Columbia University, EE Technical Report #2010-07-25, July 2010

The Resilience of WDM Networks to Probabilistic Geographical Failures

Pankaj Agarwal*, Alon Efrat†, Shashidhara Ganjugunte*, David Hay‡, Swaminathan Sankararaman†, and Gil Zussman‡
*Computer Science, Duke University. {pankaj, shashigk}@cs.duke.edu
†Computer Science, University of Arizona. {alon, swami}@cs.arizona.edu
‡Electrical Engineering, Columbia University. {hdavid, gil}@ee.columbia.edu

***Abstract*—Telecommunications networks, and in particular optical WDM networks, are vulnerable to large-scale failures of their infrastructure, resulting from physical attacks (such as an Electromagnetic Pulse attack) or natural disasters (such as earthquakes or floods). Such events happen in *specific geographical locations* and disrupt specific parts of the network but their *effects are not deterministic*. Therefore, we provide a unified framework to model the network vulnerability when the event has a *probabilistic nature*, defined by an arbitrary probability density function. Our framework captures scenarios with a number of simultaneous attacks, in which network components consist of several dependent sub-components, and in which either a 1+1 or a 1:1 protection plan is in place. We use computational geometric tools to provide efficient algorithms to identify vulnerable points within the network under various metrics. Then, we obtain numerical results for specific backbone networks, thereby demonstrating the applicability of our algorithms to real-world scenarios. Our novel approach would allow identifying locations which require additional protection efforts (e.g., equipment shielding). Overall, the paper demonstrates that using computational geometric techniques can significantly contribute to our understanding of network resilience.**

***Index Terms*—Network survivability, geographic networks, network protection, computational geometry, optical networks.**

I. Introduction

Telecommunication networks are crucial for the normal operation of all sectors of our society. During a crisis, telecommunication is essential to facilitate the control of physically remote agents, provides connections between emergency response personnel, and eventually enables reconstitution of societal functions. However, telecommunication networks heavily rely on physical infrastructure (such as optical fibers, amplifiers, routers, and switches), making them *vulnerable to physical attacks, such as an Electromagnetic Pulse (EMP) attack, as well as natural disasters, such as earthquakes, hurricanes or floods* [1]–[5].

Physical attacks or disasters affect a specific geographical area and will result in failures of neighboring components. Therefore, it is crucial to consider the effect of disasters on the *physical (fiber) layer* as well as on the (logical) network layer. Although there has been a significant amount of work on network survivability, most previous work considered a small number of isolated failures or on shared risk groups (e.g., [7]–[12] and references therein). On the other hand, work on large scale attacks focused mostly on cyber-attacks

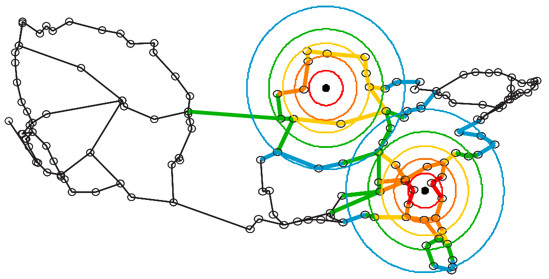


Fig. 1. The fiber backbone operated by a major U.S. network provider [6] and an example of two attacks with probabilistic effects (the link colors represent their failure probabilities).

(viruses and worms) (e.g., [13]–[15]). In contrast, we consider events that cause a large number of failures in a specific geographical region. This emerging field of *geographically correlated failures* has started gaining attention only recently [5], [16]–[22]. However, unlike most of the recent work in this field, we focus on *probabilistic attacks* and on a number of *simultaneous attacks*.

Physical attacks rarely have a deterministic nature. The probability that a component is affected by the attacks depends on various factors, such as the distance from the attack's epicenter to the component, the topography of the surrounding area, the component's specifications, and even its location within a building or a system. We consider *arbitrary probability functions* (with constant description complexity)[1] and develop algorithms that obtain the *expected vulnerability* of the network. Furthermore, while [5], [16]–[22] considered only a single disaster, our algorithms allow to assess the effect of *several simultaneous events*.

In particular, we focus on wavelength-routed WDM optical networks, especially at the backbone [8], [9]. We envision the network as a graph, embedded in the plane, in which each node corresponds to an *optical cross-connect (OXC)* and each link corresponds to an *optical fiber* (which are usually hundreds or thousands of kilometers long). Along each link there are *amplifiers*, which are spaced-out approximately equally and are crucial to traffic delivery on the fiber. Data is transmitted

[1]Characterizing the failure probability function of each component is orthogonal to this research, and we assume it is given as an input.

on that graph on *lightpaths*, which are circuits between nodes.

While lightpaths can be established by the network dynamically, lightpath-provisioning is a resource-intensive process which is usually slow. If many links fail simultaneously (as in the case of a physical attack or large scale disaster), the current technology will not be able to handle very large scale re-provisioning (see for example, the CORONET project [23]). Therefore, we assume that lightpaths are *static*, implying that if a lightpath is destroyed, all the data that it carries is lost. Our goal is to identify the most vulnerable location in the network, where vulnerability is measured either by expectation of the number of failed components or by the expected total data loss. In order to do it, our model allows considering the failure probability of a *compound component* by evaluating the effect of the event on its sub-components (e.g., the failure probability of a fiber, due to failure of some amplifiers). Under this model, we develop algorithms that identify the most vulnerable locations and trade accuracy with efficiency. Namely, we can provide arbitrary small errors, albeit high running time. Note that although these algorithms have to be executed offline in preparation to a disaster, they have to consider numerous options and topologies and therefore, their efficiency is important. Moreover, our algorithms work also for the deterministic case, with superior performance than previous suggestion [17].

For the case of k simultaneous attacks, we are interested in the k-tuple of the most vulnerable locations. In the simultaneous attack case, the problem is hard not only due to its probabilistic nature but also due to the combinatorial hardness of the deterministic problem. Hence, we develop approximation algorithms for various cases.

We also consider networks which are protected, by a *dedicated path protection* plan (either 1+1 or 1:1 plan [8], [9]). Under such plans, every (primary) lightpath has a predefined backup lightpath, on which data can be transmitted, if the primary lightpath fails. These protection plans are pre-computed before a failure event, and therefore, it is reasonable to assume that they can be applied even after a large scale set of failures. For these networks, we provide approximation algorithms that identify pairs of vulnerable locations that will have high effect both on primary and backup paths.

With the current technology, large scale dynamic restoration is mostly infeasible. However, this capability will emerge in future optical networks [23]. For future networks capable of dynamic restoration, the network resilience can be measured as the maximum post-attack flow. However, we show that computing this measure is in $\#P$ and hence cannot be found in any reasonable time. We discuss options for mitigating this evaluation barrier.

Finally, we provide numerical results that demonstrate the applicability of our algorithms to real backbone networks. Among other things, we show that even when the approximation algorithms only guarantee low accuracy (thereby, having low running time), the obtained results are very close to optimal. This would allow checking various scenarios and settings relatively fast.

The main contributions of this paper are three fold. First, this is the first paper to present a general probabilistic model for geographically-correlated attacks, as well as efficient approximation algorithms for finding the most vulnerable locations in the network. Second, we provide the first set of algorithms that deal with simultaneous attacks. Third, we provide algorithms that take into account pre-computed protection plans. Finally, the paper demonstrates that computational geometric techniques can significantly contribute to our understanding of network resilience.

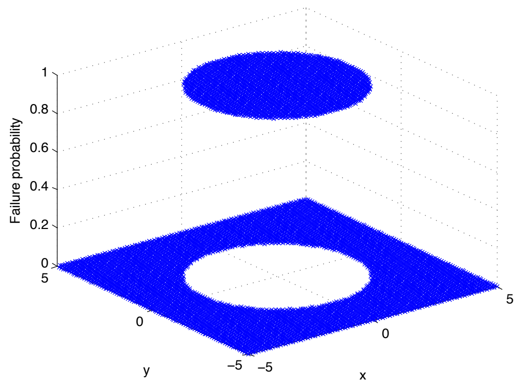
The rest of the paper is organized as follows. In Section II we review related work and in Section III we present the network model and formulate the problem. In Section IV we develop algorithms for failures centered at a single location and extend them to multiple locations in Section V. We study the effect of protection and restoration plans in Sections VI and VII. We present experimental results in VIII and conclude and discuss future work in Section IX.

II. Related Work

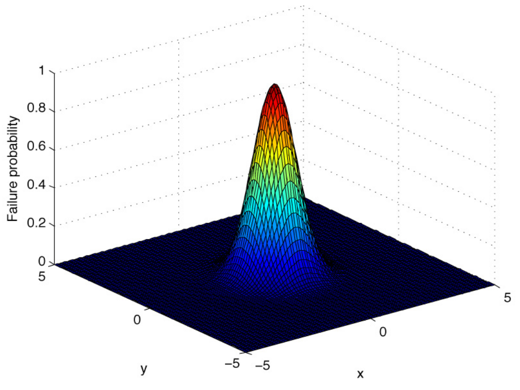
Network survivability and resilience is a well-established research area (e.g., [7]–[9], [12] and references therein). However, most of the previous work in this area and in particular in the area of physical topology and fiber networks (e.g., [10], [11]) focused on a small number of fiber failures (e.g., simultaneous failures of links sharing a common physical resource, such as a cable, conduit, etc.). Such correlated link failures are often addressed systematically by the concept of *Shared Risk Link Group* (SRLG) [24]. In contrast with these works, we focus on failures within a specific geographical *region* (e.g., [2], [3], [25]), implying that the failed components do not necessarily share the same physical *resource*. To the best of our knowledge, *geographically correlated failures* have been considered only in a few papers and under very specific assumptions [5], [16]–[18], [20]–[22]. In most cases, the assumption is that the failures of the components are deterministic and that there is a single failure. Perhaps the closest to the concepts studied in this paper are the problems studied in [17], [19], [25]. In particular, [17] recently obtained results regarding the resilience of fiber networks to geographically correlated failures where disasters are modeled as circular areas in which the links and nodes are affected. However, [17] considers only a *single disaster* scenario where failures are *deterministic*.

Another closely related theoretical problem is the *network inhibition problem* [26]–[29], in which the objective is to minimize the value of a maximum flow in the graph, where there is a cost associated with destroying each edge, and a fixed budget is given for an orchestrated attack (namely, removing a set of edges whose total destruction cost is less than the budget). However, previous works dealing with this setting and its variants (e.g., [29], [30]) did not study the removal of (geographically) neighboring links.

Notice that when the logical (i.e., IP) topology is considered, wide-spread failures have been extensively studied [13], [14]. Most of these works consider the topology of the Internet as



(a) Deterministic circular attack



(b) Probabilistic gaussian attack

Fig. 2. Examples of spatial failure probability distributions of two attack that occurs in $(0, 0)$. (a) A component fails if and only if it is within radius 3 of the attack. (b) A component fails with a probability that corresponds to its distance from the event.

a random graph [15] and use percolation theory to study the effects of random link and node failures on these graphs. These studies are motivated by failures of routers due to attacks by viruses and worms rather than physical attacks.

III. Problem Statement

Let $G = (V, E)$ be a graph representing the optical network, where V is a finite set of nodes in the plane, and E is a set of links, each of which is a non self-intersecting, connected curve that can be expressed by a constant number of polynomial inequalities of constant maximum degree (e.g, a segment or a B-spline). Let $c_e \geq 0$ be the capacity of link e. A lightpath π consists of an ordered sequence of links (where two consecutive links in the path share the same node in one of their endpoints). Let t_π be the amount of data traversing over π. Generally, we consider the nodes and links as *simple components*, and the lightpaths as *compound components*. Let $Q = \{q_1, \ldots, q_m\}$ be a given set of network components, all of the same type. Let w_q be the weight associated with a component q, which indicates either the amount of traffic along q or the capacity of the component.

Each attack induces a spatial probability distribution on the plane, specifying the damage probability at each location (see Fig. 2.) Taking the perspective of a (simple) component q, given an attack location $p \in \mathbb{R}^2$, let $f(q, p)$ be the probability that q is affected by p. For example, in a deterministic setting, $f(q, p)$ is either 0 or 1. In many applications $f(q, p)$ is given, or can be computed as a function of the distance from p to q. However, in certain types of attacks (e.g., EMP attacks), fiber links are not damaged directly. In such cases, we can treat each link as a compound component which consists of several amplifiers (which are simple components represented as points in the plane); destroying an amplifier along the fiber makes it unusable. More generally, we can regard each link to be comprised of a finite set of sample points R, and the probability that a link e is damaged by an attack at p is the probability that some point in R is damaged, i.e, $f(e, p) = 1 - \prod_{r \in R}(1 - f(r, p))$.

Furthermore, for a lightpath π (or any other compound component), denote by V_π the set of components that constitute the compound component, and for each simple component $v \in V_\pi$, let $f(v, p)$ be the probability that v is affected by p. Thus, the probability that π is affected by an attack in p is

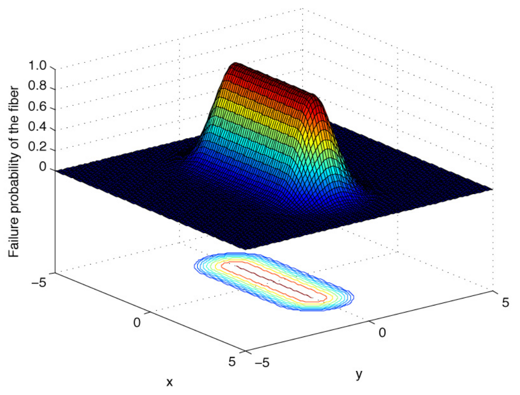
$$f(\pi, p) = 1 - \prod_{v \in V_\pi} (1 - f(v, p)). \tag{1}$$

Fig. 3 illustrates two such distributions, where the component q is either a fiber link or a lightpath, and f decreases exponentially with the distance.

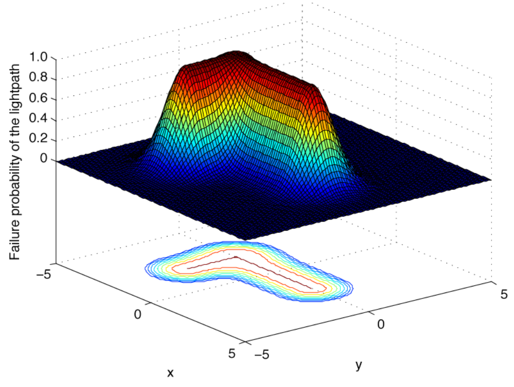
Our goal is to find the location (or set of locations) which causes the highest expected impact on the network, where the impact is measured either by the number of failed components (amplifiers, OXCs, or fibers), by the total capacity of failed fibers, or by the total traffic carried by failed lightpaths. For a set of attack locations $\mathcal{P}$, let $\Phi(Q, \mathcal{P})$ denote the expected number (or alternatively, weighted sum) of components of Q affected by the attacks in $\mathcal{P}$; by linearity of expectation, we get

$$\Phi(Q, \mathcal{P}) = \sum_{q \in Q} w_q (1 - \prod_{p \in \mathcal{P}} (1 - f(q, p))),$$

When Q is clear from the context, we use $\Phi(\mathcal{P})$. In case a protection plan is in place (see Section VI), we count data as lost, if and only if both the primary and backup lightpaths are affected. The weight w_q of each components enables us to define various measures in a unified manner: if Q is the set of amplifiers and w_q is set to 1 (for all q), then $\Phi(Q, \mathcal{P})$ is the expected number of affected amplifiers had the attacks occurred in $\mathcal{P}$. Similarly, if Q is the set of fibers and for any fiber q, $w_q = c_q$ (q's capacity) then $\Phi(Q, \mathcal{P})$ yields the expected capacity loss of attacks in $\mathcal{P}$. Finally, if Q is the set of lightpaths and $w_q = t_q$, then $\Phi(Q, \mathcal{P})$ is the expected loss in traffic, unless there is a protection (or restoration) plan in place. It is important to notice that, by linearity of expectation,



(a) Failure probability of a fiber



(b) Failure probability of a lightpath

Fig. 3. Examples of the failure probability of a Gaussian attack, taking the component's perspective. (a) Function $f(q, p)$, where q is a fiber placed between $(-2, 0)$ to $(2, 0)$. (b) Function $f(q, p)$, where q is a lightpath that consist on two fibers $[(-2, 0), (2, 0)]$ and $[(-2, 0), (-2, -2)]$.

$\Phi(Q, \mathcal{P})$ corresponds to the expected value of the measure under consideration, *regardless of any dependency between the various components in* Q. Therefore, even in the extreme situations in which two components share the same physical resource (e.g., lightpaths that share the same fiber, or fibers that share the same conduit), one can evaluate $\Phi(Q, \mathcal{P})$ by considering each component *separately*.

In this paper, for simple components, we consider only probability functions $f(\cdot, \cdot)$ with a *constant description complexity*. Intuitively speaking, these are functions that can be expressed as constant number of polynomials of constant maximum degree, or simple distributions like the Gaussian distribution. Thus, the description complexity of $\Phi(\cdot)$ is $O(|\mathcal{P}|)$ and of $f(q, \cdot)$ is $O(|V_q|)$, where q is a compound component that consists of $|V_q|$ simple components. In particular, our results holds also for the following important classes of distributions:

(i.) f is a function of the Euclidean distance, i.e, $f(q, p) = \max\{0, 1 - d(q, p)\}$, where $d(q, p)$ is the Euclidean distance between p and q; more generally, d could be any norm.

(ii.) Gaussian distribution, i.e, $f(q, p) = \beta e^{-d(q,p)^2 \alpha}$ for constants $\alpha, \beta > 0$, chosen appropriately to normalize the distribution.

When the components are points in the plane (that is, amplifiers or OXCs) and $f(q, p) = \max\{0, 1 - d(q, p)\}$, the problem is related to the Fermat-Weber problem [31], [32]; (i.e, finding a point that minimizes the average distance to a given set of points). However, the approximate solutions to Fermat-Weber problem and our problem can be quite different.

IV. The Single Location Scheme

A. Intuition

We first assume that $\mathcal{P}$ is a singleton and will denote the location of the (single) attack by p.

Our algorithm divides the plane surrounding each network element q to Y *levels*, defined as a monotonically-decreasing vector, such that a point p is in the i-th level if and only if $f(q, p)$ is at least the i-th element in the vector. Note that each point can be in multiple levels. Next, each level defines a geometric region surrounding q; note that the region corresponding to some level y contains all regions of levels $y' < y$. Moreover, if f is monotonic with the distance from q, these regions are contiguous in the plane; however, our algorithm does not require this property. Intuitively, the number of levels determines the accuracy and the running time of our algorithm: the more levels we have, the more accurate our results are, however our algorithm requires additional time to complete. Fig. 3 depicts two simple examples of levels induced by a Gaussian attack on a fiber and on a lightpath; each contour defines the edge of a level.

The next step is to find the point which maximizes Φ (up to some error). In the uniform case (all weights correspond to 1), this is the point that belongs to the largest number of levels, each might correspond to a different network element (depending on the vector Y, we might count the number of levels as a weighted sum). In the non-uniform case, each region of component q can be viewed as w_q coinciding regions; thus, having essentially the same problem. Finding the point which belongs to the highest number of regions (usually, referred to as the point with the highest depth) is a well-studied problem in computational geometry.

B. Detailed description of the simple component case

We first present an algorithm for simple components, and then describe the modifications required to handle the case of compound components. Our algorithm (see pseudo-code in Algorithm 1) is similar to a recent algorithm by Vigneron [33] and returns a point p such that $\Phi(p) \geq (1 - \varepsilon)\Phi(p^*)$, where $p^* = \arg\max_{p \in \mathbb{R}^2} \Phi(p)$ (namely, the optimal attack location), and $0 < \varepsilon < 1$.[2]

The vector Y, which determines how the plane is divided, is defined in an exponentially decreasing manner, such that its

[2]The algorithm of [33] returns a point p' such that $\Phi(p^*) \leq (1 + \varepsilon)\Phi(p')$. In addition, our analysis is slightly simpler, and, in certain cases, we show how to achieve a better running time.

Algorithm 1 MaxImpactLocation algorithm that finds a point $p \in \mathbb{R}^2$ that causes $(1-\varepsilon)$ of the maximum impact of a set of simple components Q, with weight vector W and spatial probability function vector f.

```
1: point $p$ procedure MaxImpactLocation($Q$,$W$,$f$,$\varepsilon$)
2:     $wm \leftarrow \max_{q\in Q, p\in\mathbb{R}^2} w_q f(q,p)$
           ▷ $w_q$, $f(q,\cdot)$ are elements of $W$, $f$ corresponding to $q$
3:     for each $q \in Q$ do
4:         $w'_q \leftarrow w_q/wm$
5:     $i \leftarrow 0$, $Y \leftarrow \emptyset$, $\Lambda \leftarrow \emptyset$
6:     while $(1-\varepsilon)^i \geq 1/2|Q|$ do
7:         $Y \leftarrow Y \cup \{(1-\varepsilon)^i\}$
8:         $i++$
9:     for each $q \in Q$ do
10:        $\Lambda_q \leftarrow \emptyset$
11:        for each $y \in Y$ do
12:            $\lambda_{qy} \leftarrow \{p \in \mathbb{R}^2 \mid f(p,q) \geq y\}$
13:            $\Lambda_q \leftarrow \Lambda_q \cup \{\lambda_{qy}\}$
14:        $\Lambda \leftarrow \Lambda \bigcup \{\Lambda_q\}$
15:    compute arrangement $\mathcal{A}(\Lambda)$                ▷ e.g., as in [34]
16:    $\mathcal{P} \leftarrow \mathcal{A}(\Lambda)$.vertices
           ▷ add leftmost-lowest points of faces without vertices
17:    return $\arg\max_{p\in\mathcal{P}} \Phi(Q,p)$
18: end procedure
```

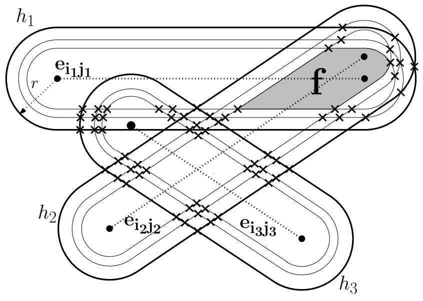


Fig. 4. The arrangement which correspond to probabilistic attacks of 3 links e_{i_1,j_1}, e_{i_2,j_2}, and $e_{i_3 j_3}$, such that each has 3 levels. The shaded region is an example of one of the faces of the arrangement. All vertices of the arrangement are marked with 'x'.

i-th element ($i \geq 0$) is $(1-\varepsilon)^i$. The number of elements of Y is the smallest s satisfying $(1-\varepsilon)^s < 1/(2m)$, where m is the number of components. Note that $s = O\left(\frac{\log m}{\log(1/1-\varepsilon)}\right)$.

We first scale (linearly) the weights associated with the network components, so that $\max_{q\in Q, p\in\mathbb{R}^2} f(q,p)w_q$ is 1; we call this weight the *normalized weight* of the component q and denote it by w'_q. The *y-th level* λ_{qy} of $f(q,\cdot)$ is the set of all points with f-value of at least y; namely, $\lambda_{qy} = \{p \in \mathbb{R}^2 \mid f(q,p) \geq y\}$. Let $\Lambda_q = \{\lambda_{qy} \mid y \in Y\}$ be all the levels of component q, and $\Lambda = \bigcup_q \Lambda_q$. Notice that $|\Lambda| = O(ms)$. Furthermore, let $\Lambda(p) = \{\lambda_{qy} \in \Lambda \mid p \in \lambda_{qy}\}$ be the set of all levels that contain p. The *arrangement* $\mathcal{A} = \mathcal{A}(\Lambda)$ of levels is the subdivision of the plane into vertices, arcs and faces, where the vertices are intersection points of the boundaries of the levels, and the arcs are the maximally connected portions of the boundaries between the vertices and faces are maximally connected regions bounded by arcs; see Fig. 4.

Our algorithm computes the arrangement $\mathcal{A}(\Lambda)$ of these $O(ms)$ levels. For the faces of the arrangement that do not have vertices, we designate the leftmost lowest (lexicographically) point of the face as a vertex. The algorithm then evaluates the function $\Phi(\cdot)$ for each vertex of the arrangement, and chooses the best one. The arrangement can be computed in time $O(m\log(m) + |\mathcal{A}(\Lambda)|)$, where $|\mathcal{A}(\Lambda)|$ is the total number of vertices, arcs and faces in $\mathcal{A}(\Lambda)$ [34], [35]. For simple network components, one can evaluate the function Φ at each vertex naïvely, implying a total complexity of $O(m|\mathcal{A}(\Lambda)|\log(m))$.

One can improve the running time by refining the faces of the arrangement $\mathcal{A}(\Lambda)$ into cells of constant description complexity, i.e, cells that can described by constant number of polynomial inequalities each of constant maximum degree. This ensures that, in order to compute the function Φ on each cell, there only a constant number of updates to perform when moving from one cell to another. Thus, by traversing the cells systematically, one can compute the value at each vertex in a constant time, implying a total complexity of $O(m\log m + |\mathcal{A}(\Lambda)|)$; see [33] for details. Notice that $|\mathcal{A}(\Lambda)| = O\left(m^2 \frac{\log^2 m}{\log^2(1/(1-\varepsilon))}\right)$, when the levels themselves have constant description complexity (i.e, as determined by the function f).

Next, we prove the correctness of our algorithm. First, the optimal point is contained within one of the faces of the arrangement $\mathcal{A}(\Lambda)$, because its Φ-value is at least 1 (that is, the maximum normalized weight). On the other hand, one can verify that Φ's value outside the arrangement is at most $1/2$, since the probability to hit any component is less than $1/2m$. We prove our approximation ratio by fixing a specific network element and looking at a specific face.

Lemma 1. *If two points p_1 and p_2, such that $f(q,p_1) \geq f(q,p_2)$, are in the same face of $\mathcal{A}(\Lambda)$, then, for every component q, $w'_q f(q,p_2) \geq (1-\varepsilon)w'_q f(q,p_1)$, where w'_q is the normalized weight of q.*

Proof: Fix a component q, let y be the level corresponding to the highest index in Y such that $y \geq f(q,p_1)$. Let y' be the next level, i.e, $y' = (1-\varepsilon)y$. The face which contains p_1, p_2 is contained in level y'. Thus, $y \geq f(q,p_1) \geq f(q,p_2) \geq y'$. Since $y' = (1-\varepsilon)y$, we have $f(q,p_2) \geq y' = (1-\varepsilon)y$, but $y \geq f(q,p_1)$, so we get $f(q,p_2) \geq (1-\varepsilon)y \geq (1-\varepsilon)f(q,p_1)$. The proof now follows by rearranging and multiplying by w'_q on both sides. ■

Applying Lemma 1 over all the network elements q, we get $\sum_q w'_q f(q,p_2) \geq (1-\varepsilon)\sum_q w'_q f(q,p_1)$. Since, this approximation is valid for any two points in any face of $\mathcal{A}(\Lambda)$, it also holds for a vertex of the face of $\mathcal{A}(\Lambda)$ and the optimal point.

When the function $f(q,p)$ is given by $f(q,p) = \max\{0, 1-d^2(q,p)\}$, where q is either an amplifier or a link, we can compute the point p^* that maximizes Φ exactly; see Appendix A.

C. Extensions to the compound component case

For compound components, one can apply the algorithm MaxImpactLocation using the function $f(q,\cdot)$ for each compound component q, as defined in Equation (1). Function f takes into account that a single component failure within the compound component is enough to break down the entire component. However, the description complexity of f may not be a constant, thus computing the levels might be difficult (compare the contours of the simple component in Fig. 3(a) with those of the compound component of Fig. 3(b)).

To circumvent this problem, we present an algorithm that finds the $(1-\varepsilon)$-approximated optimal location in two conceptual steps:

1) Compute an arrangement $\mathcal{A}(\Lambda)$ on the $f(v,\cdot)$ for the simple components $v \in V_q$, as defined in Section IV-B, albeit with a slightly finer granularity of $\varepsilon' = 1-\sqrt{1-\varepsilon}$. This arrangement deals with the simple component within each compound component directly. Since $\varepsilon' = \Theta(\varepsilon)$,[3] both the arrangement size and its computation complexity is the same as in Section IV-B.
2) For every vertex p of a face of $\mathcal{A}(\Lambda)$, we compute the value of $f(q,p)$ using Equation (1). Each such computation requires up to $\kappa = \sum_q |V_q|$ updates to previously obtained value, where κ is the sum of the sizes of all compound components. Thus that total complexity of our algorithm is $O(m\log(m) + |\mathcal{A}(\Lambda)|\kappa)$.

We now prove the correctness of the above algorithm. Let p be a vertex in the face of the arrangement which contains an optimal point p^*. From Lemma 1, for every simple component v, we have $f(v,p) \geq (1-\varepsilon')f(v,p^*)$, i.e, $1-f(v,p) \leq 1-(1-\varepsilon')f(v,p^*)$ implying that, for a compound component q, with V_q as components,

$$1-\prod_{v\in V_q}(1-f(v,p)) \geq 1-\prod_{v\in V_q}(1-(1-\varepsilon')f(v,p^*)). \quad (2)$$

We now prove an arithmetic lemma, which enables us to show that $f(q,p) \geq (1-\varepsilon')f(q,p^*)$, for every component q.

Lemma 2. *For every point $p' \in \mathbb{R}^2$, and a compound component q comprised of simple components V_q,*

$$1-\prod_{v\in V_q}(1-(1-\varepsilon')f(v,p')) \geq (1-\varepsilon')f(q,p').$$

Proof: Let $|V_q| = j$ and denote by $u_j = \prod_{v\in V_q}(1-(1-\varepsilon')f(v,p'))$ and $z_j = \prod_{v\in V_q}(1-f(v,p'))$. Hence, we need to show that $1-u_j \geq (1-\varepsilon')(1-z_j)$, or equivalently $u_j-(1-\varepsilon')z_j \leq \varepsilon'$.

We prove the inequality by an induction on j.

For $j=1$, $u_j = (1-(1-\varepsilon')f(q,p'))$ and $z_j = (1-f(q,p'))$, implying that $u_j-(1-\varepsilon')z_j = \varepsilon'$, and the claim follows.

Assume that the claim holds for j. Next we show it also holds for $j+1$, namely:

$$\begin{aligned}
&u_{j+1}-(1-\varepsilon')z_{j+1} = \\
&\quad = u_j(1-(1-\varepsilon')f(v,p')) - (1-\varepsilon')z_j(1-f(v,p')) \\
&\quad = u_j-(1-\varepsilon')z_j-u_j(1-\varepsilon')f(v,p')+(1-\varepsilon')z_jf(v,p') \\
&\quad \leq \varepsilon'-u_j(1-\varepsilon')f(v,p')+(1-\varepsilon')z_jf(v,p) \\
&\quad \leq \varepsilon'-(1-\varepsilon')f(v,p')(u_j-z_j) \\
&\quad \leq \varepsilon',
\end{aligned}$$

where the third inequality is by the induction hypothesis, and the last inequality is since $u_j \geq z_j$, for every j. ∎

Using Equation 2 and applying the Lemma 2 for p^*, we get that $f(q,p) \geq (1-\varepsilon')^2 f(q,p^*)$. The correctness of the algorithm follows by multipying by w'_q and summing over all compound components [4].

D. Improving running time by sampling

In practice, the arrangement $\mathcal{A}(\Lambda)$ does not have too many vertices, i.e, we expect that the number of vertices that appear on the boundary of the union of Λ is $O(|\Lambda|) = O(ms)$. We present a sampling based algorithm which is significantly faster in such practical situations.

Up until now, the errors introduced by our algorithms are a result of *a discretization of function f*, as captured by the different levels and the corresponding arrangement. Once this discretization is done, we find the *optimal solution with respect to the discrete arrangement.* In this section, we relax the requirement of optimal solution within the arrangement, and require only $(1-\delta)$–approximation. Specifically, we show that by carefully choosing the values of Y and δ, we can obtain a $(1-\varepsilon)$ overall approximation with significantly faster algorithm. Our $(1-\delta)$–approximation of the second stage is conceptually equivalent to finding a point with *approximately maximum depth* within the specific arrangement; this problem was solved by Aronov and Har-Peled [36] through sampling, and therefore we will build on their technique.

Specifically, fix a level vector Y, such that the approximation obtained by running the algorithm MaxImpactLocation is $1-\varepsilon'$. Let y_i denote the i-th value in Y, for $1 \leq i \leq s$. We associate the following *weights* α_{qy_i} to each level λ_{qy_i}:

$$\alpha_{qy_i} = \begin{cases} w_q(1-\varepsilon')^{y_s} & i = s \\ w_q(1-\varepsilon')^i - \alpha_{qy_{i+1}} & i < s \end{cases}$$

Furthermore, for each $p \in \mathbb{R}^2$, let $\textsc{Depth}_{\alpha}(p) = \sum_{\lambda_{qy_i}\in\Lambda(p)} \alpha_{qy_i}$. We call this value the *weighted depth* of p with respect to weights α. By the choice of α_{qy_i}, we get $\textsc{Depth}_{\alpha}(p) \geq (1-\varepsilon')\Phi(p)$. Thus, the problem of finding a point that approximately maximizes $\Phi(\cdot)$ reduces to finding a point of maximum weighted depth in $\mathcal{A}(\Lambda)$.

Let $\alpha_{\max} = \max_{q\in Q, y_i\in Y} \alpha_{qy_i}$, we scale the weights of the levels so that the new weights are, $\zeta_{qy_i} = \frac{\alpha_{qy_i}}{\alpha_{\max}}\frac{|\Lambda|}{\varepsilon'}$. This ensures that $\textsc{Depth}_{\zeta}(p^*) \geq |\Lambda|/\varepsilon'$, where p^* is a point that maximizes $\Phi(\cdot)$. We round the weights by setting $\beta_{qy_i} = \lfloor \zeta_{qy_i} \rfloor$. Therefore, for any point p, whose depth with respect to weights ζ is at least $|\Lambda|/\varepsilon'$, we have $\textsc{Depth}_{\beta}(p) \geq$

[3]For every $\varepsilon \in (0,1)$, $\varepsilon > \varepsilon' = 1-\sqrt{1-\varepsilon} \geq \varepsilon/2$.

[4]We can actually prove $f(q,p) \geq (1-\varepsilon')f(q,p^*)$, which is a stronger result and eliminates the need to construct a finer arrangement.

$\textsc{Depth}_{\zeta}(p) - \varepsilon'|\Lambda|/\varepsilon' \geq \textsc{Depth}_{\zeta}(p) - \varepsilon' \textsc{Depth}_{\zeta}(p) = (1-\varepsilon') \textsc{Depth}_{\zeta}(p)$. Thus we can assume that the weights of the levels are integers between 0 and $\lfloor |\Lambda|/\varepsilon' \rfloor$. Let Λ^c be the multiset of levels obtained by making β_{qy_i} copies for each level λ_{qy_i}. The (unweighted) *depth* of a point p in $\mathcal{A}(\Lambda^c)$, which we call $\textsc{Depth}\,(p)$, is the number of copies that contain p. We can now use the algorithm of Aronov and Har-Peled [36], which works with unweighted depth, to compute a point p such that $\textsc{Depth}\,(p) \geq (1-\delta) \textsc{Depth}\,(p^*)$. This implies, $\textsc{Depth}_{\zeta}(p) \geq \textsc{Depth}\,(p) \geq (1-\delta) \textsc{Depth}\,(p^*) \geq (1-\delta)(1-\varepsilon') \textsc{Depth}_{\zeta}(p^*)$, and after rescaling we get $\textsc{Depth}_{\alpha}(p) \geq (1-\delta)(1-\varepsilon') \textsc{Depth}_{\alpha}(p^*) \geq (1-\delta)(1-\varepsilon')^2 \Phi(p^*)$. We choose $\delta = \varepsilon' = \varepsilon/8$, to get the desired $(1-\varepsilon)$-approximation. However, if the copies are stored explicitly, we would need $\Omega(|\Lambda^2|/\varepsilon)$ copies in the worst case. But we show that the copies can be maintained implicity to achieve a faster expected running time, near-linear in $|\Lambda|$.

Conceptually, the algorithm of Aronov and Har-Peled works by first generating a random sample $R^c \subseteq \Lambda^c$, by choosing each copy of a level in Λ^c with probability ρ. A decision procedure is then invoked to check if the maximum (unweighted) depth in the arrangement of $\mathcal{A}(R^c)$ is at least a threshold $\tau = O(\varepsilon^{-2} \log \frac{|\Lambda|}{\varepsilon})$. If the depth is less than the threshold the value of ρ is doubled and this process is is repeated until either the depth is more than τ or the entire arrangement $\mathcal{A}(\Lambda^c)$ is computed, in which case a point of maximum depth is returned. Thus, the number of iterations is $O(\log \frac{|\Lambda|}{\varepsilon})$, and the total running time is $O(T_D \log \frac{|\Lambda|}{\varepsilon})$, where T_D is the expected time to choose the random sample and execute the decision procedure to see if the maximum depth is at least τ. We now show that one can maintain the copies implicitly, and execute the decision procedure so that, $T_D = O(\frac{|\Lambda|}{\varepsilon^2} \log^2 \frac{|\Lambda|}{\varepsilon})$. We use the fact that the number of copies of any level λ_{qy_i} chosen in the random sample R^c follows a *binomial distribution*, $B(\beta_{qy_i}, \rho)$, with parameters β_{qy_i} and ρ. So, for each level λ_{qy_i} we generate a binomial random variate $\nu_{qy_i} \sim B(\beta_{qy_i}, \rho)$ in $O(\log \beta_{qy_i})$ expected time [37], and associate ν_{qy_i}, as the weight of level λ_{qy_i}. Repeating this for each level, we can generate a set $R = \{\lambda_{qy_i} \mid \nu_{qy_i} > 0\}$, of distinct levels in expected time $O(|\Lambda| \log \frac{|\Lambda|}{\varepsilon})$. We then use a *randomized divide-and-conquer* algorithm to check if the maximum weighted depth (with respect to weights ν) in the arrangement $\mathcal{A}(R)$ is at most τ; for e.g., the algorithm of Agarwal et.al [38] can be adopted for this purpose. Since the number of distinct levels in R is at most $|\Lambda|$, the expected running time of this procedure is $O(\frac{|\Lambda|}{\varepsilon^2} \log^2 \frac{|\Lambda|}{\varepsilon})$. Thus, the overall running time to determine a point p, such that $\Phi(p) \geq (1-\varepsilon)\Phi(p^*)$, is $O(\frac{|\Lambda|}{\varepsilon^2} \log^3 \frac{|\Lambda|}{\varepsilon})$.

V. Simultaneous Attacks

We now consider scenarios in which k attacks may happen simultaneously. Our goal is therefore to identify the set $\mathcal{P}$ of k locations, for which $\Phi(Q, \mathcal{P})$ is maximized over all possible choices of k locations.

In general, finding the optimal set $\mathcal{P}$ is NP-hard, since maximizing the value of Φ is a generalization of the well-known *maximum set cover problem* [39]. Nevertheless, we show that the function Φ satisfies certain interesting properties (namely, monotonicity and submodularity). We then present a fast approximation algorithm with a constant approximation ratio, relying on these properties.

A. Algorithm definition

In this section, we formally define the *ε-greedy algorithm* that selects the locations one by one, so as to maximize the gain in Φ (given all past selections).

Specifically, the algorithm works iteratively. Let $\mathcal{P}_{k'} = \{p_1, \ldots, p_{k'}\}$ be the set of locations that were chosen in the first k' iterations. Let $p^* \notin \mathcal{P}_{k'}$ be the location that maximizes $\Delta\Phi(p, \mathcal{P}_{k'}) = \Phi(Q, \mathcal{P}_{k'} \cup \{p\}) - \Phi(Q, \mathcal{P}_{k'})$ over all points $p \in \mathbb{R}^2$; namely, the location that maximized the *revenue* in terms of Φ. The ε-greedy algorithm choses at iteration $k'+1$, a location p such that $\Delta\Phi(p, \mathcal{P}_{k'}) \geq (1-\varepsilon)\Delta\Phi(p^*, \mathcal{P}_{k'})$.

Notice that $\Delta\Phi(p, \mathcal{P}_{k'}) = \sum_{q \in Q} \mu(q, \mathcal{P}_{k'}) f(q,p)$, where $\mu(q, \mathcal{P}_{k'}) = w_q \prod_{p' \in \mathcal{P}_{k'}} (1 - f(q,p'))$. This implies that finding the location that maximizes $\Delta\Phi(p, \mathcal{P}_{k'})$ within factor $(1-\varepsilon)$ can be done by applying the algorithms of Section IV after modifying the weights of the components to $\mu(q, \mathcal{P}_{k'})$ (instead of w_q).

We now show that this greedy algorithm yields a constant factor approximation.

B. Performance evaluation

As mentioned previously, computing Φ exactly is NP-hard. However, the function $\Phi(Q, \cdot)$ has two key properties, namely, monotonicity and submodularity, which are useful in developing an approximation algorithm. Intuitively, the expected number of links destroyed only increases when there are more attacks, and hence $\Phi(Q, \cdot)$ is monotonically non-decreasing, i.e., $\Phi(Q, \mathcal{P}_1) \leq \Phi(Q, \mathcal{P}_2)$, for any set $\mathcal{P}_2 \supseteq \mathcal{P}_1$. (More formally, this property stems from the fact that $\mu(q, \mathcal{P}_2) \leq \mu(q, \mathcal{P}_1)$, for any $q \in Q$.) The function $\Phi(Q, \cdot)$ also exhibits the "law of diminishing returns" property or *submodularity*. That is, for a given attack p and two sets of attacks $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}_2 \supseteq \mathcal{P}_1$, the incremental gain when applying p after $\mathcal{P}_2$ is smaller than the incremental gain when applying p after $\mathcal{P}_1$. This property is captured by the following lemma.

Lemma 3. *$\Phi(Q, \cdot)$ is a submodular function. Namely, for any two set of points $\mathcal{P}_1$ and $\mathcal{P}_2$, such that $\mathcal{P}_2 \supseteq \mathcal{P}_1$, and any point $p \in \mathbb{R}^2$, $\Phi(Q, \mathcal{P}_1 \cup \{p\}) - \Phi(Q, \mathcal{P}_1) \geq \Phi(Q, \mathcal{P}_2 \cup \{p\}) - \Phi(Q, \mathcal{P}_2)$, i.e, $\Delta\Phi(p, \mathcal{P}_1) \geq \Delta\Phi(p, \mathcal{P}_2)$.*

Proof: If $p \in \mathcal{P}_2$, then $\Phi(Q, \mathcal{P}_2 \cup \{p\}) - \Phi(Q, \mathcal{P}_2) = 0$ and the claim follows trivially. So, assume $p \notin \mathcal{P}_2$. Notice that $\Delta\Phi(p, \mathcal{P}_2) = \sum_{q \in Q} \mu(q, \mathcal{P}_2) f(q,p)$. Similarly, $\Delta\Phi(p, \mathcal{P}_1) = \sum_{q \in Q} \mu(q, \mathcal{P}_1) f(q,p)$. Since $\mu(q, \mathcal{P}_2) \leq \mu(q, \mathcal{P}_1)$ for any $q \in Q$, the claim follows. ■

These two properties immediately imply that, a perfect greedy algorithm (that is, ε-greedy algorithm with $\varepsilon = 0$) achieves a $(1 - 1/e)$-approximation [40]. Since our selection at each step is approximated, the overall approximation ratio of ε-greedy is $(1 - \frac{1}{e^{1-\varepsilon}})$ [41], for any $0 < \varepsilon < 1$. Notice that our proof holds for any type of components (simple or compound).

VI. Networks with a Protection Plan

When building a resilient network, a common paradigm is to provide a *protection plan* for the significant lightpaths, as to ensure their continuous service in face of network failures. Common approaches include 1+1 dedicated protection, in which, conceptually, the data is sent twice along a primary and backup lightpaths, implying that data is lost only when these two lightpaths fail simultaneously. A 1:1 dedicated protection, on the other hand, allows using the backup lightpath for low-priority traffic. Once the primary lightpath fails, traffic is shifted to the backup lightpath, and the low-priority traffic is disregarded.

When designing a protection plan, geographical correlations are often taken into account. The primary and backup lightpaths tend to be fiber-disjoint or even not to share the same shared risk group (SRG); for example, the fibers should not be close in physical proximity. Thus, it is likely that a reasonable protection plan will cope with a single attack. In this section, we are evaluating the resilience of a protection plan to *two simultaneous attacks*.

Formally, we are given pairs of lightpaths (π_i, π'_i), where π_i is the primary path and π'_i is the backup path. Let T_i be the high-priority traffic on these lightpaths, and t_i be the low priority traffic on these lightpaths (for 1+1 protection, t_i is always 0). Thus, one loses t_i when either π_i or π'_i fails, and $T_i + t_i$ if both fail at once. Hence, given two locations p_1 and p_2, the *expected loss on the i-th pair* (obtained by case analysis) is

$$\begin{aligned}\Phi_i(p_1,p_2)= \; & t_i\left[1-g(\pi_i,p_1)g(\pi_i,p_2)g(\pi'_i,p_1)g(\pi'_i,p_2)\right] \\ & +T_i\left[f(\pi_i,p_1)f(\pi'_i,p_1)+f(\pi_i,p_2)f(\pi'_i,p_2)\right. \\ & \quad -f(\pi_i,p_1)f(\pi'_i,p_1)f(\pi_i,p_2)f(\pi'_i,p_2) \\ & \quad +g(\pi_i,p_1)f(\pi'_i,p_1)f(\pi_i,p_2)g(\pi'_i,p_2) \\ & \quad \left.+f(\pi_i,p_1)g(\pi'_i,p_1)g(\pi_i,p_2)f(\pi'_i,p_2)\right], \end{aligned} \tag{3}$$

where $g(\pi, p)$ denotes $1 - f(\pi, p)$.

For the entire network, we get $\Phi(p_1, p_2) = \sum_i \Phi_i(p_1, p_2)$. We next show how to compute the pair of attacks p_1, p_2 that maximizes $\Phi(p_1, p_2)$. Alternatively, one can measure the *worst-case vulnerability* of the protection plan by the value of $\Phi(p_1, p_2)$ and use this value to compare the resilience of alternative plans.

The algorithm is a generalization of the algorithm of MaxImpactLocation , with the following modification: Instead of computing the value of Φ for each vertex of a face of the arrangement, we consider all possible pairs of vertices and compute the value of Φ as though the attacks happened in these locations. This implies that the running time is quadratic in the size of the arrangement. Moreover, the approximation ratio can degrade up to a factor of $(1-\varepsilon)^4$, as we multiply four terms in Equation (3). This can be solved by a refined arrangement defined with $\varepsilon' = 1 - \sqrt[4]{1-\varepsilon} = \Theta(\varepsilon)$, with no extra complexity penalty.

VII. Networks with Restoration Algorithms

An alternative approach to network survivability is to devise *dynamic restoration scheme*, which, upon a network failure, is able to re-route the traffic so as to avoid data loss. In general, devising efficient restoration algorithms, especially when required to handle large-scale failure, is a challenging task. Dynamic restoration schemes are more efficient in utilizing network capacity, but have slower recovery time and often cannot guarantee quality of restoration.

Clearly, the optimal quality of restoration (in terms of post-attack traffic carried by the network between predetermined source nodes and target nodes) is the *maximum flow* of residual network, and therefore finding the most vulnerable location in such setting is equivalent to finding the location whose corresponding attack minimizes the *expected maximum flow*. However, under a probabilistic setting, finding the expected maximum flow of a graph is #P-complete. This is true even if all edges have unit weight (that is, a connectivity problem), and even if the graphs are planar. It is important to notice that although one is not directly required to compute the exact *value* of the expected maximum flow in order to find the most vulnerable location, and in some cases one can compare the effect of two location without such computation (e.g., when the failure probability of one location dominates the other), in the general case such computation is needed (for example, when two locations affect disjoint sets of links and there is no third location that can be used as a comparison), yielding the following result:

Theorem 1. *Computing the most vulnerable location in term of expected maximum flow is #P-complete.*

Proof: The proof is by reduction from the $s{-}t$ expected maximum flow problem, in which one need to compute the expected maximum flow from node s to node t. We restrict our graphs to be with capacity 1 and all edge failure probabilities to be the same (in our case, $1/2$). It is known that the problem is still #P-complete [42]. In addition, by enumerating over all possible combinations, and since for each instance the maximum flow is integral, one can verify that the *value* of the expectation is a multiply of $\frac{1}{2^n}$. Trivially, the expected maximum flow is bounded by n.

Let G be such a network of n links (all their weights are 1) and let R be its physical diameter. Assume, without loss of generality, that $d(s, t) = R$. We define the following f-function to determine geographical failures:

$$f(p,q) = \begin{cases} \frac{1}{2} & d(p,q) \le R \\ 0 & \text{otherwise} \end{cases}$$

Note that f induces a uniform reliability problem on G.

We next show how to construct a family of graphs so that by finding their most vulnerable location (in terms of expected $s{-}t$ flow), one can compute the value of the expected $s{-}t$ flow on the original graph, hence establishing that finding the most vulnerable location in in #P. Note that our family graphs will contain exponential number of graphs (each of polynomial size), however we will need to consider only a polynomial number of them (and can construct them on-the-fly).

We first consider a simple serial-parallel construction: Given a graph G, where the probability that s' is connected to t' is C,

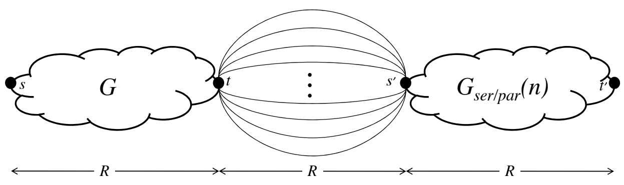


Fig. 5. Illustration of the proof of Theorem 1.

then *(i)* if one add an edge (s'', s') then the probability that s'' is connected to t' is $C/2$; *(ii)* if one add an edge (s', t') then the probability that s' is connected to t' is $1/2 + C/2$. With a combination of x serial-parallel compositions, one build a sequence of graphs, $G_{ser/par}(i)$, (for any $1 \leq i < 2^x$) whose distinguished nodes are connected with probability $i/2^x$. We scale the physical length of the edges so that the physical size of the entire graph is R (implying that a single attack can affect all edges). In the rest of the proof, we fix x to be $2n$.

Let F be the value of the maximum flow of G had all the edges do not fail (this can be computed in polynomial time). At each iteration, we use some graph $G_{ser/par}(i)$, where the capacity of all its edges is F. We connect nodes t and s' by n^2 parallel edges of capacity F and length larger than R (see Fig 5). Note that the expected minimum cut size (and hence induced bottleneck) on $G_{ser/par}(i)$ is exactly $F\frac{i}{2^{2n}}$. Our reduction follows by a binary search on the parameter i of the family of the graphs: In general, at each iteration, we compute the most vulnerable location in term of expected maximum flow between s to t' and distinguish between five different cases:

1) The location affects only G. This implies that the expected s-t flow is strictly less than $F\frac{i+1}{2^{2n}}$. We will continue in the next iteration with smaller value of i (in a binary search manner).
2) The location affects only G and the n^2 parallel edges. Failure on the parallel edges affects the maximum flow if and only if all edge fail. Since this happens with probability $\frac{1}{2^{n^2}}$, the expected s-t flow is strictly less than $F\frac{i+1}{2^{2n}}$. We will continue in the next iteration with smaller value of i (in a binary search manner).
3) The location affects only the n^2 parallel edges. This implies that the expected maximum flow is more than $F(1 - \frac{1}{2^{n^2}})$, which by the granularity of the values of the expected maximum flow implies that it is F (and the algorithm finishes).
4) The location affects only $G_{ser/par}(i)$. This implies that the expected s-t flow is strictly more than $F\frac{i-1}{2^{2n}}$. We will continue in the next iteration with higher value of i (in a binary search manner).
5) The location affects $G_{ser/par}(i)$ and the n^2 parallel edges. This, again, implies that the expected s-t flow is strictly more than $F\frac{i-1}{2^{2n}}$. We will continue in the next iteration with higher value of i (in a binary search manner).

We start with $i = 2^{2n-1}$. When the binary search completes, we get the accurate value of G's expected maximum s-t flow. The number of iterations is bounded $O(\log 2^{2n}) = O(n)$ and therefore our reduction is polynomial, as required. ■

Essentially, this hardness result implies that finding the most vulnerable location requires exponential-time algorithm *in the number of affected links*. Such algorithms might be feasible to implement when the number of these links is bounded by a small constant s. The most intuitive algorithm is by a *complete state enumeration*. Such an algorithm considers one candidate location at a time (obtained by the corresponding arrangement, as in Section IV); each location defines a probabilistic graph $G = (V, E)$ where for every edge $e \in E$, $\Pr_e$ denote the probability that e fails. Let E_1 denote the edges with zero failure probability, and E_2 the rest of the edges. The algorithm will enumerate on all subsets of E_2 and for each such subset S, compute the probability for such failure pattern $\Pr_S = \prod_{e \in S} \Pr_e \prod_{e \in E_2 \setminus S}(1 - \Pr_e)$. Then, it computes the maximum flow F_S in $G_S = (V, E_1 \cup S)$. The expected maximum flow is therefore $\sum_{S \subseteq E_2} \Pr_S \cdot F_S$, and the computation requires $2^{|E_2|} \leq 2^s$ maximum-flow computations.[5]

Alternative techniques, such as graph simplification, graph factoring, and inclusion-exclusion based approaches were also studied in the past [43]. However, all the suggested algorithms still requires exponential running time.

VIII. Numerical Results

We have run the algorithms of Section IV on three different networks within the continental USA: Level3's network of 230 links [44], Qwest's fiber-optic network of 181 links [6], and XO Communication's long-haul network of 71 links [45]. We obtained lightpaths information of the last two networks. In addition, for Qwest's network, we also have the transmission rate of the individual lightpaths, and we used them to determine the values of t_π.

We conducted our simulations with the five different accuracy values ε for the simple component: $0.1, 0.2, \ldots, 0.5$. For compound components, we used three values of ε' (recall Section IV-C): 0.1, 0.2 and 0.3; these translate to roughly 0.8-, 0.65, and 0.5-approximation. In addition, we considered five different attack radii, ranging between 60 and 300 miles. Finally, two f functions were used: a function that decreases *linearly* with the distance, and a function that follows a *Gaussian* distribution (see Section III).

We first compared the values of Φ for different accuracy levels ε of our algorithms. Table I shows the results for simple and compound components respectively when the attack radius (resp., standard deviation of the attack radius) is 180 miles for the linear (resp., gaussian) f-function. Here, Φ_L denotes the utility function Φ under linear probability function and Φ_G denotes the utility function under a Gaussian one. Our results show *no perceptible change in Φ when ε is changed, neither*

[5]Note that the arrangement of Section IV induces only an approximate solution. In this case, we need to scale the error parameter ε inversely with s to avoid accumulating errors in the computation

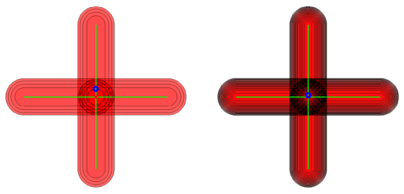


Fig. 6. Example where Φ varies significantly with ε. MaxImpactLocation selects attack location with $\Phi = 2.677$ for $\varepsilon = 0.5$ (see arrangement on the left) and $\Phi = 3.788$–approximately 40% more–for $\varepsilon = 0.1$ (arrangement on the right). Notice that the fiber-links (in green) do not intersect (but their end-points are in close proximity).

TABLE I
Values of Φ_L (Linear f-function) and Φ_G (Gaussian f-function) for an attack radius/standard deviation of 180 miles.

	Level3		**Qwest**		**XO**	
ε	Φ_L	Φ_G	Φ_L	Φ_G	Φ_L	Φ_G
0.1	20.52	69.41	14.13	37.24	6.1	15.68
0.2	20.52	69.41	14.13	37.24	6.09	15.68
0.3	20.52	69.41	14.13	37.24	6.09	15.68
0.4	20.52	69.41	14.13	37.24	6.08	15.68
0.5	20.52	69.41	14.13	37.24	6.09	15.68

(a) Simple components (fibers)

	Qwest		**XO**	
ε	Φ_L	Φ_G	Φ_L	Φ_G
0.2	475.75	615.18	11.15	15.82
0.35	475.75	615.18	11.15	15.82
0.5	475.75	615.18	11.15	15.82

(b) Compound components (lightpaths)

for links nor for lightpaths. This conclusion holds for the three networks under considerations, for both f-functions and for various attack radii. This may be explained by the fact that, in these networks, the location found by the algorithm lies on, or extremely close to, a fiber link, thus avoiding the worse-case (in terms of approximation ratio). However, there do exist cases in which Φ vary significantly with ε: in Fig. 6, four links are placed as shown; when the f-function is Gaussian with a standard deviation of 2.2 units and $\varepsilon = \{0.1, 0.5\}$, the values of Φ computed by MaxImpactLocation are 3.788 and 2.677, respectively, which is a significant difference. While cases of this sort are certainly possible, our results show that, *in practice, the dependence on ε is very limited*. This implies that for realistic fiber-optic networks, one can apply the much faster 0.5-approximation algorithms and obtain very close to optimal results.

Finally, Fig. 7 and Fig. 8 show the change in Φ with the attack radius for a linear f-function and with the standard deviation of attack's radius, for Gaussian f-functions respectively. We normalized the value of Φ, so that 100% implies the sum of the weights of all network components. As can be seen, the marginal gain in increasing the attack radius is limited, and even small attacks with radius of 60 miles can cause large damage if they are placed in vulnerable locations.

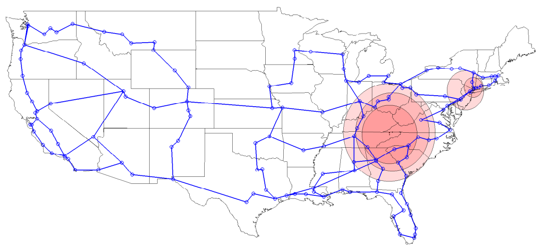


Fig. 9. Locations of attacks found by our algorithms, for various attack radii.

Fig. 9 depicts an example of these locations on the Qwest network, a Gaussian f-function on simple components, and various attack radii.

IX. Conclusions

In this paper, we provided a unified framework to identify vulnerable point(s), given a WDM network embedded in the Euclidean plane. A unique feature of our framework is its ability to cope with a *wide range of probabilistic attack and failure models*.

The basic building block of our framework is the algorithm MaxImpactLocation, which locates efficiently a point in the plane that causes arbitrarily close to maximum impact on a network comprised of simple components. By its tolerance factor ε, MaxImpactLocation trades accuracy with running time. We further extended and improved MaxImpactLocation in various ways that allow it to deal with compound components, simultaneous attacks, networks equipped with a protection plan and to deal faster with simpler networks or probabilities. We evaluated its performance also by simulation on three real WDM networks. Our numerical results show, quite surprisingly, that MaxImpactLocation finds a location very close to optimal, even when taking a high tolerance factor ε (e.g., when it runs very fast but with a loose guarantee on the quality of its output). This makes MaxImpactLocation an even more attractive tool for assessing network resilience.

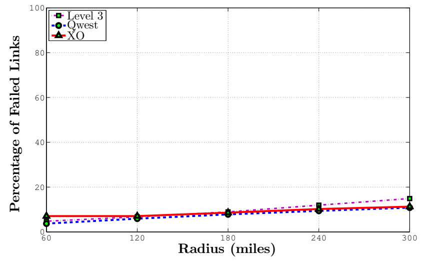
Future research directions include developing efficient planning methods for geographically-resilient networks and investigating the effect of adding minimal infrastructure (e.g., lighting-up dark fibers) on network resilience. Finally, we plan to determine how to use low-cost shielding for existing components to mitigate large scale physical attacks.

References

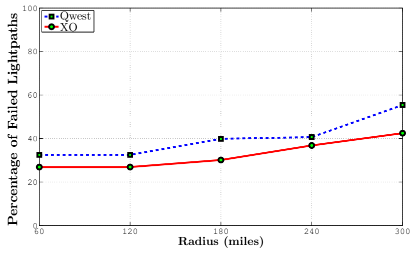
[1] J. Borland, "Analyzing the Internet collapse," *MIT Technology Review*, Feb. 2008. [Online]. Available: http://www.technologyreview.com/Infotech/20152/?a=f

[2] J. S. Foster, E. Gjelde, W. R. Graham, R. J. Hermann, H. M. Kluepfel, R. L. Lawson, G. K. Soper, L. L. Wood, and J. B. Woodard, "Report of the commission to assess the threat to the United States from electromagnetic pulse (EMP) attack, critical national infrastructures," Apr. 2008.

[3] C. Wilson, "High altitude electromagnetic pulse (HEMP) and high power microwave (HPM) devices: Threat assessments," CRS Report for Congress, July 2008. [Online]. Available: http://www.ntia.doc.gov/broadbandgrants/comments/7926.pdf

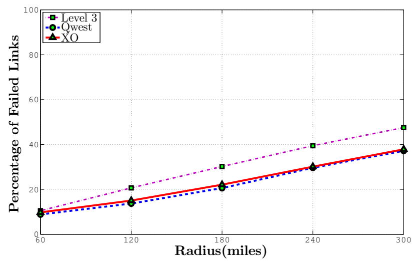


(a) *Simple Components*

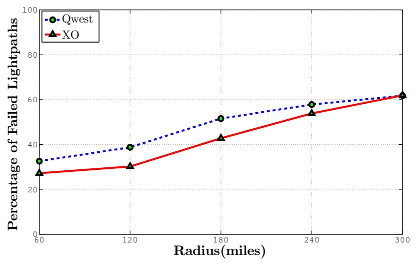


(b) *Compound Components*

Fig. 7. Variation of Φ, normalized by the sum over the entire network, with the attack radius of a *linear* failure probability function.



(a) *Simple Components (Individual Links)*



(b) *Compound Components (Lightpaths)*

Fig. 8. Variation of Φ, normalized by the sum over the entire network, with the standard deviation of a *Gaussian* failure probability function.

[4] W. R. Forstchen, *One Second After*. Tom Doherty Associates, 2009.

[5] W. Wu, B. Moran, J. Manton, and M. Zukerman, “Topology design of undersea cables considering survivability under major disasters,” in *Proc. WAINA*, May 2009.

[6] Qwest, Network Map. [Online]. Available: http://www.qwest.com/largebusiness/enterprisesolutions/networkMaps/

[7] R. Bhandari, *Survivable networks: algorithms for diverse routing*. Kluwer, 1999.

[8] C. Ou and B. Mukherjee, *Survivable Optical WDM Networks*. Springer-Verlag, 2005.

[9] A. K. Somani, *Survivability and Traffic Grooming in WDM Optical Networks*. Cambridge University Press, 2005.

[10] O. Crochat, J.-Y. Le Boudec, and O. Gerstel, “Protection interoperability for WDM optical networks,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 384–395, 2000.

[11] A. Narula-Tam, E. Modiano, and A. Brzezinski, “Physical topology design for survivable routing of logical rings in WDM-based networks,” *IEEE J. Sel. Areas Commun.*, vol. 22, no. 8, pp. 1525–1538, Oct. 2004.

[12] D. Zhou and S. Subramaniam, “Survivability in optical networks,” *IEEE Network*, vol. 14, no. 6, pp. 16–23, Nov.-Dec. 2000.

[13] D. Magoni, “Tearing down the Internet,” *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 949–960, Aug. 2003.

[14] L. K. Gallos, R. Cohen, P. Argyrakis, A. Bunde, and S. Havlin, “Stability and topology of scale-free networks under attack and defense strategies,” *Phys. Rev. Lett.*, vol. 94, no. 18, 2005.

[15] A. L. Barabasi and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, October 1999.

[16] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, “Assessing the impact of geographically correlated network failures,” in *Proc. IEEE MILCOM*, Nov. 2008.

[17] ——, “Assessing the vulnerability of the fiber infrastructure to disasters,” in *Proc. IEEE INFOCOM*, Apr. 2009.

[18] S. Neumayer and E. Modiano, “Network reliability with geographically correlated failures,” in *Proc. IEEE INFOCOM*, Mar. 2010.

[19] A. Sen, B. Shen, L. Zhou, and B. Hao, “Fault-tolerance in sensor networks: a new evaluation metric,” in *Proc. IEEE INFOCOM*, 2006.

[20] A. Sen, S. Murthy, and S. Banerjee, “Region-based connectivity: a new paradigm for design of fault-tolerant networks,” in *Proc. HPSR*, 2009.

[21] A. F. Hansen, A. Kvalbein, T. Cicic, and S. Gjessing, “Resilient routing layers for network disaster planning,” in *Proc. ICN*, Apr. 2005.

[22] M. M. Hayat, J. E. Pezoa, D. Dietz, and S. Dhakal, “Dynamic load balancing for robust distributed computing in the presence of topological impairments,” *Wiley Handbook of Science and Technology for Homeland Security*, 2009.

[23] G. Clapp, R. Doverspike, R. Skoog, J. Strand, and A. V. Lehmen, “Lessons learned from CORONET,” in *OSA OFC*, Mar. 2010.

[24] IETF Internet Working Group , “Inference of Shared Risk Link Groups,” November 2001, Internet Draft. [Online]. Available: http://tools.ietf.org/html/draft-many-inference-srlg-02

[25] D. Bienstock, “Some generalized max-flow min-cut problems in the plane,” *Math. Oper. Res.*, vol. 16, no. 2, pp. 310–333, 1991.

[26] C. A. Phillips, “The network inhibition problem,” in *Proc. STOC*, 1993.

[27] C. Burch, R. Carr, S. Krumke, M. Marathe, C. Phillips, and E. Sundberg, “A decomposition-based pseudoapproximation algorithm for network flow inhibition,” in *Network Interdiction and Stochastic Integer Programming*. Springer, 2003, ch. 3, pp. 51–68.

[28] A. Schrijver, “On the history of combinatorial optimization (till 1960),” in *Handbook of Discrete Optimization*. Elsevier, 2005, pp. 1–68.

[29] A. Pinar, Y. Fogel, and B. Lesieutre, “The inhibiting bisection problem,” in *Proc. ACM SPAA*, June 2007.

[30] R. L. Church, M. P. Scaparra, and R. S. Middleton, “Identifying critical infrastructure: the median and covering facility interdiction problems,” *Ann. Assoc. Amer. Geographers*, vol. 94, no. 3, pp. 491–502, 2004.

[31] R. L. Francis, *Facility Layout and Location: An Analytical Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[32] Z. A. Melzak, *Companion to Concrete Mathematics; Mathematical Techniques and Various Applications*. Wiley, New York, 1973.

[33] A. Vigneron, “Geometric optimzation and sums of algebraic functions,” in *Proc. SODA*, 2010.

[34] M. Sharir and P. Agarwal, *Davenport-Schinzel Sequences and their Geometric Applications*. Cambridge University Press, 1995.

[35] N. M. Amato, M. T. Goodrich, and E. A. Ramos, “Computing the arrangement of curve segments: divide-and-conquer algorithms via sampling,” in *Proc. ACM-SIAM SODA*, 2000, pp. 705–706.

[36] B. Aronov and S. Har-Peled, “On approximating the depth and related problems,” in *Proc. ACM-SIAM SODA*, Jan. 2005.

[37] L. Devroye, *Non-Uniform Random Variate Generation*. Springer-Verlag, 1968.

[38] P. K. Agarwal, D. Z. Chen, S. K. Ganjugunte, E. Misiołek, M. Sharir,

and K. Tang, "Stabbing convex polygons with a segment or a polygon," in *Proc. ESA*, Sept. 2008.

[39] D. Hochbaum and A. Pathria, "Analysis of the greedy approach in problems of maximum k-coverage," *Naval Research Logistics (NRL)*, vol. 45, no. 6, pp. 615–627, 1998.

[40] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions - i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, December 1978.

[41] P. R. Goundan and A. S. Schulz, "Revisiting the greedy approach to submodular set function maximization," *Working paper*, 2008.

[42] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network reliability," in *Network Models*, ser. Handbooks in Operations Research and Management Science. Elsevier, 1995, vol. 7, ch. 11, pp. 673 – 762.

[43] C. J. Colbourn, *The Combinatorics of Network Reliability*. Oxford University Press, 1987.

[44] Level 3 Communications, Network Map. [Online]. Available: http://www.level3.com/interacts/map.html

[45] XO Communications, Network Map. [Online]. Available: http://www.xo.com/about/network/Pages/maps.aspx

APPENDIX A
EXACT SOLUTION FOR SIMPLE COMPONENTS UNDER SUM OF SQUARE DISTANCE f-FUNCTION

In this appendix, we present the solution when the components are optical fibers (that is, the links in the network graph), and for each link e and point $p \in \mathbb{R}^2$, $f(e,p) = \max\{0, 1 - d^2(e,p)\}$, where $d^2(\cdot,\cdot)$ denotes the square of the Euclidean distance. The same technique works also when the simple components are amplifiers (namely, points in the plane).

For a link e, the *support line* of e is the (infinite) line that coincides with e (namely, e's continuation beyond its endpoints). Furthermore, let $h_e = \{p \in \mathbb{R}^2 \mid d^2(e,p) \leq 1\}$. Notice that h_e is shaped as a *hippodrome* (see, for example, Fig. 4), which can be refined into three regions: the two semicircles around the endpoints of e and the rectangle in between. Given a point $p \in h_e$, the nearest point on e to p is either one of the endpoints (in case p is in one of the semicircle) or the nearest point to p on e's support line (in case p is in the rectangle area). Thus, we associate to each refined region (semicircle or rectangle) either the corresponding endpoint or the support line. Let $\tilde{H}$ denote the set of refined regions obtained from all the hippodromes. Let $\mathcal{A}(\tilde{H})$ be the arrangement of the refined regions. For each cell c of $\mathcal{A}(\tilde{H})$, let L_c (resp., B_c) be the set of all support lines (resp., endpoints) associated with points in c.

We first describe a procedure that given a cell L_c and B_c (for a cell c) finds the point that maximizes Φ. More specifically, let each support line $l_i \in L_c$ be $\alpha_i x + \beta_i y + \gamma_i = 0$, and denote each point $b_i \in B_c$ by (b_{ix}, b_{iy}). Our goal is therefore to find the following point $p_c \in \mathbb{R}^2$:

$$p_c = \text{argmax}_{p \in \mathbb{R}^2} \left(\sum_{l_i \in L_c} w_i \left(1 - \frac{(\alpha_i p_x + \beta_i p_y + \gamma_i)^2}{(\alpha_i^2 + \beta_i^2)}\right) + \sum_{b_i \in B_c} w_i \left(1 - (p_x - b_{ix})^2 - (p_y - b_{iy})^2\right) \right),$$

where w_i is the weight of link i. The gradient vector for the objective function above is given by:

$$-\begin{bmatrix} \sum_{l_i \in L_c} 2w_i \frac{(\alpha_i p_x + \beta_i p_y + \gamma_i)\alpha_i}{\alpha_i^2 + \beta_i^2} + \sum_{b_i \in B_c} 2w_i (p_x - b_{ix}) \\ \sum_{l_i \in L_c} 2w_i \frac{(\alpha_i p_x + \beta_i p_y + \gamma_i)\beta_i}{\alpha_i^2 + \beta_i^2} + \sum_{b_i \in B_c} 2w_i (p_y - b_{iy}) \end{bmatrix}$$

Hence, we have two linear equations which can be equated to zero to determine a closed-form expression and the value of p_c. With this procedure at our disposal, we compute p_c for each cell c of the arrangement and return the point which attains the maximum objective value.

To improve the running time, instead of recomputing p_c for each cell from the scratch, we first compute a vertical decomposition of the arrangement and traverse the cells systematically [33]. There are a constant number of changes between adjacent cells, and therefore by optimum for the new cell can be computed in constant time, by updating the the closed form expression for the optimal point and recompute the optimal value for the new support lines and end points (this takes constant time, as there are only a constant number of changes). The total running time is in the order of the time it takes to construct the arrangement, namely, $O(m \log(m) + |\mathcal{A}(H)|)$.

We next prove the optimality of the algorithm. Let p^* be the optimal solution, and let c be the refined cell containing p^*; p^* must be in one of the cells of $\mathcal{A}(\tilde{H})$, since otherwise $\Phi(p^*) = 0$. Let p^c be the point optimal for L_c, B_c. For any point within the refined cell the weighted sum of square distance to the links is exactly the weighted sum of square distances to the support lines L_c and the end-points B_c, thus $\Phi(p_c) \geq \Phi(p^*)$. Moreover, p_c must lie within the arrangement $A(\tilde{H})$, as otherwise the distance to the end-points and the support lines only increases for points outside the arrangement. Therefore p_c is a valid solution, and is optimal.

We note that extending the above solution to handle compound components as lightpaths, is difficult as one would have to potentially solve a polynomial with high degree, on the order of maximum number of links along a path.