Telecommunication Systems Journal, 2011 manuscript No.
(will be inserted by the editor)

Optimization Methods for Improving IP-level Fast Protection for Local Shared Risk Groups with Loop-Free Alternates

Máté Nagy · János Tapolcai · Gábor Rétvári

Received: date / Accepted: date

Abstract Lately, demand for fast failure recovery in IP networks has become compelling. The Loop-Free Alternates (LFA) specification is a simple IP Fast ReRoute (IPFRR) scheme proposed by the IETF that does not require profound changes to the network infrastructure before deployment. However, this simplicity comes at a severe price, because LFA does not provide complete protection for all possible failure cases in a general topology. This is even more so if network components are prone to fail jointly. In this paper, we study an important network optimization problem arising in this context, the so called LFA graph extension problem, which asks for augmenting the topology with new links in an attempt to improve the LFA failure case coverage. Unfortunately, this problem is NP-complete. The main contributions of the paper are a novel extension of the bipartite graph model for the LFA graph extension problem to the multiple-failure case using the model of Shared Risk Groups, and a suite of accompanying heuristics to obtain approximate solutions. We also 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.

Keywords IP Fast ReRoute · Loop-Free Alternates · link and node protection · heuristics

The work was partially supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009.

Máté Nagy
HSNLab, Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics
E-mail: nagym@tmit.bme.hu

János Tapolcai
HSNLab, Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics
E-mail: tapolcai@tmit.bme.hu

Gábor Rétvári
HSNLab, Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics
E-mail: retvari@tmit.bme.hu

1 Introduction

Throughout the last few years, the amount of streaming media traffic transmitted over the Internet has increased significantly. As this kind of traffic is more sensitive to delay than to packet loss, the need to improve service reliability and availability has become more and more stressing to operators. Consequently, operators strike to maintain five-nines availability (99,999% uptime) in order to stay competitive in the fierce telecommunications market. Unfortunately, the state-of-the-art IP protocol suite, the prevailing platform on top of which operators are delivering their services to customers today, is not adequate to meet these needs. One of the principal reasons for this inadequacy is the IP control plane's inability to rapidly reconverge after a network component has failed. 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), like the Open Shortest Path First (OSPF, [36]) or the Integrated IS-IS (IS-IS, [5]) routing protocol, takes about ten times more than that [14].

To tackle this challenge, the Internet Engineering Task Force (IETF) has initiated the IP Fast ReRoute (IPFRR [43]) framework, with the purpose of reducing the critical routing convergence time to the order of 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 of the routers, 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. In fact, the IP Fast ReRoute standards essentially switch IP networks from what was formerly a predominantly *restoration* scheme to an intrinsically faster *protection* mechanism [45].

Note that IPFRR is not the only option to realize fast protection in IP networks. For operators that settled on MultiProtocol Label Switching (MPLS) with the Resource Reservation Protocol–Traffic Engineering extensions (RSVP-TE), the MPLS Fast ReRoute specification [38] provides a standardized and broadly implemented fast protection scheme [19]. However, there are many operators that have not deployed MPLS at all, or instead of RSVP-TE use the lighter weight Label Distribution Protocol (LDP) that also relies on the IP control plane for routing information. In such cases, IP Fast ReRoute is the only alternative available today.

Not surprisingly, researchers and device vendors have come up with a wide selection of IPFRR mechanisms this far. Some proposals introduce tunnels to route around the failed component [3, 4, 11], others apply explicit or implicit failure signaling [20, 24, 29], and still others straight-out call for altering IP's destination-based forwarding [26]. Unfortunately, the majority of existing IPFRR proposals, in some way or another, require functionality not yet available in existing IP network gear or impose significant extra management burden on network operations [13, 16, 28] (or both), which makes device vendors reluctant to implement them and discourages operators from deploying IPFRR all together.

As of today, there is only one IPFRR proposal that has gained considerable vendor and operator support: Loop-Free Alternates (LFA [2]). This is thanks to its remarkable simplicity. 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 unobtrusive extension to IGPs, thus it is appealingly easy to deploy. In fact, in most modern high-end routers turning LFA on is just a straight-forward configuration command away [7, 18, 21], and needs no extensive prior testing or pilot deployments. This simplicity, however, comes at a huge price: depending on the network topology and link costs there might exist failure scenarios in the network for which no alternate next-hop (i.e., LFA) is available and so no fast protection can be provided. This deficiency is further exacerbated by the fact that modern IP networks run on top of complex MPLS and/or optical substrates, which introduce subtle inter-dependencies between seemingly independent failure scenarios. For instance, when an optical fiber shared by multiple virtual connections goes down, the IP layer perceives this single physical failure as multiple simultaneous failures, because it sees the virtual connections as separate IP links. The existence of such *Shared Risk Groups* in the network, as shall be argued later in this paper, further amortizes LFA's potential to provide complete failure coverage.

With its standardization and appearance in off-the-shelf routers, LFA has received renewed interest both from theoreticians and practitioners recently. The main motivation behind the corresponding *LFA network optimization* research area is to optimize some aspect of the underlying network (i.e, the topology, IGP link costs, etc.) in order to improve the LFA failure coverage, facilitating to *achieve high quality protection in an IP network intervening only at the management plane, leaving the data plane and the control plane intact*. Since the control and the data planes are embedded into the IP routers themselves, altering them promises itself a particularly long and tedious fight through standardization bodies, router vendors, operator forums, etc. On the other hand, changing the underlying network instead of changing the very IP protocol seems way simpler, and with LFA already implemented in the control and data planes of IP routers LFA network optimization promises with a solution that is readily deployable even on today's installed network infrastructure.

There have been various attempts at LFA network optimization recently [34, 40, 48], one of the most recent of which is LFA graph extension [37, 41]. 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 service providers [12, 46, 47]. A closely related problem is *LFA graph improvement*, where the task is merely to raise the level of protection provided by LFA instead of shooting for 100% failure coverage. As it turns out, these problems are all NP-complete. Apart from the NP-completeness proofs, [41] also presents an Integer Linear Program (ILP) to get an exact solution plus a greedy heuristic. Unfortunately,

their treatment is limited to the case of 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 greedy heuristics is suitable to efficiently approximate the optimal solution.

We initiated the work towards remedying these shortcomings in the conference version of this paper [37], and below we conclude this work. We provide a detailed and worked out bipartite graph model for the LFA graph extension and the LFA graph improvement problems, which makes it possible to obtain a solution under basically any failure model relevant to practice, and we present a collection of optimal algorithms and approximation heuristics, each with different relative error and running time, facilitating for picking the most appropriate optimization strategy for the problem under consideration.

The main contributions of this paper are as follows.

- We extend the LFA graph extension and the LFA graph improvement problem formulations from the single link failure model to the more sophisticated multiple failure model of local Shared Risk Groups (SRGs). Under this model, links and/or nodes adjacent to a router that are prone to fail jointly can be configured into SRGs, and the task is to find an LFA that is not only loop-free but is also SRG-disjoint.
- We generalize the model given in [41] for the above extended SRG failure model, presenting a new formal framework as well as illustrative examples. We also give hints as to how to augment the model to treat Equal-Cost MultiPath (ECMP) and broadcast media (LANs).
- We study the solvability of the LFA graph extension problem and we give a general negative result for the multiple-failure case.
- We collect a number of 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. In Section 2 we outline the literature that is closely related to our work. Section 3 gives an overview of LFA, Shared Risk Groups (SRGs) used to describe multiple failure models, and we state the LFA network optimization problems formally. Section 4 introduces the mathematical model and describes the optimal backtracking, and the approximate SBT, RSBT, and MSBT algorithms. Numerical results are described in Section 5 and finally, Section 6 concludes the paper.

2 Related work

Besides Loop-Free Alternates, many other fast protection methods have surfaced for IP during the recent years. The common theme shared by all these efforts is that they attempt to reduce recovery time below some tens of miliseconds in order to eliminate slow convergence of regular IGPs for *all* possible single failure cases.

With *Failure Insensitive Routing* (FIR), routers perform interface specific packet forwarding [9, 26, 49]. If a packet arrives on an unusual input port, the router infers that due to some network element failure the packet has been redirected to a detour at an earlier stage of its forwarding path. Based on this information, it tries to retransmit the packet on an output interface that it knows is not affected by the fault. Unfortunately, in current commercial routers interface specific packet forwarding is not available.

As its name foreshadows, the *Not-via addresses* [4] IPFRR scheme introduces tunneling to route around the failed component. When a router loses contact with one of its neighbours, it encapsulates packets in a new IP header with marking the destination as a special not-via address. The semantics of this not-via address can be translated like "forward me to the destination not-via the failed element". All the messages affected by the failure are transmitted through this tunnel afterwards. Unfortunately, computational and management complexity of handling the numerous not-via addresses the proposal requires can be daunting [28]. To reduce these costs, authors in [11], and the subsequent Internet draft [1], introduce a *lightweight version of Not-via*. The idea is based on maximally redundant trees [6, 10], a pair of trees for each node with the useful property that any single link or node failure will leave at least one of the trees intact, under the assumption that at least two disjoint paths existed in the network. Unfortunately, neither Not-via addresses nor Lightweight Not-via have been standardized yet, neither commercially available implementations exist.

Multiple Routing Configurations (MRC, [24]) calculates a small set of backup network configurations (or overlays), maintaining the invariant that for any source-destination pair at least one of the configurations remains connected after any single failure. When a link or node fails, packets are marked with the proper backup configuration identifier that enables routers to use the appropriate overlay topology. Unfortunately, marking packets would consume invaluable bits from

the IP header. Another approach, called *O2* routing [42], is to keep two distinct loop-free next-hops towards each destination. In case of link failure affecting one of the next-hops a router can immediately switch packet forwarding to the other one. The disadvantage of this concept is that it breaks shortest path routing, fundamental to IGP operations today.

Protection routing [25] is different from the others in the sense that routers are only responsible for packet forwarding but routing information is stored in a central server. The server calculates a set of different routing trees, not necessarily coincident with shortest paths, accompanied with a carefully chosen set of secondary next-hops for each node that can be used when the primary next-hop disappears. After downloading precomputed information to the routers, if any next-hop becomes unreachable routers can quickly switch to their standby next-hop. The main difficulty is in calculating the routing trees (more precisely, routing DAGs) and the corresponding secondary next-hops to guarantee loop-free forwarding and 100% protection coverage. This problem, similarly to most resilience maximization network optimization problems, turns out NP-complete. Apart from this difficulty, implementing protection routing would also require centralized control, which would mandate a deep reorganization of the way today's IP networks are managed [8].

Most of the existing IPFRR proposals strike to provide 100% failure case coverage (for some recent reviews on the extent to which they attain this goal, see [13] and [16]). It seems, however, that the price for this is significant management overhead and/or new, currently non-standardized and unimplemented features in the IP control and data plane. Loop-Free Alternates, on the other hand, does not require any of these. LFA is a simple, standardized, and well-tested IPFRR mechanism, implemented by most major router vendors in their products [7, 18, 21]. This is thanks to the deliberate decisions of the designers of LFA who, instead of aiming for 100% failure coverage at the cost of insurmountable deployment barriers, from the beginning made clear that LFA is optimized for simplicity and it is not intended to guarantee fast protection for all possible failure scenarios in all network topologies. Indeed, as simulation studies confirm, depending on the topology and link cost settings, LFA can usually protect only about 50-80% of the possible link failure scenarios, and the level of node protection is even worse [13,16,35,39].

Consequently, the main motivation behind the LFA network optimization problems we study in this paper is to improve the network's topology in a way as to maximize the level of protection provided by LFA. This gives a dependable and economic IPFRR solution to operators, ready to be deployed until one of the IPFRR proposals implementing complete protection becomes available (if ever).

3 Preliminaries

Throughout this paper, we model the network topology by a simple, weighted, symmetric directed graph $G(V, E)$ where V is the set of nodes and E is the set of directed arcs. Let $n = |V|$ and $m = |E|$, let $\overline{E}$ denote the complement arc set, let $\deg(v)$ denote the node degree of $v \in V$ and let $\text{neigh}(v)$ be the set of out-neighbors of v in G. Let the IGP link costs be represented by a cost function $c : E \mapsto \mathbb{Z}^+$. The cost of an edge (i, j) is denoted by $c(i, j)$. We assume that costs are symmetric: $c(i, j) = c(j, i)$. In addition, let $\text{dist}(x, y)$ denote the shortest path distance between some nodes $x \in V$ and $y \in V$.

Initially, we assume that the network contains point-to-point links only and there are no broadcast LANs. We also presume that there is no support for Equal-Cost MultiPath (ECMP), and ties between equal cost shortest paths are broken arbitrarily. These assumptions will be relaxed later.

3.1 Loop-Free Alternates

Below, we introduce a basic formalism for Loop-Free Alternates under the single failure model. In this model, it is assumed that in every IPFRR cycle either at most 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. Even though this failure model is in line with current IP practice [32], it might prove overly restrictive in a multi-layer environment where physical outages in the lower layers can present themselves as multiple failures in the IP layer. Therefore, in the subsequent section we overview a simple model for handling such multiple-failure scenarios, local Shared Risk Groups, and we extend the LFA formalism for this particular failure model.

Perhaps the easiest way to understand LFA is through an example. Consider the network in Fig. 1 and, initially, assume that only a single link can fail at a time. 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 has 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

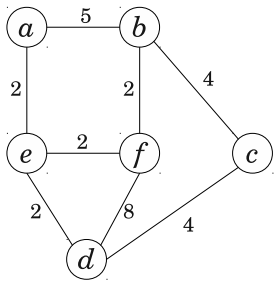


Fig. 1: A simple network without Shared Risk Groups

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 [2]. Formally:

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

i) $n \neq e$, and
ii) the loop-free condition applies:

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

Less formally, we find that n is an $s-d$ LFA if it is not upstream to s in the shortest path tree rooted at d. In the above example, $\text{dist}(b,d) = 6$, $\text{dist}(a,d) = 4$ and $\text{dist}(b,a) = 5$, and therefore b is a link-protecting LFA from a to d. Note, however, that in the sample network f does not have a link-protecting LFA to b as both its candidate neighbors d and e are upstream. The same applies to the $e-a$ pair. The level of LFA link protection $\eta_{\text{LP}}(G)$ in a network G is measured as the proportion of the protected vs. all source-destination pairs [41]:

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

For our sample topology, $\eta_{\text{LP}}(G) = 0.87$.

Let us return to the node-pair (a,d). As we have seen, the neighbor b provides a link-protecting LFA for this node-pair. Observe, however, that b only protects against the failure of link (a,e) but not against the failure of the next-hop e itself. This is because the shortest (b,d) path traverses e. In fact, b in this example does not have an LFA to d providing protection for the failure of the default next-hop. On the other hand, such a *node-protecting LFA* does exist for instance from a to c, because should its primary next-hop e go down, a could still forward traffic destined to c towards b. The below definition formalizes this property.

Definition 2 For some source s, destination d, and default $s-d$ next-hop e, a neighbor n of s is a node-protecting LFA for s to d if

i) $n \neq e$,
ii) $\text{dist}(n,d) < \text{dist}(n,s) + \text{dist}(s,d)$, and
iii) the node-protection condition applies:

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

In fact, we simply applied the extra constraint (3) in addition to Definition 1. Special care must be taken, however, when evaluating the above condition for the case 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 [4].

In our example, $\text{dist}(b,c) = 4$, $\text{dist}(b,e) = 4$ and $\text{dist}(e,c) = 6$ and so (3) holds. Some quick calculation yields that there are six source-destination pairs that are without node-protecting LFA in the above example. For instance (f,b), and (e,a) are unprotected as the last-hop exception applies and no link-protecting LFA is available, and (d,a) does not have node-protection as its LFA only fulfills (1) but not (3). LFA coverage $\eta_{\text{NP}}(G)$ 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, $\eta_{\text{NP}}(G) = 0.8$.

3.2 Shared Risk Groups

So far, we have assumed that component failures in the network are independent and affect only a single network element at a time. Though, in practice this is usually not the case. A common example of how this assumption can fail is the case of high capacity trunks that carry connections between several logically independent networks. If the trunk goes down, all virtual connections established on top of it go down as well, which is perceived by the IP layer as multiple simultaneous link failures. Similar is the case of lightpaths in optical networks, Dense Wavelength-Division-Multiplexing wavelength channels, Generalized MPLS label switched paths, etc.; these low level connections may aggregate multiple IP links and therefore can cause multiple simultaneous failures in the IP layer when being disconnected [17]. The main reason behind this issue is that

in a multi-layer protocol architecture the logical representation of the network as seen by the IP control plane is often quite different from the physical layout.

A convenient way to express the statistical dependency between a certain group of links or nodes that are expected to fail jointly is Shared Risk Groups (SRGs). In the SRG model, if a link (or node) fails then all the links (nodes) that share an SRG with it will also fail, and correspondingly all traffic will be interrupted on these links (nodes).

In a general multi-layer network, SRGs can contain arbitrary set of network components, even ones residing in far-away portions of the network. The LFA standard, however, restricts the set of SRGs considered to so called *local SRGs* [2]. This sub-category of SRGs consists of some set of links connected at one end to the same router. The reason for this limitation is that local SRGs, apart from being frequent in practice, can be configured at each router locally and independently from other routers, and routers can thus calculate local-SRG-disjoint LFAs without being aware of the local-SRGs configured at other routers. If, on the other hand, we were to allow arbitrary SRGs, a separate signaling protocol would need to be introduced to distribute SRG information between routers, plus the computational problem of selecting SRG-disjoint LFAs would become substantially more difficult as well [2]. In what follows, therefore, we shall restrict our attention to local-SRGs exclusively.

According to the above considerations, we seek LFA candidates that can be reached on a link that is not contained in the same local-SRG set as the link between the source node and its primary next-hop. We formalize this requirement below for the case when SRGs can only contain links (Shared Risk Link Groups). The treatment is straight-forward to generalize to the case when SRGs can contain nodes as well.

Let $S = \{(i,j) \in E\}$ be an SRG containing a set of links (i,j). The number of links can be arbitrary, but for S to be a local SRG we require that for any two $(i,j) \in S$ and $(u,v) \in S$: $i = u$. Note that SRGs in our model can, and usually are, asymmetric, that is, $(i,j) \in S$ does not imply $(j,i) \in S$. Now, for each directed arc $(i,j) \in E$ we can create the union of SRGs that include (i,j):

$$S(i,j) = \bigcup_{S:(i,j)\in S} S \ .$$

Using this notation, we say that a candidate node n is an *SRG-disjoint link protecting LFA* for some node pair (s,d) if, in addition to the constraints in Definition 1, the link (s,d) is also SRG-disjoint from the link of s to its next-hop. Formally:

Definition 3 For some source s, destination d, and default $s-d$ next-hop e, a neighbor n of s is an SRG-disjoint link-protecting LFA for s to d if

i) $n \neq e$,
ii) $\mathrm{dist}(n,d) < \mathrm{dist}(n,s) + \mathrm{dist}(s,d)$, and
iii) $(s,n) \notin S(s,e)$.

Similarly for the node-protecting case:

Definition 4 For some source s, destination d, and default $s-d$ next-hop e, a neighbor n of s is an SRG-disjoint node-protecting LFA for s to d if

i) $n \neq e$,
ii) $\mathrm{dist}(n,d) < \mathrm{dist}(n,s) + \mathrm{dist}(s,d)$,
iii) $\mathrm{dist}(n,d) < \mathrm{dist}(n,e) + \mathrm{dist}(e,d)$, and
iv) $(s,n) \notin S(s,e)$.

Easily, these definitions are trivial extensions of the corresponding definitions given for the single-link failure case above. To demonstrate these definitions in practice, consider the sample network introduced above extended with 3 local SRGs as depicted in Fig. 2. For instance, S_3 is a local-SRG configured at node e containing the links (e,d) and (e,f), with the interpretation that if link (e,d) fails then (e,f) is expected to fail as well, and *vice versa*. Recall that when SRGs are not considered, then f is a link-protecting LFA of e towards c. However, as the link to the next-hop (e,d) and the link to the candidate LFA (e,f) belong to the same local-SRG, this LFA is indeed not SRG-disjoint. Similar is the case for node-protection. For instance, f does not have SRG-disjoint node-protecting LFA to c due to SRG S_2. The LFA-coverage metrics η_{LP} and η_{NP} defined for the single failure case are now easy to extend to account for SRGs; we only need to swap the underlying LFA definitions. In what follows, unless otherwise stated the term "LFA" will refer to SRG-disjoint LFAs as of Definition 3 and Definition 4, and "LFA coverage" will denote the extended interpretation of η_{LP} and η_{NP}. In our example, $\eta_{\mathrm{LP}}(G) = 0.83$ and $\eta_{\mathrm{NP}}(G) = 0.73$. Evidently, LFA coverage decreases when SRGs are considered because the SRG-compliant LFA definitions are stricter than their non-SRG counterparts.

3.3 Problem formulation

There are several ways to improve the LFA protection in a network. A plausible choice would be to optimize the IGP link costs [34, 40, 48], but this would alter shortest paths. In many deployments, touching shortest paths is decidedly ruled out by operators who usually have their own intricate (and often proprietary) operational concerns they want to materialize in the way forwarding

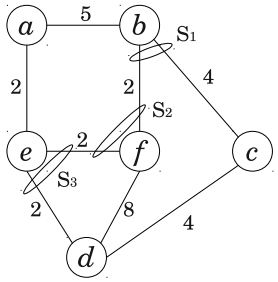


Fig. 2: The sample network with local Shared Risk Groups S_1, S_2, and S_3 each containing two links

paths are provisioned [12, 46, 47]. Consequently, in this paper we adopt the approach from [41] and [37], and we aim for increasing the LFA coverage by cleverly adding new links to the network without touching the shortest paths in any way. The *LFA graph extension* problem is defined as the task to augment a weighted graph with the minimum number of new links with properly selected costs, so that LFA coverage becomes 100% and shortest paths remain in place.

Definition 5 *LFA graph extension problem (minLFA):* Given a simple, weighted, symmetric digraph $G(V, E)$, a set of SRGs $\mathcal{S} = \{S\}$, and an integer l, is there a symmetric arc set $F \subseteq \overline{E}$ with $|F| \leq l$ and properly chosen costs, so that *(i)* for the SRG-disjoint link-protecting LFA coverage $\eta_{\text{LP}}(G(V, E \cup F)) = 1$ and *(ii)* the shortest paths in $G(V, E)$ coincide with the shortest paths in $G(V, E \cup F)$?

Here, a symmetric arc set F is a set of arcs such that $(i, j) \in F \Rightarrow (j, i) \in F$. In other words, when augmenting the digraph with a new arc we immediately add the reverse arc (j, i) as well. This case seems to be more relevant in practice, but we note that our model does not mandate this limitation in any way. The above definition is straightforward to adapt to the the node-protecting case as well by substituting $\eta_{\text{LP}}(G)$ with $\eta_{\text{NP}}(G)$.

In [41], the following complexity characterization is given for the non-SRG case.

Proposition 1 *The LFA graph extension problem for the link-protecting case is NP-complete when* $\mathcal{S} = \emptyset$.

Easily, this characterization remains valid for the case when we allow arbitrary local-SRGs, as the latter is a more general problem than the former. Furthermore, the optimization version, which asks for the minimal l for which the above condition holds, is also intractable, and similar is the case for the node-protecting version of the problem.

LFA graph extension asks for a link extension to attain 100% failure coverage. This, as it turns out, is often case too ambitious a goal. Therefore, we also consider a relaxed version of the problem, called the *LFA graph improvement* problem, which asks for realizing the highest improvement possible by adding only a limited number of new links.

Definition 6 *LFA graph improvement problem:* Given a simple, weighted, symmetric digraph $G(V, E)$, a set of SRGs $\mathcal{S} = \{S\}$, and two integers l and k, is there a symmetric arc set $F \subseteq \overline{E}$ with $|F| \leq l$ and properly chosen costs, so that *(i)* at least k source-destination pairs have an SRG-disjoint link-protecting LFA in $(G(V, E \cup F))$ and *(ii)* the shortest paths in $G(V, E)$ coincide with the shortest paths in $G(V, E \cup F)$?

Easily, the LFA graph improvement problem is also NP-complete, because otherwise we could solve minLFA with solving LFA graph improvement with setting $k = |V^2| = n(n-1)$.

As both LFA network optimization problems we aim to solve are intractable, it seems hopeless to obtain optimal solutions in large networks within acceptable running time. This calls for efficient heuristics. Therefore, 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.

4 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 large cost, say, larger than the length of the longest shortest path.

Below, we first discuss if LFA graph extension is solvable in general. As it turns out, when there are no SRGs provisioned in the network then a simple preprocessing step is enough to guarantee solvability, but when SRGs do exist in the network then there are cases when we can not add new links in any way to reach perfect LFA coverage. Next, we give an elaborate graph model of the problem and then we turn to the algorithmic strategies.

4.1 Conditions for solving minLFA

Before trying to solve the problems, we need to know whether they are solvable in the first place. In [41], the authors find that there are cases when LFA coverage

cannot be improved to 100% just by adding new links of large cost to the network. The reason is that there might exist nodes to which all other nodes are upstream and therefore no LFA candidates might exist. Such is the case when the network contains a node into which all traffic enters via a single last-hop router. Let us consider the example in Fig. 3a. There is no LFA from c to b as a is upstream, neither we can create one by adding a new link because the graph is complete and so the complement arc set is empty. In such cases, we need to resort to changing some link costs in the network.

When there are no SRGs provisioned, this is fairly simple. In our example, we only need to ensure that not all shortest paths to b traverse c. This can be done by changing the cost of, say, link (a, b) to 1. This immediately makes all nodes protected. The authors in [41] propose a polynomial time preprocessing algorithm, which changes at most two 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. This substantiates the below proposition:

Proposition 2 *Given a graph G on n nodes and m arcs in which no SRGs are provisioned and a cost function c, there is a modified graph G' containing at most $m + n$ arcs and a modified cost function c' differing from c on at most n arcs, so that minLFA is solvable in G' over c'. In addition, G' and c' can be obtained in polynomial time.*

In the course of our numerical evaluations, we found that it is usually not necessary to add as many as n new arcs and change n costs. In fact, in most cases adding some $2-4$ new links and/or changing a similar number of costs is enough. Contrarily, when the network contains SRGs there are cases when not even changing link costs help. Consider the below observation.

Observation 1 *There exist a graph G with properly chosen local SRGs and a corresponding cost function c, so that there is no cost function c' and no extension arc set $F \subseteq \overline{E}$ for which minLFA is solvable in $G(V, E \cup F)$ over c'.*

The counter-example is depicted in Fig. 3b. Easily, there are some node pairs (e.g., (c, b)) between which no LFA exists, plus we cannot change this situation with altering link costs and we cannot add new links either. Interestingly, we have met this issue only a few times during our numerical evaluations. As shall be seen, in all the networks examined only less than 3 nodes per network were affected by this issue, and the eventual LFA coverage was always above 99%. It seems thus that the above pathological case rarely arises in practice.

(a) Graph w/o SRGs (b) Graph with SRGs

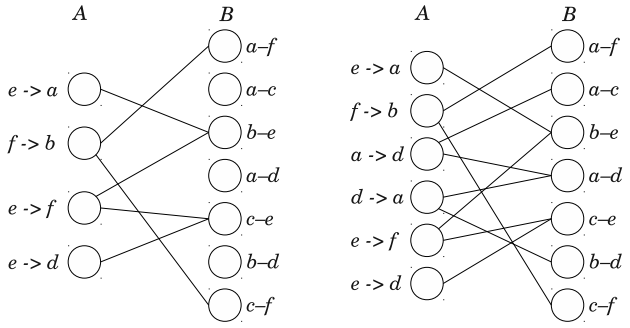
Fig. 3: Networks without (a) and with (b) SRGs on which minLFA is not solvable

4.2 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 (e, b) is suitable, as with this link in place e could use b as an LFA. Similarly, adding link (e, c) would create an LFA for other unprotected source-destination pairs, (e, d) and (e, f). Observe, however, that not all the complement arcs 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 can obtain an LFA from several complement arcs. Our task is to find the smallest symmetric subset of the complement arc set $\overline{E}$ so that each unprotected source-destination pair gets an LFA. The following bipartite graph model for the LFA graph extension problem is based on the idea that this task can be represented as a minimum cover problem over a suitably defined bipartite graph.

Suppose first that no SRGs are configured in the network, let $(s_i, d_i) : i \in 1, \ldots, k$ be the set of unprotected source-destination pairs and let $\{(u_j, v_j) : j \in 1, \ldots, l\}$ be the set of complement arcs $\overline{E}$ from which reverse arcs were eliminated, i.e., the set contains either (u_j, v_j) or (v_j, u_j), but not both. Let $G'(A, B, F)$ be an undirected bipartite graph with node set $A \cup B$ and edge set F, where we add a node $a_i \in A$ corresponding to each unprotected $(s_i, d_i) : i \in 1, \ldots, k$ and a node $b_j \in B$ for each arc $(u_j, v_j) : j \in 1, \ldots, l$, and we connect some $a_i \in A$ to some $b_j \in B$ in G' if and only if arc (u_j, v_j) or (v_j, u_j), when added with suitably large cost to G, would create a link-protecting LFA to



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

Fig. 4: Sample bipartite graph representations.

(s_i, d_i). This amounts to checking whether Definition 1 would hold for (s_i, d_i) on the graph augmented with (u_j, v_j) and (v_j, u_j).

One easily sees that $G'(A, B, F)$ has $O(n^2)$ nodes and $O(n^4)$ arcs, 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 arcs. Furthermore, the operation of adding a link (u_j, v_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 of the LFA graph extension problem on the augmented graph. We conclude that the LFA graph extension problem in some G is equivalently posed as a *Minimum set cover* problem over the corresponding bipartite graph $G'(A, B, F)$, a well-known NP-complete problem (SP5, [15]):

Definition 7 *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?

Note that the problem is directly equivalent to the Minimum hypergraph transversal problem as well [33].

The bipartite graph representation for the link-protecting LFA graph extension problem over the sample network of Fig. 1 is depicted in Fig. 4a. As it turns out, we need to add 3 complement arcs (and the corresponding reverse arcs): (b, e), (c, e), and either (a, f) or (c, f).

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) that has no node-protecting LFA. Additionally, we add a node to B for each complement arc in $\overline{E}$ eliminating the reverse arcs. 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 invariant to adding high cost arcs to G and, on the other hand, an (u, v) arc, if added to G with high cost, can provide an LFA only for source-destination pairs whose source coincides with u. In particular, for some $a_i \in A$ and some $b_j \in B$ we add an (a_i, b_j) arc to F if for the corresponding source-destination pair (s_i, d_i), complement arc (u_j, v_j) and next-hop n of s_i towards d_i, one of the below conditions holds:

- $n = d_i$, $u_j = s_i$ and $\text{dist}(v_j, d_i) < \text{dist}(v_j, s_i) + \text{dist}(s_i, d_i)$; or
- $n = d_i$, $v_j = s_i$ and $\text{dist}(u_j, d_i) < \text{dist}(u_j, s_i) + \text{dist}(s_i, d_i)$; or
- $n \neq d_i$, $u_j = s_i$, $v_j \neq n$, $\text{dist}(v_j, d_i) < \text{dist}(v_j, s_i) + \text{dist}(s_i, d_i)$ and $\text{dist}(v_j, d_i) < \text{dist}(v_j, n) + \text{dist}(n, d_i)$; or
- $n \neq d_i$, $v_j = s_i$, $u_j \neq n$, $\text{dist}(u_j, d_i) < \text{dist}(u_j, s_i) + \text{dist}(s_i, d_i)$ and $\text{dist}(u_j, d_i) < \text{dist}(u_j, n) + \text{dist}(n, d_i)$.

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.

There are some very appealing properties for solving minLFA over the above bipartite graph model. For instance, the below claim is immediate.

Observation 2 *The LFA graph extension problem is solvable, if and only if there are no isolated nodes in A.*

A further attractive aspect is easy extendability. First, we observe that the model is completely indifferent to whether there are local SRGs configured in the network. When SRGs are to be considered, we only need to verify the SRG-disjointness conditions as of Definition 3 and Definition 4 in addition to the standard conditions. This does not affect either the size of the bipartite model or the time needed to build it, neither it has effect on the minimum set cover algorithms later executed on the model to obtain a solution.

Second, for simplicity we 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, and would not be an LFA at all to the third. Fortunately, ECMP can be seamlessly incorporated into the above model: we add a node to A in G' for each unprotected source-destination-next-hop tuple and connect this node to a node in B if the corresponding complement arc 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.

4.3 Algorithms

The authors in [41] 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. The idea is that 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 [30], the SBT algorithm from [23], the RSBT, MSBT algorithms from [33] and a straightforward backtracking algorithm. An added benefit of using these heuristics is that they immediately provide solutions to the LFA improvement problem beside LFA graph extension; we only need to terminate the algorithms after a pre-defined number of iterations, i.e., after adding a pre-defined number of new links.

4.3.1 The Lovász-Johnson-Chvatal (LJC) method

In [41], 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, [30]). 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 on graph $G'(A, B, F)$

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$
3: $\quad v \leftarrow \text{argmax}_{b \in B} \deg(b)$
4: $\quad B^c \leftarrow B^c \cup \{v\}$
5: $\quad A \leftarrow A \setminus \text{neigh}(v)$
6: $\quad B \leftarrow B \setminus \{v\}$
7: **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 [30]. This means that the algorithm solves the LFA graph extension problem by adding only logarithmically more new links than necessary. 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.

4.3.2 The SBT algorithm

SBT was proposed in [23] 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 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 on graph $G'(A, B, F)$

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$
3: $\quad v \leftarrow \text{argmin}_{b \in B} \deg(b)$
4: $\quad$ **if** $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$ **then**
5: $\quad\quad B^c \leftarrow B^c \cup \{v\}$
6: $\quad\quad A \leftarrow A \setminus \text{neigh}(v)$
7: $\quad$ **end if**
8: $\quad B \leftarrow B \setminus \{v\}$
9: **end while**

4.3.3 The RSBT algorithm

The Reverse SBT algorithm [33], as the name says, does the reverse of SBT 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 $v \leftarrow \text{argmax}_{b \in B} \deg(b)$.

Algorithm 3 RSBT algorithm on graph $G'(A, B, F)$

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$
3: $\quad v \leftarrow \text{argmax}_{b \in B} \deg(b)$
4: $\quad$ **if** $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$ **then**
5: $\quad\quad B^c \leftarrow B^c \cup \{v\}$
6: $\quad\quad A \leftarrow A \setminus \text{neigh}(v)$
7: $\quad$ **end if**
8: $\quad B \leftarrow B \setminus \{v\}$
9: **end while**

4.3.4 The MSBT algorithm

The Modified SBT algorithm [33] 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 4 MSBT algorithm on graph $G'(A, B, F)$

```
1: B^c ← ∅
2: while A ≠ ∅
3:     v ← argmin_{b∈B} deg(b)
4:     if ∃n ∈ neigh(v) with deg(n) = 1 then
5:         B^c ← B^c ∪ {v}
6:         A ← A \ neigh(v)
7:     else
8:         for each a ∈ neigh(v) with deg(a) = 2
9:             w ← u ∈ neigh(a) \ {v}
10:            B^c ← B^c ∪ {w}
11:            A ← A \ neigh(w)
12:        end for
13:    end if
14:    B ← B \ {v}
15: end while
```

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

4.3.5 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 uses an arbitrary complete order ($\prec$) on the nodes of B to perform a recursive lexicographic traversal on the subsets of B. The subsets are examined in ascending order and the smallest cardinality cover found along the way is stored, which is eventually returned as the optimal cover. The pseudo-code for the backtracking algorithm is presented in Algorithm 5.

In every iteration, the recursive procedure BACKTR_R gets the set of nodes W yet to be visited, the current best estimate on the minimum cover B^c, and the node v that was last removed from the cover, and it attempts to remove every $w \in W$ that is larger than v according to $\prec$ checking if the resultant set $W \setminus \{w\}$ is still a cover. If it is, then the algorithm recurses with the new cover. Otherwise, the search is stopped at this point as removing further elements from $W \setminus \{w\}$ cannot result in a cover either. Essentially, this pruning of the search tree is what differentiates the backtracking algorithm from a naive exhaustive search. Note also that the algorithm does not need to maintain a map of the visited subsets, as the lexicographic traversal guarantees that each subset of B is visited at most once. The iteration starts with calling BACKTR_R with the complete node set B, the trivial cover $B^c = B$, and a virtual node v' that is smaller than all nodes in V according to the order ($\prec$), i.e.: $v' \prec v$ for all $v \in V$.

Algorithm 5 Backtracking algorithm on graph $G'(A, B, F)$

```
1: return BACKTR_R(B, B, v')
2: procedure BACKTR_R(W, B^c, v)
3:     for w ∈ W : v ≺ w
4:         if W \ {w} is still a cover then
5:             if |W \ {w}| < |B^c| then
6:                 B^c ← W \ {w}
7:             end if
8:             return BACKTR_R(W \ {w}, B^c, w)
9:         end if
10:    end for
11:    return B^c
12: end procedure
```

Unfortunately, the complexity of the algorithm is exponential as there are $O(2^{|B|})$ potential covers that might all need to be visited in the worst case.

5 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 the LEMON graph library [27] and we compared their performance on numerous real-life ISP topologies. The evaluations were run on a Linux PC with an Intel Xeon 2.53GHz CPU and 3G RAM.

The topologies we tested were as follows. We used the collapsed AS1221, AS1755, AS3257, AS3967 and AS6461 topologies from the Rocketfuel dataset [31]. These graphs are the maps of real service provider networks and come with inferred link costs. We also used the Abilene, Italy, NSF, Germany, AT&T and the extended German backbone (Germ_50) from [44]. 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 [22]. For this dataset, we set costs randomly wherever link costs were not available. The topologies were chosen so as to ensure that the ILP proposed in

Table 1: Link-protecting LFA graph extension results: topology name, number of nodes (n) and arcs (m); number of link costs and arcs added in the preprocessing phase ("Pre. c/e"), initial LFA coverage (η_0), number of new arcs in the optimal solution (ILP), and the number of added arcs ("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
Germ_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			

[41] 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 arcs were removed, links and costs were symmetrized and the preprocessing algorithm from [41] was executed in order to ensure that the optimization problems were always solvable, at least in the case when no SRGs were provisioned. For the case when SRGs were present, we also observed the number of source-destination pairs that could not be covered with an LFA (c.f., Observation 2). First, we discuss the non-SRG case and then we turn to the results for networks with Shared Risk Groups.

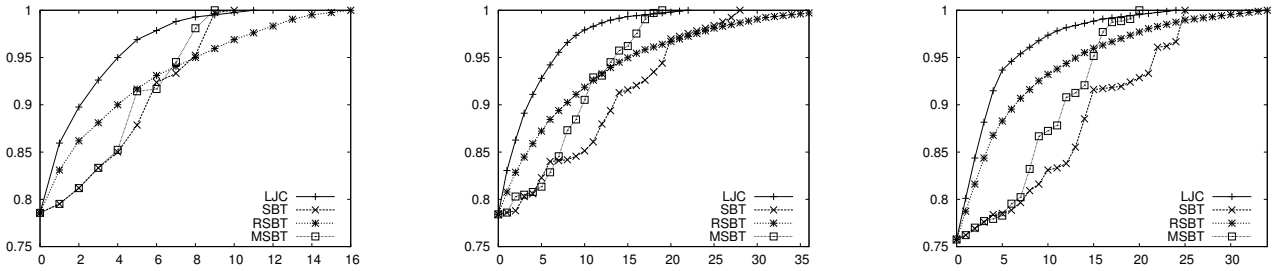
5.1 Results in networks without SRGs

In the first round of the numerical studies, we assumed single link and single node failures. The number of new links added and the running time by the different algorithms for link-protecting LFAs are given in Table 1, and the same results for the node-protecting case are presented in Table 2. The tables also report on the average number of new links added by the different algorithms and the mean deviation from the optimum.

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 the SBT and LJC algorithms also working reasonably, while RSBT is the worst performer. Finally, we find that the execution time of the heuristics is acceptable except for backtracking. LJC is obviously the fastest, while the backtracking algorithm did not

Table 2: Node-protecting LFA graph extension results: topology name, number of nodes (n) and arcs (m); number of link costs and arcs added in the preprocessing phase ("Pre. c/e"), initial LFA coverage (η_0), number of new arcs in the optimal solution (ILP), and the number of added arcs ("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
Germ_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

Fig. 5: 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.

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

5.2 Results in networks with SRGs

In the second round, we have extended our investigation to the case when certain sets of links are configured into local SRGs. Local SRG sets were generated according to an SRG-density parameter $\delta \in [0, 1]$, denoting the fraction of all possible adjacent dual-link sets to be selected as local SRGs. For $\delta = 0$ we add no SRGs at all (i.e., this case corresponds to the single failure scenario), and for the settings $\delta = 0.1$ ($\delta = 0.5$, $\delta = 0.9$, respectively), we add every adjacent link pair with probability 0.1 (respectively 0.5 and 0.9) as an SRG. Since δ is in fact only a statistical expectation on the density of SRGs, we generated 10 different sets of SRGs to each network and we executed the algorithms on all these scenarios, eventually reporting the average of the results.

As it was mentioned in Section 4.1, when local SRGs are present in the network then sometimes we cannot protect every source-destination pair with adding new links. Therefore, before executing the algorithms we used Observation 2 to identify such pairs and we eliminated the corresponding nodes from the bipartite graph model. In our results, we also report the maximum attainable LFA coverage in the network $\eta_{\max}$ as well as the number of unprotectable source-destination pairs in the worst case. The results themselves for some select topologies are in Table 3 for the link-protecting case and Table 4 for node protection under different choices of the SRG density δ. As a consequence of the complexity of the ILP, we have omitted calculating the optimal results for each SRG set and thus the tables do not highlight the mean deviation.

Our observations are as follows. First, we see that as SRG density increases the initial LFA coverage drops drastically. When the SRG density is only 10% the initial LFA coverage lags behind the single-failure case by only a mere 2-6% in both the link- and the node-protecting cases. However, for $\delta = 0.5$ the difference increases to about 20-25% for the better protected networks and about 10% for the less protected ones, to the point that only about half of the source-destination pairs can be protected by an LFA. Finally, when the SRG density is set to $\delta = 0.9$ the initial link-protecting LFA coverage falls to about 10-20%, and a bare 3-12% for the node-protecting case. Surprisingly, we find that the number of new links to be added does not grow at a similar pace: in the link-protecting case for $\delta = 0.1$ we need about 2-3 more links than in the no-SRLG case, while $\delta = 0.5$ introduces about 3-4 more links and even in the case of very large SRG density $\delta = 0.9$ we only need about 7-12 links more (depending on the network size) than when we do not have SRGs at all. This essentially means that even when 90% of every possible adjacent link pair is configured as an SRG we can still improve link-protecting LFA coverage to 100% with only a dozen or so new links. A possible cause behind the insensitivity of our results to SRG density might be that the new links we add never appear in an SRG, and therefore can provide massive improvement in LFA coverage. Similar observations can be made for the node-protecting case as well. We also observe that the rank of the algorithms, in terms of efficiency, does not change under the SRG model: the MSBT algorithm is still the most efficient. Finally, we note that in the majority of the cases the LFA graph extension problem could be solved to optimality (or very close to it), suggesting that the pathological case identified in Observation 2 rarely appears in practice.

The results for the LFA graph improvement problem under the SRG model for some select topologies are shown in Fig. 6. The observations are similar as in the non-SRG case: when the goal is merely to improve the coverage instead of shooting for 100% the LJC algorithm is clearly the best algorithmic strategy.

6 Conclusions

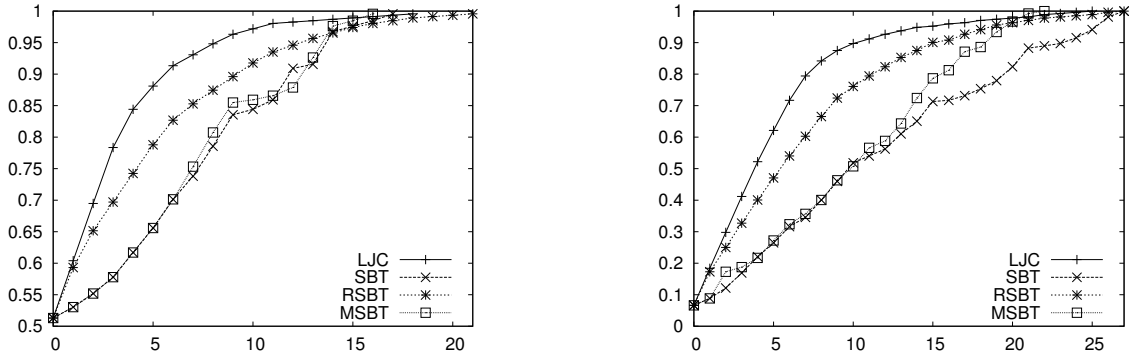
Currently, Loop-Free Alternates seems the most plausible choice to provide fast protection in IP networks. By using LFA, ISPs can gain a solid level of protection with minimal effort and with the help of the LFA-based network optimization algorithms presented in this paper the failure case coverage can be improved even further. We have extended the bipartite graph model from [41] to solve this problem under essentially any failure model relevant to practice, and we applied several heuristics from the literature to this representation to obtain approximate and optimal solutions. The most important conclusion is that, even-though NP-complete, the LFA graph extension problem is efficiently approximable. To support this claim, we presented a logarithmic upper bound on the worst case performance of the heuristic based on Lovász-Johnson-Chvatal (LJC) minimum set-cover method and we also reported on the results of 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

Table 3: Link-protecting LFA graph extension results for some select topologies with SRGs: topology name; maximum attainable LFA coverage ($\eta_{\max}$) and the worst-case number of unprotectable source-destination pairs in parentheses; initial LFA coverage (η_0); and the average of number of added arcs ("ext") for each algorithm for small ($\delta = 0.1$), medium ($\delta = 0.5$), and large ($\delta = 0.9$) SRG density

	$\delta = 0.1$						$\delta = 0.5$						$\delta = 0.9$					
Topology	$\eta_{\max}$	η_0	LJC	SBT	RSBT	MSBT	$\eta_{\max}$	η_0	LJC	SBT	RSBT	MSBT	$\eta_{\max}$	η_0	LJC	SBT	RSBT	MSBT
AS1239	1(0)	0.849	8.3	8.4	13.4	7.8	0.999(1)	0.662	15.3	15.8	24.5	14.6	0.999(1)	0.216	21.2	20.6	32.5	18.4
AS1755	1(0)	0.844	7.9	9.2	12.1	7.8	1.000(0)	0.599	12.1	12.9	15.0	11.5	1.000(0)	0.151	15.1	12.9	17.7	12.1
AS3967	1(0)	0.748	11.1	10.9	17.0	9.6	1.000(0)	0.510	12.9	13.3	21.0	12.0	1.000(0)	0.159	15.2	13.7	24.2	13.5
AT&T	1(0)	0.769	13.1	12.8	13.9	12.5	1.000(0)	0.506	17.2	15.0	19.8	14.6	0.996(2)	0.105	20.1	17.3	23.3	16.5
Digex	1(0)	0.274	26.9	27.4	45.9	25.9	1.000(0)	0.171	28.4	28.2	44.8	26.1	1.000(0)	0.036	28.3	26.3	42.9	25.4
Germ_50	1(0)	0.850	25.4	35.7	49.5	27.6	1.000(0)	0.594	33.0	40.4	64.1	35.9	1.000(0)	0.139	38.6	35.2	69.0	34.8
Germany	1(0)	0.631	13.5	12.0	14.2	11.5	1.000(0)	0.409	14.4	13.2	14.7	12.6	1.000(0)	0.085	16.1	13.9	16.1	13.0
Italy	1(0)	0.738	22.4	27.9	39.2	20.9	0.998(2)	0.489	27.4	28.7	42.6	24.9	0.998(0)	0.112	31.6	29.1	43.8	27.4
NSF	1(0)	0.811	13.5	16.1	29.0	13.0	1.000(0)	0.503	18.1	20.7	32.5	16.9	1.000(0)	0.116	21.3	20.0	34.5	19.3
Average:	1	0.723	15.8	17.8	26.0	15.2	0.999	0.493	19.9	20.9	31.0	18.8	0.999	0.124	23.1	21.0	33.8	20.0

Table 4: Node-protecting LFA graph extension results for some select topologies with SRGs: topology name; maximum attainable LFA coverage ($\eta_{\max}$) and the worst-case number of unprotectable source-destination pairs in parentheses; initial LFA coverage (η_0); and the average of number of added arcs for the LJC, SBT, RSBT, and MSBT heuristics for small ($\delta = 0.1$), medium ($\delta = 0.5$), and large ($\delta = 0.9$) SRG density

	$\delta = 0.1$						$\delta = 0.5$						$\delta = 0.9$					
Topology	$\eta_{\max}$	η_0	LJC	SBT	RSBT	MSBT	$\eta_{\max}$	η_0	LJC	SBT	RSBT	MSBT	$\eta_{\max}$	η_0	LJC	SBT	RSBT	MSBT
AS1239	0.999(1)	0.730	24.7	26.4	36.9	21.8	0.999(1)	0.568	30.7	34.8	45.9	29.4	0.999(1)	0.170	37.9	37.1	52.1	35.5
AS1755	1.000(0)	0.730	18.9	18.4	27.0	17.7	1.000(0)	0.494	22.7	22.8	28.4	20.9	1.000(0)	0.115	26.6	25.6	30.8	24.1
AS3967	1.000(0)	0.604	19.4	21.0	31.7	19.3	1.000(0)	0.402	21.6	24.6	31.5	21.9	1.000(0)	0.117	24.5	26.4	32.4	25.1
AT&T	0.996(2)	0.538	43.0	43.6	52.6	40.1	0.996(2)	0.314	45.2	45.2	50.0	41.7	0.996(2)	0.065	47.1	47.0	51.5	42.3
Digex	1.000(0)	0.270	37.2	38.8	50.3	34.6	1.000(0)	0.170	38.2	39.9	52.2	37.5	1.000(0)	0.036	39.4	39.3	49.0	38.9
Germ_50	1.000(0)	0.770	44.9	62.6	81.7	49.1	1.000(0)	0.512	51.0	64.4	85.1	56.2	1.000(0)	0.113	56.8	56.5	88.3	54.6
Germany	1.000(0)	0.509	21.5	22.2	26.6	19.1	1.000(0)	0.295	23.2	23.9	26.7	20.3	1.000(0)	0.050	25.2	25.0	26.3	22.3
Italy	0.996(2)	0.534	43.2	48.7	61.3	39.2	0.994(2)	0.348	46.6	49.4	61.2	44.4	0.998(2)	0.084	48.8	47.0	59.9	46.3
NSF	1.000(0)	0.572	25.4	31.3	38.2	23.4	1.000(0)	0.334	28.9	30.4	40.6	26.7	1.000(0)	0.070	32.0	30.7	39.9	28.4
Average:	0.999	0.584	30.9	34.8	45.1	29.4	0.998	0.381	34.2	37.3	46.8	33.2	0.998	0.091	37.6	37.2	47.8	35.3



(a) AT&T (b) Germany

Fig. 6: LFA coverage in each iteration of different heuristics in the link-protecting case for AT&T with SRG density $\delta = 0.5$, and node-protecting case for Germany topology with $\delta = 0.9$

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. Atlas, A., Kebler, R., Konstantynowicz, M., Enyedi, G., Császár, A., Shand, M.: An architecture for IP/LDP fast-reroute using maximally redundant trees. Internet Draft (2011)
2. Atlas, A., Zinin, A.: Basic specification for IP fast reroute: Loop-Free Alternates. RFC 5286 (2008)
3. Bryant, S., Filsfils, C., Previdi, S., Shand, M.: IP Fast Reroute using tunnels. Internet Draft (2007)

4. Bryant, S., Shand, M., Previdi, S.: IP fast reroute using Not-via addresses. Internet Draft (2010)
5. Callon, R.: Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195 (1990)
6. Čičic, T., Hansen, A.F., Apeland, O.K.: Redundant trees for fast IP recovery. In: Broadnets, pp. 152–159 (2007)
7. Cisco Systems: Cisco IOS XR Routing Configuration Guide, Release 3.7 (2008)
8. Császár, A., Enyedi, G., Hidell, M., Rétvári, G., Sjödin, P.: Converging the evolution of router architectures and IP networks. IEEE Network Magazine, Special Issue on Advances in Network Systems Architecture **21**(4), 8–14 (2007). DOI 10.1109/MNET.2007.386464
9. Enyedi, G., Rétvári, G., Cinkler, T.: A novel loop-free IP fast reroute algorithm. In: EUNICE (2007)
10. Enyedi, G., Rétvári, R.: Finding multiple maximally redundant trees in linear time. to appear in Periodica Polytechnica, available online: `http://opti.tmit.bme.hu/~enyedi/ipfrr/distMaxRedTree.pdf` (2010)
11. Enyedi, G., Szilágyi, P., Rétvári, G., Császár, A.: IP Fast ReRoute: lightweight Not-Via without additional addresses. In: INFOCOM Mini-conf (2009)
12. Fortz, B., Rexford, J., Thorup, M.: Traffic engineering with traditional IP routing protocols. IEEE Comm. Mag. **40**(10), 118–124 (2002)
13. Francois, P., Bonaventure, O.: An evaluation of IP-based fast reroute techniques. In: ACM CoNEXT, pp. 244–245 (2005)
14. Francois, P., Filsfils, C., Evans, J., Bonaventure, O.: Achieving sub-second IGP convergence in large IP networks. SIGCOMM Comput. Commun. Rev. **35**(3), 35–44 (2005)
15. Garey, M., , Johnson, D.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1990)
16. Gjoka, M., Ram, V., Yang, X.: Evaluation of IP fast reroute proposals. In: IEEE Comsware (2007)
17. Gomes, T., Simoes, C., Fernandes, L.: Resilient routing in optical networks using SRLG-disjoint path pairs of min-sum cost. Springer Telecommunication Systems Journal (2010)
18. Hewlett-Packard: HP 6600 Router Series: QuickSpecs (2008). Available online: `http://h18000.www1.hp.com/products/quickspecs/13811_na/13811_na.PDF`
19. Hock, D., Hartmann, M., Menth, M., Piro, M., Tomaszewski, A., Zukowski, C.: Comparison of IP-Based and Explicit Paths for One-to-One FastReroute in MPLS Networks. Springer Telecommunication Systems Journal (2011)
20. Hokelek, I., Fecko, M., Gurung, P., Samtani, S., Cevher, S., Sucec, J.: Loop-free IP Fast Reroute using local and remote LFAPs. Internet Draft (2008)
21. Juniper Networks: JUNOS 9.6 Routing protocols configuration guide (2009)
22. Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M.: The Internet Topology Zoo. `http://www.topology-zoo.org`
23. Kulaga, P., Sapiecha, P., Sej, K.: 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)
24. Kvalbein, A., Hansen, A.F., Čičic, T., Gjessing, S., Lysne, O.: Multiple routing configurations for fast IP network recovery. IEEE/ACM Trans. Netw. **17**(2), 473–486 (2009)
25. Kwong, K.W., Gao, L., Guerin, R., Zhang, Z.L.: On the feasibility and efficacy of protection routing in IP networks. In: INFOCOM 2010, long version appears as Tech. Rep. 2009, University of Pennsylvania (2010)
26. Lee, S., Yu, Y., Nelakuditi, S., Zhang, Z.L., Chuah, C.N.: Proactive vs reactive approaches to failure resilient routing. In: INFOCOM (2004)
27. LEMON – Library for Efficient Modeling and Optimization in Networks. `http://lemon.cs.elte.hu/` (2009)
28. Li, A., Francois, P., Yang, X.: On improving the efficiency and manageability of NotVia. In: ACM CoNEXT (2007)
29. Li, A., Yang, X., Wetherall, D.: SafeGuard: safe forwarding during route changes. In: ACM CoNEXT, pp. 301–312 (2009)
30. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Mathematics **13**(4), 383–390 (1975)
31. Mahajan, R., Spring, N., Wetherall, D., Anderson, T.: Inferring link weights using end-to-end measurements. In: ACM IMC, pp. 231–236 (2002)
32. Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C., Ganjali, Y., Diot, C.: Characterization of failures in an operational IP backbone network. IEEE/ACM Trans. Netw. **16**(4), 749–762 (2008)
33. Mazbic-Kulma, B., Sep, K.: Some approximation algorithms for minimum vertex cover in a hypergraph. In: M. Kurzynski, E. Puchala, M. Wozniak, A. Zolnierek (eds.) Computer Recognition Systems 2, *Advances in Soft Computing*, vol. 45, pp. 250–257. Springer Berlin / Heidelberg (2007)
34. Menth, M., Hartmann, M., Hock, D.: Routing optimization with IP Fast Reroute. Internet Draft (2010)
35. Menth, M., Hartmann, M., Martin, R., Čičić, T., Kvalbein, A.: Loop-free alternates and not-via addresses: A proper combination for IP fast reroute? Comput. Netw. **54**(8), 1300–1315 (2010)
36. Moy, J.: OSPF Version 2. RFC 2328 (1998)
37. Nagy, M., Rétvári, G.: An evaluation of approximate network optimization methods for improving ip-level fast protection with loop-free alternates. In: Proc. of the 4th International Workshop on Reliable Networks Design and Modeling (RNDM) (2011)
38. Pan, P., Swallow, G., Atlas, A.: Fast reroute extensions to RSVP-TE for LSP tunnels. RFC 4090 (2005)
39. Previdi, S.: IP Fast ReRoute technologies. APRICOT (2006)
40. Rétvári, G., Csikor, L., Tapolcai, J., Enyedi, G., Császár, A.: Optimizing IGP link costs for improving IP-level resilience. In: Proc. International Workshop on Design Of Reliable Communication Networks (DRCN) (2011)
41. Rétvári, G., Tapolcai, J., Enyedi, G., Császár, A.: IP Fast ReRoute: Loop Free Alternates revisited. In: INFOCOM 2011, pp. 2948–2956 (2011)
42. Schollmeier, G., Charzinski, J., Kirstadter, A., Reichert, C., Schrodi, K., Glickman, Y., Winkler, C.: Improving the resilience in IP networks. In: High Performance Switching and Routing, 2003, HPSR. Workshop on, pp. 91–96 (2003)
43. Shand, M., Bryant, S.: IP Fast Reroute framework. RFC 5714 (2010)
44. SNDlib: Survivable fixed telecommunication network design library. `http://sndlib.zib.de`
45. Sterbez, J., Cetinkaya, E., Hameed, M., Jabbar, A., Qian, S., Rohrer, J.: Evaluation of Network Resilience, Survivability , and Disruption Tolerance: Analysis, Topology Generation, Simulation and Experimentation. Springer Telecommunication Systems Journal (2011)

46. Swallow, G., Bryant, S., Andersson, L.: Avoiding equal cost multipath treatment in MPLS networks. RFC 4928 (2007)
47. Thorup, M., Roughan, M.: 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`
48. Viet, H.T., Francois, P., Deville, Y., Bonaventure, O.: 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)
49. Zhong, Z., Nelakuditi, S., Yu, Y., Lee, S., Wang, J., Chuah, C.N.: Failure inferencing based fast rerouting for handling transient link and node failures. In: INFOCOM (2005)

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 first-year PhD student at the High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, BME and takes part in researching IP FastReRoute solutions. He has experience in C/C++/LEMON, Java, Python and Linux.

János Tapolcai received his M.Sc. ('00 in Technical Informatics), and Ph.D. ('05 in Computer Science) degrees in Technical Informatics from Budapest University of Technology and Economics (BME), Budapest, Hungary. Currently he is an Associate Professor at the High-Speed Networks Laboratory at the Department of Telecommunications and Media Informatics at BME. His research interests include applied mathematics, combinatorial optimization, mathematical programming, optical and IP level routing and survivability, availability analysis and distributed computing. He has been involved in several related European and Canadian projects. He is an author of over 80 scientific publications, and is the recipient of the Best Paper Award in ICC'06 and in DRCN'11.

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.