The Hazard Value: A Quantitative Network Connectivity Measure Accounting for Failures

Pieter Cuijpers, Stefan Schmid, Nicolas Schnepf, Jiří Srba

The Hazard Value: A Quantitative Network Connectivity Measure Accounting for Failures

Pieter Cuijpers*†, Stefan Schmid‡, Nicolas Schnepf†§ and Jiří Srba†
*Eindhoven University of Technology, Netherlands
†Aalborg University, Denmark
‡University of Vienna, Austria
§Université de Lorraine, CNRS Inria, Loria, France

***Abstract*—To meet their stringent requirements in terms of performance and dependability, communication networks should be "well connected". While classic connectivity measures typically revolve around topological properties, e.g., related to cuts, these measures may not reflect well the degree to which a network is actually dependable. We introduce a more refined measure for network connectivity, the *hazard value*, which is developed to meet the needs of a real network operator. It accounts for crucial aspects affecting the dependability experienced in practice, including actual traffic patterns, distribution of failure probabilities, routing constraints, and alternatives for services with preferences therein. We analytically show that the hazard value fulfills several fundamental desirable properties that make it suitable for comparing different network topologies with one another, and for reasoning about how to efficiently enhance the robustness of a given network. We also present an optimised algorithm to compute the hazard value and an experimental evaluation against networks from the Internet Topology Zoo and classical datacenter topologies, such as fat trees and BCubes. This evaluation shows that the algorithm computes the hazard value within minutes for realistic networks, making it practically usable for network designers.**

***Index Terms*—network, fault-tolerance, routing, resilience, metric**

I. Introduction

Communication networks have become a critical infrastructure of our digital society, as also highlighted during the ongoing pandemic. In order to meet the resulting stringent dependability and performance requirements, networks should be "well connected". However, defining the notion of connectivity is challenging: while classic measures, related to cuts [22], number of disjoint paths [3], or expansion [1], [30], provide interesting insights into the topological robustness of a network, they do not account for several additional aspects which matter in practice, and their usefulness may hence be limited in specific scenarios.

We identify the following aspects of availability and dependability to be of relevance to network service providers, and aim to develop a more general notion of connectivity that takes these aspects into account:

- *Traffic patterns.* Traffic patterns over a large network are often skewed in practice. Certain endpoint pairs may need to communicate much more frequently, or more reliably, than others. For example, in datacenters, traffic matrices are often sparse [4], which implies that not all endpoint pairs are equally important from a connectivity point of view.
- *Distribution of failure probabilities.* With the increasing scale of communication networks, failures are becoming more likely [12]. These failures may either occur randomly or depend on each other, e.g., in shared risk link groups [26]. A practical connectivity measure should hence account for link failures and reflect the likelihood of corresponding failure scenarios and their effect on the connectivity within the network.
- *Routing constraints.* Routing paths in networks are often constrained, for example due to network policies and business considerations [18], or due to the type of routing mechanism that is being used. These constraints may limit the connectivity within a network, even though the underling physical network may be highly connected.
- *Alternatives for service and preferences therein.* Networks often come with choice. First, services are typically offered at multiple places, which is for example leveraged by DNS anycast; other examples include key-value stores which allow for replica selection and content distribution applications [21], [28]. Redundancy may also be offered in terms of different routes from a given entry point to a given exit point, in terms of alternative entry and exit points, or because a client requesting a traffic flow is connected to multiple nodes. Given such alternatives, network operators and their clients may have a preference for one alternative over another. Using a different alternative may influence the value of the service.

A. Operator's Distributed Datacenter

As a use case and running example throughout this paper, we consider a geographically distributed datacenter which we obtained from our collaboration with a network operator, see Figure 1. The network consists of two sites, where each site relies on a 3-level topology. The two sites (visually separated via the dashed vertical line) are connected via two wide-area links (using a Layer-2 network, so services/VMs can be migrated transparently). In the lowest level of the left site are the leaf switches l_i, in the middle level the spine switches s_i, and at the top the datacenter edge router c (henceforth simply called core switch); these switches are connected to the corresponding switches l'_i, s'_i, c' on the right

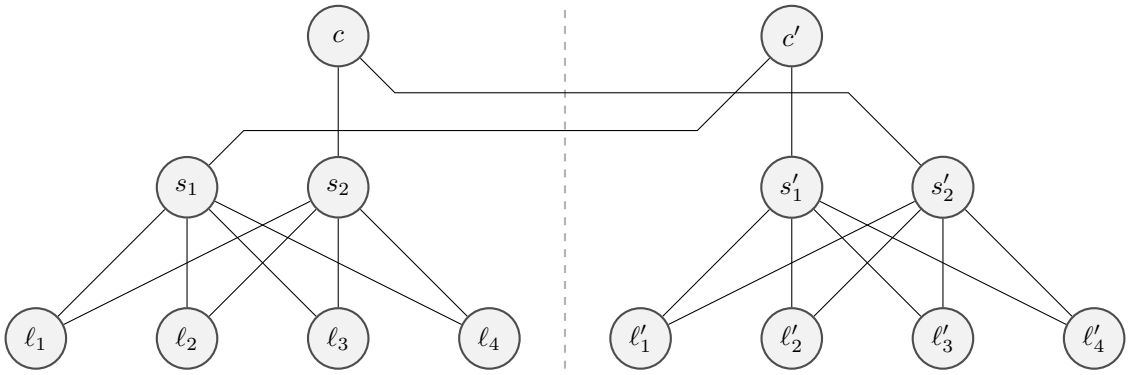

Figure 1: Use case: geographically distributed datacenter network (two sites)

site. Traffic is usually either *datacenter internal*, i.e., originates and terminates at the leaf switches (where the servers are attached), or *external*, arriving/leaving through one of the core switches from/to the Internet.

The operator is interested in knowing how failures can affect the services provided by the network. This depends on a number of parameters. First, connectivity is restricted due to routing constraints: a typical routing constraint in networks is to ensure *valley freedom*, i.e., traffic inside the Layer-2 network could stay on the leaf switch, or be switched on the spine switch; a packet between two leaf switches will first only travel up the network, up to a certain level, and afterwards only travel down towards the destination. Second, the possibility of *alternative routes* enhances the dependability of a network. In our example, a leaf may route traffic toward the Internet through either of the two core switches; hence, it is sufficient if one of the two core switches is available. However, network operators may specify *preferences* among alternatives; for example, it may be preferable to route traffic through core switch c instead of c' because it is physically closer, because it provides a higher bandwidth, because it is part of a data replication group, or because of legislation or financial concerns. The preference among alternatives can also be dependent on the *demand* and actual amount of traffic arising between endpoints. If there is a significant demand for traffic between leaf pairs on the left site, but hardly any traffic in the right site, a link failure in the left site can have more severe consequences than in the right site.

B. Our Contributions

We introduce a novel connectivity metric, the *hazard value*, to assess the dependability of a network accounting for all the above properties. As an input, this metric takes a description of the network (a directed graph), a description of the routing constraints (a regular language over sequences of links in the graph), a probability distribution over possible link failure scenarios, and a family of service weight functions that model the preferences between routing alternatives mentioned earlier. As an output, the hazard value returns the expected percentage of the total service weight that is lost due to connectivity problems arising from node or link failures and routing constraints.

A network operator can use the hazard value to compare different options to fortify a network. By comparing the hazard value for different network topologies (e.g. adding redundant links or nodes), routing strategies (e.g. enforcing valley freedom or not), and measures that influence failure probabilities (e.g. choosing more reliable hardware), one can make quantitative statements about how much a certain fortification is expected to improve the dependability of a network.

In order to verify that the hazard value provides a suitable metric for this purpose, we start out by proving a number of desirable mathematical properties:

- it is a *topological* notion, meaning that isomorphic graphs have equal hazard value;
- it is a *compositional* notion, in that the hazard value for a given set of service weights is the same as the weighted sum of the hazard values for each of the individual service weights;
- it is *monotonically* decreasing when introducing additional links, and monotonically increasing when tightening the routing restrictions (adding link redundancy or removing routing restrictions decreases the hazard);
- it is *not strictly monotone*, as the removal of links or nodes that are not used to achieve any of the serviced connections, does not affect the hazard value;
- it increases when independent sources of failure are added to the failure model;
- and finally, the hazard value is a generalization of traditional connectivity, as assuming a non-zero service weight between all pairs of nodes, and assuming that there is no probability of failure, leads to a hazard value of 0 if and only if the network is totally connected.

As a second step in verifying the usefulness of our metric, we perform a number of experiments. We present an efficient way to compute the hazard value by restricting the enumeration of failure scenarios to only those cuts that can disconnect maximally rewarded pairs of nodes, and considering subsets of those cuts for refinement of the approximation. Even though we show that already deciding whether the hazard value is equal to 0 is (in the worst-case) NP-hard, our algorithmic approach renders the hazard value computationally feasible for realistic scenarios from the Internet Topology Zoo

database [19] as well as some classical datacenter topologies including fat-tree [6] and BCube [14]. The performance is particularly good if we consider failure probability distributions in which all link failures are independent and the maximum number of link failures is restricted. However, our theory is developed for any probabilistic distribution of failures that includes node failures, shared-risk link groups as well as chained failures.

II. Preliminaries

We view a network as a directed (multi-)graph, in which the vertices represent nodes of the network and edges represent links from one node to another.

Definition 1 (Network)**.** *A* network *is a directed (multi)-graph $G = \langle V, E, src, dst\rangle$ consisting of a finite set of* nodes *(*vertices*) V, a finite set of* links *(*edges*) E, a source mapping function $src : E \mapsto V$ and a target mapping function $dst : E \mapsto V$.*

We regard a network as a service that provides connectivity between nodes to tenants (henceforth simply called customers) by choosing a path from some node s (the source), to a node t (the target).

Definition 2 (Path)**.** *Given a network $G = \langle V, E, src, dst\rangle$, a* path *is a sequence of links $\sigma = \sigma_1\sigma_2\cdots\sigma_n \in E^+$ where $|\sigma| = n$ such that $dst(\sigma_i) = src(\sigma_{i+1})$ for all i, $1 \leq i < |\sigma|$. The set of all paths of a graph is denoted by $Paths$.*

- *A path σ* starts *in node $s \in V$ if $src(\sigma_1) = s$ and it* ends *in node $t \in V$ if $dst(\sigma_{|\sigma|}) = t$. We write $Paths(s, t)$ for the set of all paths that start in s and end in t.*
- *A path σ* traverses *a link e, written as $e \in \sigma$, if there is i, $1 \leq i \leq |\sigma|$, such that $\sigma_i = e$.*
- *A path σ* visits *a node v if there is an i, $1 \leq i \leq |\sigma|$ such that $src(\sigma_i) = v$ or $dst(\sigma_i) = v$.*
- *A path σ is* simple *if it visits any node $v \in V$ at most once.*

In a network with routing policies, the number of paths that are available for connecting two nodes is often restricted by the routing mechanisms, and perhaps by other routing policies as well. Instead of describing the routing mechanisms that are used in a network explicitly, we abstract from the precise technology and assume the possible routes through a network are given simply as a subset of all paths.

Definition 3 (Routing language)**.** *Given a* routing language *$R \subseteq E^+$, i.e. a set of allowed routes, we say an entry point v can be routed to an exit point w if there exists a path $\sigma \in R \cap Paths$ from v to w.*

When we consider the actual computation of the hazard value, we usually find that realistic routing scenarios can be captured as *regular expressions* over E. For example, to describe that all routes in a network must traverse some link e, we can write $R = E^* \cdot e \cdot E^*$. Similarly, to describe that all routes in a network must traverse some node n (e.g. a firewall or other waypoint), we can write $R = E^* \cdot \left(\sum_{n=dest(e)} e\right) \cdot E^*$. In our experimental section, we consider connectivity in the topology of a large set of real-world networks (the Topology Zoo [19]), to which we apply additional constraints of 0, 1 or 2 waypoints (with 0 waypoints corresponding to pure connectivity). We also consider service chaining in those networks, where two given nodes must be visited in a predefined order. All such path constraints can be defined using regular expressions.

Definition 4 (Failures)**.** *Given a set $F \subseteq E$ of failed links we write $Paths^F$ for the maximum set of paths that do not traverse any edge from F.*

We treat failures in a probabilistic fashion, assuming that there is a distribution function that determines the likelyhood of a certain set of failures occurring at the same time.

Definition 5 (Distribution of failure probabilities (DFP))**.** *A* distribution of failure probabilities or DFP *is a probability distribution $\phi : 2^E \to [0, 1]$ that assigns a likelihood $\phi(F)$ to each possible set of failed edges $F \subseteq E$, such that $\sum_{F\subseteq E} \phi(F) = 1$.*

In our experiments, we assume that all link failures are independent and have a probability $p = 0.01$ of occurring. This gives rise to the distribution

$$\phi_p(F) = p^{|F|}(1-p)^{(|E|-|F|)}$$

where $|H|$ denotes the number of links in a set $H \subseteq E$. However, the theory is developed for arbitrary probability distributions. Furthermore, we consider the independent failure of links under the condition that at most k failures occur in total, which (according to Bayes' theorem) is given by $\phi_p^k(F) = 0$ when $|F| > k$ and

$$\phi_p^k(F) = \frac{p^{|F|}(1-p)^{(|E|-|F|)}}{\sum_{|H|\leq k} p^{|H|}(1-p)^{(|E|-|H|)}}$$

when $|F| \leq k$.

In Section IV we discuss how independent causes of failures, each with their own distribution, can be convoluted into a combined failure distribution, and how the hazard value which we define in Section III is compositional with respect to this convolution.

III. Definition of the Hazard Value

We shall now define the connectivity requirements, provide a formal definition of the hazard value and instantiate the definition to our running example.

A. Connectivity Requirements

Considering a network as a service, a customer (or tenant) may express a demand X for connectivity and assign a certain weight w to (or offer a reward for) the connection of a pair of nodes (s, t) that satisfy this demand. The weight then indicates the importance of having that demand met through a route from s to node t. In this paper we denote this by $W_X(s, t) = w$. In the face of failures, it may be possible that the missing connectivity between s and t can be replaced by sending the

traffic from an altenative source s' to an alternative target t'. The customer may also specify a weight w' for such alternative connections, usually smaller than the weight w if the customer has a preference for the original connection. This ultimately leads to a function $W_X : V \times V \to \mathbb{R}^{\geq 0}$ representing the demand reward of X.

In our running example, the Internet traffic generated at one of the leaf nodes l_i is intended to be delivered to core c, but can alternatively also be delivered to core c'. Hence for every i where ℓ_i is a leaf node we define the demand reward as $W_i(x, y) = 3$ if $x = \ell_i$ and $y = c$ or $y = c'$, and $W_{(i,I)}(x, y) = 0$ elsewhere. The value 3 is the weight of having such a connection (the higher the value the higher is the importance of having the connection).

Given a weight function W_X, the network operator attempts to maximize the weight by providing the best route possible in case of failures. For an implemented route $\sigma \in E^+$ starting in v and ending in v', we overload notation and simply define $W_X(\sigma) = W_X(v, v')$ to denote the weight associated with the start and end point of σ. For implementing a set of routes $S \subseteq E^+$, the obtained weight for a customer X then is written as $W_X(S) = \max_{\sigma \in S} W_X(\sigma)$. Given a routing language R in the face of a failure scenario F, the obtained weight for X is then $W_X(R \cap Paths^F)$.

Finally, a network may serve multiple customers, or multiple demands from the same customer. In our running example, there is traffic expected between each of the leaf nodes in the same site and from each of the leaf nodes to the Internet. We define a separate weight function for each of these demands given by the set D, leading to a *family* of weight functions W_X indexed by the set of individual demands $X \in D$.

Definition 6 (Connectivity requirement)**.** *On a network $G = \langle V, E, src, dst \rangle$, a* connectivity requirement *is a (finite) family $W = \{W_X : V \times V \to \mathbb{N} \mid X \in D\}$ of weight functions over an index set D of all demands.*

As mentioned in the introduction, we consider as inputs to our metric: a network G, a routing language R, a DFP ϕ, and a family W_X of weight functions indexed over a set $X \in D$ of demands. We call such an input a connectivity scenario.

Definition 7 (Connectivity scenario)**.** *A* connectivity scenario *$S = \langle G, R, \phi, W, D \rangle$, consists of a network G, routing language R, distribution of failure probabilities ϕ, and connectivity requirement W over an index set D.*

B. The Hazard Value

Given a connectivity scenario, the network operator tries to optimize the sum of the weights of each of the individual customers considering the distribution of failures. In other words, the network operator tries to optimize the expected connectivity weight.

Definition 8 (Expected connectivity weight)**.** *Given a connectivity scenario $\langle G, R, \phi, W, D \rangle$, the* expected connectivity weight *is given by*

$$\mu(G, R, \phi, W, D) = \sum_{F \subseteq E} \sum_{X \in D} \phi(F) \cdot W_X(R \cap Paths^F) \ .$$

We can express the *expected efficiency* of a network as a percentage by dividing the expected connectivity weight by the maximum achievable weight in case all endpoints are connected: $\mu(G, R, \phi, W, D) / \sum_{X \in D} \max_{v,w \in V} W_X(v, w)$. In practice, as we strive to get the expected efficiency close to 1, we obtain a more human-readable metric if we consider instead the *hazard value*, i.e., the expected loss of efficiency due to connection failures.

Definition 9 (Hazard value)**.** *Given a connectivity scenario $\langle G, R, \phi, W, D \rangle$ its* hazard value *is defined as:*

$$\gamma(G, R, \phi, W, D) = 1 - \frac{\mu(G, R, \phi, W, D)}{\sum_{X \in D} \max_{v,w \in V} W_X(v, w)} \ .$$

Hence if the hazard value is 0, we expect no connectivity loss in the given connectivity scenario and a value higher than 0 gives the potential percentage loss of connectivity reward due to failures in the network.

C. The Hazard Value of the Operators Distributed Datacenter

Let us now consider an application of the hazard value for the datacenter network in Figure 1. The operator aims to improve the connectivity of this network by investing in an additional link that can connect the two distributed parts of the datacenter. The operator has the following information about the current network:

- The network G topology is given in Figure 1.
- The routes R in the network are "valley-free", meaning that traffic originating from a leaf is first only forwarded upward in the datacenter hierarchy (if at all) and subsequently (if at all) only routed downward to its destination. Traffic originating from a core switch is only routed downwards and traffic from leaves to core only upwards. Formally, this is achieved by splitting the set E into two subsets, the upwards-edges $Up = \{(l_i, s_k), (s_k, c), (s_k, c'), (l'_i, s'_k), (s'_k, c'), (s'_k, c) \mid 1 \leq i \leq 4, 1 \leq k \leq 2\}$ and the downward-edges $Down = \{(s_k, l_i), (c, s_k), (c', s_k), (s'_k, l'_i), (c', s'_k), (c, s'_k) \mid 1 \leq i \leq 4, 1 \leq k \leq 2\}$, and defining R as the regular routing language
$$R = Up^* \cdot Down^* \ .$$
- As a DFP, we assume independent link failures where each link has the failing probability of $p = 10^{-3}$, and use the distribution given in the preliminaries: $\phi_p(F) = p^{|F|}(1-p)^{(|E|-|F|)}$ for all $F \subseteq E$.
- Finally, we define the demand set $D = \{(l_i, l_j), (l'_i, l'_j), (l_i, \{c, c'\}), (l'_i, \{c, c'\})) | 1 \leq i, j \leq 4, i \neq j\}$ assuming a traffic between any two leaves in each of the two parts of the datacenter as well as between each leaf and core router (here the operator allows for alternatives). For any two leaf indexes $1 \leq i, j \leq 4$ with $i \neq j$ there is medium demand

(assigned the weight 10) of traffic both in the left site and in the right site given by $W_{(l_i,l_j)}(x,y) = 10$ if $x = l_i$ and $y = l_j$, and $W_{(l_i,l_j)}(x,y) = 0$ otherwise, and similarly $W_{(l'_i,l'_j)}(x,y) = 10$ if $x = l'_i$ and $y = l'_j$, and $W_{(l_i,l_j)}(x,y) = 0$ otherwise.

The demand rewards for the traffic between leaves and core routers in the left-hand part of the datacenter are given by $W_{(l_i,\{c,c'\})}(l_i,c) = 20$ and $W_{(l_i,\{c,c'\})}(l_i,c') = 15$ and $W_{(l_i,I)}(x,y) = 0$ otherwise, for any $1 \leq i \leq 4$. This assigns a higher reward of 20 for maintaining the connection to the core router c and a lower reward of 15 for the core router c' that is located in the right-hand part of the datacenter. In the right-hand site, the demand for Internet traffic is less, and there is no routing preference between the core routers, expressed by $W_{(l'_i,\{c,c'\})}(x,y) = 5$ whenever $x = l'_i$ and $y = c$ or $y = c'$, and $W_{(j,I)}(x,y) = 0$ otherwise, for any $1 \leq i \leq 4$.

Between the other nodes in the network, no traffic is generated, and no weight functions are needed to describe the demand. In particular, the spine switches and cores do not generate any traffic themselves, and there is no leaf-to-leaf connectivity requirement across sites.

Given this datacenter network and configuration, we compute the hazard value to be $1.21349 \cdot 10^{-4}$, meaning that we expect a 0.01213% loss of reward due to failure in the network, compared to the reward gained by a fully connected network.

The operator has two options how to increase the connectivity of the network:

1) *Scenario 1:* Add a link from s_2 to c'.
2) *Scenario 2:* Add a link from s'_1 to c.

Computing the hazard value for the two options, the operator obtains an expected 0.01203% reward loss for Scenario 1 and 0.01205% reward loss for Scenario 2, both of them clearly improving the expected loss in the current network topology. Hence, as intuitively expected, both additions lead to an improvement of the dependability of the network, but adding an extra link from the left part of the datacenter where there is more traffic towards the core routers (reflected by a higher weight) is more beneficial as it causes a more significant drop in the hazard value and hence allows for more reliable operation of the datacenter.

In the next section, we shall substantiate the intuition that adding links leads to a decrease in hazard value, as part of our results regarding expected mathematical properties of the metric.

IV. Mathematical Properties and Analysis

In this section we establish monotonicity and compositionality properties of the hazard value.

A. Monotonicity

As indicated, the hazard value is intended for comparing different network layouts, different routing strategies, different DFP, and changing demands. Intuitively, the hazard value should not increase when edges or nodes are added to a network, as long as the remaining connectivity scenario remains the same. Also, when routing restrictions are lifted, or failure probabilities decrease, one expects the hazard value not to increase. When additional sources of failure are considered, hazard values may increase. And finally, when certain links in a network are not traversed by any path with positive weight (given by the set of demands), and similarly when nodes are never visited by such paths, they can be removed from the routing language and even from the network itself, without affecting the hazard value at all.

We now argue that the chosen definition of hazard value indeed satisfies these intuitions. As we show, there is no need to consider each of these properties one-by-one. Instead, we can state many of them in one general claim by exploiting homomorphisms, i.e. link preserving mappings, between networks.

Definition 10 (Homomorphism)**.** *A* homomorphism *from network $G = \langle V, E, src, dst\rangle$ to network $G' = \langle V', E', src', dst'\rangle$ is a pair (f, g) of functions $f : V \to V'$ and $g : E \to E'$ such that $f(src(e)) = src'(g(e))$ and $f(dst(e)) = dst'(g(e))$ for all $e \in E$.*

Mapping one network to another using a homomorphism allows us to consider the relationships between the routing tables, DFPs, and weight functions of the two networks. For example, if we remove an unconnected node x from the set of nodes V of a network G, this leads to a set of nodes $\hat{V} = V/\{x\}$ in a network $\hat{G}$, while keeping the set of edges $\hat{E} = E$ unchanged. This removal can be captured by a natural homomorphism from $\hat{G}$ to G, taking the identity functions for $f : \hat{V} \to V$ and $g : \hat{E} \to E$. Similarly, if we add an edge e to the set of edges E of a network G, this leads to a new network $\overline{G}$ with the same set of nodes $\overline{V} = V$ and an extended set of edges $\overline{E} = E \cup \{e\}$, and we obtain a natural homomorphism $f : V \to \overline{V}$, $g : E \to \overline{E}$ by again using the identity functions. Having identity functions as homomorphisms simply means that all the paths in the left-hand network are also paths in the right-hand network. Having more complicated functions as homomorphisms, means that some nodes or edges are merged, i.e. the routing functions of two nodes or edges in the left network are merged in a single node or edge with the combined functionality in the right network.

As an example, Figure 2 shows a network homomorphism in which two nodes are merged, and one node is added, but no link merges take place. This results in a homomorphism in which g is injective but h is not. Note, that when thinking about failure scenarios in such networks, it is easy to imagine how separate link failures in the left network result in corresponding link failures in the right network. However, to understand how node failures are transferred, one needs to realize that—in our way of modelling failures—a node failure cannot be distinguished from a simultaneous failure of all corresponding links in the network. A network homomorphism treats node failures simply as if they are coinciding failures of all connected links. This means that, if there are additional

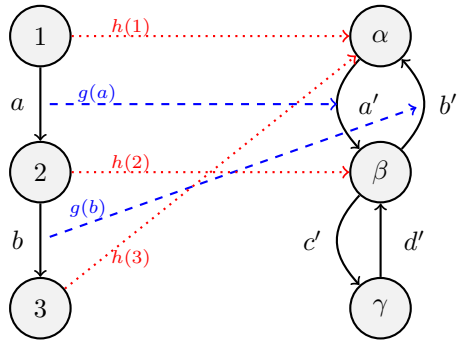

Figure 2: Example of a homomorphism between two networks. Notice how a route from 1 to 3 in the left network is mapped to a route from α via β back to α in the right network.

links in the right network, a node failure of the lower node on the left in Figure 2 may lead to a failure scenario on the right that cannot be immediately interpreted as a node failure as well.

Regardless of what a homomorphism looks like, it tells us how the paths in the left network compare to the paths in the right network. So, given a homomorphism $f : V \to V'$ and $g : E \to E'$, from a network G to a network G', and a set of routes R in network G, the function g of a homomorphism maps these routes to a set $g(R)$ of paths the right network. We can then verify that these paths are still routes by making the comparison $g(R) \subseteq R'$ to the routes R' in network G' (or more precisely $g(R \cap Paths) \subseteq R' \cap Paths'$ if we want to capture that we only consider routes that are not only part of the regular language of allowed routes, but also actually present as paths through the graph). We can also verify that the weight of a connection between nodes s and t has not changed, by comparing the weight functions: $W_X(s,t) = W'_X(f(s), f(t))$. If a homomorphism between two networks G and G' lives up to a number of such checks regarding the connectivity scenarios associated with G and G', we can draw conclusions about possible changes in the hazard value from this.

Theorem 1 (Hazard decrease)**.** *Let $\langle G, R, \phi, W, D\rangle$ and $\langle G', R', \phi', W', D\rangle$, with W and W' having the same index set D are two connectivity scenarios, and let (f, g) be a homomorphism from G to G' where g an injective function. If the connectivity scenarios are such that for all $X \in D$, $(s,t) \in V \times V$:*

i) $W_X(s,t) \leq W'_X(f(s), f(t))$;
ii) $\max_{v,w \in V} W_X(v,w) = \max_{v',w' \in V'} W'_X(v',w')$;
iii) $g(R \cap Paths) \subseteq R' \cap Paths'$, *lifting g to E^*;*
iv) $\phi(F) \geq \sum_{g^{-1}(F')=F} \phi'(F')$, *for all $\varnothing \subset F \subseteq E$*

then the hazard value of G' is not larger than that of G, i.e. $\gamma(G, R, \phi, W, D) \geq \gamma(G', R', \phi', W', D)$.

Theorem 1 states that, if we do not merge any edges (i.e. the function g in the homomorphism is injective) and if the weight between any two nodes and for any demand increases (condition i: $W_X(s,t) \leq W'_X(f(s), f(t))$), while the maximum attainable weight for any demand remains the same (condition ii: $\max_{v,w} W_X(v,w) = \max_{v',w'} W'_X(v',w')$), then the hazard value does not increase. Also, if we do not merge any edges and the number of valid routes increases (condition iii: $g(R \cap Paths) \subseteq R' \cap Paths'$), the hazard value cannot increase. Finally, if we do not merge any edges and the probability of any set of failures decreases (condition iv: $\phi(F) \geq \sum_{g^{-1}(F')=F} \phi'(F')$, except when $F = \varnothing$), the hazard value cannot increase.

The reason for requiring that we do not merge any links (i.e. require g to be injective), is that merging links may directly result in merging of set of failures that were previously independent. This increased dependency between failure scenarios may cause a rise in the hazard value, as illustrated next.

A rise in hazard value can occur when two nodes are merged, but also when the weight between any two nodes and for any demand decreases, while the maximum attainable weight for any demand remains the same. Also, if we have a homomorphism from G to G' and all the 'new' routes in G' have weight 0, the hazard value does not decrease because of those new routes. And finally, if the probability of all failures increases (except the empty failure in which nothing fails), the hazard value increases as well. Note, however, that these increases can even be concluded if edges are merged in the process of going from G to G'.

Theorem 2 (Hazard increase)**.** *Let $\langle G, R, \phi, W, D\rangle$ and $\langle G', R', \phi', W', D\rangle$ with W and W' having the same index set D be two connectivity scenarios, and let (f, g) be a homomorphism from G to G'. If the connectivity scenarios are such that for all $X \in D$ and $(s,t) \in V \times V$:*

i) $W_X(s,t) \geq W'_X(f(s), f(t))$;
ii) $\max_{v,w \in V} W_X(v,w) = \max_{v',w' \in V'} W'_X(v',w')$;
iii) $W'_X(\sigma') = 0$ *unless there exists $\sigma \in R \cap Paths$ with $g(\sigma) = \sigma'$;*
iv) $\phi(F) \leq \sum_{g^{-1}(F')=F} \phi'(F')$, *for all $\varnothing \subset F \subseteq E$*

then the hazard value of G' is not smaller than that of G, i.e. $\gamma(G, R, \phi, W, D) \leq \gamma(G', R', \phi', W', D)$.

As a corollary, when the conditions of both theorems hold, the hazard value remains equal. In particular, this is the case if we have an isomorphism between networks, i.e. if the networks are the same up to the router and link names.

B. Compositionality

Apart from the monotonicity properties discussed in the previous section, our definition of hazard value is also compositional.

Theorem 3 (Compositional connectivity requirements)**.** *Consider connectivity scenario $S = \langle G, R, \phi, W, D\rangle$ with hazard value γ, and connectivity scenario $S' = \langle G, R, \phi, W', D'\rangle$ with hazard value γ'. Note that the scenarios share the same network, routing, and DFP, and assume D and D' disjoint. The*

hazard value of the combined scenario $S'' = \langle G, R, \phi, W \cup W', D \cup D' \rangle$ *is given by*

$$\gamma'' \quad = \quad \frac{\overline{W}}{\overline{W} + \overline{W'}}\gamma + \frac{\overline{W'}}{\overline{W} + \overline{W'}}\gamma'$$

where $\overline{W} = \sum_{X \in D} \max_{v,w \in V} W_X(v,w)$ *denotes the maximum achievable sum of weights in a scenario.*

Furthermore, the hazard value is—to a certain degree—compositional with respect to the combination of independent DFPs. To show this, we first have to consider what it means to combine two different DFPs.

Let ϕ and ϕ' represent two DFPs defined on the same network G. When both failures can occur independently of one another, this leads to a combined DFP defined by the convolution:

$$(\phi \otimes \phi')(F) = \sum_{K \cup H = F} \phi(K) \cdot \phi'(H) \ .$$

While it is impossible to consider at a general level what the exact effect is of combining two DFPs in this way on the hazard value, it is possible to determine a bound. After all, we expect that introducing a new source of failure may increase the hazard value. Furthermore, we can show that the combined hazard value is never larger than the weighted sum of hazard values of the separate scenario, where the weights represent the condition that the failures occur in isolation.

Theorem 4 (Compositional DFPs)**.** *Consider connectivity scenario* $S = \langle G, R, \phi, W, D \rangle$ *with hazard value* γ*, and connectivity scenario* $S' = \langle G, R, \phi', W, D \rangle$ *with hazard value* γ'*. Note that the scenarios only differ in their failure distribution. The hazard value* γ'' *of the combined scenario* $S'' = \langle G, R, \phi \otimes \phi', W, D \rangle$ *is then bounded by:* $\max\{\gamma, \gamma'\} \leq \gamma'' \leq \phi'(\varnothing)\gamma + \phi(\varnothing)\gamma'$.

Note, that given a DFP ϕ, an additional source of failure for a single link $e \in E$ with probability p leads to the convolution $\phi \otimes \psi_{e,p}$, where $\psi_{e,p}$ is given by $\psi_{e,p}(\varnothing) = 1 - p_e$, $\psi_{e,p}(\{e\}) = p$, and $\psi_{e,p}(F) = 0$, elsewhere. Interestingly, from the fact that $\max\{\gamma, \gamma'\} \leq \gamma''$ in the theorem above, we can then already conclude that this added source of failure may lead to a larger or equal hazard value. This gives us another type of monotonicity that cannot be readily deduced from Theorems 1 and 2. This monotonicity can, for example, be exploited in the calculation of the hazard value, as it tells us that we can approximate complex DFPs by, for example, first only considering only upto n failures. This will give an increasingly good approximation as n rises.

V. Computing the Hazard Value

We shall now discuss the algorithmic issues related to computing the hazard value for a given connectivity scenario. In order to be able to pass a connectivity scenario as an input to an algorithm, we shall assume that the routing language R is regular and represented by a regular expression, or equivalently by a nondeterministic finite automaton (NFA). All examples of routing languages that we use in this paper are regular. We shall also study more fine-grained DFPs, where we consider only up-to-k concurrent link failures, $\phi_k(F) = \frac{\phi(F)}{\sum_{|H| \leq k} \phi(H)}$ when $|F| \leq k$ and $\phi_k(F) = 0$ elsewhere, as discussed in Section II.

Algorithm 1 Baseline algorithm for the hazard value

Input: A connectivity scenario $\langle G, R, \phi_k, W, D \rangle$, where $G = (V, E, src, dst)$ is the network, the path constraint R is represented by an NFA A, and k is the maximum number of failed links.
Output: The hazard ratio of the network
$sum := 0.0$
for all $F \subseteq E$ s.t. $|F| \leq k$, and $X \in D$ **do**
  $m := 0$
  for all $(s,t) \in V \times V$ **do**
    if $W_X(s,t) > m \wedge \mathbf{reach}^F(s,t)$ **then** $m := W(s,t)$
  $sum := sum + \phi_k(F) \cdot m$
$optimum := \sum_{X \in D} \max_{s,t \in X} W(s,t)$
return $1 - \frac{sum}{optimum}$

We are now interested in the question, how expensive is it to compute the hazard value for up-to-k failures in a given connectivity scenario $\langle G, R, \phi_k, W, D \rangle$. The problem whether $\gamma(G, R, \phi_k, W, D) \leq \delta$ for a given δ is clearly decidable in PSPACE by brute-force enumeration of all failure scenarios (while reusing the space). In this way the exact hazard value can be computed and compared to the given threshold value δ. On the other hand, we can prove by reduction from 3SAT that already the question whether the hazard value is nonzero becomes NP-hard (and this decision problem is also in NP as we can guess the failure that makes the hazard value nonzero and verify this in a polynomial time).

Theorem 5. *Consider a connectivity scenario* $\langle G, R, \phi_k, W, D \rangle$*, with* ϕ_k *the up-to-k independent DFP. The problem to decide whether* $\gamma(G, R, \phi_k, W, D) \neq 0$ *is NP-complete.*

A. Efficient Computation of Hazard Value

A direct way of computing the hazard function consists in enumerating all possible failure scenarios of size at most k and to sum the weight of the demands which remains connected. This is the approach described in Algorithm 1. To ensure that paths from a source to a destination respect a given set of constraints captured by a given nondeterministic automaton, we propose a standard Algorithm 2 that in an on-the-fly manner returns true if and only if it is possible to reach t from s under the given path constraints, by iteratively annotating each node in the network by the set of control states the NFA can be in when reaching the node. The correctness of the baseline algorithm should be clear as it very closely mimics the definition of the hazard value.

As already checking if the hazard value is nonzero is NP-hard, we cannot expect to find a polynomial-time algorithm for computing the value of the hazard function. However, we

Algorithm 2 Boolean function $\mathbf{reach}^F(s,t)$

Input: Fault F and two nodes $s,t \in V$ such that $s \neq t$. The NFA $A = (Q, E, \delta, q_0, Q_f)$ representing regular language over E and the graph $G = (V, E, src, dst)$ are implicitly given.
Output: $\mathbf{reach}^F(s,t) = true$ if t is reachable from s under the path constraints defined by the language of A and without using edges in F, $false$ otherwise
Let $\tau : V \to 2^Q$ be initialized s.t. $\tau(s) = \varnothing$ for all $s \in V$
$pending := \{s\}$
$\tau(s) := \{q_0\}$
while $pending \neq \varnothing$ **do**
    Remove u from $pending$
    for $e \in E \setminus F$ $s.t.$ $src(e) = u$ **do**
        $v := dst(e)$
        $X := \cup_{q \in \tau(u)} \delta(q, e)$
        if $X \not\subseteq \tau(v)$ **then**
            $pending := pending \cup \{v\}$;
            $\tau(v) := \tau(v) \cup X$
            if $v = t$ *and* $\tau(v) \cap Q_f \neq \varnothing$ **then return** $true$
return $false$

can significantly speed up the computation for average case complexity as we now show and document by our experiments.

Our efficient algorithm relies on enumerating all possible cuts with at most k edges that decrease the overall reward in the network. The algorithm is based on the set of all cuts $Cuts_A(s,t) = \{F \subseteq 2^E \mid Paths^F(s,t) \cap L(A) = \varnothing\}$ that disconnect the nodes s and t under the constraint A. Given a set of failed edges $F \subseteq E$ and a set of unfailing edges $U \subseteq E$ such that $F \cap U = \varnothing$, we also define the capacity of a cut $c \in Cuts_A(s,t)$ as follows:

$$C_F^U(c) = \begin{cases} \sum_{e \in c} C_F^U(\{e\}) & \text{if } |c| > 1 \\ \infty & \text{if } |c| = 1 \text{ and } c \subseteq U \\ 0 & \text{if } |c| = 0 \text{ or } c \subseteq F \\ 1 & \text{otherwise.} \end{cases}$$

Based on this definition, we can now introduce the strategic cut enumeration described in Algorithm 3 where $mincut$ is a function returning the min cut (set of edges that disconnet a source s from the target t) under a capacity function C.

The algorithm uses a stack storing sets of edges to enumerate all possible failure scenarios. Initially, all edges have an unknown status and once they are pushed on the stack, they are marked as absent (part of a failure). During the backtracking, all edges are one by one swapped from absent to present (line 31 to 33), meaning that we now assume that they are present in the network and can be used for forwarding packets. Instead of pushing individual edges on the stack, we push sets of them (representing the minimal cuts that disconnet the largest achievable rewards for the selected demand X) at the same time. This allows us to speed up the computation and skip the enumeration of a large number of failure scenarios that either (i) cannot disconnect the highest achievable reward of the demand X, in which case these accumulated rewards can be added to sum at line 15 by calling the function $compute\_prob$, or (ii) those that already disconnected the source and target nodes in the demand X and hence cannot contribute to the overall reward, irrelevant whether the edges with unknown status are present or absent (line 20).

Theorem 6. *Algorithm 3 returns the same value as the baseline Algorithm 1 and hence computes correctly the hazard value for the given connectivity scenario.*

VI. Hazard Value Experiments

We implemented our strategic search algorithm for computing the hazard value as well as the baseline brute-force search algorithm. Our empirical results show that our algorithm is by several orders of magnitude faster and that it is suitable for computing the hazard values for network topologies of realistic sizes. We also analyze the computed hazard values for the Internet Topology Zoo networks and datacenter networks, and show that the computed values correspond to the intuitive understanding of connectivity in networks.

A. Methodology

To evaluate the practical performance of our strategic search algorithm, we conduct experiments on a wide range of wide-area networks (ISP networks) from the Internet Topology Zoo [19] representing sparse and irregular types of topologies with several hundred of nodes, as well as on the classic datacenter topologies fat-tree [6] and BCube [14].

In order to generate flow demands for the Topology Zoo networks, we select 10% of all node pairs with the largest distance between them and consider two types of regular path constraints: (i) waypointing with 0, 1 and 2 waypoints (no waypoint corresponds to pure reachability) and (ii) service chaining with two given nodes that must be visited in a predefined order.

For the fat-tree, we distinguish between core, aggregator and edge nodes, using a parameter n: we initialize a fat tree with $\frac{n^2}{4}$ core nodes and n disjoint pods of $\frac{n}{2}$ aggregators and $\frac{n}{2}$ edges. Each aggregator is connected to $\frac{n}{2}$ core routers; similarly each edge router is connected to all $\frac{n}{2}$ aggregators of its pod. We consider the BCube topology in a hierarchical manner, with core routers and sites: for an integer n we create n core nodes and n sites containing one router and n leaves connected to it; the i-th core node is again connected to the i-th leaf of each site. For both datacenter topologies, we create demands between any pair of two distinct leafs/edge nodes in the same site and from each leaf to the core routers. We consider two types of regular path constraints: (i) reachability without any restriction and (ii) valley-free routing as described earlier in this paper.

In all experiments, we consider failures for varying numbers of failed links k ranging from 1 up to 10 concurrently failed links for ISP topologies and up to 20 failed links for the datacenter topologies. The probability of each failure depends on the probability of 0.001 that a single edge fails (independent of the failure probability of the other edges).

Algorithm 3 Strategic algorithm for the γ hazard function

Input: A connectivity scenario $\langle G, R, \phi_k, W, D\rangle$ where $G = (V, E, src, dst)$ is the network, the path constraint R is represented by an NFA A, and k is the maximum number of failed links.
Output: The hazard value of the network
$sum := 0.0$
for all $X \in D$ **do**
    $E' := \{e \in E \mid \exists (s,t) \in X.\ \exists \pi \in Paths(s,t) \cap L(A).\ e \in \pi\}$
    Let $status : E' \mapsto \{unknown, absent, present\}$ be s.t. $status(e) := unknown$ for all $e \in E'$
    Initialize $stack$ to empty stack where its elements are sequences of edges from E'
    Define $unused(stack) := E' \setminus \bigcup_{c \in stack} c$
    Define $F(stack) := \{e \in stack \mid status(e) = absent\}$
    Define $U(stack) := \{e \in stack \mid status(e) = present\}$
    repeat
        $(s,t) := \arg\max_{(s,t)\in X}\{W_X(s,t) \mid (s,t) \in X. \exists \pi \in Paths(s,t) \cap L(A). \forall e \in \pi. e \notin F(stack)\}$ *** $\max \varnothing = 0$
        $c := \arg\min_{c \in Cuts_A(s,t)} C^{U(stack)}_{F(stack)}(c)$
        if $|F(stack)| + C^{U(stack)}_{F(stack)}(c) > k$ **then**
            $sum := sum + W(s,t) \cdot compute\_prob(unused(stack), F(stack))$
            $backtrack(stack)$
        else
            $status(e_i) := absent$ for all $e_i \in c$
            $stack.push(c)$
            if $\neg\big(\exists s,t \in X$ s.t. $\mathbf{reach}^{F(stack)}(s,t)\big)$ **then** $backtrack(stack)$
    until $|stack| = 0$
$optimum := \sum_{X \in D} \max_{(s,t)\in X} W(s,t)$
return $1 - \frac{sum}{optimum}$

$backtrack(stack) =$
while $|stack| > 0 \wedge status(e) = present$ for all $e \in stack.peek()$ **do**
    $status(e) := unknown$ for all $e \in stack.peek()$
    $stack.pop()$
if $|stack| > 0$ **then**
    Let $c := stack.peek()$
    Let $e_i \in c$ s.t. $status(e_i) = absent$ and $status(e_j) = present$ for all $i < j \leq |c|$
    $status(e_i) := present$
    $status(e_j) := absent$ for all $i < j \leq |c|$

$compute\_prob(unused, F)$= **return** $\sum_{F' \subseteq unused, |F \cup F'| \leq k} \phi_k(F \cup F')$

For each scenario and different k values, we run both our baseline brute-force enumeration algorithm and the strategic enumeration algorithm with a 10 minute timeout and 16 GB memory limit. The experiments are executed on AMD EPYC 7551 processors running at 2.55 GHz with boost disabled.

For visualizing the relative performance of the baseline algorithm and our strategic enumeration, we use *cactus plots* (see e.g. [5]), where for each of the methods, we record the runtime on each instance of the problem. Then we (independently for each algorithm) sort the instances by increasing runtime and plot them as two curves. Note that the y-axis is on logarithmic scale. While cactus plots do not provide instance-to-instance comparison, they deliver an overall picture of the relative performance of the algorithms.

B. Performance Results

In Figure 3 we can see the performance of the baseline algorithm and our strategic one for the CPU time (in seconds) needed to compute the hazard values on the networks from the Topology Zoo database, both for the service chaining routing language as well as the waypointing with up to 2 waypoints. Already in the service chaining scenario, we are two orders of magnitude faster than the baseline on the largest instance that the baseline can solve. The advantage of our algorithm is even more clear for the waypointing path restriction scenario where we see several orders of magnitude improvement. The situation is analogous for the datacenter experiments presented in Figure 4 where both for the basic reachability as well as for valley-free routing (discussed earlier) the two curves open

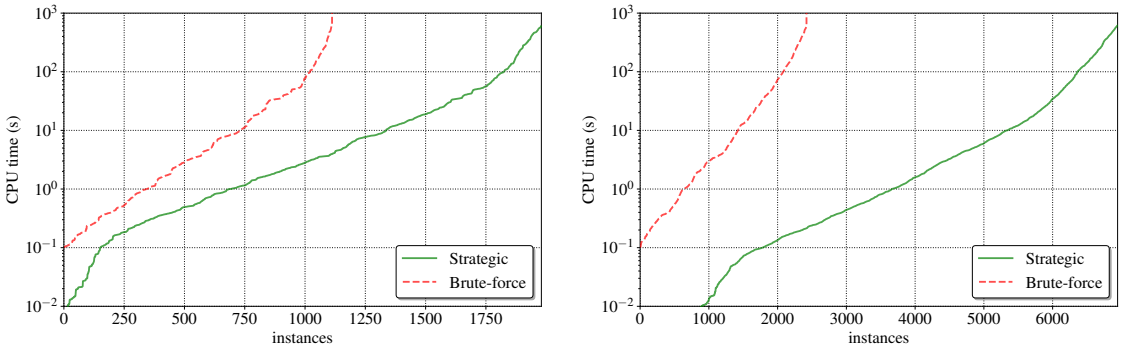

Figure 3: Topology Zoo with service chaining (left) and waypointing (right)

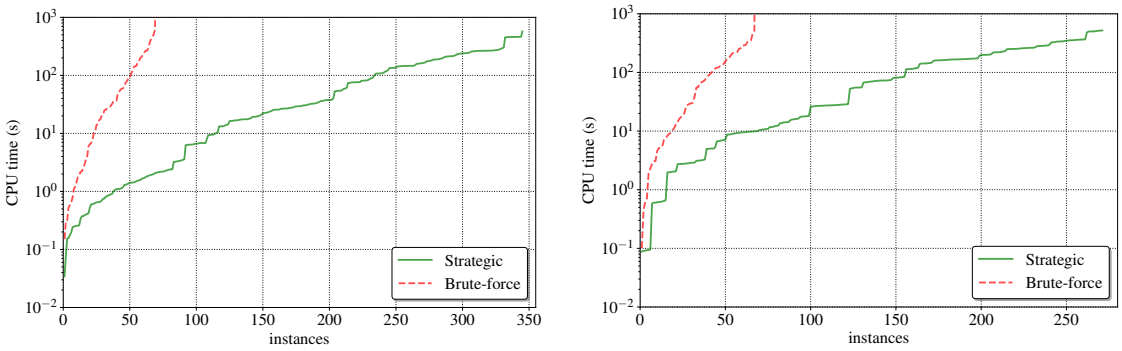

Figure 4: Datacenter with basic reachability (left) and valley-free routing (right)

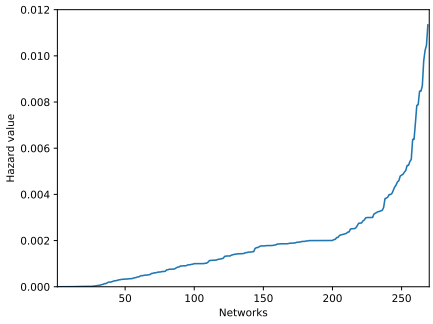

Figure 5: Hazard values for the Topology Zoo ($k = 3$)

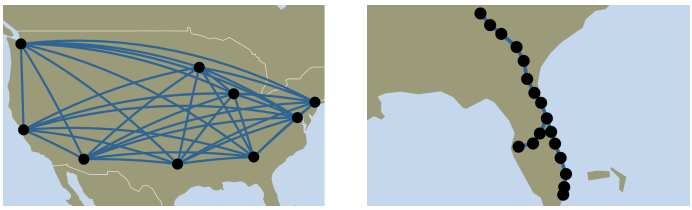

Figure 6: GlobalCenter (left) with γ=0.00000 for $k = 3$ and Sogo (right) with γ=0.01134 for $k = 3$

even faster than in the case of ISP topologies.

In conclusion, our strategic algorithm is applicable to real ISP and datacenter topologies, and the current implementation (in Python) allows us to compute in about 5 minutes the hazard value for 2 to 3 failed links on Topology Zoo networks of sizes up to couple of hundred of nodes and 300-400 links. For the more regular datacenter topologies, we can within 5 minutes compute the hazard value for similarly sized topologies in the number of nodes but with up to 1000 links and for a larger number of failed links (between 10 to 20). As a result, our method is (in its prototype implementation) already applicable to medium size networks, and we expect that additional optimizations can further improve the scaling.

C. Hazard Values for ISP Topologies

Finally, we analyze the hazard values for all ISP topologies from the Topology Zoo database for $k = 3$ number of failed links. The distribution of the hazard values for these network topologies is depicted in Figure 5. We can notice that there are few highly connected networks with hazard value zero, meaning that there is no failure scenario with up to 3 failed links that can disconnect any of the demands, meaning the these networks are highly resilient for failures. An example of such a network is GlobalCenter with the hazard value zero as depicted[1] in Figure 6. More than 200 out of 260 topologies have a hazard value below 0.002, meaning they only suffer a loss in value of 0.2% compared to the maximal achievable reward by a fully connected network. However, there are a few topologies with relatively high hazard value. In case of the

[1]The Topology Zoo graphs are taken from http://www.topology-zoo.org/

Sogo network in Figure 6, the hazard value is 0.01, meaning that connection failures are expected to lead to a 1% loss in reward value.

It is clear that in such a network topology, there is a large number of failure scenarios that can completely disconnect the end-points in the network. Hence, the computed hazard values correspond to the intuitive understanding of more or less resilient types of network topologies.

VII. Related Work

There already exists interesting literature on the empirical characteristics of failures, e.g., in datacenters [12], [31], statewide networks [29], or IP backbones [16]. This literature is highly valuable for the comparison of existing networks, but does not directly solve the problem of comparing network designs that are not yet implemented. Empirical research does provide valuable input to the method described in this paper, as it can provide us with more realistic probability distributions for link failures.

In the graph-theory community, the connectivity of a graph is often measured in terms of its minimum cut or the number of available disjoint paths, which are common measures not only for the throughput achievable in a network but also for its resilience [2], [3], [30]; cuts also form the basis for the frequently used expansion measures [15]. Another well-studied approach, primarily used by the parallel-computing community, is to measure how many failures a network can sustain while still being able to emulate its ideal counterpart with a certain maximal overhead (e.g., a constant slowdown) [20], [24]. Both worst-case scenarios—in which adversarial failures cause a loss of connection—as well as average-case scenarios—in which the probability of loss of connection is computed—are well understood [2]. However, these generic graph-theoretic metrics treat all the connections in a graph as equal, and do not consider the fact that for a network operator some connections are more important than others.

Within the networking community, additional specific metrics are considered. Some of them, such as the protection ratio [11], revolve around single failures (asking how many individual failures are protected), while we in this paper are particularly interested in multiple failures; other metrics, such as the loop ratio [8], are concerned with the detailed network behavior during convergence after failures, while during the consideration of alternative topologies, the details of the routing mechanism are not always known in sufficient detail to consider such metrics—which is why we assume fast rerouting in this paper. Two interesting connectivity measures in the context of scenarios with multiple failures, and accounting for the locality constraints imposed by fast rerouting, are the perfect [9], [10] and the ideal [7] resilience. However, these measures only account for locality constraints, but not any of the other aspects considered in this paper, nor do they provide similar powerful properties.

Our work is also related to survivable network design, a classic topic in operations research [13]. A general and powerful approach to design networks which are robust to failures, uses mathematical programming, and in particular, (integer) linear programming. A well-known platform which also provides benchmarks for telecommunication networks is SNDlib [23]. However, we are not aware of any existing approach in this context which accounts, e.g., for alternatives or more complex routing constraints and DFPs.

There further exists interesting literature on the impact of routing constraints on connectivity, for example, in the context of inter-domain routing (in the absence of failures), where routing policies typically need to be compatible with business considerations [18]; or in the context of fast rerouting where failover rules are inherently local and can harm connectivity [9], [10]. Our paper accounts for the physical topology explicitly, among other properties, and we model the routing with constraints. Our approach is hence orthogonal to work on resilient routing mechanisms [25]. It is also orthogonal to work studying how to verify and maintain routing and policy constraints under failures [17], [27].

We are not aware of any connectivity measure explicitly accounting for differences in demand on different connections, choice between connections to resolve a given demand, preferences and priorities in resolving that choice, and specific failures. With this work, we aim to fill this gap and make a proposal for a measure which meets a number of desirable properties, and which can be computed efficiently.

VIII. Conclusion

Assessing whether a network is sufficiently robust often depends on more than its mere topological connectivity. Motivated by this observation, in collaboration with a local network operator, we developed a more general network connectivity measure, the hazard value, which allows to account for specific demands and DFPs, alternatives, routing constraints and priorities. We have shown how the hazard value can be computed efficiently, and can provide interesting new insights into the connectivity of existing networks.

Regarding the required input parameters to the hazard value computation, the traffic patterns can be derived from network monitoring and historical data, using either the worst-case approach scenario or dividing the daily traffic into a finite number of time slices and analyzing them separately. DFPs can be derived also from historical data and combining the independent link, node and shared-link groups failure probabilities. The routing constraints are by default assuming basic reachability but it can be further restricted by the input from the network operators that may require waypointing on some routers, valley-free routing policies, blacklisting of certain routers etc. Finally, the alternative destinations and the weights of each demand also require an input from the network operator but can be in the first iteration approximated by e.g. the amount of traffic for each demand (more traffic implying higher weight). A further research may try to automate the collection of the input data needed for the computation of the hazard value.

REFERENCES

[1] A. Bagchi, A. Bhargava, A. Chaudhary, D. Eppstein, and C. Scheideler. The effect of faults on network expansion. *Theory of Computing Systems*, 39(6):903–928, 2006.

[2] A. Bagchi, A. Bhargava, A. Chaudhary, D. Eppstein, and C. Scheideler. The effect of faults on network expansion. *Theory of Computing Systems*, 39(6):903–928, 2006.

[3] A. Bagchi, A. Chaudhary, C. Scheideler, and P. Kolman. Algorithms for fault-tolerant routing in circuit switched networks. In *Proc. 14th Annual ACM symposium on Parallel Algorithms and Architectures (SPAA)*, pages 265–274, 2002.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.

[5] M. N. Brain, J. H. Davenport, and A. Griggio. Benchmarking solvers, SAT-style. In *Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation co-located with the 42nd International Symposium on Symbolic and Algebraic Computation (ISSAC'17)*, volume 1974 of *CEUR*, pages 1–15. CEUR-WS.org, 2017.

[6] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. volume 34, pages 892–901, 1985.

[7] M. Chiesa, I. Nikolaevskiy, S. Mitrović, A. Gurtov, A. Madry, M. Schapira, and S. Shenker. On the resiliency of static forwarding tables. *IEEE/ACM Transactions on Networking*, 25(2):1133–1146, 2016.

[8] F. Clad. *Disruption-free routing convergence: computing minimal link-state update sequences*. PhD thesis, Strasbourg, 2014.

[9] J. Feigenbaum, B. Godfrey, A. Panda, M. Schapira, S. Shenker, and A. Singla. Brief announcement: On the resilience of routing tables. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 237–238, 2012.

[10] K.-T. Foerster, J. Hirvonen, Y.-A. Pignolet, S. Schmid, and G. Tredan. On the feasibility of perfect resilience with local fast failover. In *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.

[11] P. Francois and O. Bonaventure. An evaluation of ip-based fast reroute techniques. In *Proceedings of the 2005 ACM conference on emerging network experiment and technology*, pages 244–245, 2005.

[12] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 conference*, pages 350–361, 2011.

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

[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, and G. Lv. Bcube: A high performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM*. Association for Computing Machinery, Inc., August 2009.

[15] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[16] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an ip backbone. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 237–242, 2002.

[17] P. G. Jensen, M. Konggaard, D. Kristiansen, S. Schmid, B. C. Schrenk, and J. Srba. Aalwines: A fast and quantitative what-if analysis tool for mpls networks. In *Proc. 16th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2020.

[18] R. Klöti, V. Kotronis, B. Ager, and X. Dimitropoulos. Policy-compliant path diversity and bisection bandwidth. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 675–683. IEEE, 2015.

[19] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29:1765 – 1775, 11 2011.

[20] F. T. Leighton, B. M. Maggs, and R. K. Sitaraman. On the fault tolerance of some popular bounded-degree networks. *SIAM Journal on computing*, 27(5):1303–1333, 1998.

[21] B. M. Maggs and R. K. Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, 2015.

[22] W. Najjar and J.-L. Gaudiot. Network resilience: A measure of network fault tolerance. *IEEE Transactions on Computers*, 39(2):174–181, 1990.

[23] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007. http://sndlib.zib.de, extended version accepted in Networks, 2009.

[24] C. Scheideler. Models and techniques for communication in dynamic networks. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 27–49. Springer, 2002.

[25] S. Schmid, N. Schnepf, and J. Srba. Resilient capacity-aware routing. In *Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'21)*, volume 12651 of *LNCS*, pages 411–429. Springer-Verlag, 2021.

[26] P. Sebos, J. Yates, G. Hjalmtysson, and A. Greenberg. Auto-discovery of shared risk link groups. In *OFC 2001. Optical Fiber Communication Conference and Exhibit. Technical Digest Postconference Edition (IEEE Cat. 01CH37171)*, volume 3, pages WDD3–WDD3. IEEE, 2001.

[27] S. Steffen, T. Gehr, P. Tsankov, L. Vanbever, and M. Vechev. Probabilistic verification of network configurations. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 750–764, 2020.

[28] L. Suresh, M. Canini, S. Schmid, and A. Feldmann. C3: Cutting tail latency in cloud data stores via adaptive replica selection. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 513–527, 2015.

[29] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage. California fault lines: understanding the causes and impact of network failures. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 315–326, 2010.

[30] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *Proceedings of the eleventh annual ACM symposium on Principles of distributed computing*, pages 83–89, 1992.

[31] D. Zhuo, M. Ghobadi, R. Mahajan, K.-T. Förster, A. Krishnamurthy, and T. Anderson. Understanding and mitigating packet corruption in data center networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 362–375, 2017.