Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics

Novel Network Optimization and Data Compression Methods

Máté Imre Nagy

Summary of PhD Dissertation

Advisor:

Gábor Rétvári, Ph.D. (Senior Research Fellow)

Dept. of Telecommunications and Media Informatics and
Hungarian Academy of Science Future Internet Research Group,
Budapest University of Technology and Economics

Budapest, Hungary
2021.

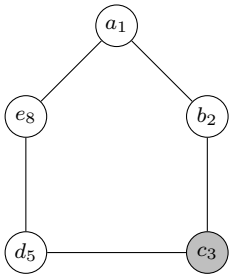
1 Introduction

Today, thanks to the appearance of smartphones, we all have daily hands-on experience about latency-sensitive services like telephony, instant messaging or video streaming [1]. However, any system built to enable communication between endpoints has major challenges to face. A well-known example is video streaming which often annoys the user with sudden decrease in quality (e.g. due to limited bandwidth) or frequent disconnections caused by malfunctioning network devices. Therefore, throughout the entire lifetime of a network, the main goals of the operators are **(I) to mitigate the possible failure events** in the infrastructure and **(II) to ensure that all devices operate with maximum efficiency**.

Getting prepared for arbitrary network failures is already a complex task on its own. Unfortunately, the Internet Protocol (IP), is built to work in an unreliable, called "best-effort" manner that poses difficult challenges to operators when they try to guarantee the five nines (99.999%) availability. The distributed system, managed by routers, can easily get into an inconsistent state where forwarding tables of routing devices show contradicting entries due to failures in the network. This can often lead to the creation of forwarding loops where packets keep bouncing between the interfaces of devices that eventually eats up all their processing capacity.

The remedy for this phenomenon is to stop handling all traffic until the routers resolve the situation by a so-called restoration process [2, 3]. As soon as it is finished, the traffic is good to go — but only if it survived the downtime at all. This can often take hundreds of milliseconds [4] that is usually unacceptable for highly-sensitive traffic. To overcome this problem, the networking community introduced the IPFRR framework [5], which aims to get prepared for the failures *in-advance* and takes action only *local to the failure*. The challenge here is to find alternates that guarantee the loop-free delivery of packets during this inconsistent state of the network. One particular solution, called the *Loop-Free Alternate*(LFA) [6], emerged lately from the wide variety of proposals.

Let us overview this concept in Fig. 1a. The five routers are connected with links of uniform cost and packets follow the shortest possible path between two endpoints. Assuming the failure of link (c, b) the $c \to a$ traffic gets stopped until the restoration of the network is finished, or unless we find a proper secondary path to which c can redirect its packets. Luckily, it is easy to notice that regardless the failure of link c-b the neighboring router, d, is still able to deliver packets to a along the path d-e-a. And most importantly, this path is guaranteed to be loop-free as it does not contain the source of the packet, c, so there is no risk of triggering a ping-pong effect between c and d. Unfortunately, there are no valid LFAs



(a) Sample topology

dest.	nh.	LFA
–	d	–
a	b	d
b	b	–
d	d	–
e	d	b

(b) Forwarding table of c

Figure 1. Demonstration of Loop-Free Alternates in a selected topology.

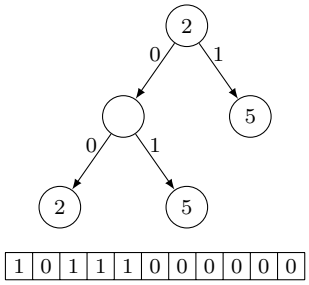
for all the source-destination pairs as it is depicted on Fig. 1b. Therefore, the first goal of the Dissertation is to *formulate different network optimization problems and design novel algorithms incorporating the most common failure models in order to bring LFA protection perfect in communication networks.*

Our intention to increase the efficiency of networking related data processing is not in the least easier task either. Not just the growth in scale of data, but also the patterns and types of queries went through significant changes lately. Obviously, the way of representing information followed these changes that gave rise of a wide set of novel data structures, but without never losing sight of the ultimate goal: *providing the fastest possible answer by consuming the smallest amount of storage space.* A practical example is the poorly performing routing device that can hugely add to the overall latency of media streams or introduce jitter in the network. To demonstrate this problem, we switch to a more realistic representation of network addresses that is based on ids, and so we present the corresponding forwarding table of router c in (Fig. 2a). The prefix column shows how to merge multiple rules by using the binary format of destination addresses and the wildcard character ($*$) to match any bits. Thanks to this trick, the number of rules is now reduced to 4, but we still need to step through all the rules if the entry for the destination of a packet resides in the last row of the table.

Since this is rather inefficient, the networking community turned to the prefix-tree representation of the forwarding table that is available on Fig. 2b. Undoubtedly, the tree gives a more simple and eye-catching overview on the rules, but also raises the question of how to encode it binary? In our example we pick the level-order encoding that walks through the leaves from the top layer to the bottom, and writes 1s for nodes holding value and 0s otherwise (see the resultant bitvector underneath the tree). The encoding reflects the structure of the tree, and we store the symbols alongside in an array. This requires only 11 bits,

dest.	prefix	nh.	LFA
-	*	5	2
1,2	00*	2	5
5	0101	5	2
8	1*	5	2

(a) Forwarding table



(b) Prefix tree

Figure 2. Demonstration of forwarding table compression.

but *to be able to answer queries on the encoded format some special operations are needed.* Since the layout of the structure is defined by the position of 0s and 1s in the bitvector, we must support queries like "what is the position of the i-th 0?" or "return the number of 1s until the i-th position". Researchers found that the bitvector can be freely compressed as long as they are still able to provide fast answers for these types of queries. However, space reduction and fast access to the data require several trade-offs. Consequently, the last goal of the Dissertation is to propose *a novel succinct data structure that offers noticeable performance gain in queries compared to the state-of-the-art implementations while achieves entropy-constrained space reduction on the storage size.*

2 Research Goals

Latency sensitive media streams require ultra fast reaction to failures. The distributed scheme of IGP networking prevents to heal the network fast enough, therefore a local and proactive IPFRR scheme is necessary to be applied. The only IPFRR method that was able to gain remarkable attention so far is the Loop-Free Alternates that does not provide protection against all the failure cases. On the other hand, since LFA selection is already available in major commercial router implementations [7, 8, 9], there is a huge potential for the instant and seamless upgrade to a full IPFRR protected network once we can get rid of this imperfect limitation of LFA. To improve the level of LFA protection in the network, we take advantage of the dependency between the number of protected node pairs and the layout of the topology.

Our first goal is to add SRG-disjoint complement edges to the network that render available alternates to otherwise unprotected source-destination pairs. In particular, we would like to know *if the problem is solvable at all for any given topology*, and if so, then

how to find the minimum set of complement edges that by adding to the network gives perfect protection. Finally, it is also important to know *how to generate the set of complement edges that increases LFA coverage the most with a given upper limit on the number of new links that can be added to the network*. Second, we aim at the same target but this time without touching the physical topology. We take advantage of *virtualization* techniques in order to construct a protective overlay on top of the physical topology. Accordingly, we *identify and compare different cost assignment strategies on virtual links* to be able to maximize the number of protected node pairs with minimum number of virtual devices.

Besides networking reliability, the performance of information retrieval operations also have great impact on the overall end-user experience. Therefore, our final goal is to present a novel compression scheme, called R3D3 ("RRR–Developed Data structure for big Data"), that by taking advantage of larger block sizes achieves better execution time compared to the state-of-the-art solutions. The idea is to resolve one of the major shortcoming of current compression methods: squeezing the index as well along the useful data. We call it a *doubly-opportunistic data structure* that allows random `access`, `rank` and `select` queries in $O(\log n)$ time.

3 Research Methodology

The methodology of the research follows the scheme where theoretical results are obtained with analytical techniques and then get verified with numerical evaluations. In many cases the complexity of the problems is *nondeterministic polinomial (NP)* that requires to formalize Integral Linear Programs to find the optimal solution. Since LP solvers usually run with very poor performance, we propose *approximation algorithms* that perform close to the optimum. The set of *simulation* tools that we developed to support our research is publicly available on GitHub [10]. Most of the source files are written in C++ and they build on open source libraries like LEMON [11], Gurobi [12] or the BOOST unit test framework [13].

We use a wide variety of sample networks as inputs. The collapsed AS1221, AS1755, AS3257, AS3967 and AS6461 topologies are taken from the Rocketfuel dataset [14]. These graphs come from real service provider networks and inferred link costs. We also use the Abilene, Italy, NSF, Germany, AT&T and the extended German backbone (Germ_50) from [15]. Further topologies are obtained from the Topology-Zoo project's dataset [16] and we set costs randomly wherever link costs were not available. We have removed all parallel arcs and made links and costs symmetric.

For evaluating our novel compression scheme, R3D3, we took different sources of textual

data from the Pizza-Chili data set [17], the UCSC Genome database [18] and from The Calgary Corpus set [19]. The input for evaluating R3D3 on IP forwarding tables (FIBs) were taken from [20]. The source code is freely available at [10].

4 New Results

4.1 LFA Graph Extension under Correlated Failures

Thesis Group 1. *[J3, J4, C2, C3] I have formally defined the failure tolerant LFA graph extension problem,* $\text{minLFA}_{\text{SRG}}$. *I have proposed pre-processing and approximation algorithms with provable worst-case error bound, and I have provided extensive empirical evidence suggesting that the proposed algorithms are able to attain 100% LFA protection by adding 30% of the links already existing in the network in the link-protecting case and 60% in the node-protecting case. I have shown that there is a dependency between the number of necessary complement edges and the density of Shared Risk Groups.*

As the title suggests the first goal is to smartly extend the network *with the minimum number of complement edges that increase LFA coverage the most.* We also take into consideration jointly failing links, as the logical representation of networks seen by the IP control plane is often different from the physical layout.

4.1.1 Definition and Attributes

Let $S = \{(i,j) \in E\}$ be an SRG containing a set of links (i,j). The number of links is at most $\deg(i) - 1$, 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):

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

For some source s and destination d, let e be the default next-hop of s towards d. Then, some neighbor t of s is an *SRG-disjoint link-protecting LFA* [6] for s to d if

Definition 1. *For some source s, destination d, and default $s-d$ next-hop e, a neighbor t of s is an SRG-disjoint link-protecting LFA for s to d if*

i) $t \neq e$,

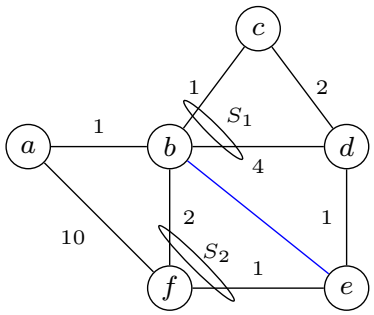


Figure 3. Sample network with an additional link (marked with blue) for improving LFA coverage.

ii) $\text{dist}(t,d) < \text{dist}(t,s) + \text{dist}(s,d)$, *and*
iii) $(s,t) \notin S(s,e)$.

Let us overview these principles on the topology depicted on Fig. 3. The traffic from a to d flows on the default shortest path a-b-c-d, and we assume the failure of link (a,b). Due to the failure, the only remaining neighbor of a is f but the question if it is an LFA to d? The answer is yes, since the shortest path f-e-d does not include the source node, a. The node pair, b-c, is unprotected on the other hand against the failure of link (b,c), since the only available neighbors of b, a and f, are upstream to the destination c. Now if we insert the link (b,e) with sufficiently high link cost setting that prevents the traffic to flow back to b, then e becomes an SRG-disjoint link-protecting $b \to c$ LFA.

Similarly to link protection, there are LFAs that are able to protect against the failure of the next-hop. For instance, node f is a link-protecting LFA for $a \to d$ since the shortest-path from f does not pass the source node, a. What is more, node f also protects against the failure of the next-hop b as well, since the $f \to d$ shortest path does not traverse b either. We call this property the *SRG-disjoint node-protecting LFA* condition.

Definition 2. *For some source s, destination d, and default $s-d$ next-hop e, a neighbor t of s is an SRG-disjoint node-protecting LFA for s to d if*

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

What is left is to provide an adequate measure for the level of LFA protection of a given network. Simply we use the ratio of protected versus unprotected node pairs that is

expressed as follows:

$$\eta_{\mathrm{LP\,/\,NP}}(G) = \frac{\#\ (s,d) \text{ pairs with link/node-protecting LFA}}{\#\text{all } (s,d) \text{ pairs}}\ . \tag{1}$$

We call $\eta_{\mathrm{LP}}(G)$ and $\eta_{\mathrm{NP}}(G)$ link and node-protecting *LFA coverage* respectively. This value depends on the layout of the topology, as well as the link cost settings and finally the density of SRGs in the network. Our example on Fig. 3 holds $\eta_{\mathrm{LP}}(G) = 0.73$ and $\eta_{\mathrm{NP}}(G) = 0.66$.

We define the correlated failure tolerant LFA graph extension problem. Here, the task is 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. Formally:

Definition 3. Correlated failure tolerant LFA graph extension problem ($\mathrm{minLFA_{SRG}}$): *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 $\eta_{\mathrm{LP}}(G(V, E \cup F)) = 1$ and* (ii) *the shortest paths in $G(V,E)$ coincide with the shortest paths in $G(V, E \cup F)$?*

The above definition is straightforward to adapt to the the node-protecting case as well by substituting $\eta_{\mathrm{LP}}(G)$ with $\eta_{\mathrm{NP}}(G)$. Note that newly inserted links are not part of any existing SRGs and they never form new ones. This consideration comes from the fact that the role of the additional links is to provide ultimate reliability in the network without being affected by any other failure of network elements. Next, we define the complexity of $\mathrm{minLFA_{SRG}}$.

Thesis 1.1. *The* $\mathrm{minLFA_{SRG}}$ *problem is NP-complete.*

Asking for link extension that attains 100% failure coverage is often too ambitious goal. Therefore, we also consider a relaxed version of the problem, called the correlated failure tolerant *LFA graph improvement* problem, which asks for realizing the highest improvement possible by adding only a limited number of new links.

Definition 4. Correlated failure tolerant 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 $G(V, E \cup F)$?*

The LFA graph improvement problem is also NP-complete, because otherwise we could solve $\mathrm{minLFA_{SRG}}$ with solving LFA graph improvement with setting $k = |V^2| = n(n-1)$.

Next, before jumping to the solution of the problem we examine if it is solvable at all for any arbitrary input.

4.1.2 The pre-processing problem

LFA is based on the idea to pass packets *to a neighbor that is not upstream* towards the destination. We take one step further by proposing minLFA$_{\text{SRG}}$ that settles for *any node in the graph that is not upstream* towards the destination that we can turn into an LFA by adding a complement edge to it. Unfortunately, just like sometimes proper neighbors do not exist, it is also possible that there isn't a single node in the graph which would not be upstream to a given destination.

Accordingly, for being able to provide an LFA from s to d, one must ensure that d is accessible through at least two different shortest paths. The algorithm that makes each destination accessible from two different neighbors is called *pre-processing*, and it either inserts new links into the network or modifies the cost of existing links.

Definition 5. LFA graph extension pre-processing problem: *Given a simple, weighted, symmetric digraph $G(V, E)$ containing S set of SRGs, find a modified graph $G'(V, E')$ and a modified weight set, so that the* minLFA$_{\text{SRG}}$ *is* solvable over G'. *Is there an E' with the minimum set of new and altered edges that results in a minimum difference of shortest paths in G and G'?*

We present a novel algorithm that exhibits considerable performance gain compared to the best-known pre-processing scheme [21]. Let us denote the nodes need to be pre-processed with $V_p \in V$, and the arcs $E_p : \forall(i, j) \in E$ where $i, j \in V_p$. Hereby we get $G_p(V_p, E_p)$ as a subset of $G(V, E)$. The following algorithm takes $G_p(V_p, E_p)$ as an input, and returns $G'(V, E')$ in four steps:

Algorithm 1 Improved pre-processing algorithm

1.) Find minimum edge cover in $G_p(V_p, E_p)$.
2.) Pair remaining nodes of V_p and insert edges between them.
3.) Insert new edge between $u, v \in V_p$ where u is the last not preprocessed node.
4.) Assign cost to the links selected above so that only shortest paths between $v \in V_p$ alter.

Thesis 1.2. *The algorithm presented in Alg. 1 has polynomial running time and terminates with optimal result.*

4.1.3 Solving minLFA$_{\text{SRG}}$ with the Bipartite Graph Model

Thesis 1.3. *There is an algorithm to build the bipartite representation for any instance of the node-protecting* minLFA$_{\text{SRG}}$ *problem in* $O(n^3)$ *steps.*

The first step to solving the minLFA$_{\text{SRG}}$ problem is to build a suitable model. As the proof of Theorem 1.1 suggests the problem can be reduced to finding the minimum set cover in bipartite graphs [22]. The idea is therefore to build a construction, a suitable bipartite graph model, which then can be used to solve minLFA$_{\text{SRG}}$ as well.

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 an SRG-disjoint link-protecting LFA to (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).

The bipartite graph $G'(A, B, F)$ 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. What is left is to solve the minimum set cover problem ($minSC$) over G'.

4.1.4 Algorithms

We have found in Thesis. 1.1 that the minLFA$_{\text{SRG}}$ problem is NP-complete. This level of complexity makes uncertain to obtain the optimal results in reasonable time for larger networks. On the other hand, we recognize that finding minimum set cover in a bipartite graph is equal to the problem of finding minimum vertex cover in hypergraphs [23] for which several efficient heuristics are available from the literature.

The algorithm of Lovász-Johnson-Chvatal (LJC, [24] 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. SBT was proposed in [25] to find an approximate cover that is, in contrast to LJC, minimal in the sense of inclusion.

Algorithm 2 LJC

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$ **do**
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**

Algorithm 3 SBT

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$ **do**
3: $\quad v \leftarrow \text{argmin}_{b \in B} \deg(b)$
4: $\quad$ **if** $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$
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**

Algorithm 4 RSBT

1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$ **do**
3: $\quad v \leftarrow \text{argmax}_{b \in B} \deg(b)$
4: $\quad$ **if** $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$
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**

Algorithm 5 MSBT

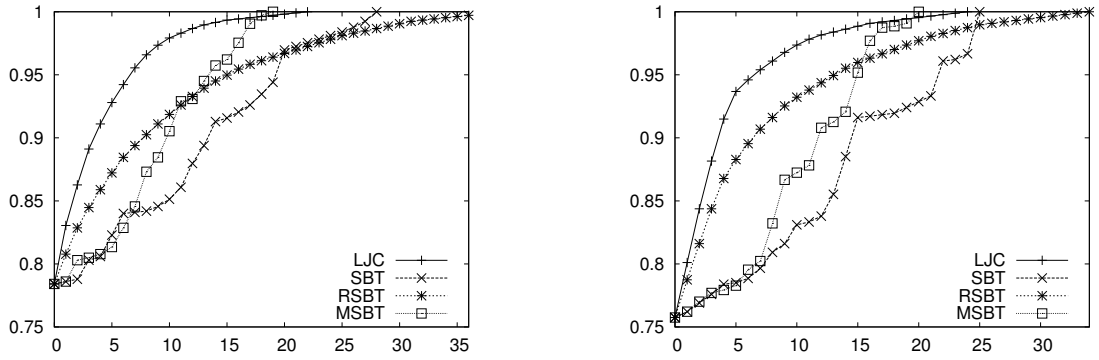
1: $B^c \leftarrow \emptyset$
2: **while** $A \neq \emptyset$
3: $\quad v \leftarrow \text{argmin}_{b \in B} \deg(b)$
4: $\quad$ **if** $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$
5: $\quad\quad B^c \leftarrow B^c \cup \{v\}$
6: $\quad\quad A \leftarrow A \setminus \text{neigh}(v)$
7: $\quad$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

Figure 4. Pseudo-codes of the LJC and SBT, RSBT and MSBT algorithms on graph $G'(A, B, F)$

The Reverse SBT algorithm [23], 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 3 with the slight modification that instead of line 3 we write $v \leftarrow \text{argmax}_{b \in B} \deg(b)$. The Modified SBT algorithm [23] applies a small optimization step to SBT. Note that the SBT, RSBT and MSBT algorithms generate covers that are minimal in the sense of inclusion.

4.1.5 Numerical Evaluations

Thesis 1.4. *[J3, C2, C3] By extensive simulation studies, I have shown that the proposed*



(a) Italy

(b) AS1239

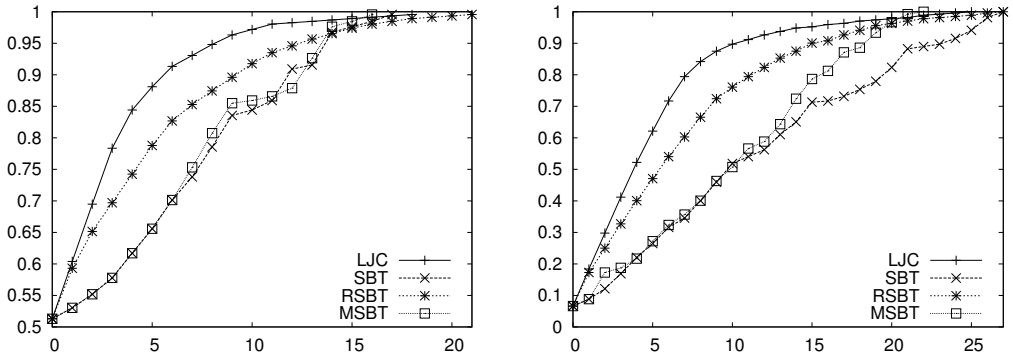
Figure 5. LFA coverage in each iteration of different heuristics in the link-protecting case for Italy, and node-protecting case for the AS1239 topology.

heuristics are capable to attain perfect link-protecting (node-protecting, respectively) LFA coverage with adding only a 10–40% of the number of links in small and middle size networks and overshoots the optimum with at most 5–15% in most cases, and the LJC and MSBT heuristics are the most efficient amongst the proposed approximating algorithms.

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. We executed our measurements under different failure models. The topologies were chosen so as to ensure that the ILP proposed in [21] still runs — at least in the non-SRG case — and so we can compare the performance of the heuristics to each other as well as to the optimum. Before actually running the algorithms, the improved pre-processing algorithm was executed in order to ensure that the optimization problems were always solvable.

In the no-SRG case, 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.

Then, in the second round, 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. Surprisingly, we find that the number of new links to be added does not grow at a similar



(a) AT&T (b) Germany

Figure 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 $\delta = 0.9$

pace: in the link-protecting case for $\delta = 0.1$ we need about 2-3 more links than in the non-SRG 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. 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. For further details, see the Dissertation.

4.2 Node Virtualization for IP Level Resilience

Thesis Group 2. *I have defined the Resilient IP Overlay Problem, called* RIOD*, which aims to provision a virtual overlay on top of IP networks in order to achieve perfect LFA protection. I have shown that the complexity of the problem is NP-complete, and it is always solvable for any given input. I have introduced an algorithmic framework that can be parameterized either to return minimal number of virtual elements or to terminate in polynomial time. By performing numerical simulations, I have found that on average a single virtual per physical router is enough to reach full protection.*

Router virtualization is a technique used for sharing the resource of a single IP routing device between multiple virtual instances [26]. Accordingly, virtual routers are indistinguishable from physical routers, each instance having its own forwarding and control planes, which allows us to assign virtual routers as LFAs to routers that originally did not have one.

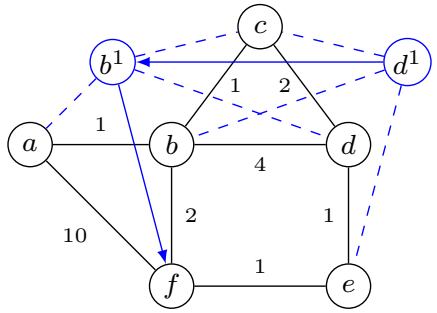


Figure 7. A sample network with a possible LFA loop (the shortest path is marked with arrows in the virtual layer): suppose that b is about to send packets to f. If, for some reason, link (b, f) goes down, b may choose to redirect its traffic to the LFA d^1. However, said traffic will never arrive to f as the $d^1 \to f$ detour degrades into the LFA loop $b - d^1 - b^1 - c - b$. Here, b^1 also switches to its LFA c realizing that its link to f has disappeared due the physical failure which node d^1 is unaware of.

4.2.1 Definition and Attributes

First of all, we augment the LFA definition to the case of virtual layers.

Definition 6. *For source s, destination d, and $s \to d$ next-hop e, node n is an SRG-disjoint link-protecting $s \to d$* LFA *if*

i) $n \in N_V(s)$ *and* $n \neq e$, *and*
ii) $\mathrm{dist}(n, d) < \mathrm{dist}(n, s) + \mathrm{dist}(s, d)$, *and*
iii) $(s, n) \notin S_{s,e}$ *(local SRG condition), and*
iv) $\mathrm{dist}(n, d) < \mathrm{dist}(n, s^i) + \mathrm{dist}(s^i, d)$ *for* $i = 1, \ldots, k_s$.

We show an example in Fig. 7 for this problem where, after a failure, the traffic ends up in an LFA loop. Therefore, to completely rule out this phenomena we require *iv)* by prohibiting "cascade LFAs" in the virtual layer.

Proposition 1. *In our model, a packet can be deflected to an LFA only at most once during its journey from the source to the destination.*

With these notations in place, we can now pose the *Resilient IP Overlay Design* (RIOD) problem. Here, the task is to compute the overlay that maximizes LFA-coverage, using only a given number of virtual routers.

Definition 7. RIOD$(G_S, c, U, k, \eta_{\min})$*: given a graph $G_S = (V_S, E_S)$, link costs c, node set $U \subseteq V_S$, and positive integer k, design a graph $G_V = (V_V, E_V)$ and link costs c_V so that:*

- *$V_S \subseteq V_V$ and virtual nodes provisioned only inside U,*

- $E_S \subseteq E_V$ *and virtual links are only between physically connected routers* (see Eq. (**??**)),
- *shortest paths between node pairs in* V_S *do not change* (the substrate is unaltered),
- $|V_V \setminus V_S| \leq k$ (no more than k virtual instances), *and*
- $\eta(G_V, c_V) \geq \eta_{\min}$ (the LFA coverage is a least $\eta_{\min}$).

4.2.2 The Solvability and Complexity of RIOD

First, we show that full LFA coverage can always be achieved by RIOD:

Thesis 2.1. *[J2, C1] For a given 2-connected graph* G_S *with positive costs* c *there always exists an overlay* G_V *and cost setting* c_V *that solves* $\mathrm{RIOD}(G_S, c, V_S, \infty)$ *with* $\eta(G_V, c_V) = 1$.

On the other hand we find the complexity intractable:

Thesis 2.2. *[J2, C1] The* $RIOD(G_S, c, U, k, \eta_{min})$ *problem is NP-complete.*

Unfortunately, today's large size of IP backbones can easily trigger unsolvable ILP instances. Correspondingly, we go on to design a heuristic algorithm that, depending on a setting of a simple configuration parameter, is *either optimal or it is guaranteed to terminate in polynomial time*, and it is possible to efficiently balance between the two according to the preferences of the operator.

4.2.3 Heuristic Algorithms to the RIOD Problem

The main idea in our heuristics is to iteratively add new virtual nodes to the network until full LFA coverage is achieved. Unfortunately, the case when a single node is added in each step may stuck in a local maximum in certain cases.

Theorem 1. *There exist cases where LFA coverage cannot be increased by adding only a single virtual node.*

Thesis 2.3. *[J2, C1] I have defined an algorithmic framework that, for any given number of virtual routers* k*, makes it possible to solve* $\mathrm{RIOD}(G, c, U, k)$ *with trading away the running time of the algorithm for the LFA-coverage attained by adding at most* k *virtual routers to the network. I have shown a construction that terminates at most* $O(n^5)$ *steps.*

We propose to add increasing number of virtual nodes in each step that we call *connected* l*-sets.*

Definition 8. *For a graph G_S, call a set of nodes in $U_l \in V_S$ a* connected l-set *if the induced subgraph of G_S spanned by U_l is connected and $|U_l| = l$. We denote the set of virtual nodes on top of U_l with U'_l.*

Then, we define the GLFAVirt(G_S, c_S, U_l, j) problem where the task is to set proper link cost settings on the virtual links so that LFA-coverage is maximized.

Definition 9. GLFAVirt(G_S, c_S, U_l, j)*: Given a substrate $G_S(V_S, E_S)$ with link costs c_S, a positive integer j, and a connected l-set $U_l \subseteq V_S$ for any $l \geq 1$ integer, inducing the virtual topology $G_V(V_V, E_V)$ with a virtual nodes $V_V = V_S \cup U'_l$, $E_V \subseteq E_S \cup \{(v, u) : v \in U'_l, u \in N_S(v)\}$, is there a cost setting c_V on E_V so that (i) the link costs and shortest paths in G_S do not change and (ii) # protected (s, d) pairs $\geq j$, $s, d \in V_S \times V_S$?*

Our heuristic is then based on simply trying increasingly larger connected sets of virtual nodes until LFA coverage eventually improves.

Algorithm 6 Greedy alg. for RIOD$(G_S, c, U, \infty, \eta_{min})$

```
while η_min > η(G_S, c) do
    for each l = 1, ..., k do
        for each connected l-set U_l ⊆ U do
            (c_{U_l}, η_{U_l}) ← solve GLFAVirt(G_S, c, U_l)
        end for
        (U'_l, η') ← choose U_l ∈ U that maximizes η_{U_l}
        if η' > η then add U'_l to G_V and set costs to c_{U'_l}
            break
        end if
    end for
end while
```

In order to reduce the running time of Alg. 6 we introduce the concept of *shortest path slices*.

Definition 10. *For a graph G_S, call a set of nodes in $U_l \subseteq V_S$ a* shortest path slice of rank l *if the induced subgraph of G_S spanned by U_l is created by shortest paths of G_S and $|U_l| = l$.*

Besides, we speed up the LFA coverage calculation by selecting the subset of nodes that can gain LFA by installing a specific virtual node set, U_l, in the network. To do so, we keep track the set of *eligible node-pairs* $\mathcal{L}$ that can gain an LFA. Clearly, a virtual router u' can provide LFA only if it is a neighbour of the source node. Let $\mathcal{L}_{U_l} \subseteq \mathcal{L}$ denote the set of eligible node-pairs with source node adjacent with U_l, formally $\mathcal{L}_{U_l} \subseteq \mathcal{L} | (s, d) \in \mathcal{L}, s \in$

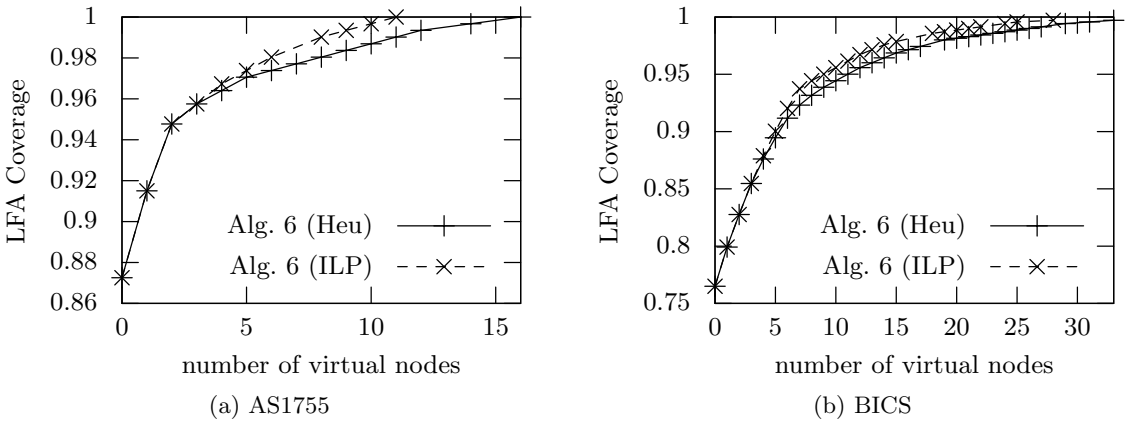


Figure 8. Progression of LFA coverage in small and middle-sized networks.

neigh(U_l). In other words, the new virtual nodes U_l can provide LFA to node pairs $\mathcal{L}_{U_l}$; thus $\frac{|\mathcal{L}_{U_l}|}{n(n-1)}$ is the upper bound in the increase of $\eta(G)$ after adding U_l. This measure helps in selecting a proper virtual nodes U_l to add.

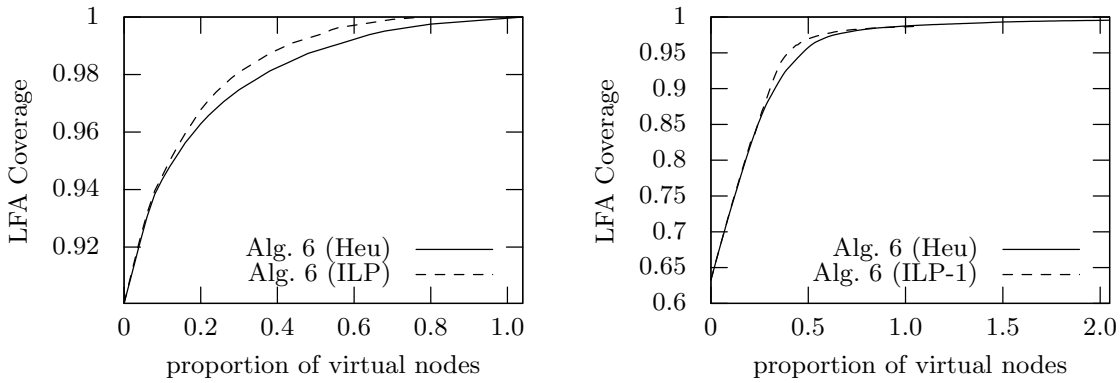
Accordingly, the theoretical worst-case complexity of Alg. 6 is $O(n^5)$ steps, however, in practice we found the all-pairs-shortest path problem to dominate running time and hence $O(n^2)$ to be a more reasonable complexity characterization.

4.2.4 Numerical Evaluations

Thesis 2.4. *[J2, C1] I have presented empirical evaluation to show that the proposed ILP adds only* 30*% of the physical nodes to reach the* 95*% level of LFA protection. At the same time, to reach perfect protection, provisioning* 70*% virtual nodes are enough. On the other hand, when the goal is to minimize running time, I have found that the heuristic reaches the same* 95*% coverage by adding* 36*% of the physical nodes. Similarly, the heuristic requires* 85*% of the physical nodes to gain perfect protection that is* 15*% approximation error. I have shown that there is only 30% difference in the length of recovery vs. default shortest paths.*

We have provided an Integer Linear Program (ILP) [Dissertation, Section 3.3.4] that serves as the baseline when evaluating the heuristic. Our major interest is to check whether the heuristic performs close to the ILP, measured by the LFA coverage metric, η, as an increasing number of virtual routers is added to the network. Obviously, we do not expect it to outperform the optimal solution, but we hope that the performance is not prohibitively worse and the improved running time makes up for the penalty.

As an average the heuristic shows some 19% overhead compared to the the ILP when adding $|V_S|$ virtual nodes, however it executes 10 times faster. The reason is that the



(a) Germany_50 (b) Deltacom

Figure 9. Progression of LFA coverage in backbone topologies.

LFA coverage improvement has a logarithmic trend, so if the goal is to solely improve LFA protection to a certain level then a couple of new nodes are usually enough. In contrast, to achieve full protection, we need to provide alternate tunnels from all sources to all destinations that can significantly increase the size of the virtual layer.

The logarithmic progression of the algorithms is clearly visible on Figs. 8-9. In the first phase there is a steep increase in the LFA protection and the performance gap between the algorithms is minimal. We also observe that in most steps the algorithms prefer to add a single virtual node, however there are cases (see e.g., Fig. 8b) when both methods need a tunnel to overcome a certain complex scenario. We also show a topology (Deltacom) where the ILP with $k = 3$ does not perform in acceptable running time; this validates the need for efficient heuristics. In this special case we relaxed $k = 1$ for the ILP and kept $k = 3$ for the heuristic that revealed a 2-3% gain on the ILP side in the first phase, but it got stuck after all, while the heuristic was still able to improve the coverage, see Fig. 9b. For further results, see Section 3.3.5 of the Dissertation.

4.3 R3D3: A Doubly Opportunistic Data Structure

In parallel to make networks more robust against failures, we also undertook the task of enhancing the efficiency of networking devices. To achieve this goal, we propose a novel succinct data structure that allows fast operations right on the compressed form.

Thesis Group 3. *I have proposed a novel succinct data structure that combines the storage scheme of RRR and the Elias-Fano encoding scheme. I have verified with analytical methods that the structure attains entropy-constrained size both on the data and the index, and I have*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0

Figure 10. A sample bitvector.

given the complexity of `access`, `rank` *and* `select` *queries. My numerical evaluation suggests that the data structure enables at most* 25% *faster operations at the price of a slight increase (up to* 15%*) in the storage size.*

4.3.1 Definitions

Suppose that there is a set of possible events with the probabilities of occurrence $p_1, p_2, \ldots, p_n$. We only know the probabilities, but nothing about the upcoming event at a certain point of time. Now the question is how uncertain we are about the outcome, and how could we measure it? The answer is called *entropy* (H):

$$H = -\sum_{i=1}^{n} p_i \log(p_i) \tag{2}$$

Note that in case of two possible events (e.g. a bit is set or zero), Eq. 4.3.1 becomes:

$$H_0 = p \log(\frac{1}{p}) + (1-p) \log(\frac{1}{1-p}) \tag{3}$$

We call a data structure *succinct*, if it can be stored on the optimal number of bits plus some little "extra space": $opt + o(opt)$. A succinct encoding of a bitmap t of length n is worst-case minimum $n + o(n)$ bits, and it also implements `rank` and `select` queries in $O(1)$ time. But why do these operations matter?

Jacobson [27] showed a new encoding scheme for compressed trees and graphs that not only consumes minimal space but also made it possible to do the traversal in constant time. His idea was to encode the tree into a bitvector where moving downwards (e.g. left or right child) requires a `rank` operation, while moving upwards (parent) goes with `select`. We define those operations on a bitvector t as follows:

- $\texttt{rank}_q(t, i)$: return the number of occurrences of symbol q in $t[1, i]$;
- $\texttt{select}_q(t, i)$: return the position of the i-th occurrence of symbol q in t.

On Fig. 10, $\texttt{rank}_1(t, 8) = 2$ gives the number of bits set to 1 up to and counting the 8-th position, and $\texttt{select}_1(t, 2) = 7$ indicates that the second set bit occurs at position 7.

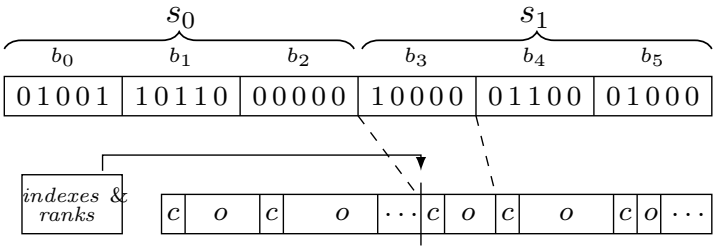


Figure 11. Sketch of the *RRR* encoding scheme.

RRR

The authors of [28], Raman, Raman and Rao, by building on the results of Brodnik and Munro [29], and Pagh [30], developed a *fully indexable dictionary (FID)* that combines the speed of hashing with the versatility of sorted arrays. It is a *compressed bitvector* representation that stores a bitmap t of length n on nH_0 bits, plus the index on $O(\frac{n \log\log n}{\log n}) = o(n)$ bits and implements `access`, `rank` and `select` queries in constant time.

The structure partitions t into blocks $b_1, b_2, \ldots$ of size $b = \frac{\log n}{2}$ bits (see Fig. 11 for an illustration). Each block b_i is encoded with a pair (c_i, o_i), where $c_i = \texttt{popcount}(b_i)$ is the *class* of b_i and o_i is the offset that is used as an index to a prefab table.

Claude and Navarro proposed an optimized RRR in [31] and [32], which attains significant space reduction by substituting RRR's universal block coding tables with *on-the-fly block decoding using combinatorial unranking* [33], and removing the relative block pointers /rank counters from the index and resorting to linear search inside superblocks.

The Elias–Fano coding scheme

Definition 11. *For a given bitmap t of length n, we call the number of set bits in t, i.e. $\texttt{rank}_1(n)$, the population count and we denote it with $\texttt{popcount}(t)$.*

The *Elias–Fano coding scheme* stores a bitvector t in $nH_0 + O(n)$ space and answers $\texttt{select}_1$ queries in $O(1)$ time, with no support for `rank` and `access`. An alternative scheme *EF* is available at [28, 34, 35] that attains $nH_0 + O(m)$ bits of space and needs $O(m)$ time for `access`, `rank`, and `select`, where $m = \texttt{popcount}(t)$.

The EF scheme encodes the characteristic vector $\{x_1, x_2, \ldots, x_m\}$ of t, where $m = \texttt{popcount}(t)$ and $x_i = \texttt{select}_1(t, i) : i \in \{1, \ldots, m\}$, instead of t itself, using a technique called MSB bucketing: group x_is according to the most significant $\log m$ bits into buckets, store the $l = \log n - \log m = \log \frac{n}{m}$ lower-order bits for each x_i verbatim in an array (called the Lower-bits Array, *LBA*), and store the significant bits as a sequence of unary encoded

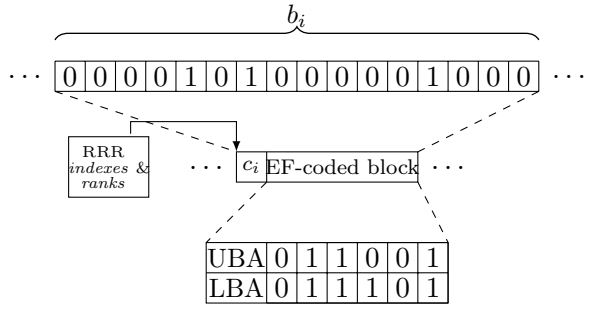


Figure 12. R3D3 encoding, with a single 16-bit block and the corresponding EF block-code. The superblock pointers and block classes are encoded in the RRR index, while the blocks are encoded with EF.

gaps in another array (the Upper-bits Array, *UBA*). The UBA is constructed as follows: for each bucket write down as many 1s as there are x_is in the bucket followed by a 0.

4.4 R3D3

R3D3 combines the storage scheme of RRR [32] with the Elias–Fano encoding scheme to create a very efficient bitmap encoding. The idea is to rely on RRR's indexing scheme, as it gives very fast access to block-codes and block-ranks, whereas the block-coding component will be changed from combinatorial unranking to the EF scheme.

R3D3 adopts a scheme we call *duplicate indexing*; it first invokes the RRR indexes to find the starting position for each block and then looks up the UBA to index the relevant entries in the LBA, and finally only a few LB entries need to be directly decoded.

Thesis 3.1. *[J1] I have shown that R3D3 encodes a t bitvector of length n of arbitrarily chosen block size, b, in*

$$nH_0 + np + \frac{n}{b}(2 + \log b) \quad \textit{bits.} \tag{4}$$

.

The query execution times for R3D3 are as follows: the time complexity for $\texttt{access}(t, i)$ is dominated by the linear search to locate the beginning of the EF-coded block containing position i and identifying the class, which, similarly to RRR, is constrained by $O(\log n)$, to which block-decoding gives another $O(pb)$ steps. To reach parity with RRR, we can choose much larger block sizes (recall, in RRR the block size is $b = O(\log n)$, while in R3D3 $pb = O(\log n)$ and $p < 1$), which brings substantial space reduction as we need to store fewer block class values in the index. The same holds for $\texttt{rank}(t, i)$. As per $\texttt{select}(t, i)$, first we binary-search superblock and block ranks in $O(\log n)$ time and then decode the block, again

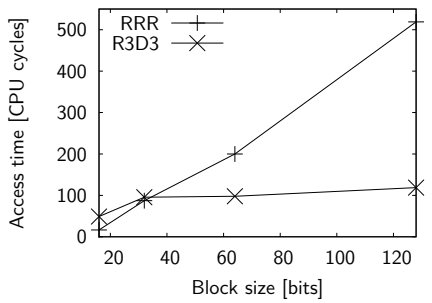


Figure 13. Average time to access a random position in RRR and EF block-codes as the function of the block size, on random bitmaps, $p = 0.1$.

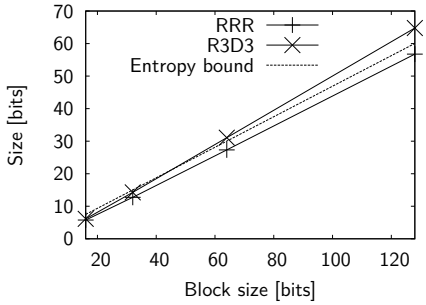


Figure 14. Average size of RRR and EF block-codes and the zero-order entropy limit as the function of the block size, on random bitmaps, $p = 0.1$.

in expected $O(pb)$ time. The total time for these queries is $O(\log n) + O(pb) = O(\log n)$ if $pb = O(\log n)$.

Thesis 3.2. *[J1] I have given a block size setting for R3D3, $pb = O(\log n)$, that achieves compression on the RRR indexing scheme by encoding t in*

$$nH_0 + nH_0\left(\frac{1}{2} + O\left(\frac{\log\log n}{\log n}\right)\right) \tag{5}$$

bits and supports `access`*,* `rank`*, and* `select` *operations in expected $O(\log n)$ time.*

4.5 Experimental Results

Thesis 3.3. *[J1] I have performed extensive numerical evaluations on a wide range of synthetic and real data to compare the performance characteristics of R3D3 to the best-known succinct compression schemes. I have found significant performance gap in the block decoding components of RRR and R3D3 in favor of R3D3. Measurements on synthetic data revealed $2-3\%$ space reduction compared to RRR that brings no significant gain for* `rank` *but improves 3 times the speed of* `select`*. Complex data structures having higher densities makes R3D3 perform 25% better, but only at the price of a $10-15\%$ (up to 40%) increase on the storage size.*

Block-coding. The goal of our first experiment is to validate our choice for EF instead of RRR's combinatorial ranking/unranking scheme to encode blocks. Recall, this choice was made because EF supports all basic block-operations in $O(\texttt{popcount}(b))$ time as opposed to $O(b)$ for RRR, where b is the block size, at the cost of slightly bigger block-codes (Fig. 14). We observe that EF block-coding is indeed much less sensitive to the block size; while R3D3

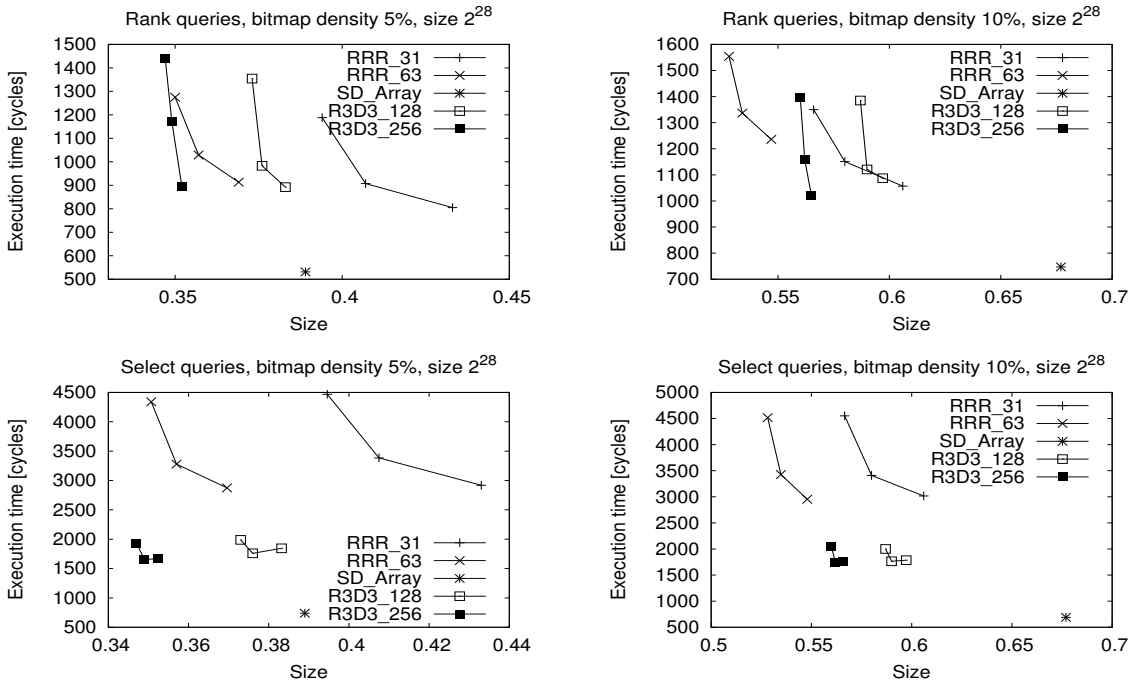


Figure 15. Average space–time efficiency of RRR and R3D3 on random rank and random select queries, on random 2^{28} bit long bitmaps.

needs only 3 times as much time to access a 128-bit block as for a 16-bit block (Fig. 13), this factor is 25-fold with RRR.

Random synthetic bitmaps. For this experiment we again use random bitmaps as input, but now we evaluate RRR and R3D3 in their entirety (not just the block-coding component), the same way as in [32]. In particular, we measure the average execution time of `rank` and `select` queries on the compressed data structures against the total compressed encoding size. Our observation is that at low density, R3D3_256 needs roughly 2-3% less space than RRR and attains similar performance in rank queries and up to 3 times better performance on select queries. In addition, R3D3 exhibits 6-8% better compression compared to the SD_Array. For a higher 10% density R3D3 achieves around 15-20% performance edge over RRR in rank queries and again multiple-times better execution times on select.

Internet forwarding tables. We carried out measurements on forwarding tables, a relatively new application of complex data structures [36]. The results for 3 real forwarding table instances, downloaded from operational Internet routers, are given in Table 1. Here, entropy is the tree entropy as of [36] and the compressed encodings were obtained by the XBW data structure, instantiated over RRR- and R3D3-coded bitmaps with different block

Table 1. RRR or R3D3 in forwarding table encoding: table sample name, number of prefixes, and entropy bound as of [36]; and compressed size and average execution time of random FIB lookups. Sizes are in Kbytes (KiB) and times in CPU cycles.

			RRR_63		R3D3_128		R3D3_256	
Name	#Prefixes	Entropy	Size	Lookup	Size	Lookup	Size	Lookup
`hbone-szeged`	453,685	70.1	63.1	21340	86.9	12670	93	11800
`access_d`	403,245	149.1	83.5	18670	115.4	11000	124	10900
`access_v`	2,970	1.08	4.6	20960	6.2	14800	6.5	11540
`mobile`	4,391	1.32	3.4	15400	3.7	10960	3.8	11580
`hbone-vh1`	453,741	222.6	160.8	20340	222	13850	238	13690

sizes. Again, we see a modest (up to 40%) size increase with R3D3, at the cost of close to twice the lookup performance. Our experiments show that most benefits already manifest themselves at the block size of 128 and 256 bits with R3D3.

5 Applicapility of New Results

In the Dissertation we introduced theoretical and experimental results to the fields of network reliability and data compression. In the first part of the Dissertation, we have pointed out that the only commercially available IPFRR scheme, LFA, suffers from the drawback that the level of protection inherently depends on the layout of the given topology. To overcome this limitation, we presented two strategies for network augmentation (Chapter 2-3).

First, our results on the graph extension problem contribute to an internal tool developed by *Ericsson Research* that is capable to analyze and optimize LFA characteristics in operator networks. Besides, our findings became integrated into a few scientific reports published by IEEE [37] and FIA [38]. For details see Section 2.4 of the Dissertation. Second, we showed in Chapter 3 how to achieve perfect LFA coverage without touching the physical topology. Based on our results, Tapolcai showed a construction in [39] that achieves full LFA coverage by adding limited number of virtual nodes. In addition, our results on router virtualization also appear in [37] in the context of *Virtual Routing Overlays* that is concluded to finally solve multiple problems of inherent failure isolation.

Last but not least, we turned our attention to compressed data structures and we introduced R3D3 for supporting time-sensitive data processing. We gave an overview in Section 4.3 on the possible fields of applications such as *text-indexing*, *genome compression*, *data mining*, *databases* and *FIB compression*. We believe that our results can directly be used in communicating networks with the purpose of making them more reliable and efficient.

Publications

Journal Papers

[J1] **M. Nagy**, J. Tapolcai and G. Rétvári. "R3D3: A Doubly Opportunistic Data Structure for Compressing and Indexing Massive Data". *Infocommunications Journal*, pp. 58-66., 2019. $(4/2 = 2)$

[J2] **M. Nagy**, J. Tapolcai and G. Rétvári. "Node Virtualization for IP Level Resilience". *IEEE/ACM Transaction on Networking*, 2018. $(6/2 = 3)$

[J3] **M. Nagy**, J. Tapolcai and G. Rétvári. "Optimization Methods for Improving IP-level Fast Protection for Local Shared Risk Groups with Loop-Free Alternates". *Telecommunication Systems 56*, pp. 103-119., 2014. $(6/2 = 3)$

[J4] L. Csikor, **M. Nagy** and G. Rétvári. "Network Optimization Techniques for Improving Fast IP-level Resilience with Loop-Free Alternates". *Infocommunications Journal*, pp. 2-10., 2012. $(4/2 = 2)$

Conference Papers

[C1] **M. Nagy**, J. Tapolcai and G. Rétvári. "On the Design of Resilient IP Overlays". In *Proc., DRCN 2014*, Ghent, Belgium, 1-3. April 2014. $(3/2 = 1.5)$

[C2] **M. Nagy** and G. Rétvári. "IP hálózatok védelmének optimalitása többszörös hibák esetére". In *Proc., Mesterpróba*, Budapest, Hungary, May 2012. $(1/1 = 1)$

[C3] **M. Nagy** and G. Rétvári. "An evaluation of approximate network optimization methods for improving IP-level fast protection with Loop-Free Alternates". In *Proc., RNDM 2011*, Budapest, Hungary, September 2011. $(3/1 = 3)$

Other Publications

[C4] M. Szalay and **M. Nagy** and D. Géhberger and Z. Kiss and P. Mátray and F. Németh and G. Pongrácz and G. Rétvári and L. Toka. "Industrial-scale Stateless Network Functions". In *Proc., IEEE Cloud*, Milan, Italy, 2019. ($3/8 = 0.37$)

Patents

[P1] **M. Nagy** and D. Fiedler and D. Géhberger and P. Mátray and G. Németh and B. Pinczel "Fast session restoration for latency sensitive middleboxes". International Application No. PCT/IB2019/060031, TELEFONAKTIEBOLAGET LM ERICSSON, 2018. ($2/6 = 0.33$)

[P2] **M. Nagy** and D. Fiedler and D. Géhberger and P. Mátray and G. Németh and B. Pinczel and A. Császár "N+1 redundancy for virtualized services with low latency failover". International Application No. PCT/IB2019/060037 , TELEFONAKTIEBOLAGET LM ERICSSON, 2018. ($2/7 = 0.28$)

Total publication score: 16.48

Bibliography

[1] L. Humphreys, T. von Pape, and V. Karnowski, "Evolving Mobile Media: Uses and Conceptualizations of the Mobile Internet," *Journal of Computer-Mediated Communication*, 2013.

[2] J. Moy, "Ospf version 2," RFC 2328, Apr 1998.

[3] D. Oran, "Osi is-is intra-domain routing protocol," RFC 1142, Febr 1990.

[4] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an ip backbone," in *ACM SIGCOMM Internet Measurement Workshop*, 2002, pp. 237–242.

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

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

[7] Cisco Systems, "Cisco IOS XR Routing Configuration Guide for the Cisco CRS Router, Release 4.2," 2012.

[8] Hewlett-Packard, "HP 6600 Router Series: QuickSpecs," 2008, available online: http://h18000.www1.hp.com/products/quickspecs/13811_na/13811_na.PDF.

[9] Juniper Networks, "JUNOS 12.3 Routing protocols configuration guide," 2012.

[10] M. Nagy, "Github homepage," https://nmate.github.io, 2019.

[11] "LEMON – Library for Efficient Modeling and Optimization in Networks," http://lemon.cs.elte.hu/, 2014.

[12] "GUROBI – Linear Programming Solver," http://www.gurobi.com, 2018.

[13] "BOOST – C++ Libraries," http://www.boost.org, 2018.

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

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

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

[17] P. Ferragina, G. Navarro, "Pizza & Chili Corpus," http://pizzachili.dcc.uchile.cl/.

[18] UCSC Genome Bioinformatics, "The UCSC Genome Browser," http://hgdownload.cse.ucsc.edu/downloads.html.

[19] I. Witten, T. Bell, and J. Cleary, "The Calgary Corpus," 1987, http://corpus.canterbury.ac.nz/descriptions/#calgary.

[20] G. Rétvári and A. Kör osi and J. Tapolcai, "The Internet Routing Entry Monitor," http://lendulet.tmit.bme.hu/fib_comp/.

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

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

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

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

[25] 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*. Springer Verlag, 2005, p. 243.

[26] Cisco, "Cisco catalyst 4500 series switch software configuration guide, 15.0," 2016.

[27] G. Jacobson, "Space-efficient static trees and graphs," in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, ser. FOCS '89, 1989, pp. 549–554.

[28] V. R. R. Raman and S. R. Satti, "Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets." *ACM Transactions on Algorithms*, vol. 3(4), 2007.

[29] A. Brodnik and I. Munro, "Membership in constant time and almost-minimum space," *Journal on Computing*, vol. 28, no. 5, pp. 1627–1640, 1999.

[30] R. Pagh, "Low redundancy in static dictionaries with constant query time," *Journal on Computing*, vol. 31, no. 2, pp. 353–363, 2001.

[31] F. Claude and G. Navarro, "Practical rank/select queries over arbitrary sequences," in *Proc. 15th International Symposium on String Processing and Information Retrieval (SPIRE)*, ser. LNCS 5280. Springer, 2008, pp. 176–187.

[32] G. Navarro and E. Providel, *Fast, Small, Simple Rank/Select on Bitmaps.* Springer Berlin Heidelberg, 2012, pp. 295–306.

[33] D. E. Knuth, *The Art of Computer Programming: Combinatorial Algorithms*, ser. Series in Computer Science. Addison-Wesley, 2011.

[34] D. Okanohara and K. Sadakane, "Practical entropy-compressed rank/select dictionary," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, 2007, pp. 60–70.

[35] S. Gog, "Compact and succinct data structures: From theory to practice," available online: http://es.csiro.au/ir-and-friends/20131111/anu_gog_seminar.pdf, 2015.

[36] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger, "Compressing IP forwarding tables: towards entropy bounds and beyond," in *ACM SIGCOMM*, 2013, paper http://lendulet.tmit.bme.hu/~retvari/publications/sigcomm_2013_tech_rep.pdf, pp. 111–122.

[37] M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, and S. Schmid, "Fast recovery mechanisms in the data plane," in *IEEE Communications Surveys and Tutorials*, 2020.

[38] L. Csikor, G. Rétvári, and J. Tapolcai, "High availability in the Future Internet," in *The Future Internet*, ser. Lecture Notes in Computer Science, A. Galis and A. Gavras, Eds. Springer Berlin Heidelberg, 2013, vol. 7858, pp. 64–76.

[39] M. Nagy, J. Tapolcai, and G. Rétvári, "Node virtualization for IP level resilience," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2018.