This paper was presented at RNDM 2014 - 6th International Workshop on Reliable Networks Design and Modeling (Nov. 17-19, 2014, Barcelona, Spain)

Cost-Efficient Network Improvement to Achieve Maximum Path Diversity

Mohammed J.F. Alenazi[*‡], Egemen K. Çetinkaya[§⋆], and James P.G. Sterbenz[*†]
[*]Information and Telecommunication Technology Center
Department of Electrical Engineering and Computer Science
The University of Kansas, Lawrence, KS 66045, USA
{malenazi, jpgs}@ittc.ku.edu
[‡]College of Computer and Information Sciences
Department of Computer Engineering
King Saud University, Riyadh, Saudi Arabia
mjalenazi@ksu.edu.sa
[§]Department of Electrical & Computer Engineering
Missouri University of Science and Technology, Rolla, MO 65409, USA
cetinkayae@mst.edu
[†]School of Computing and Communications (SCC) and InfoLab21
Lancaster LA1 4WA, UK
jpgs@comp.lancs.ac.uk
www.ittc.ku.edu/resilinets

***Abstract*—Communication networks are prone to failures due to targeted attacks or large-scale disasters. Networks can be improved to withstand challenges using mechanisms such as diversity, which can simply be improved by adding links, however achieving maximum resilience is not feasible due to limited budget. Therefore, algorithms that improve the diversity of networks cost-efficiently are necessary. In this paper, we present a heuristic algorithm that adds links to improve the diversity of the graph while minimizing the cost of link addition. We analyze the flow robustness of non- and improved graphs against targeted attacks. Our results indicate that path-diversity-improved graphs are more resilient to attacks than graphs improved by adding links to the lowest degree nodes.**

***Index Terms*—Network design, optimization, augmentation, algorithm; Network cost model; Network resilience, survivability, connectivity, robustness, reliability; Path diversity; Backbone network**

I. Introduction and Motivation

Networks in general and the Internet in particular are prone to correlated failures. Algorithms and mechanisms are necessary to defend networks and to make them resilient and survivable against challenges [1]. One such mechanism is *diversity* and it has been the subject of published works in the field of network resilience. Diversity is used to enhance bandwidth, delay, and loss rate of media streaming applications [2]. Path diversity is used in the optical domain to route around failed nodes or to split traffic for a better utilization of network resources [3]. Diverse routing is necessary for multihoming to improve the service delivery of provider networks [4], [5]. Therefore diversity is an essential mechanism for survivable networks.

⋆ Work performed while at The University of Kansas.

The design and optimization of cost-efficient networks that are resilient against challenges and attacks has been studied by many researchers over the past few decades, but the resilient network design problem is NP-hard. Moreover, networks cannot have unlimited resilience due to cost constraints, since these two objectives fundamentally oppose one another. Maximum resilience is achieved by a full-mesh, impractical from a cost perspective. Therefore careful engineering of existing networks is required under limited resources, and efficient algorithms are necessary to accomplish this.

Our contribution in this work is twofold. First, our main contribution in this work is an improvement algorithm that improves the *path diversity* of a graph. Our *heuristic* improvement algorithm considers adding links with the least cost among available choices. Our results indicate that the path diversity of improved graphs withstand centrality-based attacks better than the non- and improved graphs. Second, we introduce an algorithm for finding the optimal k-diverse paths considering both *nodes and links* using an exhaustive search of all paths.

The rest of the paper is organized as follows: We present brief background and related work on network improvement and path diversity in Section II. Measurement of path diversity and the algorithm to calculate the path diversity of a graph is explained in Section III. Our heuristic algorithm that improves path diversity of a graph in the least costly fashion is presented in Section IV. The dataset for the communication networks as well as evaluation of these topologies using our algorithm is presented in Section V. Finally, we summarize our findings as well as propose future work in Section VI.

II. Background and Related Work

In this section we present background and related work about network improvement and path diversity.

A. Network Optimization

Network design is a NP-hard problem [6] that has been studied in the past [7]–[9]. The optimization process enhances the network for one or multiple objectives. Network improvement can be accomplished by means of rewiring while keeping the number of edges constant or by means of adding new links to improve the connectivity of graphs. The design process is different for backbone and access networks, since the topological structure of these networks fundamentally differ [8], [9]. Adding a set of links or node to the graphs to optimally maximize a certain graph property is known in the literature as *graph augmentation* where several problems are proven to be NP-hard [10], for example, adding a set of links to optimally increase the algebraic connectivity [11].

Network design and optimization objectives are cost, capacity, reliability, and performance [8], [9]. Network cost is incurred by the number of nodes required, capacity of nodes required, and number of links. In our earlier work, we provided a cost model that is the *total link length* for a given network with the assumption that fiber length dominates wide-area network cost [12]. Topological connectivity is another objective that can be measured by means of many graph metrics such as average degree, betweenness, and closeness [7], [13]. Previously, we developed a heuristic algorithm that maximizes the algebraic connectivity of a graph while selecting the least cost links [14], [15].

B. Path Diversity

A path between a source s and a destination d is the set of nodes and links that form a loop-free connection. Diverse paths between node pairs strengthen the ability of a network to withstand attacks and correlated failures. If the alternative paths have no common node or link they are disjoint; if there are some common network nodes or links, they are partially disjoint. Path diversity has been studied from a topological perspective [16]–[18], as well as in terms of multipath routing [19], [20], and multipath transport [21], [22]. In this paper, we approach the path diversity subject from a graph theoretic perspective. Our objective is to calculate the path diversity of a graph as a single number between 0 and 1. A metric to capture the total graph diversity has been developed [22], [23]. We consider fully- and partially-disjoint paths between every node pair to calculate the path diversity. Then, the path diversity value of graphs are used as the objective function to optimize the graphs.

III. Path Diversity Measurement

In this section, we present definitions of path diversity, total path diversity of a graph, and our algorithm that finds k-diverse paths. We note that we use the path diversity definition from our earlier work, since it captures the diversity of a graph considering both *nodes and links* [22], [23]. We emphasize that while the definitions are summarized here to provide the necessary background on how to measure the path diversity of a given graph, we develop a new algorithm that finds the k-diverse paths as explained in Section III-C.

A. Path Diversity Definition

Given a shortest path and an alternative path between two nodes in a graph, the path diversity of the alternative is defined as the ratio of the number of disjoint elements (nodes *and* links) between the shortest path and alternative path to the number of elements in the shortest path. Given a (source s, destination d) node pair, a path P between them is a set containing all links L and all intermediate nodes N [22],

$$P = L \cup N \tag{1}$$

and the *length* of this path $|P|$ is the combined total number of elements in L and N. Let the shortest path between a given (s, d) pair be P_0. Then, for any other path P_k between the same source and destination, the definition of the diversity function [22], [24] $D(P_k)$ with respect to P_0 is:

$$D(P_k) = 1 - \frac{|P_k \cap P_0|}{|P_0|} \tag{2}$$

The path diversity has a value of 1 if P_k and P_0 are completely disjoint and a value of 0 if P_k and P_0 are identical. This measure captures the diversity with respect to both nodes and links on alternative paths [22].

B. Total Graph Diversity

TGD (total graph diversity) is the average of the EPD (effective path diversity) values of all node pairs in a given graph [22] and TGD measures the structural path diversity of a graph as a single value. EPD is the normalized sum of path diversities for a selected set of paths connecting a node pair (s, d). First, we find the k diverse paths using the algorithm presented in Section III-C. Then, we remove zero diversity paths from the list of returned paths because they do not add any additional diversity. The returned diverse path is denoted as $P_{s,d} = \{P_1, P_2, \dots, P_m\}$, where $m \le k$, since-zero diversity paths are removed from the set. To calculate EPD, we use the exponential function:

$$\text{EPD} = 1 - e^{-\lambda k_{sd}} \tag{3}$$

where k_{sd} is the sum of all non-zero diversity paths defined as:

$$k_{sd} = \sum_{i=1}^{m} D(P_i) \tag{4}$$

where $D(P_i)$ is the non-zero path diversity of the i-th path with respect to the P_0. In Equation 3, λ is an experimentally determined constant that scales the impact of k_{sd} based on the utility of this added diversity [22]. For a given pair of nodes, the range of EPD is between $[0, 1)$ where 0 means that there is no diversity in between the two nodes as there are no alternative paths connecting the pair. When the EPD approaches 1, it means high path diversity [22].

C. Finding k-Diverse Paths

In this section, we present a new k-diverse paths algorithm that determines the k paths between a source s and destination d. This algorithm has four inputs: a source node s, a destination node d, a hop-count threshold h, and a threshold for the number of returned diverse paths k. Moreover, this algorithm uses four functions: all_simple_paths(s,d,h), sort(L), path2elements(P), and p_div(P). The all_simple_paths(s,d,h) function finds all possible loopless paths between a source s and destination d, with hop-count threshold h for the path length. If h is not set, all possible paths are returned. Whereas selecting higher h value yields more accurate results,it requires more computing resources. The number of all possible simple paths can be as large as $n!$ where n is the number of nodes. This number is infeasible to compute for large size graphs, thus, the h parameter should be chosen based on the size of the graph and available computing resources. The sort(L) function sorts a list of tuples of three elements: link, diversity, and cost. The sorting is done in decreasing order of the diversity value and increasing order of the cost value for links with equal diversities. The path2elements(P) function converts a path P to a set of nodes and links elements as described in Section III-A. The p_div(P) function computes the diversity of the path with respect to the selected_elements set. The pseudo code is shown in Algorithm 1.

```
Functions:
all_simple_paths($s$,$d$,$h$) := all simple paths between node
$s$ and node $d$ with a threshold hop-count $h$
sort(l) := sorting $l$ function
path2elements($P$) := path $P$ to link and node elements
p_div($P$) := computes path diversity of path $P$
Input:
$s$ := source node
$d$ := destination node
$h$ := hop-count threshold value for examined paths
$k$ := threshold value for returned diverse paths
Output:
an ordered list of $k$diverse paths
begin
    diverse_paths = []; empty ordered list
    selected_elements = {}; empty set
    for path in all_simple_paths($s$,$d$,$h$) do
        diverse_paths.append(path,p_div(path),len(path))
        selected_elements.add(path2elements(path))
    end
    sort(diverse_paths)
    return diverse_paths[0:k]
end
```

Algorithm 1: k-diverse path algorithm

This algorithm has two phases: finding all simple paths and finding the k most diverse paths. In the first phase, all possible paths are determined between a source s and destination d with a hop-count threshold h using the function all_simple_paths(s,d,h). For the second phase, the algorithm determines the most diverse paths among the returned paths via the all_simple_paths function. The shortest path P_0 is added to the selected paths in the first iteration and its elements (nodes and links) are added to the selected_elements set. Next, the algorithm iterates over the rest of the paths by computing the diversity of the path using the p_div(P) function and adding it along with the path length to the diverse_paths list while the path elements are added to the selected_elements. Finally, using the sort(L) function, all the tuples in the diverse_paths list are sorted in decreasing order of their diversity and in case there are multiple paths with the same diversity, these paths are sorted in increasing order of their hop-count.

IV. Diversity Improvement Algorithm

In this section, we describe our algorithm that improves the TGD of a given graph with its node locations by adding new cost-efficient links.

A. Algorithm

The objective of this algorithm is to improve the TGD of a graph by adding a user-specified number of links. The algorithm adds one link that maximizes the TGD value. If there are multiple links that give the same largest TGD value, the least cost link is selected. We measure the cost of a link in terms of the Euclidean distance of that link.

The topology improvement algorithm has two inputs: an input graph G_i, a number of required links L_r. The input graph G_i has a number of nodes n_i with a number of links l_i. The number of required links L_r is the number of links that should be added to the graph. The algorithm adds links to the graph with L_r iterations. To keep track of the selected links in each iteration, the algorithm adds this link to the selectedLinks list. In each iteration, the algorithm starts by adding the selected links from previous iterations to the graph.

The candidate set contains the links that are connected to the pairs with the lowest EPD values of the graph and not currently present in the graph. To find the best candidate link, each link in the candidate set is added to the graph and the EPD of the corresponding pair is computed and mapped to that link. Then, the link with the largest EPD is selected. In case there are multiple links with the same largest EPD, the least cost link is selected. This process is repeated until the user requested number of links are added.

This algorithm uses four functions: cost(l), epd(P), candidate(G), and bestLink(L). The cost function cost(l) returns the cost of adding a link l. In this paper, the cost is defined as the Euclidean distance between the two ends of the link. The effective path diversity function epd(P) computes the effective path diversity of the path P based on Equation 3. The bestLink(L) function returns the link with the highest EPD and lowest cost in case of multiple highest EPD values. The candidate(G) takes a graph G as input and returns a set of candidate tuples of two elements. The first element is a lowest EPD pair and the second element is a candidate link.

The candidate links are the set of all links in which one end is connected to a node in the lowest EPD pairs and the other end is connected to a node in the graph given that this link does not exist in the graph. For each pair and link in the candidate set, we add the link to the graph and compute the new EPD value of that pair with its cost. Finally, the link with the highest EPD and the lowest cost is selected using bestLink(L) function and then added to the selectedLinks list. The algorithm repeats this process as many times as the user requested. The pseudo code is shown in Algorithm 2.

```
Functions:
cost($l$) := cost of link $l$
epd($P$) := EPD value of path $P$
candidate($G$) := candidate links function
bestLink($L$) := maximum EPD value of links list $L$
Input:
$G_i$ := input graph
$L_r$ := number of required links
Output:
an ordered list of the added links
begin
    selectedLinks = []; empty ordered list
    links_epd_list = []; empty ordered list
    while $L_r > 0$ do
        $G = G_i$
        $G$.addlinks(selectedLinks)
        for $P, L$ in candidate($G$) do
            links_epd_list.append(($L$,epd($P$),cost($L$)))
        end
        selectedLinks.add(bestLink(links_epd_list))
        $L_r = L_r - 1$
    end
    return selectedLinks
end
```

Algorithm 2: Topology improvement algorithm

B. Improvement Algorithm Example

In this section, we explain how our heuristic algorithm improves a topology on a small-size graph. Figure 1 shows a sample graph with 5 nodes and 5 links. In this example, the hop-count threshold h is set to 10 and the number of diverse paths k is set to 4. The initial TGD value of this sample graph is 0.2023.

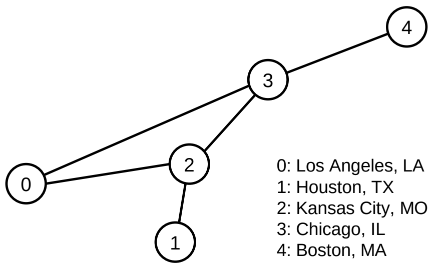


Fig. 1. Optimization graph example

Our heuristic algorithm examines the links connected to the least EPD pairs. The least EPD pairs are (1,2) and (3,4) with EPD of 0 since they have no alternative paths. Therefore, the candidate set consists of four possible links for each pair. To find the best candidate, we determine the resulting EPD of the corresponding pair after adding the candidate link and the cost incurred as shown in Table I. Then, we find the link that gives the highest pair EPD. Among the five candidate links, there are four links that give a high-pair EPD of 0.50. The next step is to find the lowest length link, which is the link (1,3). After adding this link, the new TGD of this graph is 0.4034, which is almost double the initial TGD.

TABLE I
EPD VALUES AND COST FOR CANDIDATE LINKS

Node Pair	Link	Resulting EPD	Cost
(1, 2)	(1, 0)	0.50	2,177
(1, 2)	(1, 3)	**0.50**	**1,043**
(1, 2)	(1, 4)	0.46	2,311
(3, 4)	(4, 0)	0.50	4,058
(3, 4)	(4, 2)	0.50	1,988

V. Analysis and Results

In this section, we present topological data we use. Next, we apply our heuristic improvement algorithm on three realistic backbone networks and study the results. Then, we apply three centrality-based attacks to non- and improved graphs and show how the robustness changes for each graph.

A. Topological Dataset

We study physical level communication networks that are geographically located within the continental United States. Therefore, we only include the 48 contiguous US states, the District of Columbia, and exclude Hawaii, Alaska, and other US territories. We make use of the publicly available CORONET [25], Internet2 [26], and Level 3 [27] fiber-level topologies.

TABLE II
PHYSICAL TOPOLOGICAL DATASET

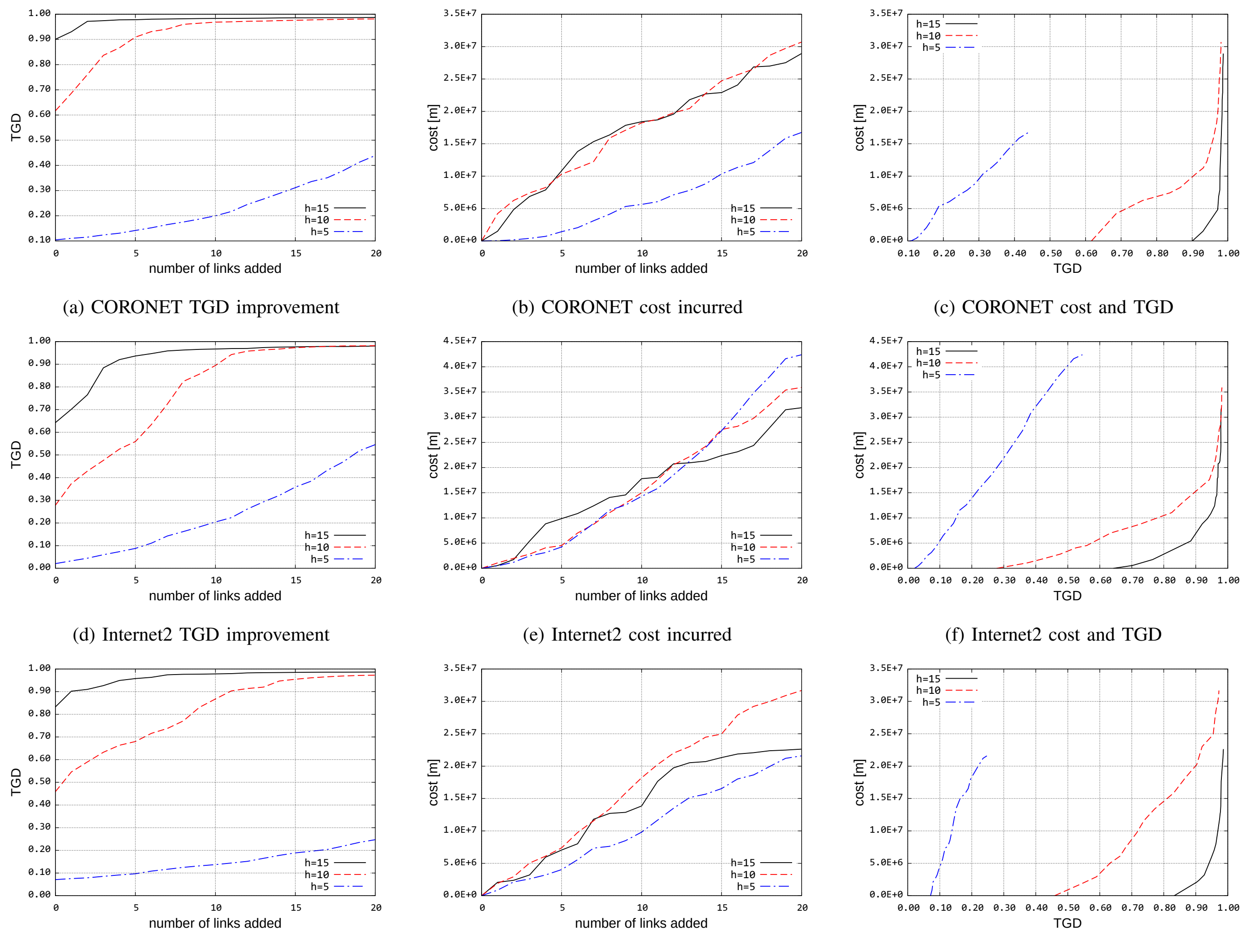
Network	Nodes	Links	Avg. Degree	Diameter	Avg. Hop
CORONET	75	99	2.64	17	6.45
Internet2	57	65	2.28	14	6.69
Level 3	99	132	2.67	19	7.65

In Table II, we list a number of relevant quantities for each of the provider networks. A detailed analysis of graph metrics for the given physical networks was presented in our earlier work [12]. Next, we apply our improvement algorithm on the CORONET, Internet2, and Level 3 fiber-level topologies.

B. Improvement Analysis

In this section, we apply the improvement algorithm on three realistic backbone service provider graphs and study the TGD improvement and the cost incurred for each graph as we add 20 links. We vary the value of h while the scaling-constant λ is set to 0.5 and the value of k is set 12.

The hop-count threshold h is a parameter that controls the length of the shortest path returned by the k diverse algorithm



(a) CORONET TGD improvement (b) CORONET cost incurred (c) CORONET cost and TGD

(d) Internet2 TGD improvement (e) Internet2 cost incurred (f) Internet2 cost and TGD

(g) Level 3 TGD improvement (h) Level 3 cost incurred (i) Level 3 cost and TGD

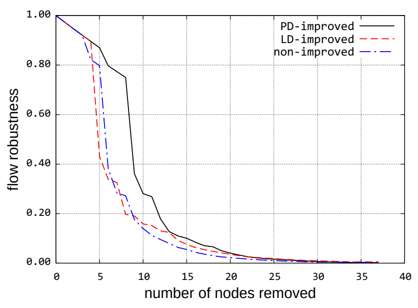
Fig. 2. Impact of varying h hop-count threshold on TGD and cost

introduced in Section III-C. Therefore, to get the optimal diverse paths, the value of h should be larger or equal to the diameter of the graph in order to examine all of the possible paths in the graph. However, for large graphs, large values of h may take an impractical time to calculate. Here, we apply the algorithm with several values of hop-count thresholds $h = \{5, 10, 15\}$. These values show how varying the parameter h affects the value of TGD. Figure 2 depicts the results of each hop-count threshold, which shows the TGD improvement as links are added, cost incurred with added links, and the TGD achieved as cost increases. As the hop-count threshold increases, the size of the candidate set also increases, which in turn increases the probability to have a higher EPD value. As a result, a 5 hop-count threshold has the lowest TGD while 10 and 15 have the median and the highest TGD respectively for all graphs as shown in Figures 2a, 2d, and 2g. However, the cost does not follow a pattern as the hop-count threshold increases since the cost for the highest EPD link for the 10 hop-count threshold could be less than the cost of the highest EPD for 15 hop threshold and vice versa as shown in Figures 2b, 2e, and 2h. Thus, the cost incurred depends on the initial topological properties such as the number of nodes and links, average degree, and node locations. The cost needed to achieve a certain TGD for all graphs are shown in Figures 2c, 2f, and 2i, which show that as the hop-count threshold increases, the cost to achieve a given TGD decreases in general.

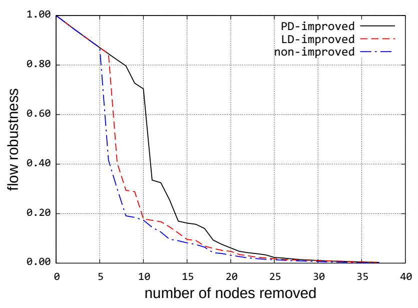
C. Robustness Evaluation

In this section, we present the set of attacks used to evaluate the robustness of the resulting non and improved. Then, we apply these attacks and show the results.

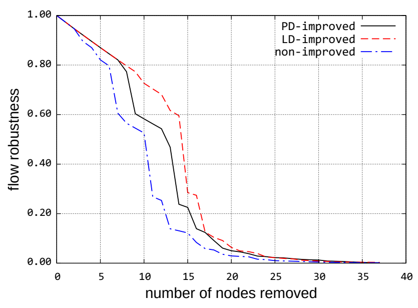
1) Flow robustness: Flow robustness [22] is a graph metric that measures the ratio of possible number of pair-connections, to the maximum number of pair-connection, $n(n-1)$. If the graph is partitioned, the possible number of pair-connections is the sum of $n(n-1)$ connections for each component. The range of flow robustness value is [0,1], and the flow robustness is 1 if the graph is not partitioned and 0 if the graph has no links.



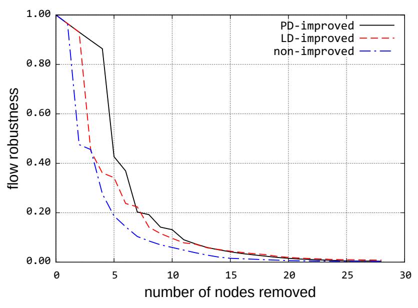
(a) CORONET betweenness-based attack



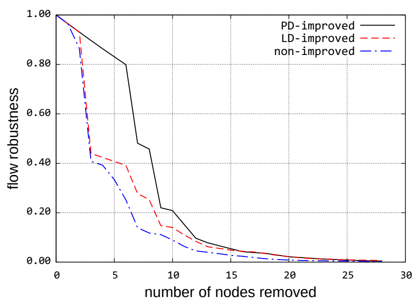
(b) CORONET closeness-based attack



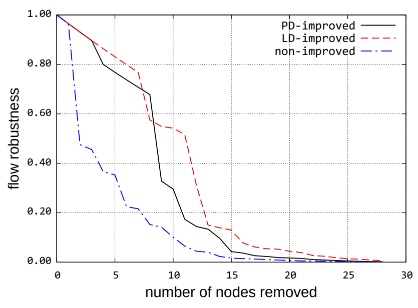
(c) CORONET degree-based attack



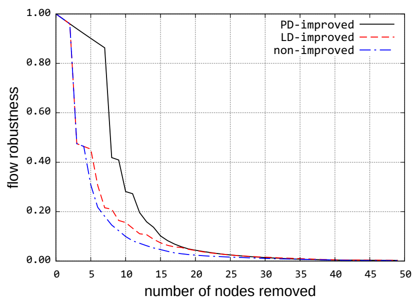
(d) Internet2 betweenness-based attack



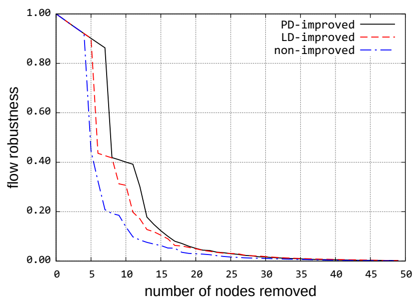
(e) Internet2 closeness-based attack



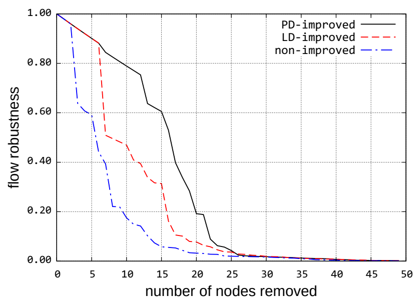
(f) Internet2 degree-based attack



(g) Level 3 betweenness-based attack



(h) Level 3 closeness-based attack



(i) Level 3 degree-based attack

Fig. 3. Flow robustness analysis of non- and improved graph

2) Graph centrality attacks: We use a graph-theoretic model to attack a given graph and show how its flow robustness changes after each node removal. We have three attack models, in which nodes with the highest centrality (i.e. betweenness, closeness, and degree) are removed. The list of removed nodes is determined *adaptively* for each attack model (i.e next-best candidate is recomputed after each removal). The adaptive removal of nodes gives a more correct selection for the highest centrality than the non-adaptive removal [28].

3) Lowest degree improvement: For comparison purposes, we introduce an intuitive improvement algorithm that improves the connectivity of a given graph by adding links to the smallest degree nodes. This algorithm adds one link repeatedly until number of links requested by the user is added. On each iteration, one end of the link is connected to the least degree node and the other end is connected to the next least degree node. If there are multiple least degree candidate links, the least cost link is selected to be added.

4) Robustness evaluation results: We show the results of applying the graph centrality attacks to path diversity improved (PD-improved), lowest degree improved (LD-improved), and non-improved topologies in Figure 3. For the set of PD-improved graphs, we choose the set generated using the hop-count threshold $h = 15$ and the number of diverse-path threshold $k = 12$ because both have more diverse and accurate results. For each graph, we apply the attack by removing half of its original number of nodes and calculate the flow robustness after each node removal. The node betweenness attack has the highest negative impact on flow robustness because it targets the most vital nodes in the graph as shown in Figures 3a, 3d, and 3g. The second highest negative impact on flow robustness is achieved by the closeness node attack since the target node has the highest closeness to all the other nodes in terms of hop count. The least negative impact on flow robustness comes from the highest degree node since it has a higher number of neighbors but is not necessarily used by most number of shortest paths.

Among the three provider graph analyses, the PD-improved graphs are more resilient than the LD-improved and non-improved graphs for betweenness and closeness attacks. For degree-based centrality attack, LD-improved graphs have higher flow robustness since links are added to the lowest degree nodes, which are targeted *the least* as shown in Figures 3c and 3f. Therefore, the links connected to the lowest degree nodes using LD-improved contribute more to flow robustness than links added using PD-improved during the degree-based attack. In Figure 3i, the degree attack for the Level 3 topology does not follow this pattern because we speculate that the links are added in the PD-improved are connected to relativity low degree nodes. Thus, these links are not prone to degree-based attacks and yet they improve the TGD of this graph. PD-improved graphs have higher flow robustness in most physical-level graphs because in PD-improved graphs, links are added to increase the number of diverse paths the most among all communicating nodes in the graph. Thus, when a node is removed from a PD-improved graph, it slightly affects the other communicating nodes since they have more alternative paths to reroute their traffic

VI. Conclusions and Future Work

Network design and optimization is a major area of research. We introduce a k-diverse path algorithm that considers both the diversity of the nodes and links in the returned paths. We present a heuristic algorithm that improves the total path diversity of a given graph. This algorithm improves the TGD of a graph by adding the cost-efficient link that increases the lowest EPD pair the most. We apply our algorithm to three service provider physical-level topologies. Using the flow robustness graph metric, the path diversity improved graphs are compared to both lowest degree non- and improved graphs as they are attacked using node removal based on highest node centrality graph metrics. The path diversity improved graphs show better resilience to these attacks compared to the lowest degree non- and improved graphs.

Acknowledgments

We would like to acknowledge Justin P. Rohrer, Yufei Cheng, and Dongsheng Zhang for discussions on this work. This research was supported in part by NSF grant CNS-1219028 (Resilient Network Design for Massive Failures and Attacks) and by NSF grant CNS-1050226 (Multilayer Network Resilience Analysis and Experimentation on GENI). Mohammed J.F. Alenazi is supported by SACM (Saudi Arabian Cultural Mission).

References

[1] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[2] J. G. Apostolopoulos and M. D. Trott, "Path Diversity for Enhanced Media Streaming," *IEEE Communications Magazine*, vol. 42, no. 8, pp. 80–87, 2004.

[3] A. Haider and R. Harris, "Recovery Techniques in Next Generation Networks," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 2–17, 2007.

[4] J. Han, D. Watson, and F. Jahanian, "An Experimental Study of Internet Path Diversity," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 273–288, 2006.

[5] J. He and J. Rexford, "Toward Internet-Wide Multipath Routing," *IEEE Network Magazine*, vol. 22, no. 2, pp. 16–21, 2008.

[6] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Transactions on Reliability*, vol. 35, no. 3, pp. 230–239, 1986.

[7] J. M. McQuillan, "Graph theory applied to optimal connectivity in computer networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 7, pp. 13–41, Apr. 1977.

[8] R. R. Boorstyn and H. Frank, "Large-scale network topological optimization," *IEEE Transactions on Communications*, vol. 25, pp. 29–47, January 1977.

[9] J. G. Klincewicz, "Hub location in backbone/tributary network design: a review," *Location Science*, vol. 6, pp. 307–335, December 1998.

[10] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 653–665, 1976.

[11] D. Mosk-Aoyama, "Maximum algebraic connectivity augmentation is NP-hard," *Operations Research Letters*, vol. 36, no. 6, pp. 677–679, 2008.

[12] E. K. Çetinkaya, M. J. F. Alenazi, A. M. Peck, J. P. Rohrer, and J. P. G. Sterbenz, "Multilevel Resilience Analysis of Transportation and Communication Networks," *Springer Telecommunication Systems Journal*, 2013. (accepted in July 2013).

[13] L. d. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas, "Characterization of complex networks: A survey of measurements," *Advances in Physics*, vol. 56, no. 1, pp. 167–242, 2007.

[14] M. J. F. Alenazi, E. K. Çetinkaya, and J. P. G. Sterbenz, "Network Design and Optimisation Based on Cost and Algebraic Connectivity," in *Proceedings of the 5th IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, (Almaty), pp. 193–200, September 2013.

[15] M. J. Alenazi, E. K. Çetinkaya, and J. P. Sterbenz, "Cost-efficient algebraic connectivity optimisation of backbone networks," *Optical Switching and Networking*, vol. 14, Part 2, pp. 107 – 116, 2014.

[16] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, 1974.

[17] J. W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, vol. 14, no. 2, 1984.

[18] R. Bhandari, "Optimal Diverse Routing in Telecommunication Fiber Networks," in *Proceedings of the IEEE INFOCOM*, vol. 3, (Toronto), pp. 1498–1508, June 1994.

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

[20] F. Wang and L. Gao, "Path Diversity Aware Interdomain Routing," in *Proceedings of the IEEE INFOCOM*, (Rio de Janeiro), pp. 307–315, April 2009.

[21] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1260–1271, 2006.

[22] J. P. Rohrer, A. Jabbar, and J. P. Sterbenz, "Path Diversification for Future Internet End-to-End Resilience and Survivability," *Springer Telecommunication Systems*, vol. 56, pp. 49–67, May 2014.

[23] J. P. Rohrer and J. P. G. Sterbenz, "Predicting topology survivability using path diversity," in *Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, (Budapest), pp. 95–101, October 2011.

[24] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path Splicing," in *Proceedings of the ACM SIGCOMM*, (Seattle, WA), pp. 27–38, August 2008.

[25] G. Clapp, R. A. Skoog, A. C. Von Lehmen, and B. Wilson, "Management of Switched Systems at 100 Tbps: the DARPA CORONET Program," in *International Conference on Photonics in Switching (PS)*, (Pisa), pp. 1–4, September 2009.

[26] "Internet2." http://www.internet2.edu.

[27] "Level 3 network map." http://maps.level3.com.

[28] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks," *Phys. Rev. E*, vol. 65, p. 056109, May 2002.