs, an elemental feature of IGPs, is also easy to add.

B. Algorithms

The authors in [11] propose an Integer Linear Program of $O(n^2 - m)$ binary variables to obtain an optimal solution for the LFA graph extension problem. Due to its complexity, the ILP is not expected to work in large networks, therefore, in this section we present several efficient heuristics. However, instead of working directly on the original network we rather solve the corresponding minimum set cover problems on the respective bipartite graph representations, as there are various well-tested heuristics available in the literature for this important class of combinatorial optimization problems. In particular, we discuss the Lovasz-Johnson-Chvatal algorithm from [20], the SBT algorithm from [21], the RSBT, MSBT algorithms from [19] and a straightforward backtracking algorithm.

1) The Lovász-Johnson-Chvatal (LJC) method: In [11], a greedy heuristic to obtain an approximate solution is proposed, which in every iteration adds the link that improves the LFA coverage the most. This algorithm, when interpreted in the bipartite graph model, corresponds to the Lovász-Johnson-Chvatal algorithm (LJC, [20]). In every iteration, LJC adds the highest degree node $v \in B$ to the cover B^c, v and its neighbors in A are deleted from G' and the algorithm proceeds to the next iteration.

Algorithm 1 LJC algorithm given a bipartite model G'.

```
B^c ← ∅
while A ≠ ∅
    v ← argmax_{b∈B} deg(b)
    B^c ← B^c ∪ {v}
    A ← A \ neigh(v)
    B ← B \ {v}
end while
```

Lovász shows that the size of the cover provided by this algorithm is within a logarithmic factor of the optimum: $t_{opt} \leq t_{LJC} \leq t_{opt} * (1 + \log_2 |B|)$ where t_{opt} is the cardinality of the optimal cover and t_{LJC} denotes the cardinality of the cover B^c from LJC [20]. Besides, the greedy algorithm is remarkably fast. Unfortunately, it is not guaranteed that the cover returned by the LJC algorithm is *minimal in the sense of inclusion*, which basically means that some proper subset of the solution would also be an adequate cover. This might render the LJC algorithm hugely impractical in certain cases.

2) The SBT algorithm: SBT was proposed in [19] to find an approximate cover that is, in contrast to LJC, minimal in the sense of inclusion. SBT seeks for the node $v \in B$ with the smallest degree and removes it from B. Additionally, if $\mathrm{neigh}(v)$ contains a node a that is covered by v only, then v is added to B^c as otherwise we could not cover A. In this case, we consider all v's neighbors as covered, remove them from A and proceed to the next iteration.

Algorithm 2 SBT algorithm given a bipartite model G'.

```
B^c ← ∅
while A ≠ ∅
    v ← argmin_{b∈B} deg(b)
    if ∃n ∈ neigh(v) with deg(n) = 1 then
        B^c ← B^c ∪ {v}
        A ← A \ neigh(v)
    end if
    B ← B \ {v}
end while
```

3) The RSBT algorithm: The Reverse SBT algorithm [19], as the name says, does the reverse of what SBT does in that in

every iteration it chooses the node with the highest degree instead of the smallest degree. Consequently, the pseudo-code is the same as given in Algorithm 2 with the slight modification that instead of line 3 we write $v \leftarrow \text{argmax}_{b \in B} \deg(b)$.

4) The MSBT algorithm: The Modified SBT algorithm [19] applies a small optimization step to SBT. Similarly to the SBT algorithm, in each iteration we choose the node $v \in B$ with the smallest degree and, if there are nodes in A covered only by v, we add v to B^c . If, on the other hand, $B \setminus \{v\}$ remains a cover, then we search for all the nodes $a \in A$ that are covered by exactly two nodes in B: v plus some other node, say, $w \neq v$, we add these ws to B^c and we remove them from B and all their neighbors from A.

Algorithm 3 MSBT algorithm given a bipartite model G'.

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$
3: $v \leftarrow \text{argmin}_{b \in B} \deg(b)$
4: **if** $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$ **then**
5: $B^c \leftarrow B^c \cup \{v\}$
6: $A \leftarrow A \setminus \text{neigh}(v)$
7: **else**
8: **for** each $a \in \text{neigh}(v)$ with $\deg(a) = 2$
9: $w \leftarrow u \in \text{neigh}(a) \setminus \{v\}$
10: $B^c \leftarrow B^c \cup \{w\}$
11: $A \leftarrow A \setminus \text{neigh}(w)$
12: **end for**
13: **end if**
14: $B \leftarrow B \setminus \{v\}$
15: **end while**

Note that the SBT, RSBT and MSBT algorithms generate covers that are minimal in the sense of inclusion.

5) An optimal backtracking algorithm: The backtracking algorithm implements a brute force strategy to solve the problem optimally. This scheme generates all possible covers and finds the one with the smallest cardinality. In order to avoid visiting a certain cover twice, the algorithm maintains a lexicographic order of the covers and examines them in ascending order. Unfortunately, the complexity is still exponential as there are $O(2^{|B|})$ potential covers. Due to space constraints, we omit the pseudo-code for this algorithm.

IV. Numerical studies

The main task we undertook in our numerical studies was to determine which of the above heuristics works best for LFA graph extension. In particular, we were curious as to how many new links are needed to achieve full LFA protection with the different algorithms both for the link-protecting and the node-protecting cases. Therefore, we implemented the bipartite graph model and the optimization algorithms in C++ with the help of LEMON graph library [22] and we compared their performance in numerous real-life ISP topologies. The evaluations were run on a Linux PC with an Intel Xeon 2.53GHz CPU and 3G RAM. We used the collapsed AS1221, AS1755, AS3257, AS3967 and AS6461 topologies from the Rocketfuel dataset [23]. These graphs come with inferred link costs. We also used the Abilene, Italy, NSF, Germany, AT&T and the extended German backbone (Germany_50) from [24]. Unfortunately, except for the last network no valid link costs were available, so we set each cost to 1. We also used some network topologies from the Topology-Zoo project's dataset [25]. For this dataset, we set costs randomly wherever link costs were not available. The topologies were chosen so as to ensure that the ILP still runs and so we can compare the performance of the heuristics to each other as well as to the optimum. Before actually running the algorithms, parallel edges were removed, links and costs were symmetrized and the preprocessing algorithm from [11] was executed in order to ensure that the optimization problems were always solvable. The number of new links added and the running time by the different algorithms for link-protecting LFAs are given in Table I, while the same results for the node-protecting case are presented in Table II.

The most important observations are as follows. First, the initial LFA coverage is usually about 70-90% in the link-protecting case and only 55-75% in the node-protecting case. This is expected, as node-protection is a stricter requirement than link-protection. Second, on most small and middle-sized networks adding only about a dozen or less new links is often enough to achieve 100% link-protection. This marks the huge potential to LFA-based network optimization. For node-protection, however, significantly more new links are needed, to the point that in larger topologies we need to virtually double or triple the number of links. Third, all heuristics perform surprisingly well, only overshooting the optimum by at most 5-15% in most cases and even finding the optimum for some networks. The MSBT algorithm is the clear winner both for link- and node-protection, with SBT also working reasonably, while LJC and chiefly RSBT are the worst performers. Finally, the execution time of the heuristics is acceptable. LJC is obviously the fastest, while the backtracking algorithm did not even run till optimum in most of the cases and had to be shut down after 5 hours of execution.

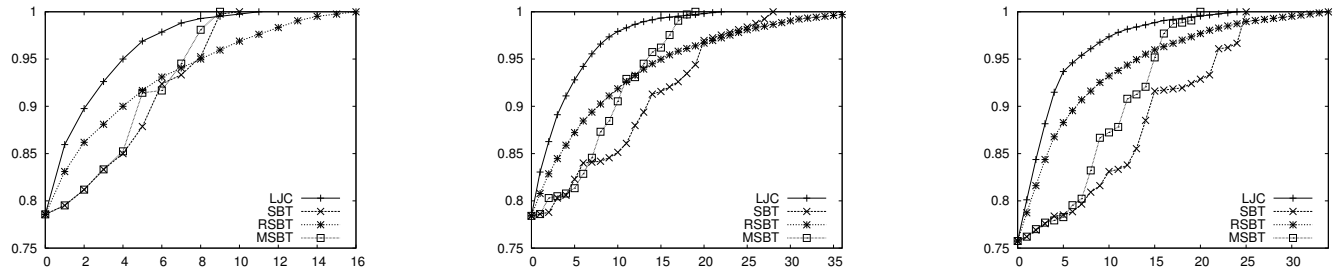
It seems that for larger networks, and especially in the node-protecting case, we need dozens of new links to achieve 100% LFA protection. This is clearly out of scope for most operators. Instead of aspiring to 100% protection, the LFA graph improvement problem therefore aims towards the more realistic goal of boosting the LFA coverage by adding only a small number of new links. Thusly, we also examined how the LFA coverage increases with each added new link in the subsequent iterations of the algorithms. The results for some select topologies are depicted in Fig. 3. The most important observation is that while MSBT is the most efficient in attaining 100% coverage with the smallest number new links, it is the LJC algorithm, by nature, that improves the LFA coverage the most in the initial steps. The RSBT algorithm also performs well in this regard. With LJC, about 10-15% improvement in the LFA coverage can be realized by adding only at most 5 new links and another 10% with the next 5

Table I: Link-protecting LFA graph extension results: topology name, number of nodes (n) and edges (m); number of link costs and edges added in the preprocessing phase ("Pre. c/e"), initial LFA coverage (η_0), number of new edges in the optimal solution (ILP), and the number of added edges ("ext") and execution time in seconds for each algorithm.

						LJC		SBT		RSBT		MSBT		Backtr.	
Topology	n	m	Pre. c/e	η_0	ILP	ext	time	ext	time	ext	time	ext	time	ext	time
AS1221	7	9	1/1	0.833	2	2	0.001	2	0.001	2	0.001	2	0.001	2	0.0007
Abilene	12	15	1/1	0.666	7	8	0.001	9	0.006	14	0.004	8	0.004	7	14592
AS6461	17	37	1/1	0.933	3	3	0.001	3	0.012	4	0.002	3	0.012	3	1.9
Germany	17	25	0/0	0.694	9	12	0.002	12	0.039	13	0.015	11	0.036	N/A	N/A
AS1755	18	33	0/0	0.873	7	7	0.001	9	0.027	12	0.009	7	0.024	N/A	N/A
InternetMCI	19	45	2/2	0.956	5	6	0.001	5	0.015	5	0.001	5	0.015	N/A	N/A
AS3967	21	36	0/0	0.786	8	11	0.003	10	0.092	16	0.036	9	0.091	N/A	N/A
AT&T	22	38	0/0	0.822	10	12	0.004	12	0.104	12	0.028	11	0.101	N/A	N/A
BtEurope	24	37	13/14	0.982	5	5	0.001	5	0.018	5	0.001	5	0.018	N/A	N/A
NSF	26	43	0/0	0.860	11	12	0.004	13	0.182	28	0.093	13	0.168	N/A	N/A
AS3257	27	64	5/5	0.930	10	11	0.002	10	0.133	12	0.020	10	0.125	N/A	N/A
BBNPlanet	27	28	16/16	0.806	17	19	0.008	17	0.278	17	0.030	18	0.257	N/A	N/A
Gambia	28	28	15/15	0.637	16	18	0.020	19	0.707	23	0.227	18	0.651	N/A	N/A
AS1239	30	69	0/0	0.874	6	6	0.003	7	0.429	11	0.086	6	0.475	N/A	N/A
Digex	31	35	0/0	0.316	22	27	0.806	27	2.020	46	1.803	27	1.745	N/A	N/A
Italy	33	56	0/0	0.784	17	22	0.025	28	1.037	39	0.434	19	0.896	N/A	N/A
BICS	33	48	8/8	0.784	20	24	0.037	24	1.038	29	0.264	22	0.820	N/A	N/A
BtNorthAm	36	76	4/5	0.847	20	22	0.004	20	1.706	27	0.420	20	1.622	N/A	N/A
GRNet	36	41	16/16	0.734	23	27	0.052	24	2.260	24	0.392	23	1.995	N/A	N/A
Geant	37	57	8/8	0.853	21	23	0.002	21	1.289	25	0.222	21	1.175	N/A	N/A
Arnes	41	65	9/9	0.819	24	29	0.071	24	3.008	30	0.450	24	2.845	N/A	N/A
ChinaTelecom	42	66	28/28	0.969	13	13	0.004	13	0.419	13	0.018	13	0.412	N/A	N/A
Carnet	44	43	34/34	0.818	16	19	0.044	16	4.332	16	0.254	16	4.210	N/A	N/A
BellCanada	48	64	9/11	0.629	32	38	0.219	35	11.869	48	4.136	34	10.673	N/A	N/A
Germany_50	50	88	0/0	0.900	18	21	0.044	29	4.804	44	1.536	25	4.323	N/A	N/A
Cudi	51	52	35/34	0.771	24	29	0.135	24	11.677	24	0.778	27	11.070	N/A	N/A
BellSouth	51	66	32/32	0.836	26	29	0.092	26	7.825	27	0.467	27	7.550	N/A	N/A
Bestel	84	93	11/12	0.343	68	91	5.575	82	378.41	128	276.22	75	312.50	N/A	N/A
Deltacom	113	183	11/10	0.614	80	100	9.226	94	1222.5	131	490.3	91	989.68	N/A	N/A
Average:	35.3	53.1		0.781	18.6	22.2	0.565	21.4	57.11	28.5	26.83	20.3	46.67	N/A	N/A
Mean deviation[%]:						115.3		114.11		147.86		107.77			

Table II: Node–protecting LFA graph extension results: topology name, number of nodes (n) and edges (m); number of link costs and edges added in the preprocessing phase ("Pre. c/e"), initial LFA coverage (η_0), number of new edges in the optimal solution (ILP), and the number of added edges ("ext") and execution time in seconds for each algorithm.

						LJC		SBT		RSBT		MSBT		Backtr.	
Topology	n	m	Pre. c/e	η_0	ILP	ext	time	ext	time	ext	time	ext	time	ext	time
AS1221	7	9	1/1	0.500	3	3	0.000	3	0.000	5	0.000	3	0.000	3	0.004
Abilene	12	15	1/1	0.591	9	12	0.001	11	0.006	17	0.004	11	0.004	N/A	N/A
AS6461	17	37	1/1	0.746	12	14	0.001	12	0.023	15	0.009	12	0.021	N/A	N/A
Germany	17	25	0/0	0.562	18	22	0.005	22	0.043	27	0.031	19	0.032	N/A	N/A
AS1755	18	33	0/0	0.765	16	19	0.003	18	0.033	27	0.016	18	0.023	N/A	N/A
InternetMCI	19	45	2/2	0.775	23	26	0.004	26	0.037	30	0.018	23	0.027	N/A	N/A
AS3967	21	36	0/0	0.643	17	19	0.013	20	0.129	32	0.087	20	0.111	N/A	N/A
AT&T	22	38	0/0	0.580	38	43	0.017	43	0.129	54	0.084	40	0.102	N/A	N/A
BtEurope	24	37	13/14	0.757	31	32	0.007	31	0.094	49	0.025	31	0.081	N/A	N/A
NSF	26	43	0/0	0.634	18	24	0.021	34	0.418	38	0.261	23	0.333	N/A	N/A
AS3257	27	64	5/5	0.768	34	36	0.015	34	0.237	50	0.115	34	0.248	N/A	N/A
BBNPlanet	27	28	16/16	0.751	35	37	0.012	35	0.207	42	0.055	35	0.165	N/A	N/A
Gambia	28	28	15/15	0.541	49	53	0.042	58	0.498	64	0.215	50	0.389	N/A	N/A
AS1239	30	69	0/0	0.757	19	24	0.025	25	0.604	34	0.226	20	0.509	N/A	N/A
Digex	31	35	0/0	0.312	29	36	0.980	39	1.722	50	1.632	34	1.556	N/A	N/A
Italy	33	56	0/0	0.570	35	43	0.082	48	1.490	60	0.949	38	1.229	N/A	N/A
BICS	33	48	8/8	0.692	37	42	0.047	44	0.986	57	0.403	40	0.742	N/A	N/A
BtNorthAm.	36	76	4/5	0.779	46	50	0.057	47	1.193	63	0.378	46	0.993	N/A	N/A
GRNet	36	41	16/16	0.607	64	70	0.116	72	1.741	83	0.596	67	1.219	N/A	N/A
Geant	37	57	8/8	0.592	58	70	0.252	70	2.971	84	1.431	59	2.518	N/A	N/A
Arnes	41	65	9/9	0.550	90	104	0.264	103	3.397	127	1.465	92	2.451	N/A	N/A
ChinaTelecom	42	66	28/28	0.866	62	66	0.033	62	0.695	82	0.253	62	0.623	N/A	N/A
Carnet	44	43	34/34	0.746	100	102	0.157	101	2.680	107	0.746	100	2.072	N/A	N/A
BellCanada	48	64	9/11	0.488	70	83	0.601	80	11.50	97	6.832	76	8.997	N/A	N/A
Germany_50	50	88	0/0	0.828	34	44	0.138	57	6.353	81	3.410	50	5.754	N/A	N/A
Cudi	51	52	35/34	0.732	75	80	0.263	77	8.115	86	1.279	77	6.993	N/A	N/A
BellSouth	51	66	32/32	0.730	93	97	0.252	94	6.527	123	1.256	93	5.158	N/A	N/A
Bestel	84	93	11/12	0.312	106	137	7.780	134	351.9	173	274.9	124	262.3	N/A	N/A
Deltacom	113	183	11/10	0.527	171	212	20.735	214	1220.2	255	599.43	197	890.21	N/A	N/A
Average:	35.3	53.1		0.644	48	55.2	1.1	55.6	56.0	69.4	30.9	51.5	41.2	N/A	N/A
Mean deviation[%]:						115.41		117.86		152.81		107.97			



(a) AS3967 (b) Italy (c) AS1239

Figure 3: LFA coverage in each iteration of different heuristics in the link-protecting case for AS3967 and Italy, and node-protecting case for the AS1239 topology.

links, putting the coverage in the 90-95% range, which may be enough in many practical scenarios.

V. Conclusions

Currently, Loop-Free Alternates is the most practical method to provide fast protection in IP networks. By using LFA, ISPs can gain high level of protection with minimal effort, and with the help of the LFA-based network optimization algorithms provided in this paper the level of protection can be improved even further. We have shown a generic bipartite graph model and we applied several heuristics from the literature to this representation. The most important conclusion is that, even-though NP-complete, the LFA graph extension problem is efficiently approximable. To support this claim, we used the logarithmic upper bound on the worst case performance of the LJC heuristic, and we also presented extensive numerical studies. We argued that, depending on the optimization objective, different approximation strategies should be pursued: when the aim is 100% LFA protection then the MSBT algorithm is a solid choice, whereas the LJC algorithm is the best option when the aim is to improve LFA coverage with only a limited number of new links. Future work involves integrating these algorithms into a common approximation framework which would be suitable to tackle both problems equally efficiently.

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

[5] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *IEEE Comsware*, 2007.

[6] Cisco Systems, "Cisco IOS XR Routing Configuration Guide, Release 3.7," 2008.

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

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

[9] M. Menth, M. Hartmann, and D. Hock, "Routing optimization with IP Fast Reroute." Internet Draft, July 2010.

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

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

[12] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Comm. Mag.*, vol. 40, pp. 118–124, Oct 2002.

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

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

[15] 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.

[16] D. Katz and D. Ward, "Bidirectional forwarding detection (bfd)." RFC 5880, March 2010.

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

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

[19] 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.

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

[21] P. Kulaga, P. Sapiecha, and K. Sej, "Approximation Algorithm for the Argument Reduction Problem," in *Computer recognition systems: proceedings of the 4th International Conference on Computer Recognition Systems, CORES'05*, p. 243, Springer Verlag, 2005.

[22] "LEMON – Library for Efficient Modeling and Optimization in Networks." http://lemon.cs.elte.hu/, 2009.

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

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

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