Critical regions and region-disjoint paths in a network

Stojan Trajanovski, Fernando A. Kuipers, Piet Van Mieghem
Delft University of Technology
Delft, The Netherlands
{S.Trajanovski, F.A.Kuipers, P.F.A.VanMieghem}@tudelft.nl

Aleksandar Ilić
Facebook Inc.
Menlo Park, CA, USA
ailic@fb.com

Jon Crowcroft
University of Cambridge
Cambridge, UK
Jon.Crowcroft@cl.cam.ac.uk

***Abstract*—Due to the importance of communication networks to society, it is pertinent that these networks can withstand failures. Improving the robustness of a network usually requires installing redundant resources, which is very costly. Network providers are consequently less inclined to take robustness measures against failures that are unlikely to manifest, like several failures coinciding simultaneously in different geographic regions of their network. Protecting against single regional failures is more realistic. Network robustness, in terms of connectivity properties, also requires survivability algorithms to quickly reroute traffic affected by a network failure.**

In this paper, we consider a network embedded in a plane and study the problem of finding a circular region with radius r in that plane that would cause the biggest network degradation if all nodes within that particular region were to be destroyed. We propose a polynomial time algorithm for finding such critical regions. In addition, we develop a region-aware network augmentation technique to decrease the impact of a critical-region failure. We subsequently consider the region-disjoint paths problem, which asks for two paths with minimum total weight between a source (s) and a destination (d) that cannot both be cut by a single circular regional failure of radius r (unless that failure includes s and d). We prove that the region-disjoint paths problem is NP-hard and propose and evaluate a heuristic algorithm for it.

***Index Terms*—Survivability; Region-disjoint paths; Determining critical regions; Network augmentation**

I. Introduction

In communication networks, when network failures occur, they usually consist of (a) one node or link failure (e.g., when a cable is cut); (b) an entire affected region (e.g., due to the failure of a base station, a natural disaster, or a targeted attack) or (c) cascading failures [1]. A fourth category of multiple network failures at several geographic locations occurring in a short time interval is less likely to take place, even though it could be the result of a series of malicious attacks. Some studies [2], [3], [4] do consider multiple network failures to test the robustness of a network. These studies typically either randomly take out a fraction of the nodes or links and study the connectivity properties of the remaining network [2] or a targeted attack is simulated in which several of the most important nodes or links (according to some criterion) are removed from the network [5]. Increasing the robustness of a network usually requires installing redundant resources or over-provisioning, which is unfortunately very costly. In general, network providers are only inclined to take preventive robustness measures for events that have a realistic chance of occurring, such as a single, rather than multiple, regional failure.

If a network is robust, in terms of connectivity, survivability algorithms are needed to exploit this robustness by quickly rerouting traffic affected by a network failure. Among the three categories, the single link- or node-failure scenario has been most studied or assumed by the research community and various link- or node-disjoint paths algorithms have been devised to find two disjoint paths, where one backup path takes over when the other primary path fails (e.g., see [6], [7]).

This paper considers the context of a single regional failure by investigating two problems. We start with the problem of finding a *critical region* - identified by a circular area of radius r - in the network whose failure (of the nodes inside that region) would be most disruptive to the network. For cases in which the network is deemed to be too vulnerable to the failure of a critical region, we propose a technique to augment the network such that the vulnerability is reduced. The second problem under consideration asks for two *region-disjoint paths*, between source s and destination d, with a minimum total weight, such that each intermediate node (between s and d) from the first path is on a distance greater than $2r$ from every intermediate node in the second path. In this case, no single regional failure can destroy both paths unless that failure affects s and d (Fig. 1).

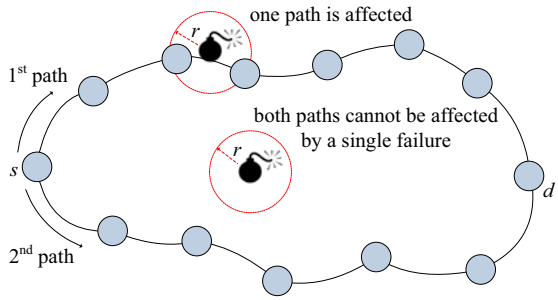


Fig. 1: An example of two region-disjoint paths that cannot both be cut by the failure of a single region with radius r.

Our main contributions are:

(a) A polynomial time algorithm that determines critical

regions in a network for any network metric;
(b) A network augmentation technique for improving the robustness against regional failures;
(c) A proof that the region-disjoint paths problem is strongly NP-hard and, subsequently, a polynomial time heuristic for the region-disjoint paths problem.

Neumayer *et al.* [8] have also considered the problem of determining critical regions. However, in their model, they assume that the failure of a region affects all nodes and links, even those that are only traversing and have no terminating nodes in that region. While this may correspond to a realistic scenario in the case of an earthquake, it may be too restrictive for countries not on a fault line and for other scenarios, e.g. for wireless or radio networks, floods, etc. In the model considered in this paper, we therefore consider that the failure of a region disrupts all nodes and their attached links in that region, but not any traversing links.

This paper is structured as follows. Section II presents the region-based model and problem definitions. In Section III, we provide a polynomial time algorithm for detecting critical regions and study a network augmentation strategy to make the network more resilient against the failure of a region. The NP-hardness of the region-disjoint paths problem is proved and a heuristic is proposed in Section IV. The evaluation of our proposed algorithms is conducted in Section V. An overview of the state-of-the-art is given in Section VI. We conclude in Section VII.

II. The Model and Problem Statement

We start with a presentation of our network model and the problems considered.

Model: We represent a network as a weighted (directed or undirected) graph $G(\mathcal{N}, \mathcal{L})$ in a plane consisting of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L links. Each node $i \in \mathcal{N}$ has two-dimensional coordinates (x_i, y_i). The distance between two nodes u and v is denoted by $d(u, v)$. The weight of a link $(i, j) \in \mathcal{L}$ is denoted by $w(i, j)$. The weight may reflect the distance, but it could also reflect another metric.

We define the *critical region* $\mathcal{C}(O(x_j, y_j), r; X)$ to be the circular area with center $O(x_j, y_j)$ and radius r for which the removal of all nodes in that area, and the links incident to them, leads to a maximum decrease in a certain network metric X. The network metric X could for instance represent the number of affected nodes, the average shortest path length, the number of connected pairs of nodes, the size of the giant component, or some service function like packet loss or average delay. Fig. 2 presents an example of a network for which a critical region is identified. There might be multiple critical regions that affect the metric X to the same degree.

For a given value r, two sets of nodes $\mathcal{A}$ and $\mathcal{B}$ are *region disjoint* if each node $a \in \mathcal{A}$ is on a distance greater than $2r$ from every node in $\mathcal{B}$. Two paths are said to be *region disjoint*, if the set of intermediate nodes in the first path is region disjoint with the set of intermediate nodes in the second path. In some cases, the first hop from s might always lie within the region of s, in which case no region-disjoint paths could ever exist. However, in that case we could consider the region of s (and similarly d) and its incoming or outgoing links as a single source node attached to those incoming and outgoing links. We are now ready to define our two main problems.

Critical region problem: For a given network $G(\mathcal{N}, \mathcal{L})$, and a given radius r, find a *critical region* $\mathcal{C}(O(x_j, y_j), r; X)$ with respect to the network metric X.

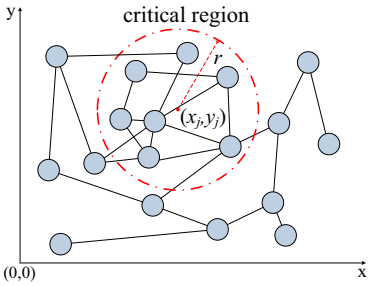


Fig. 2: Critical region.

Region-disjoint paths problem: Given a network $G(\mathcal{N}, \mathcal{L})$ with positive link weights, find two *region-disjoint paths* from s to d, with a minimum total weight as reflected by the sum of the link weights in the two paths.

III. Critical Region Detection and Mitigation

In this section, we will first demonstrate that critical regions can be found in polynomial time, after which we present a greedy network augmentation approach to reduce the impact of a failure of the critical region.

A. Polynomial time solvability

In principle, an infinite amount of circles exists within a plane. Fortunately, as we will proceed to demonstrate, the number of relevant circles is polynomially bounded in the input N. The crux is, as formalized in Lemma 1, that we may confine ourselves to only considering circles for which the circumference passes through two or more nodes.

Lemma 1: If there exists a circle with radius r that covers a set of nodes $\mathcal{S}$, then that same set $\mathcal{S}$ could also be covered by at least one of the circles with radius r that passes through two distinct nodes in $\mathcal{S}$.

Proof: Let us assume that we have an arbitrary circle C_1 that covers a set $\mathcal{S}$ of m nodes. As exemplified in Fig. 3a, we can move C_1 along the x-axis as long as it does not contain (at least) one node $A \in \mathcal{S}$ on the circumference. The resulting circle, C_2, contains the same set of nodes $\mathcal{S}$ or includes more nodes. It cannot include fewer nodes, because that would mean that a node would have crossed the circumference. We continue to rotate (either clockwise or counter-clockwise over node A) C_2 until it contains (at least one) node $B \in \mathcal{S}$ (in Fig. 3a). That circle, C_3 contains the set $\mathcal{S}$ with two nodes A and B on the circumference. ■

For two nodes A and B, it is possible to construct at most two different circles with radius r that pass through A and

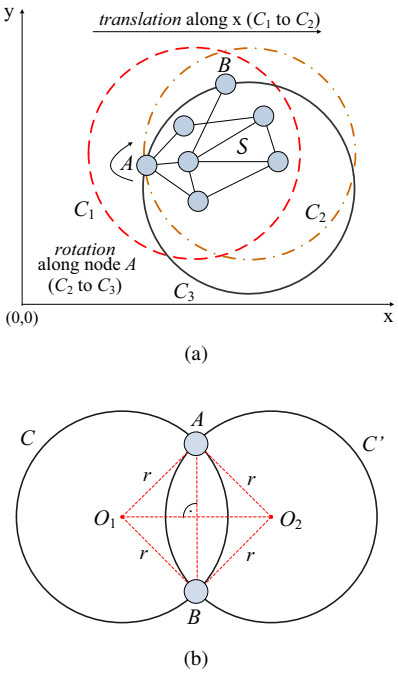


Fig. 3: (a) Visualization of (the proof of) Lemma 1. (b) Possible circles with radius r through nodes A and B.

B (in Fig. 3b): exactly two if the distance between A and B is smaller than $2r$, one if it is equal to $2r$ and zero if it is greater than $2r$. The centers of these circles can be found in $O(1)$ time. Having $\binom{N}{2}$ different pairs of nodes results in a maximum of $N(N-1)$ possible circles. If a node is on a distance greater than $2r$ to all the other nodes, then we add an arbitrary circle that contains (or passes through) this node. Consequently, an algorithm that considers all possible circles and checks which nodes are inside to find the critical region(s) will have a complexity of $O(N^3 \cdot C)$, where $O(C)$ is the complexity of finding the change in network metric X after the failure of a region.

B. Region-critical network augmentation

In order to make the network more robust, we may augment the network by adding k links, preferably of minimum total weight. Since augmenting a network to increase node- or link-connectivity is NP-complete for weighted networks [9], our network augmentation problem is also NP-complete for various metrics X. Since augmenting a network is a "design" problem, time complexity may be of secondary concern. In that case, by examining all possible combinations of k links, we could choose the best combination of links to be added. For large k, the running time may get too large. We will therefore demonstrate that a much faster greedy approach that, out of all the available links, only adds that one link which realizes the greatest reduction in network degradation after failure of the critical region (i.e., the network vulnerability as measured in the network metric X and when compared with the network vulnerability prior to the link addition), already leads to significant improvements. There might be multiple available links that equally reduce the network vulnerability. In this case and when possible, we choose the link that does not connect to any other critical region[1]. Adding one link may be insufficient to (substantially) reduce the network vulnerability. In that case, we might repeat the greedy algorithm k times, depending on the number of links to be added. The complexity of our network augmentation strategy is $O(k \cdot L_{\bar{G}} \cdot C)$, where $O(C)$ is the time complexity of finding the network metric change after a link addition and $L_{\bar{G}}$ reflects the number of links in the complement graph $\bar{G}$ of the original network.

Considering the number of connected pairs as our metric X, we demonstrate the link addition strategy in a simple network in Fig. 4a. The distances between nodes 1 and 2; 2 and 3; 3 and 4 are all three and the radius of failure $r = 1$. The network is connected, therefore the initial number of connected pairs is 6. As there is no circle with radius one that passes through two nodes, we detect two critical regions, around nodes 2 and 3, whose single failure results in only one connected pair. There are three possibilities for a link addition: $(1,3)$, $(2,4)$ and (1,4). If we add a link between nodes 1 and 3 (in Fig. 4b) to protect against the critical region around node 2, the number of connected pairs would not be decreased as the failure of the critical region around node 3 would lead to only nodes 1 and 2 to be connected). It did improve robustness in the sense that the number of critical regions reduced. The same applies to adding the link $(2,4)$ (in Fig. 4c). If we add the link $(1,4)$ that is not connected to any critical region then we always have three connected pairs after any single critical region failure as shown in Fig. 4d.

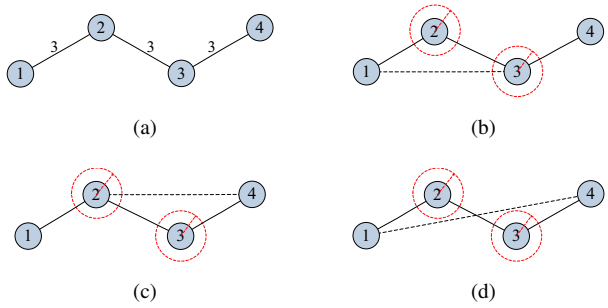


Fig. 4: Example of the network augmentation technique, with $r = 1$. (a) The original network. There are three possibilities for a link addition (shown in dotted lines). (b) Link $(1,3)$ is added. Critical region around 3 remains; (c) Link $(2,4)$ is added. Critical region around 2 remains; and (d) Link $(1,4)$ is added. Failure of any single critical region always leaves three connected pairs.

In Section V, we present an evaluation of our greedy network augmentation when applied to real-world networks. Instead of adding a fixed number of links, another variant consists of adding links as long as the network metric X does not meet a certain threshold, set by a network provider.

[1]In our simulations we take $w(u,v) = d(u,v)$. If different weights are used, one could break ties by choosing the link with lower weight.

IV. Region-disjoint paths problem

The **region-disjoint paths problem** is equivalent to finding two node-disjoint paths from s to d with a *minimum total weight*, such that each intermediate node from the first is on a distance at least $2r$ from all the intermediate nodes from the second path.

First, we show that the problem is strongly NP-hard. Subsequently, we propose a heuristic polynomial time region-disjoint paths algorithm.

A. Complexity of the problem

We provide a polynomial time reduction for the 3SAT problem[2], which is known to be NP-complete [10]. We use a graph structure called *lobe* [11]. We use the lobe to construct a graph from a 3SAT problem and on which finding two region-disjoint paths would provide a solution to that 3SAT problem, thus proving NP-hardness. We will assume undirected networks, although the results also apply to directed networks, since directed links could also have been used.

Graph Construction. For a given 3SAT instance, we create a lobe for each variable x_i, $i = 1, 2, \ldots, n$. Denoting by p_i the number of occurrences of variable x_i (in an auxiliary form x_i or as a negation $\bar{x}_i$), the lobe of x_i contains $(4p_i + 2)$ nodes: x_i, x_{i+1}, $u_j^i, v_j^i, \bar{u}_j^i, \bar{v}_j^i$ for each $j = 1, 2, \ldots, p_i$. We construct the links: $\left(x_i, u_1^i\right), \left(x_i, \bar{u}_1^i\right), \left(v_{p_i}^i, x_{i+1}\right), \left(\bar{v}_{p_i}^i, x_{i+1}\right)$ all with a weight 1; $\left(u_j^i, v_j^i\right)$ with a weight $0, \left(v_j^i, u_{j+1}^i\right)$ with a weight 1, $\left(\bar{u}_j^i, \bar{v}_j^i\right)$ with a weight 0, $\left(\bar{v}_j^i, \bar{u}_{j+1}^i\right)$ with a weight 1 for each $j = 1, 2, \ldots, p_i$. In Fig. 5, the lobe of variable x_i is depicted, which contains two parallel disjoint branches.

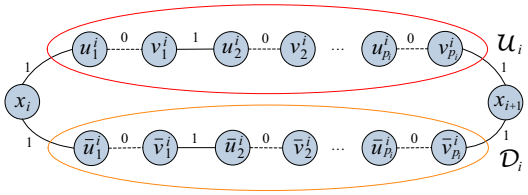


Fig. 5: Lobe of x_i.

The links with weight 0 are shown with dotted lines and the links with weight 1 with solid lines. All n lobes are connected in series (the i-th lobe is connected to the $(i+1)$-th lobe), starting from $s = x_1$ to $d = x_n$. Further, for each clause i of the 3SAT instance, we construct two nodes y_i and z_i. A link (z_i, y_{i+1}) with a weight 0 is established for each $i = 1, 2, \ldots, m-1$. Additionally, links (s, y_1) and (z_m, d), each with weight 0, are constructed. To relate the clauses with the variables, we have the following: (i) if the k-th occurrence of variable x_i exists without negation in clause C_j, then links $\left(y_j, u_k^i\right)$ and $\left(v_k^i, z_j\right)$ are added; or (ii) if the k-th occurrence of variable x_i exists with a negation $(\bar{x}_i)$ in clause C_j, then links $\left(y_j, \bar{u}_k^i\right)$ and $\left(\bar{v}_k^i, z_j\right)$ are added. The final graph is shown in Fig. 6.

[2]3SAT is a formula satisfiability problem, with an input logical expression $C_1 \wedge C_2 \wedge \ldots C_m$, where C_i are logical expressions of 3 variables (in auxiliary or negated forms) connected by $\vee$, which asks whether it is possible to assign boolean values to the variables so that the formula evaluates to TRUE.

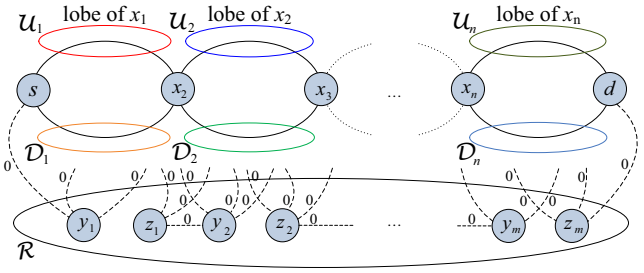


Fig. 6: Constructed graph.

The position of all the nodes in the Euclidean plane is defined in the following way. We set the nodes u_j^i, v_j^i from the upper part of each lobe x_i such that they are all on a distance not greater than $2r$ from each other and they are all on a distance greater than $2r$ from all the other nodes in the constructed graph. We denote the union of these nodes by $\mathcal{U}_i = \bigcup_{j=1}^{p_i} u_j^i \cup \bigcup_{j=1}^{p_i} v_j^i$. Similarly, for the lower part of each lobe x_i, we set the nodes $\bar{u}_j^i, \bar{v}_j^i$, such that they are all on a distance not greater than $2r$ from each other and they are all on a distance greater than $2r$ from all the other nodes in the constructed graph. We denote the union of these nodes by $\mathcal{D}_i = \bigcup_{j=1}^{p_i} \bar{u}_j^i \cup \bigcup_{j=1}^{p_i} \bar{v}_j^i$. Finally, all the nodes y_i and z_i, $i = 1, 2, \ldots, m$ are on a distance not greater than $2r$ from each other and they are all on a distance greater than $2r$ from all the other nodes in the constructed graph. We denote these nodes by $\mathcal{R} = \bigcup_{i=1}^{m} y_i \cup \bigcup_{i=1}^{m} z_i$. All the sets $\mathcal{R}, \mathcal{U}_i, \mathcal{D}_i$ for $i = 1, 2, \ldots, m$ are pair-wise region disjoint. The construction of the graph can be done in polynomial time.

An example of the aforementioned construction of the graph, for a 3SAT instance, is given in Fig. 7.

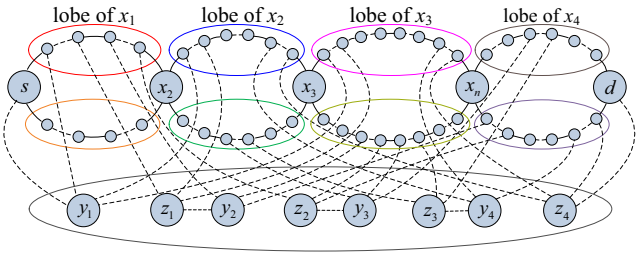


Fig. 7: Constructed graph that corresponds to $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_2 \vee x_3 \vee \bar{x}_4)$.

Theorem 1: The problem of finding two region-disjoint paths with minimum total weight is strongly NP-hard.

Proof: We will demonstrate that we could solve any instance of the 3SAT problem by solving the region-disjoint paths problem on a graph obtained via the polynomial time transformation of the 3SAT instance explained before.

3SAT to **Region-disjoint paths**. Let us assume there is an assignment τ, such that all m clauses are satisfied. For each clause C_j: (i) there exists a variable x_i (in non-negated

form) such that $\tau(x_i)$ = TRUE in which case we use links (y_j, u_k^i), (u_k^i, v_k^i), (v_k^i, z_j) or (ii) there exists a variable $\bar{x}_i$ in C_j, such that $\tau(x_i)$ = FALSE in which case we use links $(y_j, \bar{u}_k^i)$, $(\bar{u}_k^i, \bar{v}_k^i)$, $(\bar{v}_k^i, z_j)$. Together with links (s, y_1), (z_m, d), (z_i, y_{i+1}) for $j = 1, 2, \ldots, m-1$ they form a path P_1 with weight 0. In addition, there is another path P_2 that traverses through the lower part of x_i's lobe ($\mathcal{D}_i$) if $\tau(x_i)$ = TRUE or the upper part ($\mathcal{U}_i$) if $\tau(x_i)$ = FALSE, for each $i = 1, 2, \ldots, m$. Paths P_1 and P_2 are region-disjoint and have a minimum total weight, namely $\sum_{i=1}^{n}(p_i + 1)$.

Region-disjoint paths to **3SAT**. Let us assume there are two region-disjoint paths P_1 and P_2 with minimum total weight. It is important to notice that it is not possible for both P_1 and P_2 to contain nodes y_i or z_i at the same time, because there will be two paths that traverse through nodes from $\mathcal{R}$, therefore P_1 and P_2 will not be region disjoint. Consequently, at most one path, without loss of generality P_1, uses (some or all) the nodes y_i and z_i and the other, P_2, traverses through all the lobes. We denote by $w(P)$ the sum of the weights of all the links in the path P. Because the total weight of traversing links through a lobe is the same (p_i) when either using the upper ($\mathcal{U}_i$) or the lower part ($\mathcal{D}_i$), the total weight of P_2 is $w(P_2) = \sum_{i=1}^{n}(p_i + 1)$. Hence, the total weight of P_1 and P_2 is $w(P_1) + w(P_2) \geq \sum_{i=1}^{n}(p_i + 1)$, with an equality if $w(P_1) = 0$, i.e. all the link weights equal 0. This is only possible if P_1 uses links (s, y_1), (z_m, d), (z_i, y_{i+1}) and either (i) (y_j, u_k^i), (u_k^i, v_k^i), (v_k^i, z_j) or (ii) $(y_j, \bar{u}_k^i)$, $(\bar{u}_k^i, \bar{v}_k^i)$, $(\bar{v}_k^i, z_j)$ are not occupied by P_2. In (i) we set $\tau(x_i)$ = TRUE, while for (ii) we set $\tau(x_i)$ = FALSE. Finally, by this truth assignment, all m clauses become satisfied. Since only $\{0, 1\}$ weights have been used, the problem is strongly NP-hard. ■

B. Heuristic region-disjoint paths algorithm

Since a strongly NP-hard problem cannot be solved by a FPTAS, unless P=NP, we propose a polynomial time heuristic algorithm, named REGIONDISJOINTPATHS, for the region-disjoint paths problem.

REGIONDISJOINTPATHS starts by finding two node-disjoint paths P_1 and P_2 with minimum total weight, e.g. by using the Suurballe-Tarjan algorithm [12], if possible, otherwise there are no region-disjoint paths and the algorithm terminates. We denote by *critical pairs* $\mathcal{K}$ the set of pairs of nodes, such that one is in P_1 and the other is in P_2 on a distance not greater than $2r$. Then, the algorithm partitions the nodes in the network into two sets $\mathcal{S}_1$ and $\mathcal{S}_2$. A node belongs to the set $\mathcal{S}_1$ if it is closest to a node in $P_1 \setminus \{s, d\}$, otherwise the node belongs to $\mathcal{S}_2$, which means that these nodes are closer to P_2. We initialize a set of "unavailable" nodes to an empty set ($\mathcal{Q}$ in Algorithm 1). The algorithm finds a node k that appears in most of the pairs in $\mathcal{K}$, but is not in the set of "unavailable" nodes. Further, if $k \in \mathcal{S}_1$ ($k \in \mathcal{S}_2$), the algorithm makes a local improvement by finding the shortest path through the nodes in $\mathcal{S}_1$ (in $\mathcal{S}_2$) that do not appear in $\mathcal{Q}$ and in a pair in $\mathcal{K}$, between the first predecessor a of k in P_1 (in P_2) and the first successor b of k in P_1 (in P_2) that both do not appear in $\mathcal{Q}$ and in a pair in $\mathcal{K}$. The updated path P_1 (or P_2) comprises of the current part from s to a, the newly determined shortest path from a to b and the current part from b to d. REGIONDISJOINTPATHS iterates by searching back for a new k that appears in most of the pairs in $\mathcal{K}$, but is not in the "unavailable" nodes, and updates $\mathcal{S}_1$ and $\mathcal{S}_2$. If there are multiple nodes that appear in most of the pairs in $\mathcal{K}$, it chooses the one such that the resulting total weight of P_1 and P_2 is minimal. The pseudo-code of REGIONDISJOINTPATHS is given in Algorithm 1.

Algorithm 1: REGIONDISJOINTPATHS

input : Network G, radius r, source s, destination d
output: Region-disjoint paths P_1 and P_2

1 Find node-disjoint paths P_1 and P_2 between s and d, if exist; otherwise **Exit**; Find the set of *critical pairs* $\mathcal{K}$;
2 Initialize the set of "unavailable" nodes $\mathcal{Q} \leftarrow \emptyset$;
3 Divide all the nodes in the network (except s and d) into two disjoint sets $\mathcal{S}_1$ and $\mathcal{S}_2$ in the following way: $i \in \mathcal{S}_1$ if i is the closest to some node in $\mathsf{P}_1 \setminus \{\mathsf{s}, \mathsf{d}\}$ or $i \in \mathcal{S}_2$ if i is the closest to some node in $\mathsf{P}_2 \setminus \{\mathsf{s}, \mathsf{d}\}$;
4 If the set $\mathcal{K} = \emptyset$ then stop, **region-disjoint paths** are found or do not exist. Otherwise, find the node $k \in (\mathsf{P}_1 \cup \mathsf{P}_2) \setminus \mathcal{Q}$ that appears in most of the critical pairs in $\mathcal{K}$ (if many the one with a minimum resulting total weight of P_1 and P_2);
5 $k \in \mathsf{P}_j$, where $j \in \{1, 2\}$. Find the shortest path P_x through nodes in $\mathcal{S}_j$ that do not appear in $\mathcal{Q}$ and in a pair in $\mathcal{K}$, between the first predecessor a and the first successor b of k in P_j that do not appear in $\mathcal{Q}$ and in a pair in $\mathcal{K}$;
6 If P_x does not exist $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{k\}$, Go to Line 9;
7 Update P_j such that it consists of: the existing part in P_j from s to a, P_x and the current part from b to d in P_j;
8 Update $\mathcal{K}$, $\mathcal{S}_1$ and $\mathcal{S}_2$ based on P_1 and P_2;
9 $G \leftarrow G - k$, Go to Line 4.

REGIONDISJOINTPATHS always terminates, since each node can be picked for removal at most once. The complexity of REGIONDISJOINTPATHS can be determined as follows. Finding node-disjoint paths with a minimum total weight [12] requires a complexity [13] of $O(L + N\log_2 N)$. Finding the shortest path in $\mathcal{S}_1$ or $\mathcal{S}_2$ (after a node k is picked) can be done in $O(L + N\log_2 N)$. The updates of the sets $\mathcal{K}$, $\mathcal{S}_1$ and $\mathcal{S}_2$ all require a worst-case complexity of $O(N^2)$ as $O(N)$ nodes may change in one of the paths, whose distances to the nodes in the second path have to be compared. The number of nodes that appear in pairs in the set $\mathcal{K}$ and could be picked is $O(N)$, which reflects the number of iterations. Consequently, the total worst-case time-complexity of REGIONDISJOINTPATHS is $O(N^3)$.

For comparison purposes, we also present (in Algorithm 2) a naive algorithm, named DOUBLEDIJKSTRA, since it uses two iterations of the Dijkstra shortest path algorithm [14]. The first

iteration finds the shortest path between s and d. Subsequently, it removes all the nodes within a distance $2r$ from at least one node in the first path different from s and d. Finally, the second path is found by running Dijkstra's algorithm in the pruned network. Since it uses two Dijkstra algorithm iterations, the complexity [13] of DOUBLEDIJKSTRA is $O(L + N\log_2 N)$.

Algorithm 2: DOUBLEDIJKSTRA

input : Network G, radius r, source s, destination d
output: Region-disjoint paths P_1 and P_2

1 $\mathsf{P}_1 \leftarrow$ DIJKSTRA$(\mathsf{s}, \mathsf{d}, \mathsf{G})$;
2 Find set S, which contains all the nodes in $\mathsf{P}_1 \setminus \{\mathsf{s}, \mathsf{d}\}$ and all the nodes on a distance at most $2\mathsf{r}$ from a node in $\mathsf{P}_1 \setminus \{\mathsf{s}, \mathsf{d}\}$, $G' \leftarrow \mathsf{G} - S$;
3 $\mathsf{P}_2 \leftarrow$ DIJKSTRA$(\mathsf{s}, \mathsf{d}, G')$;

Since the region-disjoint paths problem is NP-hard, we resort to an Integer Linear Program (ILP) to find the exact solution. For each link $(i,j) \in \mathcal{L}$, we define two variables $x_{ij}, y_{ij} \in \{0,1\}$. If a link (i,j) is on path P_1 then $x_{ij} = 1$, otherwise $x_{ij} = 0$ and if a link (i,j) is on path P_2 then $y_{ij} = 1$, otherwise $y_{ij} = 0$. The distances $d(i,j)$ between the nodes can be calculated beforehand in polynomial time ($O(N^2)$ for all pairs). The 0-1 ILP formulation is as follows:

$$\min \sum_{(i,j)\in\mathcal{L}} w(i,j)\cdot(x_{ij}+y_{ij})$$

$$\text{s.t.} \quad (1) \sum_{j\in\mathcal{N}} (x_{ij}-x_{ji}) = \begin{cases} 1, & \text{if } i \equiv s \\ -1, & \text{if } i \equiv d \\ 0, & \text{otherwise} \end{cases}$$

$$(2) \sum_{j\in\mathcal{N}} (y_{ij}-y_{ji}) = \begin{cases} 1, & \text{if } i \equiv s \\ -1, & \text{if } i \equiv d \\ 0, & \text{otherwise} \end{cases}$$

(3) $x_{ij} + y_{kl} \leq 1$, if ($d(i,k) \leq 2r$, $i \notin \mathcal{M}$, $k \notin \mathcal{M}$) or ($d(i,l) \leq 2r$, $i \notin \mathcal{M}$, $l \notin \mathcal{M}$) or ($d(j,k) \leq 2r$, $j \notin \mathcal{M}$, $k \notin \mathcal{M}$) or ($d(j,l) \leq 2r$, $j \notin \mathcal{M}$, $l \notin \mathcal{M}$), where $\mathcal{M} = \{s, d\}$

(4) $x_{sd} + y_{sd} \leq 1$

The objective function represents the total weight of the region-disjoint paths. The equality conditions (1) and (2) are "conservation rules" and ensure that for all the nodes (different from s and d) in both P_1 and P_2, the number of incoming and outgoing links is the same. For the source node s, there is exactly one outgoing link for both P_1 and P_2, while for the destination node d there is exactly one incoming link for both P_1 and P_2. Condition (3) gives the region-disjointness constraint, as it is not possible to have two nodes different from s and d, one in link $(i,j) \in P_1$ and one in link $(k,l) \in P_2$ that are within a distance $2r$. In condition (4), if there is a direct link from s to d, that link will be used by at most one path. Condition (4) is not a sub-case of (3).

C. An example of region-disjoint paths

In Fig. 8, two region-disjoint paths between Turin and Palermo are depicted for the Italian main backbone network. The (undisplayed) weights in the network are the geographical distances between the nodes. The two node-disjoint paths of minimum total weight (which traverse through Turin-Rome-Naples-Palermo and Turin-Cagliari-Palermo) are depicted in solid lines. However, these paths cannot protect against a failure of a circular region with radius $r = 3000$.

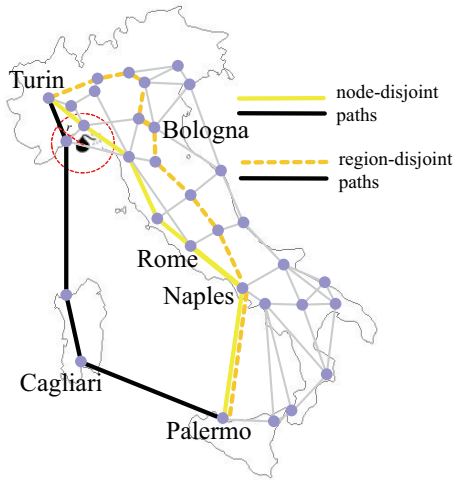
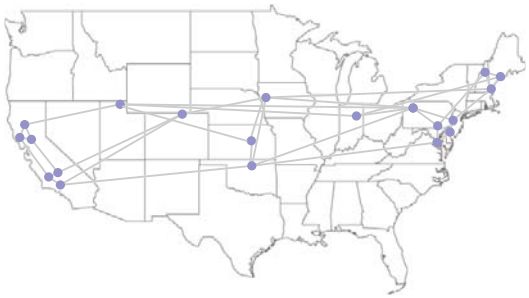


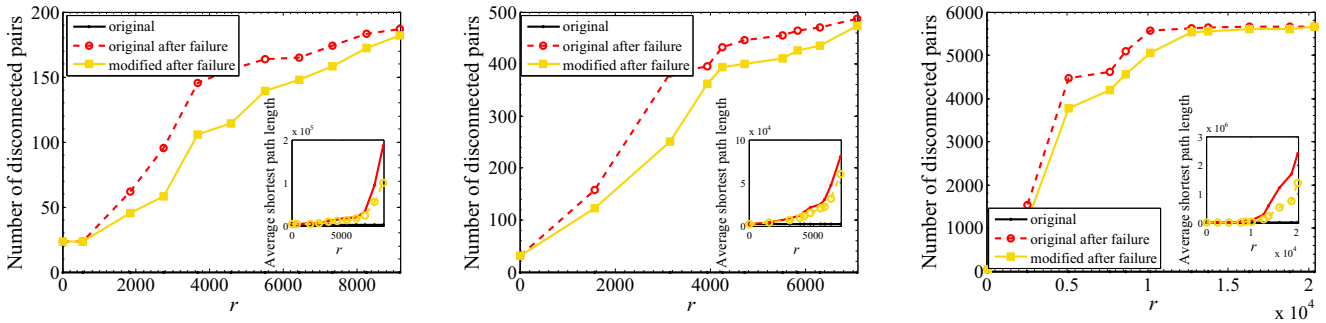
Fig. 8: Node-disjoint and region-disjoint paths in the Italian backbone network.

(a) Europe.



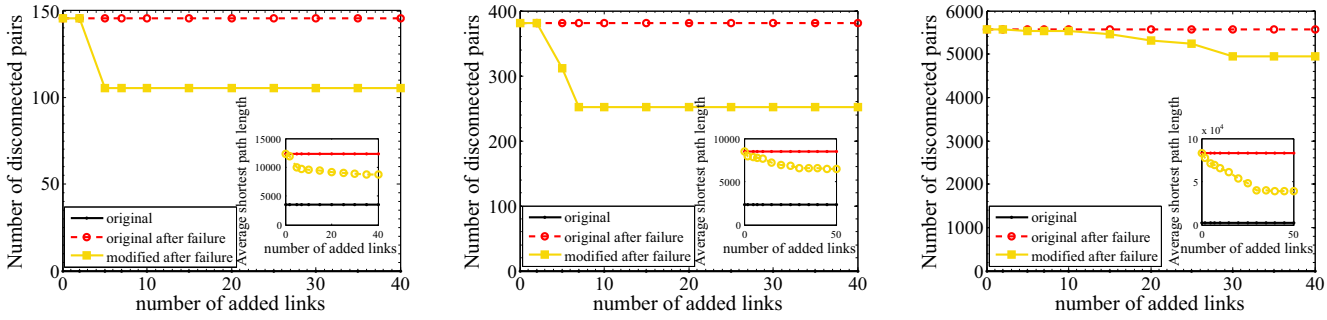
(b) ARPANET.

Fig. 9: Used network topologies.



(a) ARPANET. (b) Italy. (c) Europe.

Fig. 10: The number of disconnected pairs and the average shortest path length (inset) as a function of the failure radius r. Ten links were added to each of the modified networks. The number of disconnected pairs in the original network is 0.



(a) ARPANET ($r = 3672$). (b) Italy ($r = 3152$). (c) Europe ($r = 10185$).

Fig. 11: The number of disconnected pairs and the average shortest path length (inset) as a function of the number of added links, for a fixed value of the failure radius r.

REGIONDISJOINTPATHS returns region-disjoint paths such that one of the paths is the same as in the initial node-disjoint paths, but the other path differs (Turin-Bologna-Naples-Palermo, shown with dashed lines). Moreover, the ILP confirms that this solution is exact. DOUBLEDIJKSTRA could not find a solution.

D. Region-disjoint paths under non-circular failures

A solution to the region-disjoint paths problem for circular shaped regions is immediately applicable to regions of any shape. The *diameter* of a particular regional shape is the longest distance between any two points (not nodes) in that region. In order for two nodes to be region disjoint, they will have to lie on a distance greater than the diameter from each other. Consequently, the region-disjoint paths problem for regional failures of any arbitrary shape with a diameter D is equivalent to the problem of finding region-disjoint paths problems under circular failures with radius $r = D/2$.

V. EVALUATION STUDY

In this section, we evaluate the performance of our greedy network augmentation strategy and we evaluate the accuracy and running time of our REGIONDISJOINTPATHS algorithm.

A. Used data

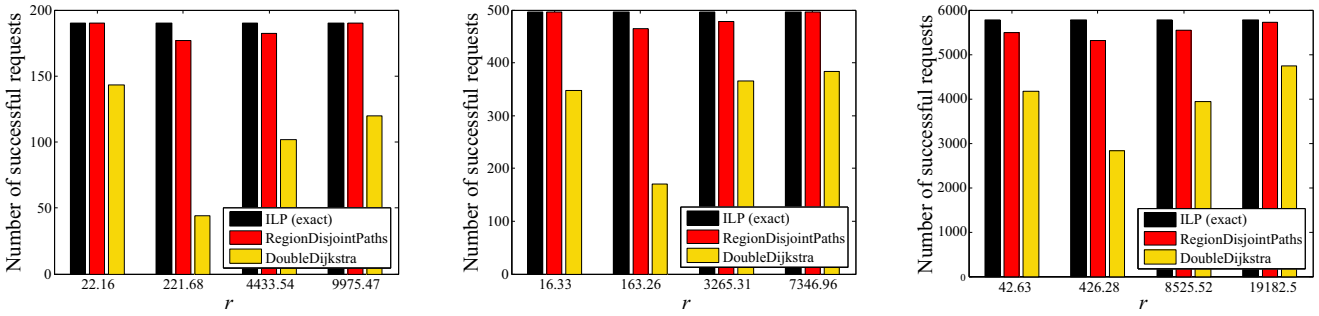
We use three real-world network data sets: the Italian main backbone network (in Fig. 8), the main backbone fiber connections in Europe [15] (in Fig. 9a) and for benchmarking purposes the infrastructure of the ARPANET network [16] (in Fig. 9b). Through longitude and latitude information, the geographical distances between the nodes can be derived. The properties of the used networks are given in Table I.

TABLE I: Real networks used in the evaluation.

Networks	N	L	Description
ARPANET	20	32	first packet switching network [16]
Italy	32	62	main fiber connections in Italy
Europe	108	151	main fiber connections in Europe [15]

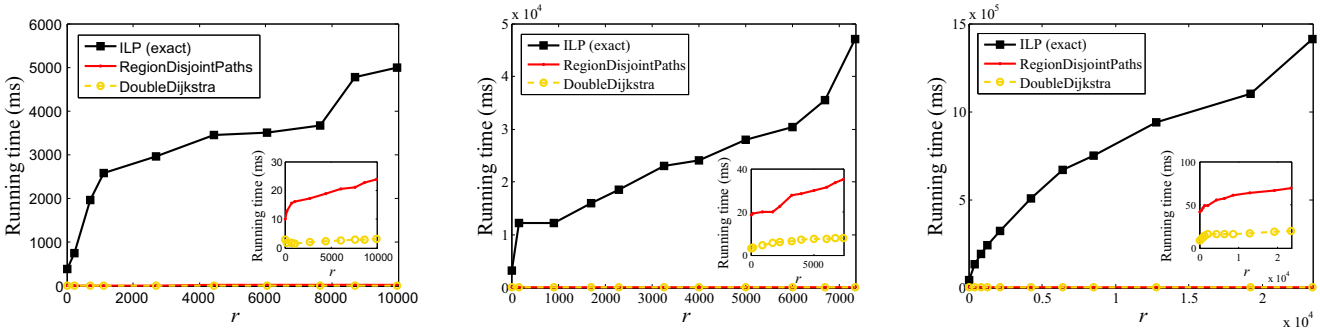
B. Evaluation of region-critical network augmentation

We examine the effect of our network augmentation strategy on the number of disconnected pairs (i.e., the number of connected pairs of the original network minus the number of remaining connected pairs) and the average shortest path length of the connected pairs. In Fig. 10, we present the numbers of disconnected pairs (main plot) and the average



(a) ARPANET. (b) Italy. (c) Europe.

Fig. 12: Number of successful requests. All pairs of nodes are requests. Variable r reflects the radius of failure.



(a) ARPANET. (b) Italy. (c) Europe.

Fig. 13: Average running times of the algorithms over all pairs of nodes. Variable r reflects the radius of failure.

shortest path lengths (inset) for (i) the original network (the solid, black, horizontal line), (ii) the original network after a critical region failure and (iii) the modified network after failure of a critical region, as a function of the radius of failure. The network augmentation strategy has been repeated ten times: consequently ten new links were added to each network. The results show a substantial reduction in the number of disconnected pairs and in the average shortest path lengths, as a result of the network augmentation. The curve for the number of disconnected pairs of the modified network approaches that of the original for high values of the radius, because most of the nodes are then affected, leaving no effective protection.

In Fig. 11, we examine the effect on the number of disconnected pairs and the average shortest path length as a function of the number of added links. For small networks (ARPANET in Fig. 11a and Italy in Fig. 11b), only few links are required to reduce the number of disconnected pairs and further link additions would not have an effect as the remaining disconnected pairs are attributed to nodes residing inside a critical region (which therefore cannot be protected against the failure of that region). For Europe (in Fig. 11c), more links are required to reduce the number of disconnected pairs. In all three networks, the average shortest path lengths are shortened as more links are added, where a few added links already realize a significant reduction.

C. Evaluation of region-disjoint paths algorithms

The accuracy of REGIONDISJOINTPATHS is evaluated by comparing it to an exact ILP solution and the naive DOUBLEDIJKSTRA algorithm. For each network, the requests consist of all the possible pairs of distinct nodes. Fig. 12 depicts the success rates of the three algorithms. The results vary for different values of r that reflects the radius of failure, but the algorithm REGIONDISJOINTPATHS correctly finds region-disjoint paths in a significant number of cases. On the other hand, DOUBLEDIJKSTRA has a much worse performance, because greedily removing a first shortest path may jeopardize finding a second path.

The average running times of the three algorithms over all possible requests (pairs of nodes) are shown in Fig. 13. The algorithms have been implemented in the same programming language, using the same libraries and the simulations have been conducted on the same machine[3]. Fig. 13 shows that solving the ILP requires significant running time, orders of magnitude greater than for the REGIONDISJOINTPATHS and DOUBLEDIJKSTRA algorithms, which is unacceptable when paths need to be computed on the fly for dynamically arriving requests. The running times of REGIONDISJOINTPATHS and

[3]Intel(R) Core 2 Duo T9600 - 2 × 2.80GHz, 4GB RAM memory; using JAVA libraries: JUNG (http://jung.sourceforge.net/) for network representations and algorithms and lpsolve (http://lpsolve.sourceforge.net/) for the ILP.

DOUBLEDIJKSTRA do not differ much, while the improved accuracy of REGIONDISJOINTPATHS clearly outweighs the slight increase in time over the DOUBLEDIJKSTRA algorithm.

VI. RELATED WORK

The problem of finding link- or node-disjoint paths in a network between two nodes has been widely explored under different scenarios, e.g. see [17], [12], [6], [18], [19], [20], [21], [7]. Suurballe [17] proposed a polynomial time algorithm to find k node- or link-disjoint paths with minimum total weight (i.e., the min-sum objective). Subsequently, an optimized algorithm was proposed by Suurballe and Tarjan [12] to find 2 link- or node-disjoint paths from a source to all other nodes. Finding disjoint paths for other objective functions has been studied in [18], [22], [23], among others. Contrary to the min-sum objective, these objectives typically lead to NP-complete problems.

Much work on the robustness of a network against geographical failures has been done by Neumayer *et al.* [8], [24], [25]. However, they use a different failure model than the one presented in this paper. Agarwal *et al.* [26] study a probabilistic geographical failure model. Sen *et al.* [3] consider the problem of finding region-disjoint paths for the case that critical regions are fixed and predetermined. Banerjee *et al.* [27] have taken a different application domain, namely that of distributed file storage in a network, in which data resiliency is provided to regional failures.

VII. CONCLUSION

This paper has studied robustness problems in relation to the geographical embedding of a network. We have provided a polynomial time algorithm to determine a circular critical region with a radius r that, after the removal of all the covered nodes (and their incident links), would cause the biggest change in a given network metric. This algorithm can be used to detect the most vulnerable parts in a network. We also proposed a region-aware network augmentation strategy to increase the network robustness to regional failures. By applying our augmentation strategy to three real networks, we have demonstrated that adding only a few links may already induce significant robustness gains. Subsequently, we have studied the problem of finding two region-disjoint paths with a minimum total weight such that they cannot both be cut by a single regional failure with a given diameter, unless that failure affects the source or the destination. Region-disjoint paths are needed to quickly reroute traffic upon the failure of a region. We have proved the NP-hardness of the region-disjoint paths problem and subsequently proposed an efficient polynomial time heuristic for it. The comparison with an exact exponential-time algorithm shows that our proposed algorithm correctly finds region-disjoint paths in most of the cases, while being orders of magnitude faster than the exact algorithm.

ACKNOWLEDGMENT

This research has been partly supported by the EU FP7 Network of Excellence in Internet Science EINS (project no. 288021) and by the GigaPort3 project led by SURFnet.

REFERENCES

[1] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, pp. 1025–1028, Apr. 2010.

[2] R. Cohen, K. Erez, D. ben-Avraham, and S. Havlin, "Resilience of the internet to random breakdowns," *Phys. Rev. Lett.*, vol. 85, pp. 4626–4628, Nov. 2000.

[3] A. Sen, S. Murthy, and S. Banerjee, "Region-based connectivity - a new paradigm for design of fault-tolerant networks," in *Int. Conference on High Performance Switching and Routing*, pp. 1 –7, June 2009.

[4] W. Zou, M. Janic, R. Kooij, and F. A. Kuipers, "On the availability of networks," in *Proc. of BroadBand Europe*, Dec. 2007.

[5] R. Cohen, K. Erez, D. ben-Avraham, and S. Havlin, "Breakdown of the internet under intentional attack," *Phys. Rev. Lett.*, vol. 86, pp. 3682–3685, Apr. 2001.

[6] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*. Kluwer Academic Publishers, 1999.

[7] F. A. Kuipers, "An Overview of Algorithms for Network Survivability," *ISRN Communications and Networking*, vol. 2012, no. 932456, 2012.

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

[9] G. Frederickson and J. JáJá, "Approximation algorithms for several graph augmentation problems," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 270–283, 1981.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[11] A. Itai, Y. Perl, and Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," *Networks*, vol. 12, no. 3, pp. 277–286, 1982.

[12] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[13] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, pp. 596–615, July 1987.

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

[15] Level 3 Communications, "Level 3 Network map." available on: http://www.level3.com/en/resource-library/maps/level-3-network-map/.

[16] "ARPANET Maps." available on: http://www.10stripe.com/featured/map/arpanet/arpanet-1972-aug.php.

[17] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974.

[18] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On finding disjoint paths in single and dual link cost networks," in *INFOCOM*, vol. 1, pp. 4 vol. (xxxv+2866), IEEE, March 2004.

[19] Y. Guo, F. A. Kuipers, and P. Van Mieghem, "Link-disjoint paths for reliable QoS routing," *Int. J. Communication Systems*, vol. 16, no. 9, pp. 779–798, 2003.

[20] R. Andersen, F. Chung, A. Sen, and G. Xue, "On disjoint path pairs with wavelength continuity constraint in wdm networks," in *INFOCOM*, vol. 1, pp. 4 vol. (xxxv+2866), IEEE, March 2004.

[21] A. A. Beshir, F. A. Kuipers, A. Orda, and P. Van Mieghem, "On-line survivable routing in WDM networks," in *21st International Teletraffic Congress (ITC 21)*, Sept. 2009.

[22] Q. She, X. Huang, and J. P. Jue, "Maximum survivability using two disjoint paths under multiple failures in mesh networks," in *GLOBECOM*, IEEE, Dec. 2006.

[23] A. A. Beshir and F. A. Kuipers, "Variants of the min-sum link-disjoint paths problem," in *16th Annual Symposium on Communications and Vehicular Technology (SCVT)*, IEEE, Nov. 2009.

[24] S. Neumayer and E. Modiano, "Network reliability with geographically correlated failures," in *INFOCOM*, pp. 1658–1666, IEEE, March 2010.

[25] S. Neumayer, A. Efrat, and E. Modiano, "Geographic max-flow and min-cut under a circular disk failure model," in *INFOCOM*, pp. 2736–2740, IEEE, March 2012.

[26] P. Agarwal, A. Efrat, S. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "Network vulnerability to single, multiple, and probabilistic physical attacks," in *MILCOM*, pp. 1824 –1829, IEEE, Nov. 2010.

[27] S. Banerjee, S. Shirazipourazad, and A. Sen, "On region-based fault tolerant design of distributed file storage in networks," in *INFOCOM*, pp. 2806 –2810, IEEE, March 2012.