Design of Robust Programmable Networks with Bandwidth-optimal Failure Recovery Scheme

Andrea Tomassilli, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Pérennes, Thierry Turletti, Ruslan Sadykov, François Vanderbeck, Chidung Lac

Design of Robust Programmable Networks with Bandwidth-optimal Failure Recovery Scheme[1]

A. Tomassilli[a], G. Di Lena[a,c], F. Giroire[a], I. Tahiri[b], D. Saucez[a], S. Perennes[a], T. Turletti[a], R. Sadykov[b], F. Vanderbeck[b], C. Lac[c]

[a] *Université Côte d'Azur, CNRS, Inria Sophia Antipolis, I3S, UMR 7271, 06900 Sophia Antipolis, France*
[b] *Université de Bordeaux, Institut de Mathématiques, France*
[c] *Orange Labs, France*

Abstract

More than ever, data networks have demonstrated their central role in the world economy, but also in the well-being of humanity that needs fast and reliable networks. In parallel, with the emergence of Network Function Virtualization (NFV) and Software Defined Networking (SDN), efficient network algorithms considered too hard to be put in practice in the past now have a second chance to be considered again. In this context, as new networks will be deployed and current ones get significant upgrades, it is thus time to rethink the *network dimensioning problem* with protection against failures. In this paper, we consider a path-based protection scheme with the global rerouting strategy in which, for each failure situation, there may be a new routing of all the demands. Our optimization task is to minimize the needed amount of bandwidth. After discussing the hardness of the problem, we develop two scalable mathematical models that we handle using both Column Generation and Benders Decomposition techniques. Through extensive simulations on real-world IP network topologies and on randomly generated instances, we show the effectiveness of our methods: they lead to savings of 40 to 48% of the bandwidth to be installed in a network to protect against failures compared to traditional schemes. Finally, our implementation in OpenDaylight demonstrates the feasibility of the approach. Its evaluation with Mininet shows that our solution provides sub-second recovery times, but the way it is implemented may greatly impact the amount of signaling traffic exchanged. In our evaluations, the recovery phase requires only a few tens of milliseconds for the fastest implementation, compared to a few hundreds of milliseconds for the slowest one.

List of Acronyms

Notation	Description	Notation	Description
APX	APproXimable complexity class	NP	Nondeterministic Polynomial Time complexity class
DP	Dedicated Path	RMP	Restricted Master Problem
GR	Global Rerouting	SDN	Software Defined Networking
ILP	Integer Linear Program	SFP	Service Function Path
IP	Internet Protocol	SRLG	Shared Risk Link Group
LP	Linear Program		
MILP	Mixed Integer Linear Program		
NFV	Network Function Virtualization		
NFVI	Network Function Virtualization Infrastructure		

1. Introduction

We address in this paper the challenge of designing an SDN/NFV-enabled network that provides SRLG-failure survivability under the global rerouting protection scheme. Network design [1], and in particular the *design of survivable networks*, is one of the fundamental problems of the networking field [2, 3]. It consists in deciding how much, and where, link capacity should be installed in a network in order to minimize the installation cost, while satisfying diverse communication constraints such as available bandwidth, delay, protection against failures, etc. Such a decision has to be taken when a new network is installed, but also when a network has to be upgraded to sustain the continuous increase of traffic demand [4].

Network failures such as cable cuts, natural disasters, faulty interfaces, or human errors are the daily routines of operations teams [5]. Faults in the IP and optical layer tend to be correlated between them [6]. Indeed, the failure of a component located in a common router, such as a linecard, or in the underlying optical infrastructure, such as a common fiber, may result in the consequential failure of multiple entities at the IP layer. The concept of *Shared Risk Link Groups* (SRLGs) was proposed to model this correlation [7]. SRLGs allow to easily express a risk relationship, and can also represent different types of failures, such as single or multiple node and link failures. Moreover, SRLGs can be grouped together in new ones to model any type of simultaneous failures. Because they simplify reasoning on complex failure scenarios, SRLGs are particularly useful to design failure protection techniques [8], among which the *unrestricted flow reconfiguration* technique, also known as *global rerouting* [9], which is of particular interest as it offers a *bandwidth-optimal* protection. This optimality is achievable as a new set of backup paths is defined - one for each

demand - in each of the possible failure situations. Unfortunately, the optimality comes at the risk that a failure may result in a completely different routing for the demands, making it hardly usable in traditional networks as it would result in high signaling overhead.
However, with the generalisation of the *Software Defined Networking* (SDN) approach [10] where network control and packet forwarding are decoupled, this may not hold true anymore and it is then time to reconsider global rerouting and how to deploy it in modern networks. In SDN, network intelligence is centralized in the controller that maintains a global view of the network. Routing decisions are taken in a single location, *the controller*, with a complete knowledge of the network state instead of resulting from a distributed algorithm. As a result, with SDN, the global rerouting protection scheme may be put in practice since control message exchanges can be minimized [11, 12], and as we demonstrate in Section 5.

Our goal is to compute for each demand *a primary and a backup path for each SRLG failure scenario*, while ensuring that the required network functions will be performed on the packets in the order specified by its Service Function Chain (SFC). The studied subject is a *dimensioning problem* for a network that needs to determine the minimal amount of resources (e.g., link capacity) it needs to deploy, while guaranteeing protection against SRLG failures. Even though, at first glance, the problem may appear easy due to the absence of link capacity constraints, we demonstrate that it is not the case. We actually show that, *even for a single demand*, the problem is NP-Hard and inapproximable within $(1-\epsilon)\ln(|R|)$ for any $\epsilon > 0$ unless P=NP, where $|R|$ denotes the number of SRLG failure scenarios.

Our contributions can be summarized as follows.

- To the best of our knowledge, we are the first to provide two scalable exact methods using Column Generation and Benders Decomposition techniques to solve the problem of global rerouting in SDN/NFV-enabled networks with SRLG constraints. Indeed, we are able to solve much larger instances than a classic compact ILP formulation.
- We also propose a fast 2-phase polynomial method. The first phase consists in solving the fractional relaxation of the problem. The second one is the building of an integral solution from the fractional one. It leads to an optimization problem we named Min Overflow Problem. We show that the problem is NP-complete, but that there exists a $(1+\frac{1}{e}+\varepsilon)$–approximation algorithm to solve it.
- We demonstrate the applicability of our proposed protection method with real implementations emulated in Mininet, and study metrics such as the burden on the network elements and time to recovery from a failure. We also discuss the technical choices to be taken into account by the network operator in order to put in practice our proposed technique.

The rest of this paper is organized as follows. In Section 2, we discuss related work. Section 3 develops the proposed optimization approaches. Specifically, in Section 3.1, we formally define the problem to be studied, as well as notations that are used in this paper. In Section 4, we validate our models by means of numerical analysis on real world and randomly generated network topologies and workloads. In Section 5, we demonstrate the feasibility of our proposal by implementing a prototype with OpenDaylight and evaluating it with Mininet. Finally, we draw our conclusions in Section 6.

2. Related Work

The problem of providing network protection against failures has been widely investigated in the last decades, see, e.g., [13]. With the advent of SDN/NFV, there are more opportunities to create, deploy, and manage networks more efficiently. Indeed, with SDN and its control–data planes decoupling, routing decisions can be done using a logically centralized approach. This paves the way for a broadening of perspective in terms of fault management [14].

A large number of works have explored how SDN allows an efficient traffic engineering of networks implementing NFV, see for example [15, 16]. In particular, Bastam et al. propose linear programming models and decomposition techniques to divide the problem model into smaller subproblems, implementing a scalable management for SDN-based data center networks [15]. Tomassilli et al. introduce scalable optimisation solutions to compute backup paths in SDN/NFV backbone networks. In this work, we consider the complementary survivable design problem, i.e., determining the amount of capacity needed when building or upgrading a network in order for it to be fault-tolerant.

Chu et al. [17] consider a hybrid SDN network and propose a method to design the network in such a way that fast failure recovery from any single link failure is achieved. Their proposal consists in redirecting the traffic on the failed link from the routers to SDN switches through pre-configured IP tunnels. Next hops are pre-configured before the failures take place, and the set of candidate recovery paths for different affected destinations is chosen by the SDN controller in such a way that the maximal link utilization after redirecting the recovery traffic through these paths is minimized.

Suchara et al. [18] propose a joint architecture for both failure recovery and traffic engineering. Their architecture uses multiple pre-configured paths between each pair of edge routers. In the event of a failure, the failover is made on the least congested path that ensures connectivity. Besides, Sgambelluri et al. [19] propose a controller–based fault recovery solution that uses OpenFlow's Fast Failover Group Tables to quickly select a pre-configured backup path in case of link failure.

Different from previous studies on failure recovery, we present a simple and bandwidth-optimal approach based on multiple backup paths to protect the network against SRLG failures where SDN switches are deployed. Our concept was previously introduced in [20, 21].

The idea of using a set of pre-configured multiple backup network configurations is not new. For instance, in [22, 23] the authors propose a pre-configured proactive IP recovery scheme that makes use of multiple routing backup configurations as a method for fast recovery. The main idea is to create a small set of backup routing configurations to be used in the case of a single link or node failure. Since the backup configurations are kept in the routers, it is necessary to reduce their number in order not to require the routers to store a significant amount of state information.

In [8, 24, 25], the authors use SRLG for modelling in order to enhance the preparedness of a given network to natural disasters, or regional failures. For example, a regional SRLG aims to characterize a failure damaging the network only in a bounded geographical area. The authors in [26] propose Mixed Integer Linear Program (MILP) for an elastic optical network protection with SRLG group and dedicated protection mechanism. Herein, we take to the extreme the idea of multiple routing configurations by allowing a completely different routing in response to an SRLG failure situation. Different from the above works, our aim is to provide a bandwidth-optimal mechanism to design a reliable network. Our work revisits optimization techniques and protection schemes introduced to design survivable optical networks in a large corpus of works of the 90s, see for example [2, 27, 3] for surveys or reference books. We generalize the results to layered graphs. Indeed, besides guaranteeing the recovery, our proposed approach also takes into consideration the SFC requirement of the flows. Moreover, we show using experiments that global rerouting, which at the time was studied just as a lower bound, now is a viable very efficient protection solution thanks to SDN control. Table 1 summarizes the most important works to be compared to our proposal. In particular, we check if the works consider optimal (bandwidth) dimensioning for protection against SRLGs, handle the VNFs of the network services, implement fast rerouting upon failures using SDN, propose scalable solutions through decomposition methods, or approximation algorithms.

3. Optimization Approaches

In this section, we first rigorously define the bandwidth-optimal failure recovery problem and prove the hardness and inapproximability results for the Global Rerouting problem (Section 3.1). We provide a compact ILP formulation (Section 3.3) which will be used as a baseline for evaluation. We then propose a scalable[2] decomposition model that relies on the Column Generation technique (Section 3.4). Both formulations are based on a layered network model described in Section 3.2. We next introduce a second scalable model using Benders decomposition approach (Section 3.5). The models led us to study the

[2]The term scalable has to be understood here in the context of exact optimization methods. The proposed optimization methods are said *scalable* in the sense that they allow to solve much larger instances than the classic compact ILP formulation solving the same problem.

Context and problem solved	[17]	[18]	[26]	[28]	[19]	[22]	[23]	[24]	[15]	Our proposal
Survivable network design	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔
Fast rerouting with SDN	✔	✘	✘	✔	✔	✘	✘	✘	✔	✔
Network services and NFV	✘	✘	✘	✔	✘	✘	✘	✘	✘	✔
SRLG	✘	✔	✔	✘	✘	✘	✘	✔	✘	✔
Theoretical methods and results										
Optimal bandwidth	✔	✔	✔	✘	✘	✘	✘	✘	✔	✔
Decomposition methods	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔
Approximation algorithms	✔	✔	✘	✔	✘	✔	✔	✔	✘	✔
Implementation results										
Implementation via emulation	✘	✘	✘	✘	✔	✘	✘	✘	✘	✔

✔=Yes, ✘=No

Table 1: Summary of related work. The table analyzes the context and the methods used in the different works.

complexity of a subproblem, the Min-Overflow Problem, and to propose approximation algorithms to solve it (Section 3.6).

3.1. Problem Statement and Notations

We model the network as an undirected graph $G = (V, E)$, where V represents the set of nodes and E the set of links. We are given a set of SRLG events $\mathcal{R}$ that can incur link failures. All the notion are summarised in Table 2

Each $r \in \mathcal{R}$ consists of a set of links that share a common physical resource. We denote by $\mathcal{D}$ the set of demands (e.g., traffic between two locations). As we are solving a dimensioning problem, we assume prior full knowledge of traffic demands, i.e., traffic matrices are known beforehand. A demand $d \in \mathcal{D}$ is modeled by a quadruple (s_d, t_d, bw_d, C_d) with s_d the source, t_d the destination, C_d the ordered sequence of network functions that need to be performed to all the packets belonging to the flow of the demand, and bw_d the required units of bandwidth. We denote by $\ell(d)$ the length of the SFC for a demand d.

Network functions need to be executed on the so-called Network Function Virtualization Infrastructure (NFVI) nodes. Not all the network nodes are enabled to run virtual functions. We denote by $V^{\text{NVFI}} \subseteq V$ the set of NFVI nodes. Moreover, we assume that an NFVI node can only run a subset of the network functions, as there may be constraints on their location in the network (e.g., geography or regulatory constraints and anti-affinity rules).

Given the network topology and the traffic rate of the demands to be supported, the purpose of the design problem is to precompute a set of paths to guarantee the recovery of all the demands in the event of an SRLG failure, while satisfying their SFC requirements. The considered optimization task is to *minimize the required bandwidth in the network*. We refer to this problem as the Global Rerouting problem.

Proposition 1. *The* Global Rerouting *problem is NP-hard even for a single demand, and cannot be approximated within*

Symbol	Description
G	Undirected graph
V	Set of nodes in the graph G
E	Set of links in the graph G
$r \in \mathcal{R}$	Set of links that share a common physical resource
$\mathcal{R}$	Set of SRLGs
$d \in \mathcal{D}$	Single demand composed by (s_d, t_d, bw_d, C_d)
$\mathcal{D}$	Set of demands
s_d	Source of the traffic
t_d	Destination of the traffic
bw_d	Required units of bandwidth G
C_d	Ordered sequence of network functions
$\ell(d)$	Length of the SFC for a demand d
$V^{\text{NVFI}} \subseteq V$	Set of NFVI nodes
$G^L(d)$	Layered graph for demand d
π	Service path
Π_d^r	Service function paths for a demand d
a_{uv}^{π}	Number of times link (u, v) is used in the service path π
$y_{\pi}^{d,r}$	Boolean value to check if demand d uses path π as a service path in the SRLG failure event r
x_{uv}	Bandwidth allocated on link (u, v)
$\varphi_{(ui,vj)}$	Boolean value to check if the flow is forwarded on link $((u, i), (v, j))$ of $G^L(d)$
α_{ω}^{sd}, β_{uv}^{r}	Dual values relative to constraints (8) and (9)
$\lambda_d(\mu)$	length of the shortest path for demand d, with respect to link metrics μ

Table 2: Notation table.

$(1 - \epsilon)\ln(|R|)$ *for any* $\epsilon > 0$ *unless P=NP, where* $|R|$ *denotes the number of failing scenarios.*

Proof. We use a reduction from the Hitting Set Problem, which is defined as follows. We are given a collection C of subsets of a finite set S and the problem consists in finding a hitting set for C, i.e., a subset $S' \subseteq S$ such that S' contains at least one element from each subset in C of minimum cardinality. Given an instance $\mathcal{I} = (S, C)$ of Hitting Set, we can build an instance $\mathcal{I}' = (G, \mathcal{D}, \mathcal{R})$ of Global Rerouting in the following way. $G = (V, E)$ is a multigraph with $V = \{s, t\}$ and $E = \{e_i,\ i = 1, ..., |S|\}$. All the edges have s and t as endpoints

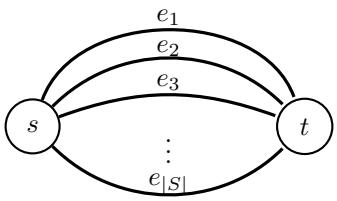


Figure 1: Multigraph resulting from the reduction from the HITTING SET PROBLEM to The GLOBAL REROUTING problem. $G = (V, E)$ is the multigraph with $V = \{s, t\}$ and $E = \{e_i,\ i = 1, ..., |S|\}$. All the edges have s and t as endpoints.

- see Fig. 1 and Example 1 below for an example. For each $C' \subseteq C$, we add a failing scenario $r_{C'} = E \setminus C'$ to R, corresponding to edges that cannot be used in the failure situation r. Finally, we add to $\mathcal{D}$, a demand d with s and t as source and destination respectively, and with charge equal to 1. The goal now consists in finding a path for each of the failure scenarios $r \in \mathcal{R}$ minimizing the needed capacity to deploy. The total capacity needed to satisfy d in each of the failure situations does not exceed $c \iff$ there exists a hitting set of cardinality not greater than c. The proposition follows immediately from the fact that HITTING SET is NP-Hard [29] and cannot be approximated within a factor of $\ln |S|$ [30], unless P=NP. □

Example 1 (Reduction)**.** Consider the following instance of the HITTING SET PROBLEM: $S = \{1, 2, 3, 4, 5\}$ and $C = \{C_1, C_2, C_3, C_4, C_5\}$ with $C_1 = \{1, 2, 3, 4\}$, $C_2 = \{1, 4, 5\}$, $C_3 = \{2, 5\}$, $C_4 = \{2, 3, 5\}$, and $C_5 = \{1, 4\}$. The multigraph resulting from the reduction of the instance is a multigraph with 5 edges (noted $1, 2, 3, 4$, and 5) linking two nodes s and t corresponding to the elements in S. The corresponding instance of the GLOBAL REROUTING problem has a single demand d with source s and destination t, an empty SFC, and requiring a single unit of bandwidth to be sent. It has 5 SRLG events $\mathcal{R} = \{r_1, r_2, r_3, r_4, r_5\}$, as many as the number of subsets in C, with $r_1 = E \setminus C_1 = \{5\}$, $r_2 = E \setminus C_2 = \{2, 3\}$, $r_3 = E \setminus C_3 = \{1, 3, 4\}$, $r_4 = E \setminus C_4 = \{1, 4\}$, and $r_5 = E \setminus C_5 = \{2, 3, 5\}$. Solving the GLOBAL REROUTING problem boils down to finding a set of edges (of minimum cardinality) for which at least an edge will be available over all failure scenarios: $\{4, 5\}$. If we install a capacity of 1 for these two edges, we obtain a survivable network of minimum cost for protected against the 5 considered SRLG events. Note that this is equivalent to finding a minimum cardinality HITTING SET for the original instance, as the sets C_1, C_2, C_3, C_4, C_5 correspond to the edges available in each failure scenario.

3.2. A Layered Network Model

In order to support any service types and service function chains, the traffic associated to each demand must be processed by an ordered sequence of network

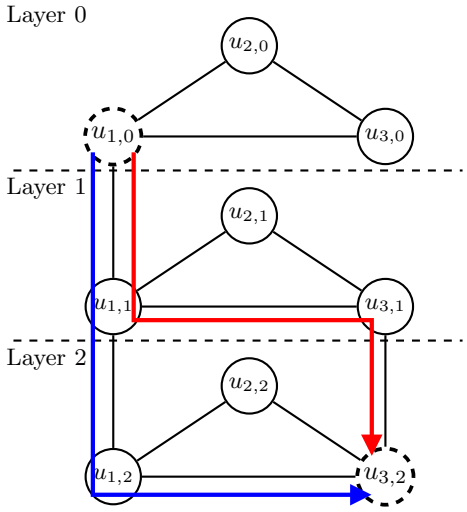


Figure 2: The layered network $G^L(d)$ associated with the demand d such that $s_d = u_1$, $t_d = u_3$, and $C_d = f_1, f_2$, with $G = (V, E)$ being a triangle network. We assume f_1 installed on node u_1 AND f_2 installed on nodes u_1 and u_3. Source and destination nodes of $G^L(d)$ are $u_{1,0}$ and $u_{3,2}$, respectively. They are drawn with dashed lines. Two possible service paths that satisfy d are drawn in red and blue.

functions defined in advance. Similarly to [31], we use a layered graph to model this constraint.

Let $G = (V, E)$ be a graph. We associate to each demand $d \in \mathcal{D}$ a layered graph $G^L(d) = (V', E')$. $G^L(d)$ is defined as follows. For each $u \in V$, V' contains the vertices $(u, 0), (u, 1), ..., (u, \ell(d))$. An edge $((u, i), (v, j))$ belongs to E' if and only if (1) $(u, v) \in E$ and $i = j$, or (2) u is a NFVI node, $u = v$, $j = i + 1$, and the j^{th} function of C_d is installed on u.

Given a demand d, let s_d and t_d be the source and the destination node, respectively. A path starting at vertex $(s_d, 0)$ and finishing at vertex $(t_d, \ell(d))$ of $G^L(d)$ defines (a) which edges of G are used to route the flow associated to the demand; and (b) on which NFVI nodes the traffic is processed by each of the requested network functions. We refer to a path in $G^L(d) = (V', E')$ as a Service Function Path (SFP) - see Figure 2 for an example.

3.3. Compact ILP Formulation

A straightforward way to model our problem consists in using an Integer Linear Program (ILP). The goal of the ILP is to find for each demand $d \in \mathcal{D}$ a Service path on the layered graph $G^L(d)$ for each SRLG event such that the total bandwidth required in the network is minimized. In order to take into account the no failure scenario, we add an SRLG associated with an empty set of links to $\mathcal{R}$. Thus, the SRLGs set is extended to $\mathcal{R} \cup \emptyset$.

Variables:

- $\varphi^{d,r}_{(ui,vj)} \in \{0,1\}$, with $\varphi^{d,r}_{(ui,vj)} = 1$ if demand d uses link $((u,i),(v,j))$ of $G^L(d)$ in the SRLG failure event r.

- $x^r_{uv} \geq 0$ is the amount of bandwidth allocated on link (u,v) of G in the SRLG failure event r.

Objective (1): minimization of the bandwidth needed in the network in order to guarantee the recovery.

$$\min \sum_{(u,v)\in E} \max_{r\in\mathcal{R}} x^r_{uv} \tag{1}$$

For each link, the needed capacity to be installed is $\max_{r\in\mathcal{R}} x^r_{uv}$ in order to guarantee the recovery for every failure scenario or SRLG in $\mathcal{R}$. We thus mininimize the sum over all links in (1).
Flow Conservation constraints (2) (3) (4): for each demand d, SRLG set $r \in \mathcal{R}$,

$$\sum_{(v,j)\in\omega((s_d,0))} \varphi^{d,r}_{(s_d0,vj)} = \sum_{(v,j)\in\omega((t_d,\ell(d))} \varphi^{d,r}_{(t_d\ell(d),vj)} = 1 \tag{2}$$

$$\sum_{(v,j)\in\omega((u,i))} \varphi^{d,r}_{(ui,vj)} \leq 2 \quad v \in V \setminus \{(s_d,0),(t_d,\ell(d))\} \tag{3}$$

$$\sum_{(v',j')\in\omega((u,i))\setminus\{(v,j)\}} \varphi^{d,r}_{(ui,v'j')} \geq \varphi^{d,r}_{(ui,vj)}$$
$$v \in V \setminus \{(s_d,0),(t_d,\ell(d))\}, \ell \in \omega((u,i)) \tag{4}$$

Equation (2) ensures that the path of demand d starts at the source node $(s_d,0)$ in layer 0 and ends at the destination node $(t_d,\ell(d))$ in layer $\ell(d)$, with $\ell(d)$ the length of the SFC of the demand d. Equation (3) forces the paths to not use twice the same link of the layered graph to avoid loops. The flow conservation of each path, i.e., a path arriving at an intermediate node has to leave this node is a result of Equation (4). Remark that the constraint is an inequality. However, we have equality for an optimal solution as a product of the minimization objective.

Unavailable links in an SRLG failure event (5): for each $r \in \mathcal{R}$,

$$\sum_{d\in\mathcal{D}} \sum_{(u,v)\in r} \sum_{k=0}^{\ell(d)} \varphi^{d,r}_{(uk,vk)} = 0. \tag{5}$$

Equation (5) ensures that a path cannot use a link involved in SRLG r when considering the failure scenario r.

Bandwidth utilization in an SRLG failure event (6): for each SRLG set $r \in \mathcal{R}$, link $(u,v) \in E$,

$$\sum_{d\in\mathcal{D}} bw_d \cdot \sum_{k=0}^{\ell(d)} \varphi^{d,r}_{(uk,vk)} \leq x^r_{uv}. \tag{6}$$

Equation 6 guarantees that the bandwidth usage on each link (which is the sum of the bandwidth usage over all layers on the link) is lower than or equal to the link capacity.

3.4. Column Generation Approach

One can apply the Dantzig-Wolfe decomposition [32, 33] to the ILP formulation, to exploit its block structure per demand $d \in \mathcal{D}$. The resulting model takes the form of a path flow formulation. In order to model the ordered sequence of network functions by which the traffic associated to a demand must be processed, we use a layered graph, similarly as in [31]. Let $G = (V, E)$ be a graph. We associate to each demand $d \in \mathcal{D}$ a layered graph $G^L(d) = (V', E')$. $G^L(d)$ is defined as follows. For each $u \in V$, V' contains the vertices $(u, 0), (u, 1), ..., (u, \ell(d))$. An edge $((u, i), (v, j))$ belongs to E' if and only if (1) $(u, v) \in E$ and $i = j$, or (2) u is an NFVI node, $u = v$, $j = i + 1$, and the j^{th} function of C_d is installed on u. We refer to a path in $G^L(d) = (V', E')$ as a Service Function Path (SFP).
We denote by Π_d^r, the set of service function paths for a demand d in the SRLG failure situation r. Each service path π is associated with an integer value $a_{uv}^{\pi} \geq 0$ telling the number of times link (u, v) is used in the service path π.
Variables:

- $y_{\pi}^{d,r} \geq 0$, where $y_{\pi}^{d,r} = 1$ if demand d uses path π as a service path in the SRLG failure event $r \in \mathcal{R}$.
- $x_{uv} \geq 0$, is the bandwidth allocated on link $(u, v) \in E$.

Objective (7): minimization of the required bandwidth

$$\min \sum_{(u,v)\in E} x_{uv} \tag{7}$$

One service path for each demand and SRLG failure event (8): for all $d \in \mathcal{D}$, $r \in \mathcal{R}$

$$\sum_{\pi\in\Pi_d^r} y_{\pi}^{d,r} \geq 1. \tag{8}$$

Inequality (8) ensures that at least one path is selected for each pair demand/SRLG event. Note that a single path is selected for optimal solutions due to the minimization objective.

Bandwidth utilization (9): for all $(u, v) \in E$, $r \in \mathcal{R}$

$$x_{uv} \geq \sum_{d\in\mathcal{D}} \sum_{\pi\in\Pi_d^r} bw_d \cdot a_{uv}^{\pi} \cdot y_{\pi}^{d,r}. \tag{9}$$

Given its very large number of variables, column generation is an efficient technique to handle the above linear integer programming model. One starts with a limited set of variables in a so-called Restricted Master Problem (RMP). At each iteration, the RMP is solved. The dual values associated to the constraints are used to generate new paths with negative reduced cost and the associated

variables are added to the RMP that may enable to improve the current solution. This process is repeated until no more columns can be added to the RMP, i.e., no more columns with negative reduced cost exist. We refer to [34] for more details regarding this technique.

Pricing Problem

The pricing subproblem is solved independently for each demand d and SRLG failure event r and it returns a service path π. It consists in finding a minimum cost service path in the layered graph where the weight of a link is defined according to the dual values of the associated constraint.
Variables:

- $\varphi_{(ui,vj)} \in \{0,1\}$, where $\varphi^{d,r}_{(ui,vj)} = 1$ if the flow is forwarded on link $((u,i),(v,j))$ of $G^L(d)$.

Let $\alpha^{sd}_{\omega} \geq 0$ and $\beta^{r}_{uv} \geq 0$ be the dual values relative to constraints (8) and (9), respectively. The service path reduced cost for a given demand d and an SRLG r can be written as:

$$\min -\alpha^{d}_{r} + bw_d \cdot \sum_{(u,v)\in E} \beta^{r}_{uv} \cdot \sum_{k=0}^{\ell(d)} \varphi_{(uk,vk)}$$

The first term is a constant for each request, and the second term corresponds to a summation over the links of the network. Therefore, the objective function (10) becomes:

$$\min \sum_{(u,v)\in E} \beta^{r}_{uv} \cdot \sum_{k=0}^{\ell(d)} \varphi_{(uk,vk)}. \tag{10}$$

Thus, for each request and for each failure situation, the pricing subproblem corresponds to a weighted shortest-path problem in the layered graph. In a given SRLG failure situation r and for all the demands $d \in D$, the weight of a link $((u,i),(v,j))$ of $G^L(d)$ is defined to be β^{r}_{uv} if $i = j$, 0 otherwise. Either one of these paths leads to a negative reduced cost column, or the current master solution is optimal for the unrestricted program. In the former case, the new configurations found are then added iteratively to the RMP. In the second case, the solution of the linear relaxation of the RMP z^{*}_{LP} is optimal. Convergence of the basic column generation procedure suffers from dual oscillations as the number of constraints (9) is large. To improve the convergence and reduce the fluctuations in the dual variables, we use a piecewise linear penalty function stabilization described in [35]. Associated to the optimal solution of the linear relaxation of the RMP, for each demand d and SRLG failure situation r, there is a set of service paths identified by all the variables $y^{d,r}_{\pi}$ with value greater than 0. These service paths guarantee the minimum cost in terms of required bandwidth to deploy to guarantee the recovery in the splittable flow case. However, if we restrict our attention to the unsplittable flow case, we have to select only one service path for each demand and SRLG failure situation. The problem

now consists in making this choice by reducing the *overflow* introduced in the network. One possible way consists in changing the domain of the variables in the last RMP from continuous to integer and use an ILP solver. We refer to this strategy as MasterILP.

3.5. Benders Decomposition Approach

Applying Benders Decomposition technique [36] to our compact model consists in splitting the original problem variables into first stage link capacity assignments on one hand, and second stage routing decisions on the other hand. The master problem is in terms of the x_{uv} variables. It takes the following form.

Objective (11): minimization of the bandwidth used in the network

$$\min \sum_{(u,v)\in E} x_{uv}. \tag{11}$$

Metric inequalities (12):

$$\sum_{(u,v)\in E} \mu_{uv} \cdot \delta_{uv,r} \cdot x_{uv} \geq \sum_{d\in\mathcal{D}} \lambda_d(\mu) \cdot bw_d \quad \forall\ \mu \in \mathbb{R}+^E \tag{12}$$

where the latter constraints are known as metric inequalities [37]. They can be separated in polynomial time by solving an LP. Hence, they can be handled in a lazy way by generating them dynamically, which allows to solve the problem using the cutting plane algorithm. These cuts are iteratively added to the master problem until the difference between the lower bound, corresponding to the solution of the master problem, and the upper bound, corresponding to the solution of the subproblems, falls under a fixed value ϵ.

Benders separation subproblem is solved given the link bandwidth vector x. This capacity assignment is globally feasible (for the splittable problem) if and only if for each vector $\mu = \{\mu_{uv} \geq 0 : (u,v) \in E\}$ and for each SRLG failure situation $r \in \mathcal{R}$, the inequality

$$\sum_{(u,v)\in E} \mu_{uv} \cdot \delta_{uv,r} \cdot x_{uv} \geq \sum_{d\in\mathcal{D}} \lambda_d(\mu) \cdot bw_d \tag{13}$$

holds, where $\delta_{uv,r} \in [0,1]$ is the available portion of link (u,v) under scenario r, and $\lambda_d(\mu)$ is the length of the shortest path for demand d with respect to link metrics μ.

Associated to the optimal solution of the Master problem, we have the optimal link capacities in the splittable flow case, as in the Column Generation case. The main difference relies in the fact that we do not have the selected paths. We thus have to find a path for each demand and failure situations trying to minimize the *overflow*, with respect to the solution found in the splittable flow case.

Algorithm 1 Iterative ILP (IterILP)

1: Solve the linear relaxation of the general problem
2: $\tilde{c} \leftarrow c^*$
3: **for each** $r \in \mathcal{R}$ **do**
4: (a) route the demands on $G' = (V, E \setminus r, \tilde{c})$ solving MinOverflowILP, an integral multicommodity flow problem minimizing the overflow
5: (b) update $\tilde{c}$ with the introduced overflow (if any)
6: **end for**
7: **return** $\tilde{c}$

3.6. The Min-Overflow Problem

The efficiency of decomposition methods such as Column generation and Benders technique is based on two main principles: (i) decomposing the problem into sub-problems which can be solved efficiently (ii) first solving fractional versions of the problems. Indeed, solving a fractional Linear Program can be done in polynomial time, when their mixed integral version often is NP-complete (as it is the case for the problems considered in the paper). An important step of the methods is to obtain good integral solutions from the optimum fractional solutions found. The Min Overflow Problem appears in this context.

As it is costly to solve (exactly) the integer version of the master program (the MasterILP strategy discussed above at the end of Section 3.4), to obtain a "good" integer solution, we could use another approach. That is, we may start by efficiently computing a fractional solution to the linear relaxation of the problem (i.e., when flows are splittable) using either the Column Generation algorithm or the Benders Decomposition technique and, then, we try to obtain a *good* integer solution to the problem (i.e., when flows are unsplittable) by minimizing the cost to pay in terms of additional capacity (i.e., the *overflow*) over all the scenarios when using a single path for each demand.

3.6.1. IterILP Algorithm

We define overflow as the total amount of additional bandwidth to be allocated to the network in order to satisfy all the demands. To this end, one possible strategy considers each scenario one at a time, and formulates a multicommodity flow problem as an ILP. The objective function consists in minimizing the overflow to be allocated to the network. We refer to this strategy as IterILP - see Algorithm 1 for its pseudo-code. The used ILP, denoted as MinOverflowILP, is given below.

MinOverflowILP:

Input a graph $G = (V, E)$, an SRLG set $r \in \mathcal{R}$, and a capacity function $\tilde{c}$.

Objective (14): minimization of the overflow (additional bandwidth) needed in the network in order to guarantee the recovery for the scenario corresponding

to SLRG r.

$$\min \sum_{(u,v)\in E} o^r_{uv} \tag{14}$$

Remark: if the objective function is equal to $|E|$, all demands can be routed with the capacity function $\tilde{c}$. All the overflow ratios are equal to 0 and no additional capacity is needed for the scenario of the SLRG r.

Flow conservation constraints (15) (16) (17): for each demand d,

$$\sum_{(v,j)\in\omega((s_d,0))} \varphi^{d,r}_{(s_d 0,vj)} = \sum_{(v,j)\in\omega((t_d,\ell(d))} \varphi^{d,r}_{(t_d \ell(d),vj)} = 1 \tag{15}$$

$$\sum_{(v,j)\in\omega((u,i))} \varphi^{d,r}_{(ui,vj)} \leq 2 \quad v \in V \setminus \{(s_d,0),(t_d,\ell(d))\} \tag{16}$$

$$\sum_{(v',j')\in\omega((u,i))\setminus\{(v,j)\}} \varphi^{d,r}_{(ui,v'j')} \geq \varphi^{d,r}_{(ui,vj)}$$
$$v \in V \setminus \{(s_d,0),(t_d,\ell(d))\}, \ell \in \omega((u,i)) \tag{17}$$

Bandwidth utilization for the SRLG set r (18): for each link $(u,v) \in E$,

$$\sum_{d\in\mathcal{D}} bw_d \cdot \sum_{k=0}^{\ell(d)} \varphi^{d,r}_{(uk,vk)} \leq x^r_{uv}. \tag{18}$$

Equation 18 guarantees that the bandwidth usage on each link is lower than or equal to the link capacity.

Definition of the overflow: the overflow ratio of the link $uv \in E$, o^r_{uv}, is defined as 1 if $x^r_{uv} \leq \tilde{c}_{uv}$ and $(x^r_{uv} - \tilde{c}_{uv})/\tilde{c}_{uv}$ otherwise, i.e., $o^r_{uv} = \max((x^r_{uv} - \tilde{c}_{uv})/\tilde{c}_{uv}, 1)$. It thus gives Inequalities (19) and (20), for each link $(u,v) \in E$,

$$o^r_{uv} \geq \frac{x^r_{uv} - \tilde{c}_{uv}}{\tilde{c}_{uv}} \tag{19}$$

$$o^r_{uv} \geq 1 \tag{20}$$

Note that the minimization of the objective function will imply the equality to the maximum for an optimal solution.

If on one hand, this strategy leads to good results, on the other hand, it may not scale well, since we have to solve an ILP for each SRLG failure scenario.

3.6.2. Algorithmic Complexity and Randomized Rounding Strategy

Another strategy consists in using an algorithm to route the demands while minimizing the overflow. The problem to be solved for an SRLG failure scenario which we refer to as Min Overflow Problem can be stated as follows.

Definition 1 (Min Overflow Problem)**.**

Input: A graph $G = (V,E)$, a collection $\mathcal{D}$ of demands, each associated with

a source, a destination and the units of flows to be routed. Also, each demand is associated with a set of paths, corresponding to the fractional solution of the splittable flow version of the problem. Lastly, a capacity function $c^* : (u, v) \rightarrow c^*_{uv}$, according to the optimal capacities found solving the linear relaxation of the general problem.
Output: A path for each demand.
Objective: Minimize the overflow, i.e., minimize $\sum_{(u,v)\in E} \frac{\tilde{c}(u,v)}{c^*_{uv}}$ with $\tilde{c}(u, v)$ defined as the maximum between c^*_{uv} and the capacity of the link (u, v) after having selected one path per demand.

Note that, contrary to the classical version of the problem, we do not have hard capacity constraints to respect while computing an integer routing. Herein, the goal is to route all the demands reducing the increase in terms of capacity over each of the links (i.e., the overflow) with respect to the *free given capacities* already available in the network.

Proposition 2. *The* Min Overflow Problem *is APX-hard (and so is NP-Hard) and cannot be approximated within a factor of* $1 + \frac{3}{320}$*, unless P=NP.*

Proof. We use a reduction from Max 3-SAT. Let $\mathcal{I}$ be an instance of Max 3-SAT with n variables $V_i, 1 \leq i \leq n$ and m clauses $C_j, 1 \leq j \leq m$. We associate each boolean variable V_i to a demand d_i asking for one unit of flow from a source s_{d_i} to a destination t_{d_i} connected by two paths $P_0(V_i)$ and $P_1(V_i)$. Selecting $P_1(V_i)$ (respectively $P_0(V_i)$) correspond to assign to V_i the true (respectively false) value.
We associate each clause C to to an edge (u_C, v_C) and we build the paths in the following way. For each variable V_i, we consider all the set $C(V_i)$ with all the clauses in which V_i appears as positive literal. $C(V_i) = C_{i_1}, C_{i_2}, ..., C_{i_m}$ with $i_1 \leq i_2 \leq ... \leq i_m$. Then, $P_1(V_i) = s_{d_i}, (u_{i_1}, v_{i_1}), (u_{i_2}, v_{i_2}), ..., (u_{i_m}, v_{i_m}), t_{d_i}$.
In a similar way, we consider now all the clauses in which V_i appears as negative literal. $C(\overline{V_i}) = C_{\bar{i}_1}, C_{\bar{i}_2}, ..., C_{\bar{i}_m}$ with $\bar{i}_1 \leq \bar{i}_2 \leq ... \leq \bar{i}_m$. $P_1(V_i)$ is defined as $s_{d_i}, (u_{\bar{i}_1}, v_{\bar{i}_1}), (u_{\bar{i}_2}, v_{\bar{i}_2}), ..., (u_{\bar{i}_m}, v_{\bar{i}_m}), t_{d_i}$.
As we build paths in this way, the load of an edge (u_C, v_C) is equal to the number of literals in the clause C assigned to the false value. There are $\sum_{i=1}^{n}(2i_m + 1)(2\bar{i}_m+1) = 6m+2n$ edges in the construction, as $\sum_{i=1}^{n} |C(V_i)|+|C(\overline{V_i})| = 3m$, the numbers of literals in the formula. We now assign each edge a capacity 2. A fractional routing always exists. Indeed, routing one half of the charge of each demand d_i on $P_0(V_i)$ and the half on $P_1(V_i)$ is feasible, since after identification an arc receives at most $3 \times \frac{1}{2} \leq 2$. The case of an integral flow is quite different, since, in such a case, only one between $P_0(x)$ or $P_1(x)$ can be chosen. Since the capacity of the edges is 2, the cost will be 2 on each identified edge $\iff$ the formula is satisfiable. This proves that the problem is NP-complete (as 3-SAT is NP-complete). Then, we derive an inapproximability result using the fact that it is NP-hard to satisfy more than $\frac{7}{8}$ of the clauses (even if the formula is satisfiable) [38]. So, we may have to pay 3 on $m/8$ edges (even though the optimal is 2 on all edges). Since the initial cost is less than 2 times the number of edges, it is less than $2 \times (6m + 2n) = 12m + 4n$. We have $n \leq \frac{m}{3}$. So, it is

NP-hard to decide if the cost is 1 or $\frac{(12+\frac{4}{3})m+\frac{m}{8}}{(12+\frac{4}{3})m} = 1 + \frac{3}{320}$. □

Proposition 3. *The* Min Overflow Problem *can be approximated with high probability within a factor of* $(1 + \frac{1}{\mathrm{e}}) + \epsilon$*, for any* $\epsilon > 0$.

Let c^*_{uv} be the optimal capacity of an edge (u, v) in the splittable flow case. After having computed a fractional flow, we have associated to each demand $d \in \mathcal{D}$ a set consisting of $n(d) \geq 1$ paths $\mathcal{P}_d = \{P_{d,i} : i = 1, ..., n(d)\}$. Each path $P_{d,i}$ is associated to a multiplier $0 \leq \lambda_{d,i} \leq 1$ such that $\sum_{i=1}^{n(d)} \lambda_{d,i} = 1$ which gives the amount of flow $\lambda_{d,i} \cdot bw_d$ routed on $P_{d,i}$. Let $\lambda_{d,i}(uv)$ be the fraction of flow routed on the edge (u, v) by a demand d. Note that for each edge (u, v) we have $\sum_{d\in\mathcal{D}} \sum_{i=1}^{n(d)} bw_d \cdot \lambda_{d,i}(uv) \leq c^*_{uv}$ since by hypothesis these capacities are feasible for the splittable flow case. In order to find an unsplittable solution, we use a rounding–based heuristic referred to as Randomized Rounding, which assigns to a demand d a path $P_{d,i}$ with probability $\lambda_{d,i}$. We consider now the impact in terms of load on an edge (u, v). Let f_{uv} be the flow on (u, v) at the end of the rounding procedure. Clearly, for each edge (u, v) $\mathbb{E}(f_{uv}) \leq c^*_{uv}$ holds. Let O_{uv} be the overflow on the edge (u, v) defined as $\max(0, f_{uv} - c^*_{uv})$. We denote by $\mathsf{P}_0(uv) = \mathsf{P}[f_{uv} = 0]$ the probability that the edge (u, v) is not used. We have

$$\mathbb{E}[O_{uv}] = \mathsf{P}_0(uv) \cdot 0 + (1 - \mathsf{P}_0(uv))\, \mathbb{E}[f_{uv}|f_{uv} > 0] - c^*_{uv} \tag{21}$$
$$= (1 - \mathsf{P}_0(uv))\, \mathbb{E}[f_{uv}|f_{uv} > 0] - c^*_{uv}(1 - \mathsf{P}_0(uv)) \tag{22}$$

Moreover,

$$\mathbb{E}[f_{uv}] = \mathsf{P}_0(uv) \cdot 0 + (1 - \mathsf{P}_0(uv))\, \mathbb{E}(f_{uv}|f_{uv} > 0). \tag{23}$$

It gives

$$\mathbb{E}[f_{uv}|f_{uv} > 0] = \frac{\mathbb{E}[f_{uv}]}{1 - \mathsf{P}_0(uv)}. \tag{24}$$

We can therefore bound the expected overflow of a link (u, v).

$$\mathbb{E}[O_{uv}] = \mathbb{E}[f_{uv}] - c^*_{uv}(1 - \mathsf{P}_0(uv)) \tag{25}$$
$$= \mathsf{P}_0(uv)c^*_{uv} - (c^*_{uv} - \mathbb{E}[f_{uv}]) \leq \mathsf{P}_0(uv)c^*_{uv}. \tag{26}$$

Let us now consider the probability $\mathsf{P}_0(uv)$ that an edge is not used after the randomized rounding. Given an edge (u, v), we define $\mathcal{P}_{uv}$ to be the paths that contain (u, v) as an edge.

$$\mathsf{P}_0(uv) = \prod_{P_{d,i}\in\mathcal{P}_{uv}} (1 - \lambda_{d,i}). \tag{27}$$

The probability for an edge not to be selected is maximized when all $\lambda_{d,i}$ are equal (i.e., $\lambda_{d,i} = \frac{1}{|\mathcal{P}_{uv}|}\ \forall\ \lambda_{d,i} \in \mathcal{P}_{uv}$). Thus,

$$\mathsf{P}_0(uv) = (1 - \frac{1}{\rho|\mathcal{P}_{uv}|})^{|\mathcal{P}_{uv}|}, \tag{28}$$

where ρ is defined to be $\frac{\mathbb{E}[f_{uv}]}{c^*_{uv}}$. This gives an upper bound for the possible value of $\mathsf{P}_0(uv)$. Indeed,

$$\mathsf{P}_0(uv) \leq \lim_{n\to\infty}(1-\frac{1}{\rho n})^n = \frac{1}{\mathrm{e}^\rho}. \tag{29}$$

The function is minimized with $\rho = 1$. We thus get

$$\mathbb{E}[O_{uv}] \leq \frac{1}{\mathrm{e}^\rho}c^*_{uv} - (c^*_{uv} - \mathbb{E}[f_{uv}]) \tag{30}$$

$$\leq c^*_{uv}(\frac{1}{\mathrm{e}^\rho} - (1-\rho)) \leq \frac{1}{\mathrm{e}}c^*_{uv} \approx 0.37c^*_{uv}. \tag{31}$$

Finally, the expected cost of the solution provided is

$$\mathbb{E}\left[\frac{\sum_{(u,v)\in E} O_{uv}}{\sum_{(u,v)\in E} c^*_{uv}}\right] = \frac{\sum_{(u,v)\in E}\mathbb{E}[O_{uv}]}{\sum_{(u,v)\in E} c^*_{uv}} \leq \frac{1}{\mathrm{e}} \approx 0.37. \tag{32}$$

By using the Markov inequality, the probability that the obtained solution has a cost larger than $1.37(1+\epsilon)$ is at most $\frac{1}{1+\epsilon}$. The overflow resulting from the execution of the randomized rounding can be checked in polynomial time. If the overflow exceeds the factor of $(1+\frac{1}{\mathrm{e}})+\epsilon$, another trial may be necessary in order to find a solution below this value. The number of trials depends on the chosen value for ϵ. For instance, if we set $\epsilon = \frac{1}{10}$, we need an average of 10 trials in order to find a solution with cost not greater than 1.507 ($= 1.37 + 0.137$) times the optimal fractional one. As just shown, the problem of minimizing the overflow can be approximated efficiently for a single scenario. The proposed scheme consists in a randomized rounding to be performed according to the value of the splittable flow solution. We may extend Randomized Rounding to the case of multiple scenarios by simply solving the scenarios in an iterative fashion. At each iteration, an SRLG $r \in \mathcal{R}$ is considered. First, a fractional capacitated multicommodity flow is solved. Then, a $(1+\frac{1}{\mathrm{e}}+\epsilon)$–approximated integer solution is found using the RandomizedRounding procedure described in Algorithm 3. The overflow introduced (if any) by the procedure is then added. We refer to this method as Iterative Randomized Rounding (IterRR)— see Algorithm 2 for the pseudo-code of our proposed algorithm.

4. Numerical Results

In this section, we evaluate the performances of our proposed algorithms on both real and synthetic network topologies and workloads. The compared methods are MasterILP, in which the last RMP is solved as an ILP by setting the domain of the paths variables from fractional to binary. IterILP, in which each scenario is solved independently with an ILP that has, as a goal, the minimization of the overflow and IterRR, in which instead of using an ILP to minimize the overflow, we use a $(1+\frac{1}{\mathrm{e}}+\epsilon)$–approximation algorithm. We show the effectiveness

Algorithm 2 Iterative Randomized Rounding

Solve the linear relaxation of the general problem
$\tilde{c} \leftarrow c^*$
for each $r \in \mathcal{R}$ **do**
    (a) route the demands on $G' = (V, E \setminus r, \tilde{c})$ solving a fractional multicommodity flow problem
    (b) use RandomizedRounding to find a $(1+\frac{1}{\mathrm{e}}+\epsilon)$–approximate integer routing
    (c) update $\tilde{c}$ with the introduced overflow (if any)
end for
return $\tilde{c}$

Algorithm 3 RandomizedRounding

Input: A network with a capacity function $\tilde{c}$ and a fractional routing for a set of demands $\mathcal{D}$
Output: A $(1 + \frac{1}{\mathrm{e}} + \epsilon)$–approximate integer routing

repeat
    for each $d \in \mathcal{D}$ **do** ▷ Build an integral routing
        select a path $P_{d,i}$ among all fractional paths p_i with probability $\lambda_{d,i}(uv)$.
    end for
    $c_i \leftarrow$ compute the capacities of the drawn integral routing
    overflow $= 0$ ▷ Compute the overflow
    for each $e \in E$ **do**
        overflow $+= \frac{\max(c_i, \tilde{c})}{\tilde{c}}$
    end for
until overflow $\leq (1 + \frac{1}{\mathrm{e}} + \epsilon)|E|$
return c_i, the capacities of the drawn integral routing

- of our algorithms in terms of *scalability*. Indeed, we are able to solve instances with much larger numbers of demands, network nodes and links than the classic compact ILP model, and
- of Global Rerouting in terms of *bandwidth usage*, i.e., the bandwidth needed to be provisioned by an operator to handle failures is minimum.

Data sets

We conduct experiments on three real-world topologies from SNDlib [39]: `polska`, (12 nodes, 18 links, and 66 demands), `pdh` (11 nodes, 34 links, and 24 demands) and `nobel-germany` (17 nodes, 26 links, and 121 demands). For these networks, we use the traffic matrices provided with the data set. No information is available about the SRLGs for these networks. Thus, the collection of network failures $\mathcal{R}$ for these instances contains single edge failures. We also conduct experiments on randomly generated instances of different sizes and

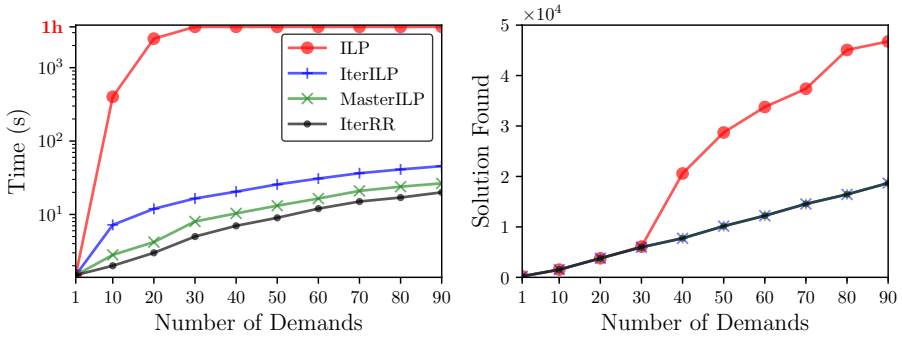


Figure 3: Time and value of the solution found by the ILP and by our proposed methods as a function of the number of demands.

with different SRLGs. We build our synthetic instances using a similar method to the one in [6]. We generate two networks in which we place nodes in a unit square. In each of them, we add links according to the Waxman model [40]. The probability of having a link (u, v) is defined as $\alpha \exp^{\frac{-\text{dist}(u,v)}{\beta L}}$ where $\text{dist}(u, v)$ is the Euclidean distance from node u to node v, L is the maximum distance between two nodes and α, β are real parameters in the range $[0, 1]$. One of the two networks represents the logical IP network, i.e., IP routers and IP links while the other represents the underlying optical network, i.e., cross-connect and fibers. Each IP node is mapped to the closest optical cross-connect and each IP link (u, v) is mapped onto the shortest path between u and v in the physical network. All the IP links using the same physical link are associated to an SRLG. In addition, we add an SRLG for each undirected link. Demands are generated using the model described in [41]. The model considers the distance factor $\exp^{\frac{-\text{dist}(u,v)}{2L}}$ between two nodes u and v. As a result, the load of the demands between close pairs of nodes is higher with respect to pairs of nodes far apart. Finally, the chain of each demand is composed of 3 to 6 functions uniformly chosen at random from a set of 10 functions. Each NFVI node can run up to 6 network functions. Indeed, a node may not be allowed to run all the network functions. Similarly as in [31], locations are chosen according to their betweenness centrality, an index of the importance of a node in the network: it is the fraction of all shortest paths between any two nodes that pass through a given node. Experiments have been conducted on an Intel Xeon E5520 with 24GB of RAM.

Limits of an ILP-based approach. To study the limits in terms of computing time of an ILP-based approach, we tested our optimization models on a small random topology with 10 nodes, 16 links, and 26 SRLGs. In Fig. 3, we show the impact of the number of demands on the execution time. We compare the time necessary to find an optimal solution (on the left) and the value of the solution found (on the right) by the ILP and by our proposed methods. For each experiment, we set a maximum time limit of one hour. If the time

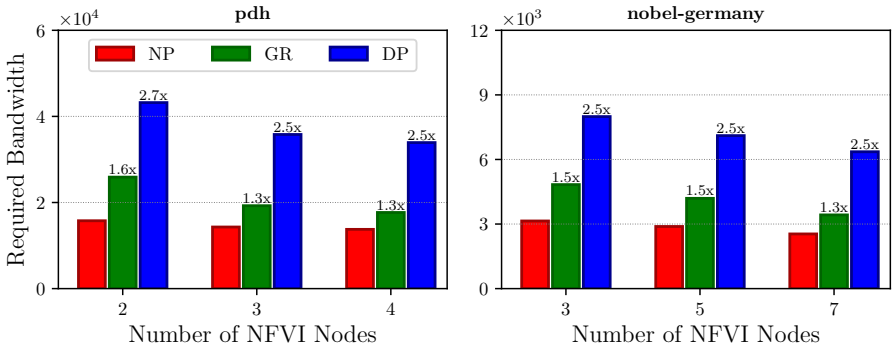


Figure 4: Bandwidth overhead comparison of the Global Rerouting (GR) and Dedicated Path protection (DP) schemes with respect to the no-protection scenario (NP) for `pdh` and `nobel-germany` networks. Labels on top of the bars indicate the overhead with respect to the unprotected case.

Network	z^*_{LP}		**MasterILP**		**IterILP**		**IterRR**	
	ColGen	Benders	time	ϵ	time	ϵ	time	ϵ
`pdh`	22s	32s	11mn	4%	1mn	4.82%	40s	12.7%
`polska`	15s	18s	40s	0.22%	1mn	0.1%	20s	1.4%
`nb-germany`	35s	1mn	40s	0.17%	4mn	0.06%	30s	3.2%
`wxm10`	10s	5s	50s	0.3%	40s	1%	10s	5.5%
`wxm20`	40s	2mn	1mn	0.6%	4mn	0.6%	30s	2.7%
`wxm30`	3mn	16mn	6mn	0.2%	21mn	0.9%	1mn	4.5%
`wxm40`	22mn	>1h	27mn	2.2%	>1h	-	2mn	9.2%

Table 3: Numerical results for the proposed optimization models. first column refers to the time needed to find the optimal fractional solution z^*_{LP}. We set a maximum time limit of 1h. The other columns refer to the proposed methods to obtain an integer solution $\tilde{z}_{ILP}$. For each method, we show the additional time needed and the quality of the solution found, expressed as the ratio $\epsilon = \frac{\tilde{z}_{ILP} - z^*_{LP}}{z^*_{LP}}$.

limit is exceeded, the solution reported represents the best solution found so far. For just 30 demands, the time needed by IBM ILOG Cplex 12.8 to find an exact solution exceeds 1 hour and for larger instances, no optimal solution can be found using an ILP approach in a reasonable amount of time. On the contrary, the algorithms we propose can compute solutions for larger instances fairly efficiently. Indeed, our algorithms only take 1 minute to solve the problem for 90 demands. As the considered network is small, the computed values by the three proposed methods are very close between them. We compare them in the following on larger networks.

Performances of the optimization models. Table 3 summarizes the results of our proposed methods for the 3 real networks and for 4 Waxman random networks. Networks are identified as `wxm_N` with `N` being the number of nodes. The number of demands is set to be $50, 100, 150$, and 200 for the $10, 20, 30$, and 40 nodes networks, respectively. Moreover, the number of resulting SRLGs for

the Waxman random networks are $22, 40, 53$, and 70, respectively. The first column compares the Column Generation (ColGen) and the Benders Decomposition [36] (Benders) techniques to find a fractional solution based on which the heuristics find an integer solution. The Column Generation technique appears to be faster in finding the optimal solution z^*_{LP}: on the largest considered network `wxm40` only takes 22 minutes to find an optimal solution, while Benders would require more than one hour. The remaining 3 columns refer to our optimization methods. For their evaluations, we started from the solutions given by the Column Generation technique, as it is more efficient in general. For each method, we present both the time needed to find a solution $\tilde{z}_{ILP}$ as well as the ratio $\epsilon = \frac{\tilde{z}_{ILP} - z^*_{LP}}{z^*_{LP}}$ with respect to the optimal fractional solution z^*_{LP}. ϵ, called *accuracy* in the following, gives an upper bound on the maximum overflow to pay in excess with respect to the optimal integer solution z^*_{ILP}, since the optimal integer solution may be larger than the fractional one. Both MasterILP and IterILP allow to find near–optimal solutions. As the size of the network increases, we begin to observe the limits of the IterILP approach, as it solves an ILP for each scenario. Although MasterILP demonstrates a better scalability and a very high accuracy, for larger networks we have a tradeoff between the time to find the solution and the quality of the solution found. Indeed, for `wxm40`, IterRR only takes 2 minutes to find a good solution with an accuracy of about 9%, while MasterILP requires 27 minutes to find a solution with an accuracy of 2.2%.

Varying the number of NFVI nodes. NFVI nodes are expensive to both purchase and maintain (e.g., hardware, software licenses, energy consumption, and maintenance). If, on one hand, an over-provisioning corresponds to undue extra costs, on the other hand, under-provisioning may result in poor service to user and in Service Level Agreement (SLA) violations. It is thus necessary to find the right trade-off in terms of NFVI nodes in the network design phase. *Bandwidth overhead.* In Fig. 4, we compare the overhead in terms of bandwidth needed in the network by the global rerouting scheme and Dedicated Path Protection with respect to the bandwidth needed in the unprotected case. For Dedicated Path Protection we compute, for each demand, two SRLG-disjoint paths, i.e., two paths such that no link on one path has a common risk with any link on the other path. In doing this, we set the bandwidth minimization as an optimization task. With an increasing number of NFVI nodes in the network, the required bandwidth decreases. However, the overhead with respect to the unprotected case tends to remain constant. Indeed, if with global rerouting we only need from 30 to 60% more bandwidth, with dedicated path protection we may need almost 3 times more bandwidth to guarantee the recovery.

Paths' delays. In Fig. 5, we show the impact of the number of NFVI nodes on the paths' latencies distribution and compare them with the ones calculated using shortest paths on the layered network. As expected, we see that the number of hops decreases as the number of NFVI nodes increases. The reason is that, the more NFVI-nodes in a network, the higher the opportunity of easily finding closer NFVI-nodes which can perform some of the required network

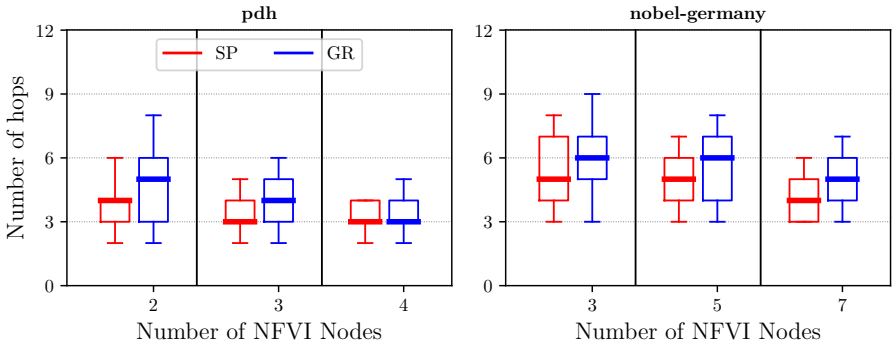


Figure 5: Hops distribution of the backup paths computed by the Global Rerouting scheme (GR) compared with the Shortest Paths (SP) for `pdh` and `nobel-germany` networks. Boxes are defined by the first and third quartiles. Ends of the whiskers correspond to the first and ninth deciles.

functions. Another result is that the paths computed using our method are almost as short (in terms of number of hops) as the shortest paths.

Number of paths. In our considered protection scheme, a demand may be rerouted on a different path in each of the possible SRLG failure situations. Even though our optimization models do not impose constraints on the number of distinct paths for a demand, the experimental results indicate that their number tends to be small in practice. In Fig. 6, we show the distribution for the number of distinct paths of the demands for our considered networks. The number of distinct paths increases with the size of the network and tends to stay within the range $(5, 10)$ for most of them. For instance, for `wxm40` we may have potentially 71 distinct paths to be used for a demand, one for each of the possible SRLG failure scenarios plus one for the no–failure case. However, as the results show, in such a case, 50% of the demands would use no more than 12 distinct paths.

5. Implementation Perspectives

In the previous sections, we followed a theoretical approach to design a bandwidth-optimal failure recovery scheme relying on the hypothesis that programmable networks should be able to implement such a scheme. In this section, we validate this assumption by implementing the scheme on production SDN appliances, namely Open vSwitch and OpenDaylight, and evaluate different trade-offs with Mininet. Our evaluation shows that implementation choices have a significant impact on the recovery time of protection mechanisms.

5.1. Implementation Options

A first option to implement the protection scheme in OpenFlow is to let the OpenFlow controller fully update the flow tables on the switches upon failure.

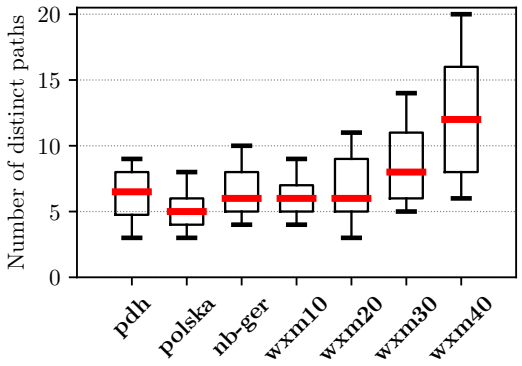
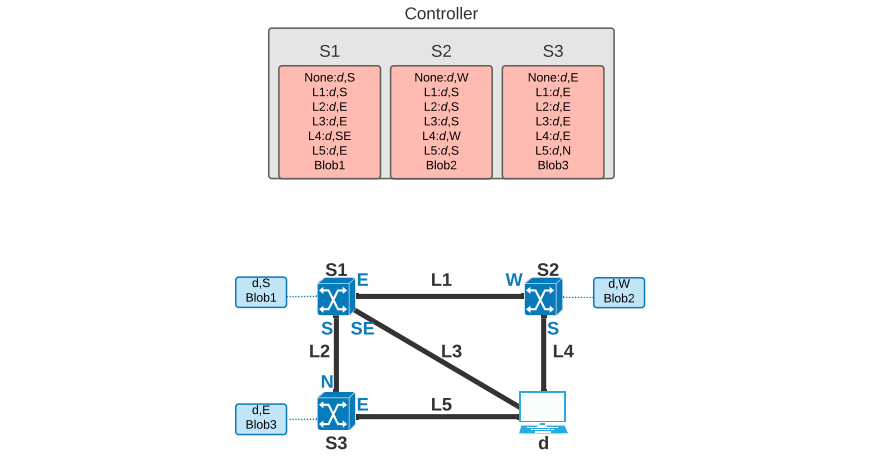


Figure 6: Distribution for the number of distinct paths for each demand. Boxes are defined by the first and third quartiles. Ends of the whiskers correspond to the first and ninth deciles. The median value is drawn in red.

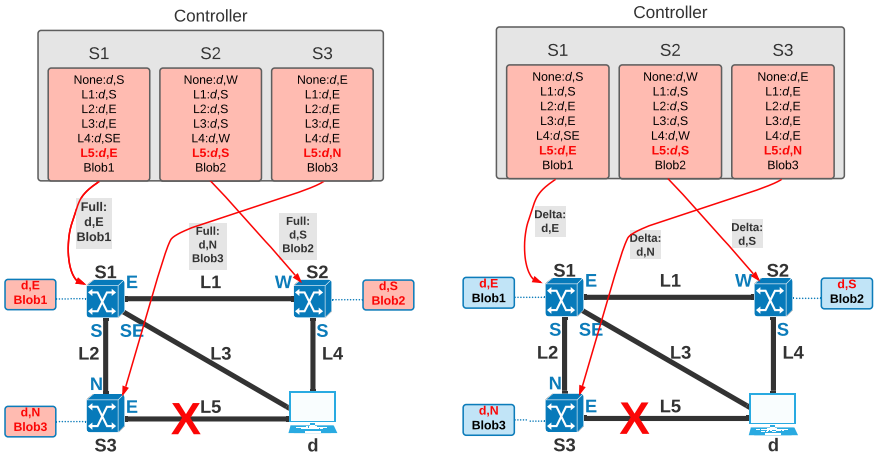
When the controller detects a failure, it sends the new flow tables to the impacted switches. This approach minimizes the memory usage on the switches but incurs high signaling overhead between the controller and the switches, and imposes the latter to install a full flow table at every network change. We refer to this option as *full*. A variation of this option is to only send the changes to be performed on the flow tables to the switches to reduce the signaling load and the number of flow table updates on the switches. We name this option *delta*. Another option is to leverage the Multiple Flow Tables capability introduced in OpenFlow 1.3 to pre-install the flow tables for each SRLG failure scenario in the switches. When the controller sends a failure notification to a switch, the switch activates the appropriate flow table in only one operation (using goto). This approach minimizes the signaling load and flow table changes but consumes more memory on the switches than the other options. This option is referred to as *notification*. In the rest of the paper, we study the impact of the technical choices on the recovery time in realistic operational scenarios.

Fig. 7 illustrates how the `full` and `delta` implementation options presented above operate. The network is composed of 3 switches (`S1`,`S2`,`S3`), one destination for traffic (`d`), and one controller. The controller has the full knowledge of the network and pre-computed the alternative flow tables for each potential failure. To save space, we only show the information related to link failures for host `d`, the rest (e.g., node failures, SRLGs, other destinations...) of the table is hidden behind the *blob* term. Figures 7b and 7c show the messages sent by the controller to the switches to recover from the failure of link `L5` and the resulting flow tables on the switch.

Fig. 8 is the counterpart of Fig. 7 for the `notification` implementation option. In this case, the controller does not store flow tables in memory. Instead, it stores the goto operations to be performed in case of failures. Fig. 8a shows the preconfigured rules and tables in the network before the failure, while Fig. 8b



(a) State of the network before any failure.



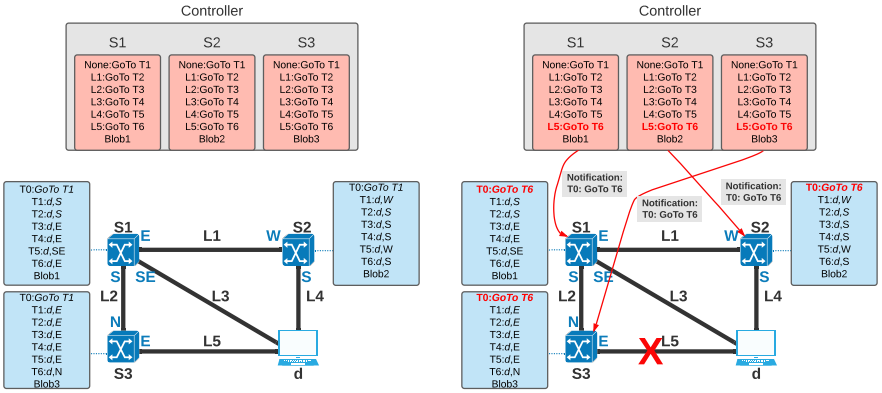
(b) *Full option.* (c) *Delta option.*

Figure 7: Switch update operations for the `full` and `delta` protection scheme implementation options in SDN networks upon one link failure. In the `full` protection scheme, the table inside the switches are completely overwritten by the new tables. In the `delta` protection scheme the controller modifies only the rules that are changing in the failure scenario

shows the goto instructions sent to the switches to recover from the failure of link `L5` and how it changes the forwarding decisions on the switches.

5.2. Experimental Setup

Our experimental platform is a dual Intel Xeon E5-2630 CPU server with 128GB of RAM running Mininet 2.2.2 [42] and the controller *OpenDaylight*



(a) State of the network before any failure.

(b) *Notification*.

Figure 8: Switch update operations for the `notification` protection scheme implementation option in SDN networks upon one link failure. The rules are preinstalled in the switches in a different table for each failure scenario, and `table 0` is pointing to the `no failure table` in the normal state. In a failure situation the controller updateS the pointer in `table 0` to use the table assigned for the failure scenario.

Oxygen [43] with OpenFlow 1.3.
The routing logic is implemented as a network application orchestrator that communicates with the controller with the HTTP OpenDaylight Northbound API. This approach is recommended as it decouples the implementation of the logic from the implementation of the controller.

We also made an *ideal* implementation to assess the best possible performance one could have. It is equivalent to the *notification* option, but is implemented directly in Mininet with Open vSwitch commands. In this case, the switches are programmed directly, without the controller overhead. Mininet emulation is centralized, so we are able to synchronize all failure notifications to the switches just after the failure occurs bypassing thus the controller.

Due to the limited number of CPU cores on our emulation server, we could only evaluate the `wxm10` and the `polska` networks.

5.3. Recovery Time

The *recovery time* is the span of time between a failure event and the moment in which all switches are updated to be in a state that circumvents the failure. To measure the recovery time, we continuously probe the end-to-end paths with UDP datagrams.

Fig. 9 shows the recovery time for our three OpenDaylight implementation options and the ideal one. It compares our Global Rerouting (GR) protection scheme to the Dedicated Path (DP) protection scheme. The figure highlights the importance of implementation choices on the recovery time: the notification option significantly outperforms the other options. The ideal implementation

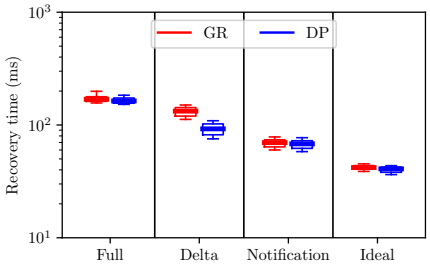


Figure 9: Recovery time comparison of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

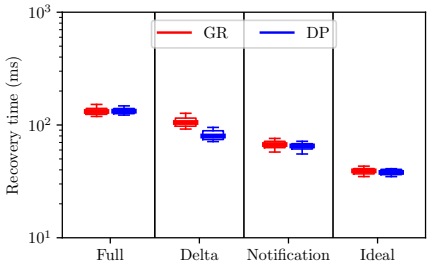


Figure 10: Recovery time comparison of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `wxm10` network.

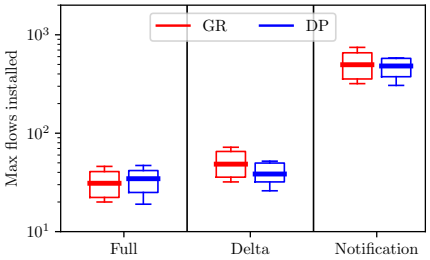


Figure 11: Comparison of the flow table sizes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

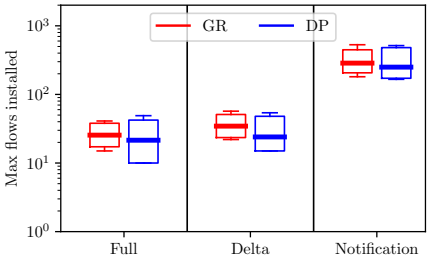


Figure 12: Comparison of the flow table sizes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `wxm10` network.

also shows that the tools used to implement the protection scheme have a significant impact on the recovery time as, all things considered, our ideal is just a way of implementing the notification option without a controller. Actually, a significant fraction of the recovery time in OpenDaylight implementations is caused by the usage of the Northbound API. All implementation options offer sub-second recovery time for the considered network.

Figures 9 and 13 show that there is a direct link between the number of changes to be performed on the switches and the recovery time in Polska network. The same observation applies to the wx10 network as highlighted by Figures 10 and 14. Figures 13 and 14 report, for each switch, the maximum number of flow table changes observed, expressed in number of flow entries for the three OpenDaylight implementation options. Dedicated path protection has similar recovery time than global rerouting when the full implementation is used. This is because the number of flows to install on switches is similar between dedicated path protection and global rerouting. Similarly, no performance difference can be observed when the notifications-based implementation is used as the controller only has to update one rule per switch. On the con-

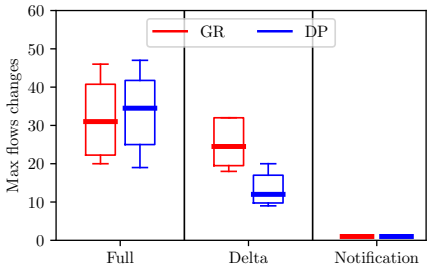


Figure 13: Comparison of the number of flow table changes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

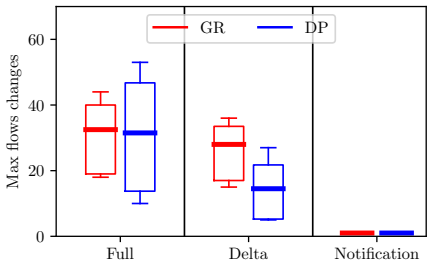


Figure 14: Comparison of the number of flow table changes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `wxm10` network.

trary, if updates are performed with the delta implementation, dedicated path protection converges faster than global rerouting as it requires much fewer path changes.

5.4. Operational Trade-offs

Based on the recovery time, one would recommend deploying the notification option. However, the reduction of the recovery time comes at the cost of increasing flow table sizes on switches as shown in Figures 11 and 12 that report, for each switch, the maximum observed flow table size expressed in number of flow entries for the three OpenDaylight implementation options in both networks. The full option minimizes the number of entries as it only requires to have the flow table for the current routing case. The delta option consumes slightly more space than the full one as the flow table always contains the "no-failure" scenario flow table and the additional flow entries needed to circumvent the current failure. Finally, the notification option has significantly larger flow tables (one order of magnitude more) as flow tables always contain all the potential failure scenarios, which may prevent it to be used in practice on low end switches or for large networks.
As the robustness of the controller is an orthogonal problem that must be treated by all SDN solutions and because it is already largely studied [44], it was not considered here.

6. Conclusion

In this paper, we studied the network dimensioning problem with protection against a Shared Risk Link Group (SRLG) failure in the light of network virtualization. We considered a path-protection method based on a global rerouting strategy, which makes the protection method optimal in terms of bandwidth. We proposed algorithms to compute the backup paths for the demands which rely on the Column Generation and Benders Decomposition techniques. We

validated them with simulations on real-world and on randomly generated network topologies and workloads. Simulations show that our algorithms can reach near-optimal solutions in a short time, even for large instances. Finally, we proposed a real implementation of our proposition in OpenDaylight and show the applicability of the global rerouting protection method when SDN is used. Our experimental results in Mininet show that technical implementation choices may have an important impact on the time to recover from a failure, or on the ability to actually deploy the protection mechanism in large networks.

References

[1] D. S. Johnson, J. K. Lenstra, and A. R. Kan, "The complexity of the network design problem," *Networks*, vol. 8, no. 4, pp. 279–285, 1978.

[2] M. Grötschel, C. L. Monma, and M. Stoer, "Design of survivable networks," *Handbooks in operations research and management science*, vol. 7, pp. 617–672, 1995.

[3] M. Stoer, *Design of survivable networks*. Springer, 2006.

[4] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni), complete forecast update, 2017–2022," *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, 2018.

[5] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an ip backbone," in *Proceedings of IEEE INFOCOM, 2004*.

[6] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A tool for failure diagnosis in ip networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM, 2005, pp. 173–178.

[7] D. Papadimitriou, "Inference of shared risk link groups," *Internet-draft: draft-many-inference-srlg-02. txt*, 2001.

[8] B. Vass, L. Németh, M. Zachariasen, A. De Sousa, and J. Tapolcai, "Vulnerable regions of networks on sphere," in *2018 10th International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE, 2018, pp. 1–8.

[9] M. Pióro and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[11] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[12] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[13] A. Fumagalli and L. Valcarenghi, "Ip restoration vs. wdm protection: Is there an optimal choice?" *IEEE network*, vol. 14, no. 6, 2000.

[14] P. Fonseca and E. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys & Tutorials*, 2017.

[15] M. Bastam, M. Sabaei, and R. Yousefpour, "A scalable traffic engineering technique in an sdn-based data center network," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 2, p. e3268, 2018.

[16] A. Tomassilli, N. Huin, F. Giroire, and B. Jaumard, "Resource requirements for reliable service function chaining," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–7.

[17] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid sdn networks," in *Proceedings of IEEE INFOCOM, 2015*.

[18] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," in *Proceedings of ACM SIGMETRICS 2011*. ACM.

[19] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow-based segment protection in ethernet networks," *Journal of Optical Communications and Networking*, vol. 5, no. 9, 2013.

[20] A. Tomassilli, G. Di Lena, F. Giroire, I. Tahiri, D. Saucez, S. Pérennes, T. Turletti, R. Sadykov, F. Vanderbeck, and C. Lac, "Poster: Design of Survivable SDN/NFV-enabled Networks with Bandwidth-optimal Failure Recovery," in *Proc. of IFIP Networking 2019*, extended abstract.

[21] A. Tomassilli, G. D. Lena, F. Giroire, I. Tahiri, D. Saucez, S. Pérennes, T. Turletti, R. Sadykov, F. Vanderbeck, and C. Lac, "Bandwidth-optimal failure recovery scheme for robust programmable networks," in *IEEE International Conference on Cloud Networking (CloudNet)*, Coimbra, Portugal, Nov. 2019.

[22] A. Kvalbein, A. F. Hansen, S. Gjessing, and O. Lysne, "Fast ip network recovery using multiple routing configurations," in *Proceedings of IEEE INFOCOM, 2006*.

[23] A. Kvalbein, T. Cicic, and S. Gjessing, "Post-failure routing performance with multiple routing configurations," in *Proceedings of IEEE INFOCOM, 2007*.

[24] B. Vass, L. Németh, and J. Tapolcai, "The earth is nearly flat: Precise and approximate algorithms for detecting vulnerable regions of networks in the plane and on the sphere," *Networks*, vol. 75, no. 4, pp. 340–355, 2020.

[25] B. Vass, J. Tapolcai, D. Hay, J. Oostenbrink, and F. Kuipers, "How to model and enumerate geographically correlated failure events in communication networks," in *Guide to Disaster-Resilient Communication Networks*. Springer, 2020, pp. 87–115.

[26] K. D. R. Assis, R. C. Almeida, H. Waldman, M. J. Reed, B. Jaumard, and D. Simeonidou, "Linear formulation for the design of elastic optical networks with squeezing protection and shared risk link group: Invited paper," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1–5.

[27] H. Kerivin and A. R. Mahjoub, "Design of survivable networks: A survey," *Networks: An International Journal*, vol. 46, no. 1, pp. 1–21, 2005.

[28] M. M. Tajiki, M. Shojafar, B. Akbari, S. Salsano, M. Conti, and M. Singhal, "Joint failure recovery, fault prevention, and energy-efficient resource management for real-time sfc in fog-supported sdn," *Computer Networks*, vol. 162, p. 106850, 2019.

[29] G. Ausiello, A. D'Atri, and M. Protasi, "Structure preserving reductions among convex optimization problems," *Journal of Computer and System Sciences*, vol. 21, no. 1, pp. 136–153, 1980.

[30] I. Dinur and D. Steurer, "Analytical approach to parallel repetition," in *Proceedings ACM STOC 2014*.

[31] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, 2018.

[32] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations research*, vol. 8, no. 1, pp. 101–111, 1960.

[33] V. Chvatal, V. Chvatal *et al.*, *Linear programming*. Macmillan, 1983.

[34] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*. Springer Science & Business Media, 2006, vol. 5.

[35] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck, "Automation and combination of linear-programming based stabilization techniques in column generation," *INFORMS Journal on Computing*, 2018.

[36] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik*, vol. 4, no. 1, 1962.

[37] A. M. Costa, J.-F. Cordeau, and B. Gendron, "Benders, metric and cutset inequalities for multicommodity capacitated network design," *Computational Optimization and Applications*, vol. 42, no. 3, pp. 371–392, 2009.

[38] J. Håstad, "Some optimal inapproximability results," *Journal of the ACM (JACM)*, vol. 48, no. 4, pp. 798–859, 2001.

[39] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks*, vol. 55, no. 3, 2010.

[40] B. M. Waxman, "Routing of multipoint connections," *IEEE journal on selected areas in communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[41] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, pp. 756–767, 2002.

[42] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010.

[43] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceedings of IEEE WoWMoM 2014*.

[44] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *J. Netw. Comput. Appl.*, vol. 103, no. C, pp. 101–118, Feb. 2018.