Availability-Aware Routing in Presence of Geographically Correlated Failures

Balázs Vass, Levente Birszki, Erika Bérczi-Kovács, Péter Babarczi, Péter Gyimesi, János Tapolcai

***Abstract*—Survivable routing is crucial in backbone networks to ensure connectivity, even during failures, including natural and man-made disasters. Service-level agreements (SLAs) often define availability targets in terms of "number of nines" (e.g.,** 99.99%, 99.999%**), and network operators aim to meet or exceed these targets by provisioning redundant paths between source-destination pairs. Nevertheless, ensuring extremely high availabilities comes with extra cost in terms of total network bandwidth usage. This study formalizes the problem of availability-aware routing based on risk zone failure probabilities. On the downside, we find the problem is $\mathcal{NP}$-hard, with minimizing unavailabilities being even inapproximable in polynomial time (supposing $\mathcal{P} \neq \mathcal{NP}$). Motivated by the hardness results, we initiate the study of heuristics providing availability-aware routing considering regional failures. Our methods are suitable for providing a range of routing plans, navigating the trade-off between availability and bandwidth usage. At the core of the heuristics, we rely on an efficiently solvable problem variant, where failure probabilities of risk zones are changed to capacities. We validate our findings through extensive simulations based on real-world input data. With reasonably low bandwidth usage, our heuristic is close to a lower bound on unavailability, the median and average gap being** 1.23% **and** 2.21%**, respectively.**

I. Introduction

Ensuring high availability in backbone networks is a fundamental requirement for mission-critical applications. In practice, network operators often define availability targets in terms of "number of nines" (e.g., 99.9%, 99.999%) in their Service-Level Agreements (SLAs), and aim to meet or exceed these targets by provisioning redundant paths between source-destination pairs [1]. Traditionally, this redundancy has been achieved by computing multiple link disjoint paths under the assumption that failures are independent. For instance, a common rule of thumb is that a single path provides 99% availability, while two link disjoint paths, e.g., with $1+1$ dedicated protection [2] yield 99.99%, allowing less than an hour downtime per year for the service.

However, recent insights into the nature of network failures challenge the assumptions underlying this practice [3], [4], revealing that multiple failures often exhibit spatial correlation: network components that fail together are frequently in close physical proximity [5]. This is especially true in the case of regional failures caused by natural disasters [6], [7], such as earthquakes, hurricanes [8] or floods, where multiple links or devices in the same geographic area can simultaneously become unavailable [4]. Therefore, the geographic layout of the paths plays a critical role: two physically distant *regionally disjoint* paths may provide better protection than three in the same area with shared risks. As a result, simply increasing the number of link disjoint paths does not guarantee a proportional increase in availability, thus, the design objective of redundant paths need to shift towards availability from their number.

Several approaches have been proposed to model the spatial correlation of multiple network failures as Shared Risk Link Groups (SRLGs) [9], using e.g., geographic [10] disaster models, but finding SRLG disjoint paths to maximize availability is a computationally hard problem [11], [12], only solvable efficiently for special cases [13]. Furthermore, owing to the uncertain nature of disasters, the introduction of stochastic failure models [14] led to the concept of Probabilistic SRLGs (PSRLGs) [15], which consist of a list of edge sets paired with probabilities, where each entry denotes that the corresponding set of edges may jointly fail with the associated likelihood. A key challenge is that even small networks can produce PSRLG lists with hundreds of entries, for which no efficient routing algorithm exists to compute paths meeting the specified availability targets declared in the SLA.

In this paper, we address this issue by proposing an algorithmic framework for *availability-aware routing* in presence of regionally correlated failures. The output of our algorithm determines how many paths are needed, where they should be routed, and how their spatial layout affects the overall availability (see Fig. 1). The framework builds on recent advances in efficient algorithms to compute regionally disjoint paths using SRLG models [13], which we generalize to PSRLGs. Moreover, we extend the original model to (1) support *directed graphs*, and (2) to allow each PSRLG to be assigned a *capacity*, i.e., a limit on how many paths may traverse it. This capacity reflects the risk level or criticality of a PSRLG: higher-risk zones should be traversed by fewer paths. Our simulations demonstrate that incorporating regional failure

Balázs Vass, Péter Babarczi, and János Tapolcai are with the Dept. of Telecommunications and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary, and HUN-REN-BME Information Systems Research Group. Balázs Vass's main affiliation is Faculty of Mathematics and Computer Science, Babeș-Bolyai University of Cluj Napoca, Romania. E-mails: balazs.vass@ubbcluj.ro, {babarczi,tapolcai}@tmit.bme.hu.

Levente Birszki, Erika Bérczi-Kovács and Péter Gyimesi are with Department of Operations Research, ELTE Eötvös Loránd University, Budapest, Hungary. Erika Bérczi-Kovács is also with HUN-REN–ELTE Egerváry Research Group on Combinatorial Optimization. Contact them on {frocor,peti1234}@student.elte.hu, and erika.berczi-kovacs@ttk.elte.hu.

This research was supported by the National Research Excellence Programme of the Hungarian National Research, Development and Innovation Office under funding schemes ELTE TKP 2021-NKTA-62, and STARTING_25 ID: STARTING 153002, and Project no. K_23 146347, respectively. This study has received funding from the European Union's Horizon Europe research and innovation programme.

Co-funded by the European Union

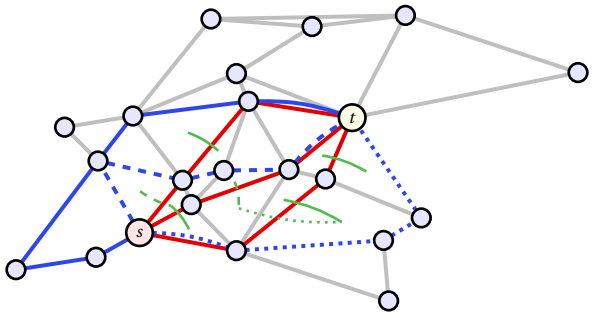


Fig. 1. 2+1 protection using shortest node-disjoint paths (in solid red) compared to using our paths (in blue solid, dashed, and dotted, resp.) on network Optic EU [18]. Although the paths in our solution are longer, and they are not even node-disjoint, under the next earthquake, ours' availability is above five-nines, while the shortest paths fall well below this threshold. This is partially because our paths avoid the 7 risk zones with the highest failure probabilities (depicted in green). Seismic hazard input taken from [15].

awareness into backbone network design leads to significantly more resilient and cost-efficient provisioning strategies [16].

Ultimately, we envision a future where backbone operators embrace such models to build smarter, availability-aware infrastructures that adapt gracefully to large-scale disruptions. We make the following contributions towards this goal:

- We present the first scalable routing algorithm for backbone networks that can operate effectively on PSRLG lists of realistic size.
- We prove that the underlying algorithmic problem – Availability-Aware Routing Based on Risk Zone Failure Probabilities (Problem 1) – is $\mathcal{NP}$-hard.
- The core of our basic heuristic algorithm is an efficient polynomial-time procedure that solves the Capacitated Region Routing (CRR) problem (Problem 2). It generalizes the algorithm of [13], [17] by allowing each PSRLG to have an associated capacity, which limits the maximum number of paths that can traverse its edges.
- Advanced heuristic algorithms are designed that optimize PSRLG capacity assignments to minimize total bandwidth consumption, subject to availability constraints on end-to-end connections.

The paper is organized as follows. §II formalizes the problem and our main findings. Next, §III investigates the algorithmic challenges of the availability-aware routing. An efficient algorithm for a slightly altered problem variant is presented in §IV. Using this subroutine, §V builds efficient heuristics for solving the original availability-aware routing problem. Then, §VI and §VII discuss some routing considerations and related works. Numerical evaluations are presented in §VIII, and finally, §IX concludes our paper.

II. Problem Formulation and Main Results

A. Inputs and Problem Formulation

The problem input consists of three parts. The first part of the input is a connected directed graph $G=(V,A)$ with vertex set V ($|V|\geq 2$) and arc set A. In this paper, sometimes we refer to arcs as *links*. We store the incident arcs for every node given in clockwise order, called a rotation system [19]. We say a subset of arcs $R\subseteq A$ is a **risk zone** if a connected geographic area a hits all links in R, i.e., every arc in R has a common point with the interior of a. The set of risk zones is denoted by $\mathscr{R}$, and we suppose the empty set (denoting the failure-free state after a disaster) is part of $\mathscr{R}$.

The other part of the problem input encodes the joint failure probabilities of link sets (i.e., risk zones). For this, for a link set $R\in\mathscr{R}$, in line with [15], [20], we define $\mathrm{FP}(R)$ ('link failure state probability of R') to denote the probability that *exactly* link set R will fail at the next disaster. Note that $\sum_{R\in\mathscr{R}}\mathrm{FP}(R)=1$, as the next disaster corresponds to exactly one risk zone. Also, while the input focuses only on common failures of *links*, if necessary, these structures can store failure probabilities of both links *and* node failures (see [15, §V.]).

The last part of the input stands for a node pair $\{s,t\}\subseteq V$, a positive integer k, and thresholds $1\geq T_0\geq T_1\geq T_2\geq\ldots,\geq T_k\geq 0$, as the problem to be defined will ask for an arbitrary number of $l\in\{1,\ldots,k\}$ st-paths such that the probability of failure of at most i of these paths should be at most T_i, for all $i\in\{0,\ldots,k\}$. Equivalently, at least $l-i$ paths should survive with probability $\geq 1-T_i$. We will refer to bounds T_i as *availability thresholds*.

Given a set of paths $P_1,\ldots,P_l$, we can define unavailabilities (that are just 1 minus the availability values) $U_0\geq U_1\geq\cdots\geq U_k$, denoting that at least $1,\ldots,k$ paths fail due to the next disaster, in the following way:

$$U_i=\sum_{R\in\mathscr{R}}\mathrm{FP}(R)\cdot I\left(i,\sum_{j=1}^{l}\texttt{intersects}(R,P_j)\right),$$

where indicator function $I(i,b)$ equals 1 if $i\leq b$, and is 0 otherwise. Here, function `intersects` indicates if a path was affected after the failure of a risk zone:

$$\texttt{intersects}(R,P_j)=\begin{cases}1 & \text{if } R\cap P_j\neq\emptyset,\\ 0 & \text{otherwise.}\end{cases}$$

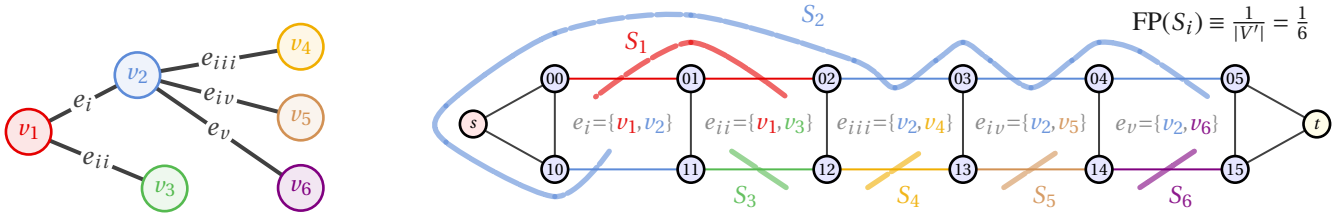
Based on the above, we can formally define the *Availability-Aware Routing Based on Risk Zone Failure Probabilities* problem tackled in this paper as Problem 1.

Problem 1: Availability-Aware Routing Based on Risk Zone Failure Probabilities

Input: A directed plane graph $G=(V,A)$, nodes $s,t\in V$, risk zones $\mathscr{R}\subset 2^E$, $\mathrm{FP}():\mathscr{R}\to(0,1]$ risk zone failure probabilities, availability thresholds $T_1,\ldots,T_k\in[0,1]$, where k is the maximal number of desired paths

Output: st-paths $P_1,\ldots,P_{l\leq k}$ such that availability thresholds $T_1,\ldots,T_k$ are met

Examples. Suppose the operator intends to have a set of l paths (where this number l is not greater than k), with the failure of a given $\lambda\in\{0,\ldots,l-1\}$ paths being tolerable. In the SLA, there is a prescribed threshold T of availability. Then, the question is how to set thresholds $T_1,\ldots,T_k$ to phrase the above task as a Problem 1 instance, as all the rest of the problem input is given by the circumstances. The most admissive setting for this is $T_{i\in\{1,\ldots,l-\lambda\}}:=1-T$, $T_{i\in\{k-l+1,\ldots,k\}}:=0$. Note that the problem formulation would admit more sophisticated settings of the thresholds. This could be important e.g., if



(a) A `Vertex Cover` instance f.

(b) Problem 1 instance created based on `Vertex Cover` instance f.

Fig. 2. Example instances of a reduction of Problem 1 to the `Vertex Cover` problem. There is an st-path of unavailability of at most U_1 if there is a vertex cover in the instance depicted in Subfig. (a) using a fraction of at most U_1 of the vertices.

the operator wanted to respect an additional threshold $T' > T$ denoting the requirement for at most 1 path failing; here $T_1 := T'$ should be applied.

Remark. In the formulation of Problem 1, we have omitted the natural objective of minimizing the bandwidth usage (i.e., 'cost') among the feasible solutions. This is because this overhead is dependent on the survivable routing approach used by the operator. Heuristics taking in consideration the minimization of the bandwidth usage will be described in §V-A, while coding considerations are detailed in §VI.

B. Main Results

We will see in Claim 3 that evaluating whether a set of paths is a feasible solution to Problem 1 can be easily done in polynomial time. In other words, Problem 1 is in $\mathcal{NP}$. However, it is $\mathcal{NP}$-hard even for a single path:

Theorem 1. *It is $\mathcal{NP}$-hard to decide whether a solution of Problem 1 exists, even for $k = 1$.*

Furthermore, the following Claim formalizes that the optimization version of Problem 1 is already hard to approximate.

Claim 2. *Using the input of Problem 1 with $k = 1$, the optimal unavailability U_1 of a single st-path P is $\mathcal{NP}$-hard to approximate within a factor of ~ 1.36, and thus does not admit a Polynomial Time Approximation Scheme (PTAS) (unless $\mathcal{P}$=$\mathcal{NP}$).*

The proofs of Thm. 1, and Claims 3 and 2 are given in §III.

Our final main result is an efficient heuristic that, based on simulation results, often matches a lower bound on unavailability, with median and average gaps of approximately 1.23% and 2.21%, respectively.

In the heart of this heuristic approach for solving Problem 1, intuitively, we will substitute the failure probabilities of risk zones with capacities on them.

III. Computing Availability-aware Routing based on Risk Zone Failure Probabilities

Fortunately, evaluating if a solution to Problem 1 is feasible is algorithmically easy:

Claim 3. *Suppose we have a collection C of a number of k st-paths. Let $\mathscr{R}^i_C \subseteq \mathscr{R}$ stand for those risk zones that intersect exactly i paths from C. Then, the probability A_i of at least i paths surviving the next disaster is $\sum_{j=0}^{k-i} \sum_{S \in \mathscr{R}^j_C} FP(S)$. Since this sum is polynomially computable, Problem 1 is in $\mathcal{NP}$.*

Proof: The proof of the claim is immediate from the definition of sets of risk zones $\mathscr{R}^i_C$. Note that sets $\mathscr{R}^i_C$ ($i \in \{0,\dots,k\}$) partition the set of risk zones $\mathscr{R}$. ∎

Sadly, the quick positive results end here, as finding a feasible solution (or proving its inexistence) is algorithmically hard, and is proved in the following.

A. Proof of the $\mathcal{NP}$-hardness of Problem 1

Proof of Thm. 1: We will provide a reduction from the $\mathcal{NP}$-complete `Vertex Cover` [21] to Problem 1. In `Vertex Cover`, the input is a graph $G' = (V', E')$ and the problem is to decide whether there exists a b-node cover of the edges, i.e., is there a subset $V'' \subseteq V'$ of cardinality $|V''| \leq b$ such that each edge in E' has at least one endpoint in V''.

Suppose we are given a `Vertex Cover` problem instance F. Suppose the nodes and edges of the instance are given as lists $V' = \{v_1,\dots,v_{|V'|}\}$ and $E' = \{e_1,\dots,e_{|E'|}\}$, respectively. Suppose each edge $e = \{v_i, v_j\} \in E'$ is given such that $i \leq j$. In the following, we describe a function f, where $f(F)$ will be an instance of Problem 1 that has an input of size $O(|V'|+|E'|)$, and that can be computed from F in $O(|V'|+|E'|)$. For the Problem 1 instance, for simplicity, we describe the construction of an undirected graph $G = (V, E)$. From edge set E, arc set A is constructed by simply directing each edge $e \in E$ in both directions. Let the set of nodes be $V := \{v_{i,j} \mid i \in \{0,1\}, j \in \{0,\dots,|E'|\}\} \cup \{s,t\}$. The set of edges is $E := \{\{v_{0,j}, v_{1,j}\} \mid j \in \{0,\dots,|E'|\}\} \cup \{\{v_{i,j-1}, v_{i,j}\} \mid i \in \{0,1\}, j \in \{1,\dots,|E'|\}\} \cup \{\{s, v_{0,1}\}, \{s, v_{1,1}\}, \{v_{0,|E'|}, t\}, \{v_{1,|E'|}, t\}\}$. Intuitively, G can be drawn as a $2 \times (|E'|+1)$ grid graph, with s and t appended to the nodes on the first and last columns, as depicted in Fig. 2. As we will see, the i^{th} column of the grid graph will roughly correspond to the i^{th} edge $e_i \in E'$. If G is drawn this way, intuitively, the links that may fail at the next disaster are exactly the horizontal ones, as explained in the following.

For each node u in V', there will be assigned exactly one risk zone $R_u \in \mathscr{R}$. For each $i \in \{0,1\}$ and $j \in \{1,\dots,|E'|\}$, $\{v_{i,j-1}, v_{i,j}\} \in R_u$ exactly if u is incident to edge $e_j \in E'$, and u is the first or second node in the description of e_i (for $i = 0$ and 1, respectively.) For each node $u \in V'$, we set the failure probability of $\mathrm{FP}(R_u)$ uniformly to $1/|V'|$.

Note that all the above link sets with $\mathrm{FP}(S) > 0$ are proper *risk zones*, i.e., each of them can be covered by a connected

geographic area (see Fig. 2b for an example). This is easy to verify, since each edge that may fail bounds the infinite face. With this, the description of function f is complete.

Now we describe function $g(P)$ that maps a given st-path P being a solution of problem instance $f(F)$ to a solution of the original `Vertex Cover` instance F. Given a risk zone R_u, if $P \cap R_u \neq \emptyset$, we add $u \in V$ to the covering vertex set.

We can observe that the unavailability of P can be optimized to be at most $b/|V'|$ only if the `Vertex Cover` instance F has a vertex cover of size at most b. ∎

B. Further Notes on the Hardness: Inapproximability

Proof of Claim 2: Observe that in the above construction, the unavailability U_1 of the single st-path is just $b/|E'|$, that is just a scaled-down version of the `Vertex Cover` parameter b, where the scaling factor is just the number of edges. This means that all the hardness results for the minimization version of `Vertex Cover` carry over to minimizing the unavailability of a single path in Problem 1. Thus, the proof of Claim 2 is immediate from [22, Theorem 1.1], that states the following: *"Given a graph G, it is $\mathcal{NP}$-hard to approximate the Minimum Vertex Cover to within any factor smaller than* $10\sqrt{5}-21 = 1.3606\ldots$ *."* ∎

Note that the construction for the hardness results did not rely on the fact that the input graph $G = (V, A)$ is directed. Thus, the $\mathcal{NP}$-hardness and inapproximability results hold in the undirected version of Problem 1 too.

IV. Routing with respect to capacitated risk zones

In previous sections, we have seen that directly solving Problem 1 is algorithmically very challenging. Thus, in this section, we present a similar problem (Problem 2) searching for a given number of $l \in \{2,\ldots,k\}$ st-paths that can be solved efficiently. The aim is to heuristically recycle the solution of Problem 2 as a possible solution to Problem 1. Heuristics on tackling Problem 1 will be described in §V.

A. The Capacitated Risk Zone Routing (CRR) Problem

Taking one step closer, the main idea is to switch from probabilities to *capacities*. Intuitively, if a risk zone R has a high probability $\mathrm{FP}(R)$ of failing, then most likely, it is not a good idea to let many paths go through it. That is, this risk zone R should have a 'low' **capacity** $\mathrm{cap}(R)$, where function cap() determines the maximum number of st-paths of the solution that are allowed to cross each risk zone. Observe that if we had allowed all l paths to cross a risk zone R, this risk zone would cease to exist as a constraint to the solutions. This way, we can formalize the capacity function as $\mathrm{cap}() : \mathscr{R} \to \{0,\ldots,l-1,\infty\}$, where ∞ is set as the capacity of all the non-constraining risk zones. Note that each risk zone R separating s from t should have a capacity of ∞ (i.e., it has to be able to host all the l paths), otherwise no solution exists.

To make the above problem efficiently solvable, we need an extra assumption: paths of the solution should be **non-crossing**. Here, two paths are non-crossing if, after contracting their common edges, there is no node where the edges of the paths are alternating; l paths are non-crossing if they are pairwise non-crossing. Non-crossingness is mostly a technical assumption, as in plane graphs, if two paths are node-disjoint, they are also non-crossing. With the above modifications applied to Problem 1, we can formalise Problem 2 as follows.

Problem 2: Capacitated Risk Zone Routing (CRR)

Input: Directed plane graph $G = (V, A)$, nodes $s, t \in V$, risk zones $\mathscr{R} \subset 2^E$, $\mathrm{cap}() : \mathscr{R} \to \{0,\ldots,l-1,\infty\}$ capacity function on the risk zones, number of desired paths $l \geq 2$.

Output: non-crossing st-paths $P_1,\ldots,P_l$ such that for every risk zone $R \in \mathscr{R}$ is intersected by at most $\mathrm{cap}(S)$ paths if they exist; or proof of inexistence.

The following claim gives a bound on the performance of a Problem 2 solution recycled for Problem 1.

Claim 4. *Given a Problem 2 input, for $j \in \{0,\ldots,k-1\}$, let $\mathscr{R}_j$ denote the set of risk zones $R \in \mathscr{R}$ for which $\mathrm{cap}(R) = j$. Then, we have the following upper bound on the probability of at most $i \geq 1$ paths failing: $U_i \leq \sum_{j=1}^{i} j \cdot \sum_{R \in \mathscr{R}_j} \mathrm{FP}(R)$.*

Proof: By definition, each risk zone R can hit at most $\mathrm{cap}(R)$ paths in the worst case. That is, in the worst case, concentrating solely on risk zone R, a number of $\mathrm{cap}(R)$ paths can be hit, with a probability of $\mathrm{FP}(R)$. The proof of the claim is imminent after adding up these worst-case values. ∎

B. Algorithmic Considerations for Solving the CRR Problem

For the undirected unit capacity case of Problem 2, an efficient polynomial-time algorithm called *DateLine* relying on a min-max theorem was given in [17]. Later, [13] generalized the results to directed graphs. In a nutshell, a minor tweak to the DateLine algorithm makes it suitable for solving the CRR problem in unchanged time complexity. We will call the resulting method the **CRR algorithm**. The main idea of the algorithm is that the existence of a feasible solution to the CRR Problem is equivalent to the non-existence of a negative cycle in an auxiliary weighted graph $D^* = (V^*, A^*)$ defined in the following (note that its arc set is different from the arc set of the usual dual graph). Here, V^* is just the set of faces of planar graph G, and and the arcs are derived from $\mathscr{R}$: for every risk zone $R \in \mathscr{R}$, we add a complete directed graph on the faces neighboring the edges of R to A^*. Parallel arcs in the auxiliary graph are possible. If an arc a was added due to risk zone R, we may indicate it with adding a subscript $_R$ to a, i.e., $a = a_R$. The function describing arc weights $c_l() : A^* \to \mathbb{Z}$ can be intuitively depicted as follows. First, we fix a directed st-path P. For an arc a_R, let $\xi(a_R)$ denote the number a curve lying inside risk zone R running from the tail to the head of a_R crosses P right-to-left minus the number it crosses P left-to-right. Then, $c_l(a_R) = \mathrm{cap}(R) + \xi(a_R) \cdot l$. Specifically, if an a_R does not cross P, $c_l(a_R) = \mathrm{cap}(R)$, if it crosses P from left to right, $c_l(a)$ will be $\mathrm{cap}(R) - l$, and in case of a right-to-left crossing, $c_l(a)$ is $\mathrm{cap}(R) + l$. To prevent a path using each arc $(a, b) \in A$ backwards, we add an extra arc (r^*, l^*)

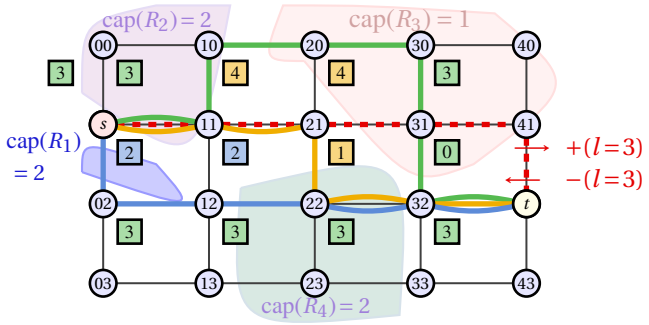


Fig. 3. A function π on the faces encoding $l = 3$ st-paths in a Capacitated Risk Zone Routing (CRR) problem instance.

to A^*, where r^* and l^* are the faces neighboring (a,b) on the right and left, respectively. If (a,b) is not on path P, is a forward arc, or a backward arc on P, then $\mathrm{cap}((r^*,l^*))$ equals $0, l$, and $-l$, respectively. Note that a more formal definition of c_l would exceed the limits of this paper; however, the cost function c_l in this paper is almost exactly the same as the one formally defined in [13], [17], with the only difference that there, instead of the capacity-dependent term $\mathrm{cap}(R)$, there was constant 1 (referring to the unit capacity).

If there is no negative cycle in D^* according to c_l, we can compute a function on the faces $\pi : V^* \to \mathbb{Z}$ such that $c_l(uv) + \pi(u) - \pi(v) \geq 0$ for all $(u,v) \in A^*$. Such a function π can be computed e.g. by choosing any face $v^* \in V^*$ of and by setting $\pi(w^*) := \mathtt{dist}_{c_l}(v^*, w^*)$ for each $w^* \in V^*$. These distances can be computed using the Bellman-Ford (B-F) algorithm [23] or, e.g., [24]–[26]. Then, a corresponding arc set F can be created, which describes the required paths $P_1, \ldots, P_l$. Intuitively, the boundaries between the $\bmod l$ classes of faces of G according to π determine l paths forming a feasible solution of Problem 2 (as depicted on Fig. 3).

On the other hand, if there is a negative cycle C' according to c_l in D^*, it is a witness for the non-existence of a solution in the Problem 2 instance. E.g., decreasing $\mathrm{cap}(R_4)$ to 1 would induce a negative cycle. Conform to this, no $l = 3$ paths could cross the third column of horizontal edges. Again, such a negative cycle can be found using the B-F algorithm or similar.

The formal proof of correctness of the CRR algorithm formulated in Problem 2 is predominantly analogous to the DateLine's; thus, we refer the interested reader to [13], [17]. The minor but important additional observation in our case is that since $\mathrm{cap}(R) < l$ for each $R \in \mathscr{R}$ that does not separate s from t, and paths $P_1, \ldots, P_l$ are non-crossing, there is no tuple R, P_i such that R would not separate s from t, but $R \cup P_i$ did so. Intuitively, no path crosses a risk zone twice, thus the output is correct. Also, as far as l is $O(|V|^2)$ (that is a very practical assumption), all complexity considerations remain the same for the CRR algorithm as those presented in [17].

In conclusion, the CRR Problem can be solved efficiently. Thus, based on Claim 4, a time-efficient family of heuristic strategies for tackling Problem 1 is to solve appropriate CRR problem instances by experimenting with the capacities of the risk zones, aiming to meet the availability thresholds defined in Problem 1. Such heuristics will be described in §V.

V. Heuristics for Solving the Availability-Aware Routing Problem

In this section, we describe heuristics for solving Problem 1. Even with the subroutine solving CRR problem instances in hand, providing feasible solutions to Problem 1 is a challenging task. Furthermore, as it will be presented in §VIII, empirically, in some settings, solutions with fewer paths may perform better in terms of availability, and can also yield arguably good availability-bandwidth usage trade-off. Thus, our strategy is to generate multiple possible solutions to the problem, with possibly fewer paths than k. More precisely, for each Problem 1 instance, we generate a number of $k-1$ different (but somewhat interconnected) collections of paths consisting of $l = 2, 3, \ldots, k$ paths, respectively.

A. The Basic Heuristics

In the following, we describe our heuristics relying on solving a sequence of CRR Problem instances for solving Problem 1 instances. We suppose, in continuation, that the maximal number of paths k we are searching for is at least 2. To initialize a CRR problem instance, we take from the inputs of Problem 1 graph G, nodes s, t, and risk zones $\mathscr{R}$, then, we initialize the capacity function as follows. First, let us denote by $\mathscr{R}^0$ the set of risk zones $R \in \mathscr{R}$ that do not separate s from t. For each $R \in \mathscr{R}^0$, we set the capacity uniformly 1; and we set $\mathrm{cap}(R)$ to ∞ for all $R \in \mathscr{R} \setminus \mathscr{R}^0$.

First, we are looking for a solution with $l = 2$ paths. Later, the value of l will be increased one by one until it reaches k. Regarding the CRR instance we got, there are two possibilities:

1) We are given a negative cycle C as a witness of infeasibility: Each arc of C corresponds to a risk zone in a set $\mathscr{R}_{\text{cutting}}$ of risk zones that make the CRR problem infeasible. To reach a CRR instance that has a feasible solution, the capacity of at least one $R \in \mathscr{R}_{\text{cutting}}$ has to be increased. For this, heuristically, we choose an $R \in \mathscr{R}_{\text{cutting}}$ with minimal $\mathrm{cap}(R) \cdot \mathrm{FP}(R)$ value. Recall that due to the inner workings of the CRR algorithm, when searching for l paths, it is important that all the finite risk zone capacities are within the range of $\{0, \ldots, l-1\}$. Thus, if the capacity of a risk zone R reached l, we set $\mathrm{cap}(R) := \infty$. With the increased capacities, we retry.

2) There is a feasible solution $P_1^l, \ldots, P_l^l$ of l paths: In this case, we return with them. If $l < k$, the value of l is increased by 1. For risk zones $R \in \mathscr{R}^0$ with $\mathrm{cap}(R) = \infty$, we set $\mathrm{cap}(R)$ to the number of paths that cross R in the actual solution (thus, for each $R \in \mathscr{R}^0$, $\mathrm{cap}(R) \leq l-1$). The CRR algorithm is re-run.

B. Shortening the Resulting Paths

The CRR algorithm on its own is insensitive to the length of the resulting paths. In most settings, however, extreme stretches compared to the length of a shortest st-path should be avoided. Thus, in this subsection, we describe three slightly interconnected strategies for optimizing path lengths.

Algorithm 1: Strategy **S1** (Making room, then use **S0**)

Input: A directed plane graph $G_c = (V, A_c)$, nodes $s, t \in V$, risk zones $\mathscr{R} \subset 2^E$, $\mathrm{FP}() : \mathscr{R} \to (0,1]$ risk zone failure probabilities, number of desired paths l, capacities $\mathrm{cap}() : \mathscr{R} \to \{0, \ldots, l-1, \infty\}$, feasible set of paths $P_1^{l,0}, \ldots, P_l^{l,0}$
Output: Set of shortened paths $P_1^l, \ldots, P_l^l$; updated capacities cap()

$i := 0$
while $\max_{j=1}^{l} \log\left(U_j^{l,0}\right) \geq \log\left(U_j^{l,i}\right) \cdot 101\%$ **do**
  increase $\mathrm{cap}(R)$ for $R = \arg\min_{R \in \mathscr{R}} \mathrm{cap}(R) \cdot \mathrm{FP}(R)$
  $i := i + 1$
  run CRR algorithm $\longrightarrow$ path set $\left\{P_1^{l,i}, \ldots, P_l^{l,i}\right\}$
$i := i - 1$ // revert the last capacity increase to likely be within violation bound
with paths $P_1^{l,i}, \ldots, P_l^{l,i}$, shorten by **S0** $\longrightarrow$ path set $\left\{P_1^l, \ldots, P_l^l\right\}$
for all $R \in \mathscr{R}^0$, set $\mathrm{cap}(R)$ to be tight w.r.t. $\left\{P_1^l, \ldots, P_l^l\right\}$
return *path set* $\left\{P_1^l, \ldots, P_l^l\right\}$, *and capacities cap()*

Algorithm 2: Strategy **S2** (Trimming)

Input: A directed plane graph $G = (V, A)$, nodes $s, t \in V$, constant $c \in \mathbb{N}^+$
Output: Trimmed graph $G_c(V, A_c)$

Fix a shortest st-path P
$A(G_c) := \{a \in A | \mathtt{dist}(P, a) \leq c\}$
return $G_c(V, A_c)$

Strategy S0: *Postprocessing an existing solution:* This strategy takes a set of l paths as input, fixes $l-1$ of them, deletes the arcs of risk zones R that are full (i.e., a number of $\mathrm{cap}(R)$ of the $l-1$ fixed paths are already going through them), and searches for a shortest path in the remaining graph. Note that such a path always exists. The heuristic stops when there is no path out of the current collection that could be shortened. It is easy to see that this heuristic is finite. In our implementation, we prioritized the shortening of the current longest, second longest, etc. paths. This strategy resembles a standard technique in the literature [17], [27]–[29].

Strategy S1: *Providing room then use* S0*:* The main problem with the above-presented shortening heuristic is that if there are many risk zones R in the CRR problem instance that are **tight** in the sense that the number of paths in the solution that cross R is just $\mathrm{cap}(R)$. Intuitively, this is a problem for shortening the path lengths because tight risk zones severely restrict the room of movement of the actual path that is being recomputed in the hope of being shortened. This means that the more successful the basic heuristic was in only increasing the capacities where needed (i.e., keeping risk zones tight) the smaller this possible room of movement of the path is.

To navigate this anticipated trade-off between the availability scores and total path lengths of the resulting paths, we propose the following. First, we search for a feasible CRR problem instance with the above-described capacity increasing strategy. Then, based on the yielded paths $P_1, \ldots, P_l$ and risk zone failure probabilities, we evaluate unavailabilities $U_i^{l,0}$ denoting the probability of at most i of them failing ($i \in \{0, \ldots, l\}$). Then, first, by possibly reducing some of the risk zone capacities, we set the capacities of the risk zones $R \in \mathscr{R}^0$ to be tight with respect to $P_1, \ldots, P_l$. Second, we have

Algorithm 3: Heuristic for tackling Problem 1

Input: A directed plane graph $G = (V, A)$, nodes $s, t \in V$, risk zones $\mathscr{R} \subset 2^E$, $\mathrm{FP}() : \mathscr{R} \to (0,1]$ risk zone failure probabilities, maximum number of desired paths k, constant $c \in \mathbb{N}^+$ for strategy S2.
Output: A collection of $k-1$ path sets with reasonably high availabilities, consisting of $2, \ldots, k$ paths

run strategy **S2**$\longrightarrow$ record graph $G_c(V, A_c)$
$\mathrm{cap}(R) := 1, \forall R \in \mathscr{R}$
for $l = 2..k$ **do**
  run CRR algorithm // described in §IV-B
  while *Output of CRR is negative cycle C* **do**
    increase $\mathrm{cap}(R)$ for for $R = \arg\min_{R \in \mathscr{R}_{\text{cutting}}} \mathrm{cap}(R) \cdot \mathrm{FP}(R)$
    run CRR algorithm
  for all $R \in \mathscr{R}^0$: set $\mathrm{cap}(R)$ to be tight w.r.t. output of CRR
  run strategy **S1**$\longrightarrow$ record its output as `Paths`$[l]$ and modified cap()
return `Paths`

to slightly change the capacity-increasing strategy, as, having a feasible problem instance, we do not have a negative cycle anymore (that would be a proof of inexistence). Thus, we choose an R now from the whole risk zone set $\mathscr{R}$ with minimal $\mathrm{cap}(R) \cdot \mathrm{FP}(R)$. With this updated rule, until the availabilities do not degrade 'too much', we repeat increasing the risk zone capacities, solving the resulting problem instance, and evaluating unavailabilities $U_j^{l,i}$ of the current solution, where $U_j^{l,i}$ denotes the probability that, in the i^{th} iteration, no more than j paths of the actual solution will fail. Here, our strategy is to stop manipulating the risk zone capacities when, for some i, after the i^{th} capacity increasement, for some path index j, $\log(U_j^{l,i})$ exceeded $\log(U_i^{l,0}) \cdot 101\%$, that is when the logarithm of one of the unavailability scores got more than 1% worse compared to the availability-wise conservative baseline solution.[1] Then, we run the shortening heuristic **S0** on the last CRR instance not violating the allowed degradation in terms of unavailability. Strategy **S1** is formalized in Alg. 1.

Strategy S2: *Trimming remote parts of G:* Even with the above-described relaxation method, one might judge that the resulting paths are still too long, thus too costly. A more aggressive way of constraining the path lengths is by banning the paths from using parts of the graph, that are 'far away' from a shortest st-path. Intuitively, e.g., making a connection more reliable in the contiguous US through Japan might be an overkill. After choosing a shortest st-path P in G, we can define graphs $G_0, G_1, \ldots$ with arc sets $A_0, A_1, \ldots$, respectively, where arc set A_i contains those arcs that are no further than in an i-hop distance from P. Note that for a sufficiently large i, A_i is just the arc set A. While trimming the input graph might be an efficient tool in constraining the lengths of the paths in the solution, its drawback is a possibly sharp decrease in availability compared to the baselines, where all the arcs on the original graph G could be used. Strategy **S2** is summarized in Alg. 2. Note that for simplicity, we suppose functions FP() and cap() still use graph G (i.e., they are not restricted to G_c).

The full-blown heuristic is summarized in Alg. 3.

[1] A larger room for unavailability degradation could be also used as a trade-off yielding possibly shorter routes, but in our experience, 101% proved to be a reasonable sweet spot. Upcoming strategy S2 can take care of path lengths.

VI. Survivable Routing Along the Paths and Bandwidth Usage

To tailor availability-aware paths to the characteristics of the provided service, the network operator can tune the level of failure resilience by selecting different survivable routing techniques to transfer data along the l st-path found by our heuristics for Problem 1. The simplest proactive method is to send the same copy of data along the paths for maximum survivability like dedicated protection [2]. Although the connection survives failures along $\leq l-1$ st-paths, it uses at least l-times the links (bandwidth) of the shortest path (original data).

Instead of duplication, in diversity coding [16] the original data is split into $l-1$ parts sent along $l-1$ different st-paths, while the l^{th} path is used to transfer redundancy data by using the eXclusive OR (XOR) operation of the $l-1$ original data parts [2]. Thus, no packet retransmission is required if one of the path fails, as the target node t can reconstruct the sent information by simply XOR-ing the data of the surviving $l-1$ paths (if needed). Diversity coding can be generalized to provide higher redundancy by using $l-\lambda$ paths for sending the original data parts while utilizing λ paths to provide redundancy. With this approach the connections survive λ path failures, but instead of XOR coding linear operations over finite fields with $2^{O(\log l)}$ size required [16], which might lead to more complex network operation than using efficient XOR coding implementations, e.g., in optical networks [30].

Claim 5. *Suppose sending a unit capacity data flow over a single data link uses a unit of bandwidth. Then, sending a unit amount of data using l paths and surviving the failure of a number of λ paths can be done using an amount of $\frac{L}{l-\lambda}$ bandwidth, where L is the total length of the paths.*

Proof: Based on the results discussed in this section, sending a unit amount of data using l paths and surviving the failure of a number of λ paths can be done via sending an amount of $1/(l-\lambda)$ data over each path. ∎

VII. Related Works

Works in the field of risk zone-disjoint routing: Risk zones are just special Shared Risk Link Groups (SRLGs). In general, many of the questions related to SRLG-disjoint routes are $\mathcal{NP}$-hard. The $\mathcal{NP}$-hardness of some problem variants was investigated in [31]–[34]. Some polynomially solvable cases were investigated in [32], [33]. ILP and Mixed ILP solutions of problem variants were given in [35]–[38]. Heuristics were also proposed [39], [40], bearing issues like possibly non-polynomial runtime or possibly arising forwarding loops. Notably, based on a probabilistic SRLG model defined in the paper, exploratory study [14] aims to find a pair of paths with low joint failure probability via an ILP, using synthetic data. For a comprehensive guide in modeling geographically correlated failures in backbone networks, we refer to [41].

Background for the CRR algorithm: Papers [13], [17], [27]–[29], [42]–[44] constitute a chain of studies on risk zone disjoint routing problems, that gradually laid the theoretical foundations for the solution of the Capacitated Risk Zone Routing (CRR) Problem. The CRR algorithm of our paper relies on a black-box algorithm that is suitable for computing shortest paths in possibly negatively weighted graphs if there is no negative cycle, and can return a negative cycle if there exists one. The Bellman-Ford (B-F) algorithm [23] suits these tasks well. Our simulations used a heuristic speedup of the B-F [24]. Papers [25] and [26] are alternatives with attractive deterministic and randomized time complexities, respectively.

VIII. Evaluation results

In this section, we present our numerical experiments demonstrating the effectiveness of the proposed heuristics, using real-world problem inputs. The algorithms were implemented in Python and C++, using various libraries. The implementation can be accessed through a public repository (see §IX). Runtimes were measured on a commodity laptop with a CPU at 1.8 GHz and 16 GB of RAM. In the simulation results, instead of concrete availability values, we present the survival/failure probabilities of the network connections *when the next disaster hits*. Based on failure probability F, it is straightforward to compute the unavailability of the connection caused by the disasters, since the yearly downtime can be estimated as F times the rate of the earthquakes considered (e.g., ~ 5.3 per year for Italy) times the Mean Time To Repair (MTTR) of the connection. The MTTR is highly dependent on the operator, thus we leave it open to be chosen arbitrarily.

A. Input data

To conduct our experiments on tuples of real-world backbone topologies and related risk zones and their failure probabilities, we decided to reach out for the data produced in [15]. Paper [15], based on a careful processing of the available real seismic hazard data, produced risk zones and related failure probabilities for a number of 7 network topologies, taken from [18], [45]. Out of these, 3 topologies are North American, 3 of them span Europe, and there is a national network from Italy. The first three columns of Table I contain the name, node, and edge count of these topologies. Column 4 depicts the number of risk zones at *intensity tolerance* equaling VI according to the MCS scale [46]. Vaguely, this means that we suppose the network equipment may fail in case of a 'strong' shaking, with peak ground accelerations exceeding around 1.2 $[\mathrm{m/s^2}]$.

B. Baseline Approaches and Lower Bounds

1) Naive baselines: To provide baseline data points, we computed $l = 1,2,\ldots$, up to the maximal number of node-disjoint st-paths. To do so, we computed a minimum-cost interiorly node-disjoint st-flow of size $l = 1,2\ldots$ with uniform arc costs. Note that for l equalling 1 and 2, respectively, this approach outputs the same paths as the fast Dijkstra [47] and Suurballe [48] algorithms, respectively. We will refer to this heuristic simply as ***Shortest***.

2) Traditional approach for more reliable paths: As we have seen in Thm. 1, Claim 2, in our setting, it is computationally hard to find an st-path that maximizes the availability. For finding a reasonably safe st-path, a folklore technique also described in [49] starts with unrealistically assuming

TABLE I
THE INVESTIGATED INPUT SETTINGS. COMPARISON OF $\text{ARZP}^2_{\text{ADV}}$ TO SHORTEST^2 AND UNAVAILABILITY LOWER BOUND. RUNTIMES.

Network name	$\lvert V\rvert$	$\lvert E\rvert$	# risk zones at VI (MCS scale)	Our $\text{ARZP}^2_{\text{adv}}$ vs. baseline Shortest^2, out of 50 st-pair per graph								$\text{ARZP}^2_{\text{adv}}$ unavailability over u_{lb} in [%]				avg. runtime of ARZP_{adv} per st-pair [sec]
				[%] st-pairs vs. Shortest^2				decrease in unav.		increase in bandwidth						
				better	m. rel.	shorter	same	avg %	safest %	avg. %	safest in %	min	max	avg	median	
Optic EU	22	45	202	4	52	80	44	16.03	67.44	14.35	14.29	0	13.8	2.5	1.4	150
Italian	25	34	308	6	56	76	38	18.67	72.88	23.52	50	0	10.3	1.8	0.7	201
US	26	43	246	0	54	84	38	10.27	74.77	14.92	42.86	0	13.5	2.7	1.9	172
Nobel EU	28	41	149	2	46	94	50	5.53	27.32	7.27	22.22	0	10.5	1.7	0.7	157
EU	37	57	212	2	40	94	50	5.12	24	4.75	22.22	0	25	2.5	0.6	156
N.-American	39	61	394	2	60	84	32	10.94	91.47	9.81	0	0	8.8	2.4	1.6	219
NFSNET	79	108	969	10	52	84	26	15.52	93.59	9.04	4.35	0	13	1.9	1.7	441

that link failures are independent. For each link $a \in A$, we will assume that the probability that a fails is less than 1, i.e., $0 \le \sum_{R\in\mathscr{R}:a\in R} \text{FP}(R) < 1$. By assuming the independence of the link failures, the availability of an st-path P can be expressed as $A_{\text{indep}}(P) = \Pi_{a\in P}(1 - \sum_{R\in\mathscr{R}:a\in R}\text{FP}(R))$. Since the logarithm is strictly monotone, the maximum of the logarithm of the original expression is reached on the same paths that also maximize $A_{\text{indep}}()$. The logarithm of a product is the sum of the logarithms, thus, after multiplying with -1, it is enough to minimize $\sum_{a\in P} w_a$, where weight w_a is set to $-\log(1 - \sum_{R\in\mathscr{R}:a\in R}\text{FP}(R))$, for each arc $a \in A$. This can be found by e.g., the Dijkstra algorithm [47], since the weights are nonnegative. To provide a similar, more refined heuristic for $l = 2,3,\ldots$ paths, we searched for a min-cost node-disjoint st-flow of size $l = 2,3,\ldots$ with the same weights w_a. Collectively, we refer to this heuristic family as ***Independent***.

3) Lower bounds on unavailability and bandwidth: A simple lower bound on the achievable minimal unavailability is yielded by simply summing up the FP(R) for risk zones $R \in \mathscr{R}\setminus\mathscr{R}^0$, i.e., the failure probabilities of regions that separate s from t: $u_{\text{lb}} = \sum_{R\in\mathscr{R}\setminus\mathscr{R}^0}\text{FP}(R)$.

Recall that in our model, sending a unit amount of data over a data link translates of a unit of bandwidth. Thus, the length $b_{\text{lb}} := \texttt{dist}(s,t)$ of a shortest st-path (in terms of hop count) is the minimum amount of bandwidth unit needed.

We will use the following terminology for comparing two solutions of the problem S and T. That is, we say: 1) S is **more reliable** than T if S has a lower unavailability than T, 2) S is **shorter** than T if S uses less bandwidth than T, and finally 3) S is **better** than T if it is both more reliable and shorter than T. Specifically, no solution can be either more reliable than u_{lb} or shorter than b_{lb}.

C. Survivable Routing Along the Paths

We investigated different diversity coding based survivable routing approaches on the l paths by using $\lambda = 1,2$ and 3 out of them as backup, shown in Fig. 4 for a selected st-pair in the Optic EU network. Here, for solving the problem of **A**vailability-Aware **R**outing Based on Risk **Z**one Failure **P**robabilities, we used our heuristic Alg. 3 without any graph trimming (i.e., setting $c = \infty$ for S2, or equivalently, omitting line 1), referred hereafter as the $\textbf{ARZP}_{\textbf{adv}}$ method. One can observe that for lower l values, each method reached the maximum achievable availability value. This is because by increasing the number of paths, the paths explore larger parts of the network, which makes them prone to more failures, thus the failure probability increases as well. Furthermore, for each l, $\lambda = 1$ backup path has the lowest bandwidth consumption among the three routing approaches (with the bandwidth usage growing as λ grows). The above is positive news as the approach using $\lambda = 1$ backup has an efficient XOR coding implementation for most network technologies [16]. Therefore, we provide our results with $\lambda = 1$ backup path in the rest of the paper as the most efficient survivable routing on our paths.

D. Comparison of the Basic and Advanced Heuristics

In Fig. 5 we compared the performance of the ARZP_{adv} heuristic to the plain $\textbf{ARZP}_{\textbf{basic}}$ heuristic (as described in §V-A) and the baselines, and investigated the effect of our path shortening strategies on the bandwidth. From now on, for each algorithm ALG tackling Problem 1, we will refer to its l-path output as $\texttt{ALG}^l$. Note that here we suppose the solution uses $\lambda = 1$ backup path if $l \ge 2$, while for $l = 1$, it can use none. Concentrating on Fig. 5, one can see that closely optimal availabilities are reached with low bandwidth by $\text{ARZP}^2_{\text{adv}}$ and $\text{ARZP}^3_{\text{adv}}$. Observe that these bandwidths

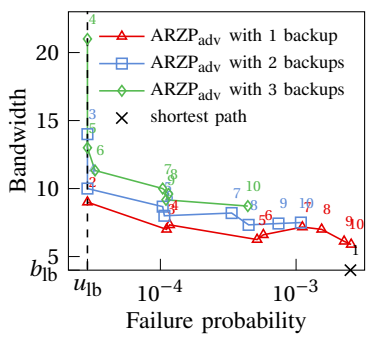


Fig. 4. Failure probability vs. bandwidth usage of path sets by ARZP_{adv} up to 10 paths with $\lambda = 1,2,3$ backups.

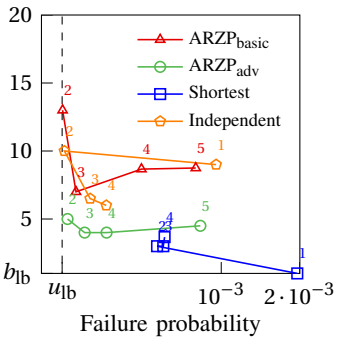


Fig. 5. $\text{ARZP}_{\text{basic}}$ versus ARZP_{adv} compared to the baselines for a node pair in Optic EU.

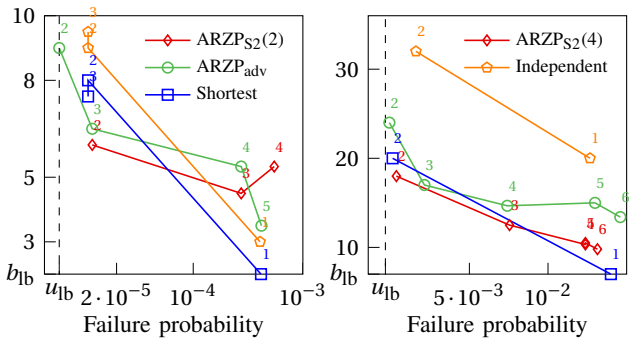


Fig. 6. Effect of graph trimming in hope of lower bandwith. On the left and right: results for selected st-pairs taken from Optic EU and NFSNET, respectively.

not exceeding 5 were achieved as the cost of sacrificing some availability while significantly shortening the solutions of $\text{ARZP}_{\text{basic}}$. In fact, the most reliable solutions in decreasing order are $\text{ARZP}^2_{\text{basic}}$, Independent^2 and $\text{ARZP}^2_{\text{adv}}$, with slight degradation of availability, but massive savings in bandwidth, In fact, for $l \geq 2$, $\text{ARZP}^{\text{l}}_{\text{adv}}$ has comparable bandwidth consumption to heuristic Shortest^l, while approaching the failure probability of Independent^l paths. As a takeaway, both Fig. 4 and Fig. 5 suggest that the path pair provided by $\text{ARZP}^2_{\text{adv}}$ is an attractive choice for survivable routing, its most direct competitors being Shortest^2 and Independent^2. We will further analyze the outputs of ARZP_{adv} in the next subsections.

E. Comparison of the Advanced Heuristic to the Baselines

Columns 5-12 of Table I depict our simulation results comparing $\text{ARZP}^2_{\text{adv}}$ to Shortest^2, run on 50 uniformly random chosen terminal pairs in each network topology. Columns 5-8 compare the quality of the solutions. If the path pair provided by $\text{ARZP}^2_{\text{adv}}$ was better or the same as Shortest^2, we counted the respective terminal pair in both col. 5 and 8, or only in col. 8, respectively. If there was no *better* solution provided by $\text{ARZP}^2_{\text{adv}}$, then if there was more reliable and shorter, we counted the terminal pair in col. 6 and 7, respectively. In the majority of the settings, $\text{ARZP}^2_{\text{adv}}$ provided shorter, more reliable, or even better solutions. Columns 9 and 10 represent the average and maximal decrease in unavailability per topology using the $\text{ARZP}^2_{\text{adv}}$ solution over the Shortest^2. Note this decrease can be significant, in some cases being over 90%. Columns 11 and 12 depict the ratio of the extra bandwidth of $\text{ARZP}^2_{\text{adv}}$; usually being modest, it never surpassed 50%.

Table II depicts the metrics discussed above, but here the baseline was substituted with the Independent^2, that is, a naive heuristic intended to minimize unavailability. Note that in some cases, $\text{ARZP}^2_{\text{adv}}$ still cuts down almost 3/4 of the baseline unavailability while using typically slightly less bandwidth.

F. Comparison to the Unavailability Lower Bound

Returning to Table I, its last columns (except the very last) depict the extra unavailability incurred by $\text{ARZP}^2_{\text{adv}}$ compared to the lower bound on the unavailability (yielded by the risk zones separating s and t). The results are convincing: many times, $\text{ARZP}^2_{\text{adv}}$ matches the lower bound, with the median and average gap being around 1.23% and 2.21%, respectively. The gap never exceeded 25%.

G. Improving Path Length with Trimming in S2

Finally, Fig. 6 showcases the effect of graph trimming with strategy S2 to limit bandwidth usage more aggressively.

TABLE II
SAME SETTING AS TABLE I, BUT INDEPENDENT^2 AS BASELINE.

Network name	[%] st-pairs Adv. vs. Ind.				decr. in unav.		inc. in bandwidth	
	better	m. rel.	shorter	same	avg%	safest%	avg.%	safest%
Optic EU	30	26	62	10	11.43	49.14	−13.41	50.00
Italian	0	62	82	36	10.96	60.74	19.59	50.00
US	28	28	68	2	7.85	74.77	−8.87	−28.57
Nobel EU	26	26	72	28	6.63	27.16	−3.12	8.33
EU	42	32	56	12	8.00	28.65	−9.52	−30.77
N.-Am.	34	12	64	12	6.62	74.77	−18.35	−13.33
NFSNET	22	8	78	10	3.31	51.73	−14.62	−8.33

We will call the full-blown version of Alg. 3 using S2 with constant c as $\text{ARZP}_{\text{S2}}(c)$ (that leaves only a shortest path P and arcs no further from P than c hops in the graph). We conducted simulations on the 22-node Optic EU, and the 79-node NFSNET networks, where the trimmed graph G_c equals to the original G for $c \geq 4$ and $c \geq 18$, respectively. Results are shown on the left and on the right of Fig. 6, respectively.

In case of Optic EU, one can observe that while producing similar unavailability, $\text{ARZP}^2_{\text{S2}}(2)$ lowers the bandwidth consumption compared to $\text{ARZP}^3_{\text{adv}}$. In NFSNET, being a larger network, we found that with a slight increase in c to compared to the EU topology (i.e., larger detours are allowed compared to the demonstrated setting on Optic EU), $\text{ARZP}_{\text{S2}}(4)$ produces potentially reasonable availability-bandwidth trade-offs. Note that this solution is also shorter than the solutions of the Shortest - this can happen since the paths in our solutions do not have to be node, or even edge disjoint. A remark to be made here is that in both the presented cases, more trimming would yield significantly higher unavailabilities, being closer to the unavailability of a shortest path.

H. Runtime analysis

The last column of Table I presents the average runtime of the ARZP_{adv} heuristic per network topology, where we searched for at most $k = 10$ paths. Our simulation was not particularly optimized for speed. Only the CRR algorithm was implemented in C++, which was repeatedly called from a Python framework. The average runtime per terminal pair was between 150 and 441 `[sec]` for all networks.

IX. CONCLUSION

In this paper, we introduced a routing algorithm for backbone networks that accounts for correlated link failures, a problem we termed **A**vailability-Aware **R**outing Based on Risk **Z**one Failure **P**robabilities (**ARZP**). This formulation incorporates the observed spatial correlations in link failures, such as those caused by natural disasters affecting geographically co-located links. We first proved that the ARZP problem is $\mathcal{NP}$-hard. To address this, we proposed an efficient heuristic algorithm whose core is a polynomial-time method for computing SRLG-disjoint paths in planar graphs, where each SRLG is assigned a capacity limiting the number of traversing paths. Our ARZP_{adv} algorithm scales to real-world network sizes and computes routing plans that meet target availability thresholds. Simulation results confirm that achieving high end-to-end availability accomplished with multiple node-disjoint paths can be suboptimal not only in terms of availability, but can use significantly more bandwidth than necessary. Instead, routing over multiple partially overlapping paths yields higher availabilities – on average within ∼2.2% of the theoretical bound – while reaching attractively low total bandwidth demand. Surprisingly, even for the widely deployed 1+1 protection scheme, the paths found by our method ($\text{ARZP}^2_{\text{adv}}$) are substantially more efficient alternatives, both in terms of availability and resource usage. The authors have provided public access to their code and data at https://github.com/beer-sky/availability-aware-routing.

REFERENCES

[1] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, and M. Jaeger, "General availability model for multilayer transport networks," in *Proc. Design of Reliable Communication Networks (DRCN)*, Lacco Ameno, Italy, 2005.

[2] A. Pašić, P. Babarczi, J. Tapolcai, E. R. Bérczi-Kovács, Z. Király, and L. Rónyai, "Minimum cost survivable routing algorithms for generalized diversity coding," *IEEE/ACM Trans. Netw.*, 2020.

[3] B. Mukherjee, M. Habib, and F. Dikbiyik, "Network adaptability from disaster disruptions and cascading failures," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 230–238, 2014.

[4] J. Rak, R. Girão-Silva, T. Gomes, G. Ellinas, B. Kantarci, and M. Tornatore, "Disaster resilience of optical networks: State of the art, challenges, and opportunities," *Optical Switching and Networking*, 2021.

[5] P. K. Agarwal, A. Efrat, S. K. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of WDM networks to probabilistic geographical failures," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, 2013.

[6] J. Oostenbrink and F. Kuipers, "Computing the impact of disasters on networks," *ACM SIGMETRICS Performance Evaluation Review*, 2017.

[7] F. Dikbiyik, M. Tornatore, and B. Mukherjee, "Minimizing the risk from disaster failures in optical backbone networks," *J. Lightw. Technol.*, vol. 32, no. 18, pp. 3175–3183, 2014.

[8] J. Heidemann, L. Quan, and Y. Pradkin, *A preliminary analysis of network outages during hurricane Sandy*. University of Southern California, Information Sciences Institute, 2012.

[9] G. Ellinas, E. Bouillet, R. Ramamurthy, J.-F. Labourdette, S. Chaudhuri, and K. Bala, "Routing and restoration architectures in mesh optical networks," *Optical Networks Magazine*, vol. 4, no. 1, pp. 91–106, January/February 2003.

[10] H. Saito, "Analysis of geometric disaster evaluation model for physical networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1777–1789, 2015.

[11] J. Liu, J. Zhang, Y. Zhao, C. Ma, H. Yang, W. Li, J. Xin, and B. Chen, "Differentiated quality-of-protection provisioning with probabilistic SRLG in flexi-grid optical networks," in *OSA Asia Communications and Photonics Conference*, 2013, pp. AF2G–8.

[12] S. Trajanovski, F. A. Kuipers, A. Ilić, J. Crowcroft, and P. Van Mieghem, "Finding critical regions and region-disjoint paths in a network," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 908–921, 2015.

[13] E. Bérczi-Kovács, P. Gyimesi, B. Vass, and J. Tapolcai, "DateLine: efficient algorithm for computing region disjoint paths in backbone networks," *IEEE J. on Selected Areas in Communications*, pp. 1–14, Jan. 2025.

[14] H.-W. Lee, E. Modiano, and K. Lee, "Diverse routing in networks with probabilistic failures," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, 2010.

[15] B. Vass, J. Tapolcai, Z. Heszberger, J. Bíró, D. Hay, F. A. Kuipers, J. Oostenbrink, A. Valentini, and L. Rónyai, "Probabilistic shared risk link groups modelling correlated resource failures caused by disasters," *IEEE J. on Selected Areas in Communications*, 2021, Data produced available at https://github.com/jtapolcai/regional-srlg/blob/master/psrlg/JSACdata.zip.

[16] E. Ayanoglu, C.-L. I, R. Gitlin, and J. Mazo, "Diversity coding for transparent self-healing and fault-tolerant communication networks," *IEEE Transactions on Communications*, vol. 41, no. 11, 1993.

[17] E. Bérczi-Kovács, P. Gyimesi, B. Vass, and J. Tapolcai, "Efficient algorithm for region-disjoint survivable routing in backbone networks," in *IEEE INFOCOM*, May 2024.

[18] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[19] J. L. Gross and S. R. Alpert, "The topological theory of current graphs," *Journal of Combinatorial Theory, Series B*, vol. 17, no. 3, pp. 218–233, 1974. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0095895674900288

[20] A. Pašić, R. G. ao Silva, F. Mogyorósi, B. Vass, T. Gomes, P. Babarczi, P. Revisnyei, J. Tapolcai, and J. Rak, "eFRADIR: An Enhanced FRAmework for DIsaster Resilience," *IEEE Access*, vol. 9, pp. 13 125–13 148, 2021. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9319646

[21] M. R. Garey and D. S. Johnson, *Computers and intractability*. San Francisco, Calif.: W. H. Freeman and Co., 1979, a guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[22] I. Dinur and S. Safra, "On the hardness of approximating minimum vertex cover," *Annals of mathematics*, pp. 439–485, 2005.

[23] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[24] F. Duan, "A faster algorithm for shortest path-spfa," *Journal of Southwest Jiaotong University*, vol. 29, no. 6, pp. 207–212, 1994.

[25] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for network problems," *SIAM Journal on Computing*, vol. 18, no. 5, 1989.

[26] A. Bernstein, D. Nanongkai, and C. Wulff-Nilsen, "Negative-weight single-source shortest paths in near-linear time," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022.

[27] Y. Kobayashi and K. Otsuki, "Max-flow min-cut theorem and faster algorithms in a circular disk failure model," in *IEEE INFOCOM*, April 2014.

[28] K. Otsuki, Y. Kobayashi, and K. Murota, "Improved max-flow min-cut algorithms in a circular disk failure model with application to a road network," *European Journal of Operational Research*, vol. 248, 2016.

[29] B. Vass, E. Bérczi-Kovács, A. Barabás, Z. L. Hajdú, and J. Tapolcai, "Polynomial-time algorithm for the regional SRLG-disjoint paths problem," in *IEEE INFOCOM*, London, United Kingdom, May 2022.

[30] H. Zhang, B. Wang, and S. Wang, "A novel method of high-speed all-optical logic gate based on metalens," *Optics Communications*, 2025.

[31] J.-Q. Hu, "Diverse routing in optical mesh networks," *IEEE Trans. Communications*, vol. 51, pp. 489–494, 2003.

[32] J.-C. Bermond, D. Coudert, G. D'Angelo, and F. Z. Moataz, "SRLG-diverse routing with the star property," in *Proc. Design of Reliable Communication Networks (DRCN)*. IEEE, 2013, pp. 163–170.

[33] ——, "Finding disjoint paths in networks with star shared risk link groups," *Theoretical Computer Science*, vol. 579, pp. 74–87, 2015.

[34] X. Luo and B. Wang, "Diverse routing in WDM optical networks with shared risk link group (SLRG) failures," in *Proc. Design of Reliable Communication Networks (DRCN)*, Oct. 16-19 2005.

[35] D. Xu, G. Li, B. Ramamurthy, A. Chiu, D. Wang, and R. Doverspike, "SRLG-diverse routing of multiple circuits in a heterogeneous optical transport network," in *Proc. Design of Reliable Communication Networks (DRCN)*, 2011, pp. 180–187.

[36] T. Gomes, M. Soares, J. Craveirinha, P. Melo, L. Jorge, V. Mirones, and A. Brízido, "Two heuristics for calculating a shared risk link group disjoint set of paths of min-sum cost," *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 1067–1103, 2015.

[37] A. de Sousa, D. Santos, and P. Monteiro, "Determination of the minimum cost pair of D-geodiverse paths," in *Proc. Design of Reliable Communication Networks (DRCN)*, Munich, Germany, March 8-10 2017.

[38] R. Girão-Silva, B. Nedic, M. Gunkel, and T. Gomes, "Shared Risk Link Group disjointness and geodiverse routing: A trade-off between benefit and practical effort," *Networks*, vol. 75, no. 4, pp. 374–391, 2020.

[39] K. Xie, H. Tao, X. Wang, G. Xie, J. Wen, J. Cao, and Z. Qin, "Divide and conquer for fast SRLG disjoint routing," in *IEEE/IFIP Dependable Systems and Networks (DSN)*, 2018, pp. 622–633.

[40] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep forwarding: Towards k-link failure resilient routing," in *IEEE INFOCOM*, 2014.

[41] 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.

[42] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1610–1623, 2011.

[43] S. Neumayer, A. Efrat, and E. Modiano, "Geographic max-flow and min-cut under a circular disk failure model," in *IEEE INFOCOM*, 2012.

[44] B. Vass, E. Bérczi-Kovács, A. Barabás, Z. L. Hajdú, and J. Tapolcai, "A whirling dervish: Polynomial-time algorithm for the regional srlg-disjoint paths problem," *IEEE/ACM Trans. Netw.*, 2023.

[45] A. Valentini, B. Vass, J. Oostenbrink, L. Csák, F. Kuipers, B. Pace, D. Hay, and J. Tapolcai, "Network resiliency against earthquakes," in *Resilient Networks Design and Modeling (RNDM)*, Oct 2019, pp. 1–7.

[46] A. Sieberg, "Erdebeben," *Handbuch der Geophysic*, vol. 4, 1931.

[47] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.

[48] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, 1974.

[49] Stack Overflow user. (2020) Answer explaining how to find a most reliable path with Dijkstra's algorithm. Accessed: 2025. [Online]. Available: https://stackoverflow.com/questions/64991937/find-the-most-reliable-path-dijkstra-s-algorithm