Realtime Robustification of Interdependent Networks under Cascading Attacks

Zhen Chen
School of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, US
Email: zhen.chen.1@asu.edu

Hanghang Tong
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, US
Email: hanghang.tong@asu.edu

Lei Ying
School of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, US
Email: lei.ying.2@asu.edu

***Abstract*—This paper studies the problem of robustifying an interdependent network by rewiring a small number of links in realtime during a cascading attack. Interdependent networks have been widely used to model interconnected complex systems such as a critical infrastructure network including both the power grid and the Internet. Realtime robustification of interdependent networks, therefore, has significant practical importance. This paper formulates the problem using the Markov decision process (MDP) framework. We first show the problem is NP-hard and then develop an effective and efficient greedy algorithm, named *R*EAL*W*IRE, to robustify the network in realtime. *R*EAL*W*IRE scores each link (and each node) based on the expected number of links failures resulted from the failure of the link (or the node), and rewires the links greedily according to the scores. Extensive experimental results show that *R*EAL*W*IRE outperforms other algorithms on multiple trobustness metrics.**

***Keywords*-Interdependent networks; network robustness; cascading failures; Markov decision processes**

I. Introduction

Interdependent networks have been widely used to model and analyze interconnected complex systems such as the Internet, online social networks, transportation systems, biochemical reactions, etc [1, 2]. For example, a two-layer interdependent network that consists of a power-grid network and a communication network has been used to explain the large-scale electrical blackout in Italy in 2003 [3]. Over the past decade, there have been a number of studies on the structures, properties and features of interdependent networks such as their robustness, stability and connectivity (see, e.g. [4, 5, 3, 6, 2, 7, 8, 1, 9, 10, 11, 12, 13]). Among these topics, network robustness is of particular practical importance because some popular network topologies are known to have low tolerance to structural damages (e.g. the diameter doubled after only 5% of the most connected nodes are removed on the scale-free network [8]), and interdependent networks are even more fragile because of the inter-network connections [3]. Research on robustness can help guide the design of resilient network topologies and the design of network control to mitigate cascading failures caused by either cyber attacks or natural diasters.

However, despite recent progresses on understanding robustness of interdependent networks, existing results almost exclusively focus on the design of robust network topologies that are resilient to cascading attacks, instead of on control and intervention mechanisms to mitigate cascading failures in realtime. The later is critically important. As we will see in this paper, even a small number of rewiring in realtime based on the state of the cascading attack can dramatically reduce the scale of the attack and minimize its damage.

In this paper, we consider the problem of robustifying an interdependent network in realtime under a localized attack [14], where attacked nodes attack their neighbors so the attack spreads in the network hop-by-hop starting from the source. We assume an interdependent network with two layers (also called subnetworks). The functioning of a node depends on a set of nodes from the other layer. By modeling propagation of a cascading attack as a Markov chain, we formulate the problem as a Markov decision process (MDP) problem where the objective is to maximize the weighted time-average of some network robustness metric by rewiring a given number of links at each time. Due to the complex interactions of different layers and the complex dynamics of cascading failures, the MDP problem suffers from the curse of dimensionality and is NP-hard. Therefore, we focus on greedy algorithms to rewire links in real-time. The main contributions of this paper are summarized below.

- **Problem Formulation:** We formulate *the interdependent network link rewiring problem* as a Markov decision process (MDP) problem, and prove that the problem is NP-hard by reducing the maximum coverage problem to our MDP problem.
- **Algorithm:** We propose a greedy algorithm, named ***R*EAL*W*IRE** to rewire the links during the attack. ***R*EAL*W*IRE** scores each link (and each node) based on the expected number of links failures resulted from the failure of the link (or the node), and rewires the links greedily according to the scores.
- **Analysis:** We compare the performance of ***R*EAL*W*IRE** with the exact solution of the MDP problem on a small network. The results show that ***R*EAL*W*IRE** provides a

nearly-optimal solution to the MDP problem.
- **Empirical Evaluations:** We evaluate the performance of the algorithm on various interdependent networks formed by the real networks including an air traffic network, the Internet Autonomous Systems (IAS) network and a power grid network under simulated localized attacks. In most cases, when a large fraction of nodes in the networks are attacked, our algorithm outperforms other algorithms.

We would like to remark that we consider an abstract interdependent network model where a few links can be rewired at each time slot. This "rewiring" may have different meanings on different networks. For a communication network, "rewiring" means establishing a new communication path. For a social network, "rewiring" means establishing a new collaboration between two persons. For a power-grid, "rewiring" may mean shutting down a transmission line and activate a back-up transmission line. The practical implementation of "rewiring" in real systems is beyond the scope of this paper. The focus of this paper is to answer how to rewire a small number of links to robustify the network when rewiring is possible.

A. Related work

Network robustness has been studied in the literature. One of the focuses is to define meaningful robustness metrics and then optimize a network topology to maximize the target robustness metric. For example, Schneider et al. [12] proposed a robustness metric called node-robustness and developed a greedy algorithm to switch links to improve this node-robustness. Zeng and Liu [13] later defined a related metric called link-robustness and proposed a similar greedy algorithm to improve the link-robustness. Chan et al. [15] used the natural connectivity as the robustness metric and proposed algorithms to modify the network structure to maximize the natural connectivity.

Another line of research is on minimizing or maximzing information diffusion process in networks [16, 17]. Tong et al. [16] proposed algorithms to modify the leading eigenvalue of the adjacency matrix by adding or removing edges to limit or facilitate the information diffusion. Zhang et al. [17] proposed to limit propagation by removing some nodes or edges at a group scale. Chan et al. [18] studied the problem of identifying a robust subgraph. There are also papers focusing on the critical threshold of information diffusion: a threshold such that when the number of nodes removed exceeds the threshold, the network does not have a giant component anymore [19, 9, 4, 14]. In particular, Cohen et al. [19] used the percolation theory to calculate the critical threshold of scale-free networks under a random attack. Yuan et al. [4] used both theoretical analysis and experimental studies to understand the relation between the breadth of the degree distribution and the critical threshold.

The robustness of interdependent networks has also been studied recently. Chen et al. [20] developed a near-optimal algorithm to identify a subset of nodes at the control layer whose failures can lead to the maximum damage to the target layers. Buldyrev et al. [3] proposed a cascading failure model for interdependent networks and developed a framework for analyzing the critical threshold. However, there are only a few work on preserving the robustness of interdependent networks during the attack. From the best of our knowledge, the only related work is Di Muro et al. [2] which proposed a method to recover a fraction of attacked nodes during a cascading attack to improve network robustness. For many physical networks, failed nodes are not recoverable or cannot be recovered in a short time period. In this paper, we focus on rewiring edges instead of recovering failed nodes.

This paper is organized as follows. Section II introduces the network model, the attack model and the MDP formulation of *the interdependent network link rewiring problem*. Section III presents our algorithm, ***REALWIRE*** and Section IV summarizes our experimental studies. Finally, Section V concludes this paper.

II. Model and Problem formulation

A. Interdependent Networks

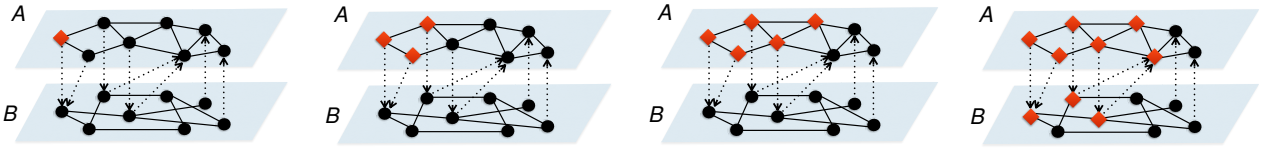
We consider an interdependent network model inspired by [3], which consists two networks, network A and network B. Let $G_a(\mathcal{V}_a, \mathcal{E}_a)$ ($G_b(\mathcal{V}_b, \mathcal{E}_b)$) denote the graph structure of network A (B), where $\mathcal{V}_a$ ($\mathcal{V}_b$) is the set of vertices and $\mathcal{E}_a$ ($\mathcal{E}_b$) is the set of links (edges) of graph G_a (G_b). Furthermore, the two networks are connected by a set of directed dependency links, denoted by $\mathcal{D}$. Each dependency link is a directed link which represents the dependency of the head node on the tail node.

The interdependent network, therefore, is defined by the tuple $(G_a(\mathcal{V}_a, \mathcal{E}_a), G_b(\mathcal{V}_b, \mathcal{E}_b), \mathcal{D})$. For each vertex $v \in \mathcal{V}_b$ ($\mathcal{V}_a$) we define its supporting node set to be $\mathcal{R}_v = \{u | \forall (u, v) \in \mathcal{D}\}$, i.e. the set of nodes on the other network, on which the functioning of node v depends. We assume node v fails if all nodes in its supporting node set $\mathcal{R}_v$ fail.

B. Localized Attack Model

We consider a localized attack model similar to that in [4]. The attack spreads in network A from the attack source. The failures of nodes in network A then lead to the failures of nodes in network B. Specifically, at the beginning of the localized attack, a single node from network A is attacked and fails. Then at each following time slot, each attacked node at network A may attack their healthy neighbors in node A. We assume the attack stops with probability p at each time slot before time T and stops at time slot T with probability one if it does not stop before that.

There are two types of failures under this cascading attack:



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$ (d) $t > 2$

Figure 1: An example of the localized attack with $p = 0$ and $T = 2$. The red diamond represents failures. At time $t = 0$, a node in network A is attacked and fails. Then, the attack starts to propagate. At time $t = 1$, the neighbors of the previous attacked nodes in network A get attacked and fail. The similar procedure goes on at time $t = 2$. Then, the attack stops and there are 4 more node failures caused by the failures of their supporting nodes.

- Type-I Failures: the failures directly resulted from the attack. A node stops functioning because it is attacked. These failures only occur at network A.
- Type-II Failures: the failures caused by failures in the interdependent network. A node stops functioning because of all its supporting nodes fail. These failures can occur at both network A and network B, and cascades in the network even after the attack stops.

We assume type-I failures occur at a faster time scale than tpye-II failures. In other words, the attack first propagates on network A and then type-II failures occur after the attack stops as shown in the example in Figure 1. Consider an interdependent network where the communication network is the first layer and the power grid is the second layer. In general, the cyber attack on the communication network spreads very fast, but it takes much longer time for generators and power lines to fail. For example, in the sequence of events that led to the cascading failures of the power grid in India in 2012, the 2nd failure in the power grid occurred one hour and twenty minutes after the first failure and the 3rd failure was 58 minutes after the 2nd [21]. This motivates our assumption that failures due to the attack occur at a faster time scale than failures due to the failures of the supporting nodes.

C. The Markov Decision Process (MDP) Formulation

Recall that the interdependent network is denoted by the tuple $(G_a(\mathcal{V}_a, \mathcal{E}_a), G_b(\mathcal{V}_b, \mathcal{E}_b), \mathcal{D})$. We next define the Markov process under the localized attack. Define $\mathcal{C}_t$ to be the set of nodes that are attacked and failed at time slot t on G_a, and $\mathcal{F}_t = \cup_{i=0}^{t} \mathcal{C}_i$, which is the set of nodes that have been attacked and failed by time slot t. Furthermore, we define $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$ to be the current graph of network A before rewiring occurs at time slot t, which is obtained by removing nodes from type-I failures and rewiring links in the previous time slots. Furthermore, we have $G_a^0 = G_a$ and $G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t) = G_b^0$ for $t < T$ under the assumption that type-II failures occur after the attack stops.

Now define the state of the Markov process at time slot t to be a tuple

$$s_t = (G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t), G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t), \mathcal{C}_t, \mathcal{F}_t, \mathcal{D}).$$

At time slot 0, we have

$$s_0 = (G_a(\mathcal{V}_a, \mathcal{E}_a), G_b(\mathcal{V}_b, \mathcal{E}_b), \{source\}, \{source\}, \mathcal{D}),$$

where "source" is a single node attacked at time slot 0. The action a_t given state s_t is link rewiring on graph G_a^t. We assume that at each time slot, we can at most rewire z links or r fraction of links that connect the current attacked nodes and healthy nodes, whichever is smaller. Let n_t denote the number of links rewired at time slot t. We have

$$n_t = \min\{\lfloor r|\mathcal{W}_t|\rfloor, z\}, \tag{1}$$

where

$$\mathcal{W}_t = \{(v, u) \in \mathcal{E}_a^t | v \in \mathcal{F}_t \text{ and } u \in \mathcal{V}_a^t \setminus \mathcal{F}_t\}$$

is the set of edges that connect the current attacked nodes and healthy nodes in network A.

We remark under the assumption above, the attacked nodes cannot be completely isolated from healthy nodes by cutting all the edges between attacked nodes and healthy nodes. If it is possible, then a trivial solution to protect the network is to isolate the attacked nodes immediately.

Given state

$$s_t = (G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t), G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t), \mathcal{C}_t, \mathcal{F}_t, \mathcal{D}) \ (t \geq 0),$$

and action a_t, define $G_a^{t\prime}(\mathcal{V}_a^{t\prime}, \mathcal{E}_a^{t\prime})$ to be the graph at time t after action a_t has been taken. Then, the state transitions of the Markov process can be described as follows:

1) If $\mathcal{C}_t = \emptyset$, then $\mathcal{C}_{t+1} = \emptyset$, $\mathcal{F}_{t+1} = \mathcal{F}_t$, $G_a^{t+1} = G_a^{t\prime}$ and $G_b^{t+1} = G_b^t$.
2) If $\mathcal{C}_t \neq \emptyset$, then $G_a^{t+1} = G_a^{t\prime}$ and $G_b^{t+1} = G_b^t$. Since the attack stops with probability p at each time slot before T, we have

$$\mathcal{C}_{t+1} = \begin{cases} \emptyset, \text{ with probability } p, \\ \{u | \forall u \in \mathcal{V}_a^{t\prime} \setminus \mathcal{F}_t, (v, u) \in \mathcal{E}_a^{t\prime}, v \in \mathcal{F}_t\}, \\ \text{with probability } 1 - p, \end{cases}$$

and $\mathcal{F}_{t+1} = \mathcal{F}_t \cup \mathcal{C}_{t+1}$.

Let $\mathcal{S}$ denote the state space and $\mathcal{A}$ denote the set of actions. We define reward function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

which is defined to be the largest connected component by considering both type-I and type-II attacks. In particular,

1) If $\mathcal{C}_t \neq \emptyset$, we define

$$\mathcal{P}_t = \{u | \forall u \in \mathcal{V}_a^{t\prime} \setminus \mathcal{F}_t, (v,u) \in \mathcal{E}_a^{t\prime}, v \in \mathcal{F}_t\},$$

which is the set of nodes that can be attacked in the next time slot. Define $\mathcal{O}_a^t$ to be the set of failed nodes caused by the failures of $\mathcal{F}_t \cup \mathcal{P}_t$ on network A and $\mathcal{O}_b^t$ to be the set failed nodes caused by the failures of $\mathcal{F}_t \cup \mathcal{P}_t$ on network B. Failures in $\mathcal{O}_a^t \cup \mathcal{O}_b^t$ are type-II failures. Given $\mathcal{F}_t \cup \mathcal{P}_t$, it may take multiple iterations to identify $\mathcal{O}_a^t$ and $\mathcal{O}_b^t$ by emulating the cascading failures.

Define f_a^t to be the robustness measure of network A after removing the nodes $\mathcal{F}_t \cup \mathcal{P}_t \cup \mathcal{O}_a^t$ (e.g. size of the largest connected component of $G_a^{t\prime}$). Similarly, define f_b^t to be robustness measure on network B after removing nodes in $\mathcal{O}_b^t$. The reward at time slot t is defined to be $f_t = f_a^t + f_b^t$.

2) Otherwise, we set $f_t = 0$.

We define a reward function here for general robustness measures, and will evaluate our algorithms under several well-known robustness metrics in the numerical evaluations. For example, we will consider the size of the largest connected component as the robustness metric, which is also used by multiple previous works [2, 4, 3].

Given the model defined above, the value function at time t is defined to be

$$V_t(s_t) = \max_{a_t}(f(s_t,a_t) + \gamma \sum_{s_{t+1}} \mathbb{P}(s_{t+1}|s_t,a_t)V_{t+1}(s_{t+1}), \tag{2}$$

where γ is the discount factor. If we set $\gamma = 1$, we have

$$V_t(s_t) = \max_{a_t}(f(s_t,a_t) + \sum_{s_{t+1}} \mathbb{P}(s_{t+1}|s_t,a_t)V_{t+1}(s_{t+1}). \tag{3}$$

Calculating the value function $V_0(s_0)$ is equivalent to solving the following problem [22]:

Problem 1 (MDP problem)**.**

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} f(s_t,a_t) \middle| s_0\right], \tag{4}$$

where π is a policy that decides which the rewiring action at each state.

III. A Low-Complexity and Greedy Algorithm

A. NP-Hardness of the Problem

Similar to other MDP problems, the problem defined in the previous section suffers from the curse of dimensionality. In particular,

1) The cardinality of the state space and the action space are both huge, which make it impossible to solve the problem by using the dynamical programming.
2) There is no closed-form expression of function f.

Theorem 1. *The MDP problem defined in Problem 1 is NP-hard.*

This theorem is proved by reducing the maximum coverage problem to Problem 1. The detailed proof can be found in our technical report [23].

B. Proposed Algorithm - *REALWIRE*

Since the MDP problem is hard to solve, we propose a heuristic algorithm, ***REALWIRE***, by estimating the expected number of link failures resulted from the failure of a given node or link.

Under ***REALWIRE*** we first construct $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$ a directed subgraph from G_a^t at time t, where the node set $\tilde{\mathcal{V}}_a^t = \mathcal{V}_a^t \setminus \mathcal{F}_{t-1}$ is the set of healthy nodes at the beginning of time slot t, and the link set

$$\tilde{\mathcal{E}}_a^t = \bigcup_{v \in \mathcal{C}_t, u \in \mathcal{V}_a^t \setminus \mathcal{F}_t} \Big\{(x,y) | \ (x,y) \in \mathcal{E}_a^t, \text{ and } (x,y) \text{ is on some shortest path on graph } G_a^t\Big\}$$

includes links on the shortest paths. The neighbors of the current attacked nodes are defined to be level 0, and the level of other nodes are defined to be their distance to the corresponding node at level 0.

After constructing $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$, for $\forall v \in \tilde{\mathcal{V}}_a^t$, we define the expected number of link failures contributed to node v at time t, denoted by f_v^t, to be

$$f_v^t = \sum_{u \in \mathcal{N}_v^t} \frac{f_u^t}{d_{in}^t(u)} + (1-p)^{l_v^t}\left(d_{out}^t(v) + w_v^t + \frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}\right). \tag{5}$$

In the equation above, p is the stopping probability of the localized attack at each time slot, $\mathcal{N}_v^t$ is the set of successors of node v on the directed graph g_a^t, $d_{in}^t(v)$ is the in-degree of node v on g_a^t, $d_{out}^t(v)$ is the out-degree of node v on g_a^t, $d_a^t(v)$ is the degree of node v on subgraph of g_a^t with node set $\tilde{\mathcal{V}}_a^t$, and l_v^t is the level of node v on graph g_a^t, where $l_v^t = \min_{u \in \mathcal{C}_t} dist^t(u,v)$, and $dist^t(u,v)$ is the distance from node u to v on graph g_a^t.

- $d_{out}^t(v)$ represents the failures of out-links caused by the attack from node v to its neighbors.
- $\frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}$ is the number of link failures caused by the attack of node v on the edges with endpoints at the same level l_v^t on graph G_a^t. Here we divide it by two because we attribute half of each failure to each endpoint for each failed link.
- w_v^t represents the links failures caused by type-II node failures, which can be calculated according to Algorithm 2.

- $\frac{f_u^t}{d_{in}^t(u)}$ is the expected link failures caused by node u, at level $l_v^t + 1$, whose shortest paths from the current attack nodes pass through node v.
- $(1-p)^{l_v^t}$ is the probability, with which the attack continues at level l_v^t.

In ***REALWIRE***, we use the total expected number of link failures,

$$F_{t+1}(s_t, a_t) = \sum_{v \in \mathcal{V}_a^t \setminus \mathcal{F}_t} (1-p)^{l_v^t} (d_{out}^t(v) + w_v^t + \frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}) \tag{6}$$

to replace $V_{t+1}(S_{t+1}(S_t, a_t))$ in equation (3), and the number of link failures brought by the current attack,

$$f_t(s_t, a_t) = \sum_{v \in \mathcal{C}_t} d_{out}^t(v) + w_v^t + \frac{1}{2} \sum_{v \in \mathcal{C}_t} (d_a^t(v) - d_{out}^t(v) \tag{7}$$

to replace $f(s_t, a_t(n_t))$. Thus, equation (3) becomes

$$\begin{aligned} F_t(s_t) &= \min_{a_t} f_t(s_t, a_t) + F_{t+1}(s_t, a_t) \\ &= \min_{a_t} \sum_{v \in \mathcal{V}_a^t \setminus \mathcal{F}_t} {}_1 (1-p)^{l_v^t} \times \\ &\left(d_{out}^t(v) + w_v^t + \frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2} \right). \end{aligned} \tag{8}$$

REALWIRE can be divided into four steps:

1) At each time slot t, we build graph $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$. An example can be found in Figure 2. Figure 2.a shows the network at time 0 with only one attacked node in network A. The result of $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$ can be found in Figure 2.b. The pseudo code can be found in [23].
2) We calculate the expected number of link failures for any node $v \in \tilde{\mathcal{V}}_a^t$ according to Equation (5). Figure 2.c shows the calculations of the expected number of link failures for the example.
3) For each edge $(u, v) \in \tilde{\mathcal{E}}_a^t$, we assign a score $w_{u,v} = \frac{f_v^t}{d_{in}^t(v)}$. The result of link scores of the example is presented in Figure 2.d.
4) Pick the top n links with the highest scores. For each one of those links (u, v), reattach node v to another node on g_a^t on the highest level with a probability proportional to its degree. The pseudo code can be found in [23]. Finally, the network after rewiring is in Figure 2.e.

The pseudo code can be found in Algorithm 1. Higher the score an edge has, more shortest paths will be affected by removing the edge. Thus, in Algorithm 1, by choosing the link with the highest score to remove, ***REALWIRE*** aims at increasing the distances between the current attacked nodes and unattacked nodes. Then, we reattach the node to the highest level in g_a^t so that the levels, l_v^t, for the nodes whose shortest paths will be impacted by the edge can be increased the most. Thus, according to equation (8), our algorithm aims to minimize $F_t(s_t)$ in a greedy manner.

C. Complexity Analysis

Lemma 1. *The complexity of our algorithm is* $O(T|\mathcal{V}_a||\mathcal{D}| + T|\mathcal{E}_a|)$.

Proof: At each time slot, the computational complexity includes three parts:

1) Calculating w_v^t for each node v, which contributes a complexity $O(|\mathcal{V}_a||\mathcal{D}|)$.
2) Finding the network g_a^t. The complexity is $O(|\mathcal{V}_a| + |\mathcal{E}_a|)$.
3) Selecting the top n_t links with highest scores. The complexity is $O(|\mathcal{E}_a| \log n_t)$.

Since n_t is upper bounded by a constant, the complexity becomes $O(|\mathcal{V}_a||\mathcal{D}| + |\mathcal{E}_a|)$. Assume T is the duration of the attack, then the complexity of the algorithm is $O(T|\mathcal{V}_a||\mathcal{D}| + T|\mathcal{E}_a|)$. ■

Figure 3 plots the wall-clock time versus the network size for the BA-BA networks with attack duration $T = 4$. From Figure 3, we can see the complexity is nearly linear.

Algorithm 1: *REALWIRE*

Input : Graph $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$, $G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t)$, current attack $\mathcal{C}_t$, number of rewired links n, stopping probability p.
Output: $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$.

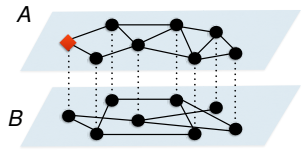
```
Construct graph g_a^t(Ṽ_a^t, Ẽ_a^t) ;
Calculate f_v^t ∀v ∈ Ṽ_a^t according to Equation (5);
w_{u,v} ← f_v^t / d_in^t(v) for ∀(u, v) ∈ g_a^t;
W ← the n edges with highest w_{u,v}, i ← 1;
for 1 ≤ i ≤ n do
    Remove edge (u, v) on G_a^t with (u, v) = W[i];
    Reattach node v to another node on g_a^t [23];
end
```

IV. Empirical Evaluations

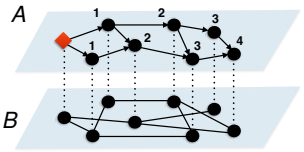
A. Robustness Metrics

We use the following performance metrics to measure the network robustness.

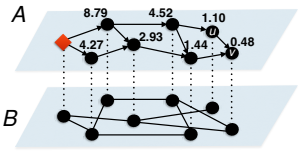
1) Largest connected component fraction: The size of the largest connected component of graph G_a (G_b) after removing all failed nodes divided by the original size of the graph.
2) Natural connectivity: It is defined in [24] as $\bar{\lambda} = \ln(\frac{1}{N} \sum_{i=1}^{N} e^{\lambda_i})$, where λ_i is the i^{th} largest eigenvalue of the adjacency matrix of a graph. It can be used to measure the number of closed walks on the graph. Since in the interdependent networks, there are two networks G_a and G_b, after removing the failed nodes,



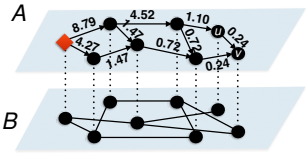
(a) In this example, assume at current time slot, in each subnetwork, there is only one attacked node, which is marked by red diamond. The dash lines between two layers are bidirectional dependency links.



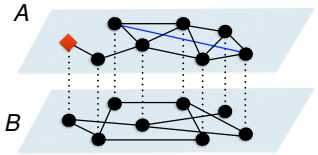
(b) In the first step, we run a breadth-first search from the current attacked node in subnetwork A. The number around each healthy node in subnetwork A represents the distance from current attack.



(c) Calculate the expected number of link failures for each healthy node in subnetwork A according to (5). For example, with assumption of $p = 0.3$, for node v, we have $f_v^t = 0.7^4 \times 2 = 0.48$. Then, we have $f_u^t = f_v^t/2 + 0.7^3 \times 2.5 = 1.10$.



(d) We assign the expected number of link failures of each node equally to the incoming links of these nodes.



(e) Finally, we choose links with highest scores to cut and reattach the endpoints to the nodes with the largest level number.

Figure 2: An example of our algorithm.

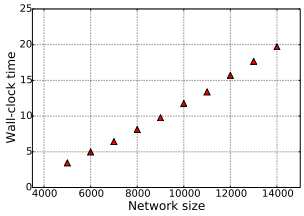


Figure 3: The wall-clock time vs network size for BA networks when attack time duration is 4.

Algorithm 2: Calculate w_v^t

Input : Graph $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$, current attack $\mathcal{C}_t$, attacked nodes $\mathcal{F}_t$, node v, the set $VisitedEdges$, the score dictionary D

Output: A score

if $v \in \mathcal{V}_a^t$ **then** $D_v \leftarrow d_a^t(v)$;
else $D_v \leftarrow d_b(v)$;
for $(v, w) \in \mathcal{D}$ **do**
  if $(v, w) \notin VisitedEdges$ *and* $(w, v) \notin VisitedEdges$; **then**
    $VisitedEdges$.add$((v, w))$;
    if D *does not contain* D_w **then**
      Calculate D_w using Algorithm 2 based current $VisitedEdges$ and D;
    $D_v \leftarrow D_v + \frac{D_w}{||\{(u,w)|\forall(u,w)\in\mathcal{D}\}||}$;
end
Return D_v;

we calculate the natural connectivity for each network and add them together.

3) Spectral radius: The spectral radius is defined as the largest absolution value of the eigenvalues of the adjacency matrix.
4) Spectral gap: The spectral gap of a graph is the difference between the largest and the second largest eigenvalues of the adjacency matrix. The spectral gap is closely related to the expansion properties of the graph. It has been used as a robustness metric in [25].

B. Optimality on Small Networks

For a small size network, we can calculate the optimal solution for the MDP problem defined in Problem (1). In this experiment, we used the Florentine families graph, which only contains 15 nodes. Since there are two networks, A and B, involved in the interdependent network model, we used the Florentine families graph to represent both A and B and randomly assign the name $0, 1, \ldots, 14$ to each node. Then, the functioning of one node in B depends on the node with the same name in A. Then, we let the attack start from each node in the network A and propagate for two time slots. The result is listed in Figure 4. The bars of the optimal solution were generated based on the maximum objective value defined in (4) over all possible rewiring traces. For other algorithms, we repeated the evaluation 500 times for each source and took the average. Based on Figure 4, we can see that the value of the MDP objective generated by our algorithm is close to the optimal solution.

To calculate the optimal solution of the MDP problem, we need to consider every possible rewiring of the networks, which means as the increase of the propagation time, T,

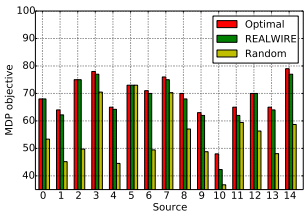


Figure 4: Florentine families network

the computation time grows exponentially, which makes the calculation of the optimal solution for large networks impossible. For example, the average number of possible rewiring combinations over all possible attack sources is 312 for $T = 1$, $103,342$ for $T = 2$, and $13,718,030$ for $T = 3$.

C. General Networks

We tested our algorithm on the following networks:

1) BA network: The network is generated according to the Barabasi-Albert preferential attachment model. Each new node brings three edges.
2) Air traffic network: The network has been built based on one year (2016) of interval USA air traffic data[1], which includes 1,243 airports connected by 16,106 routes (links).
3) IAS network: The IAS network is based on the Internet Autonomous Systems peering information inferred from the Oregon route-views [26]. The network includes 6,474 nodes and 13,895 edges[2].
4) Power grid network: This network is used to represent the topology of the Western States Power Grid of the United States, which includes 4,941 nodes and 6,594 edges [7].

From each of the network mentioned above, we built the following interdependent networks.

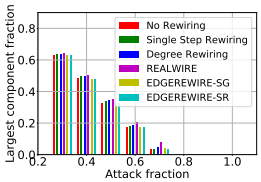
- The BA-BA network: Both the first layer and second layer are BA networks with 2,000 nodes generated randomly. We generated a one-to-one mapping between nodes from both layers as the dependency links.
- The IAS-PG network: For each node v in the IAS network or the power grid network, we uniformly picked an integer number d from 0 to 2 and randomly chose d nodes from the other layer as the supporting nodes of node v.
- The IAS-Air network: Since the communication network can impact the air traffic network and not the other way around, we generated a single direction interdependent network in this case. For each node v in the air traffic network, we randomly picked $0 \sim 2$ nodes from the other layer as the supporting node of node v.

[1]https://www.transtats.bts.gov/TRAFFIC/
[2]http://snap.stanford.edu/data/as.htmls

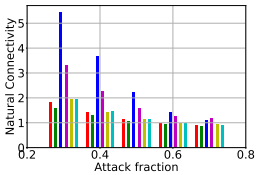
We compared our algorithm with the following algorithms:

- Single step rewiring: Instead of rewiring during the attack, it rewires the links before the attack. This algorithm aims to balance the degree of two endpoints connected by each edge.
  - For each link (u, v) on G_a, we define the degree difference to be $|d_a(u) - d_a(v)|$. We rank all the links according to their degree differences.
  - We pick the top n links with largest degree differences and remove those links.
  - For each removed link, we rewire the endpoint with the smaller degree to another node in the network with a small degree.
- Degree rewiring: This algorithm consists of the following steps:
  - At each time slot t, we only rank all the links that connect attacked nodes and unattacked nodes and rank them according to a score based on the degree of their unattacked endpoints. For example, assume there is an edge $(u, v) \in G_a^t$, while $u \in \mathcal{F}_t$ and $v \in \mathcal{V}_a^t \setminus \mathcal{F}_t$. Then the score of edge (u, v) is $d_a^t(v) + \sum_{w \in \mathcal{V}_b, \text{ while } v \in \mathcal{R}_w} \frac{d_b(w)}{|\mathcal{R}_w|}$, where $d_b(w)$ is the degree of node w on G_b, and $d_a^t(v)$ is the degree of node v on the subgraph of G_a^t with node set $\mathcal{V}_a^t \setminus \mathcal{F}_{t-1}$.
  - We pick n_t links with highest scores to remove and then reattach the unattacked endpoint of each link to another unattacked with a higher score. Here n_t is defined in Equation (1).
- EDGEREWIRE: The algorithm is proposed in [27].
  - For each rewiring, the algorithm picks an edge that increases the robustness measure of network A the most to add and then choose an edge between an endpoint of the newly added edge and another node that decreases the robustness metric of network A the least to cut.
  - At each time slot, the algorithm repeats the previous step for n_t times. The robustness metrics used here are spectral gap (SG) and spectral radius (SR). Since the algorithm requires us to consider each possible edge to add to the network, which is proportional to $|\mathcal{V}_a|^2$ for a sparse network, this algorithm runs slow for large networks compared to others. Thus, we only include the results of this algorithm for the BA-BA network.

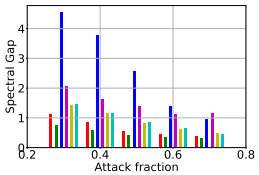
We evaluated our algorithm with a more general SI model for cascading attacks, where at each time slot each attacked node in network A can attack its unattacked neighbors with certain probability p_a. In the experiments, we set $p_a = 0.7$. The attack stops after a certain fraction of nodes have been attacked. For the number of rewired links at each



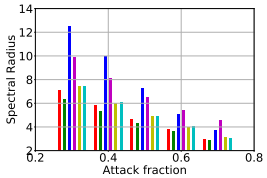
(a) Largest component fraction of network A.



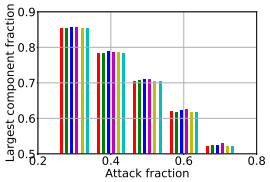
(b) Natural connectivity of network A.



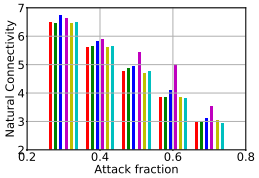
(c) Spectral gap of network A.



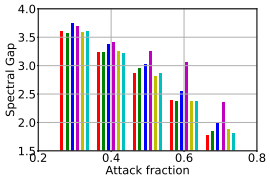
(d) Spectral radius of network A.



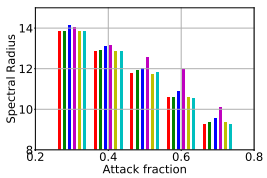
(e) Largest component fraction of network B.



(f) Natural connectivity of network B.

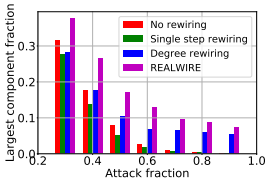


(g) Spectral gap of network B.

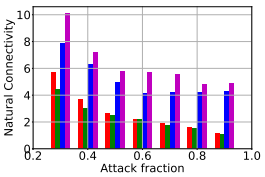


(h) Spectral radius of network B.

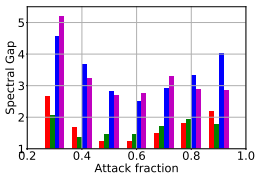
Figure 5: The BA-BA network.



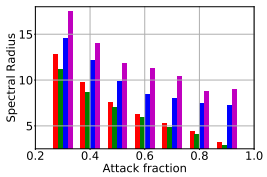
(a) Largest component fraction of network A.



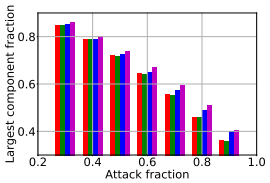
(b) Natural connectivity of network A.



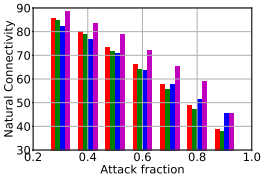
(c) Spectral gap of network A.



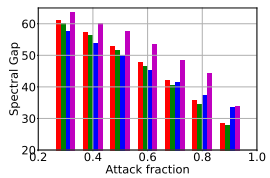
(d) Spectral radius of network A.



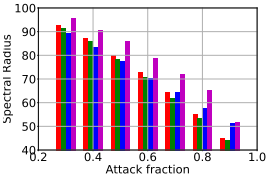
(e) Largest component fraction of network B.



(f) Natural connectivity of network B.



(g) Spectral gap of network B.



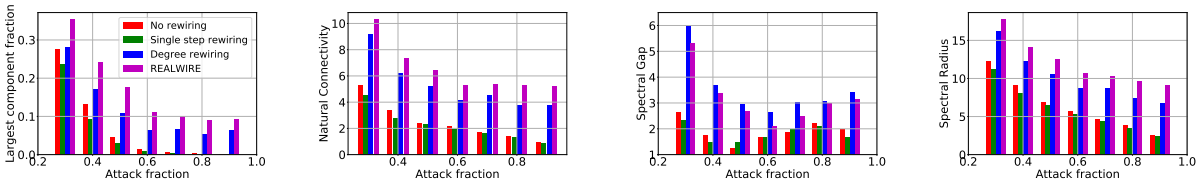
(h) Spectral radius of network B.

Figure 6: The IAS-Air network.

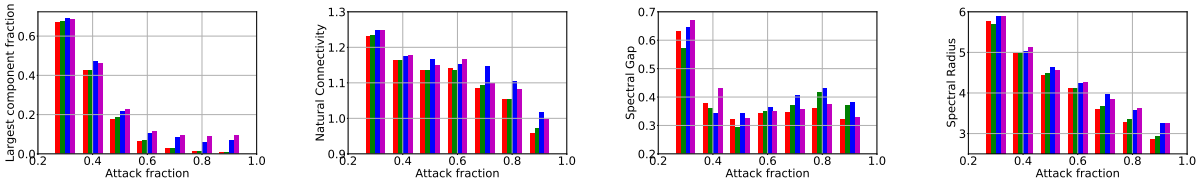
	***R*EAL*W*IRE**	Degree rewiring	Single step rewiring
IAS-PG	131.68	126.10	694.75
IAS-Air	134.78	119.63	694.74

Table I: The average number of rewired links for IAS-PG and IAS-Air.

time slot, we set $r = 0.5$ and $z = 20$ in Equation (1). For the single step rewiring algorithm, we chose to rewire 5% of the total number of links on network A. Table I shows the average number rewired links under ***R*EAL*W*IRE** and degree rewiring, the numbers were calculated when the attack fraction is 0.9. From Table I, the number of rewired links resulted by ***R*EAL*W*IRE** is close to the number of rewired links from the degree rewiring, which is much smaller than the number of links rewired by the single step rewiring algorithm. From Figure 3, we can see that the wall-clock time is almost linear to the size of the network. From Figure 5-7, we can see that in terms of the largest connected component fraction, our algorithm performs the best. For example, in the IAS-Air network, when the attack fraction is 0.3, the largest connected component fraction under ***R*EAL*W*IRE** is 0.39, which is much higher than that under the degree rewiring, which is 0.28. In the BA-BA network, the degree rewiring heuristic is hard to beat when the attack fraction is small. However, as the attack fraction increases, our algorithm starts to outperform others in both network A and network B. In the IAS-Air network, our algorithm outperforms the others for most metrics. In the IAS-PG network, the performance of our algorithm and the degree heuristic is close. Under all of the experiments, ***R*EAL**

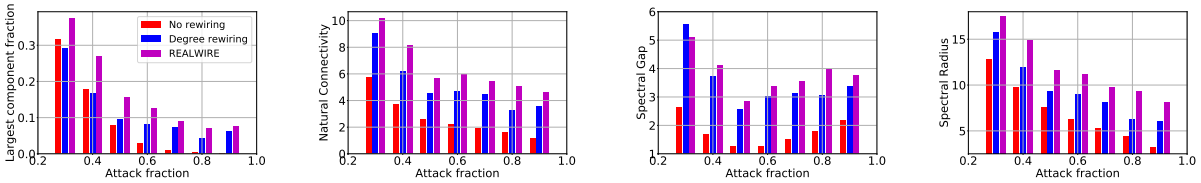


(a) Largest component fraction of network A.
(b) Natural connectivity of network A.
(c) Spectral gap of network A.
(d) Spectral radius of network A.

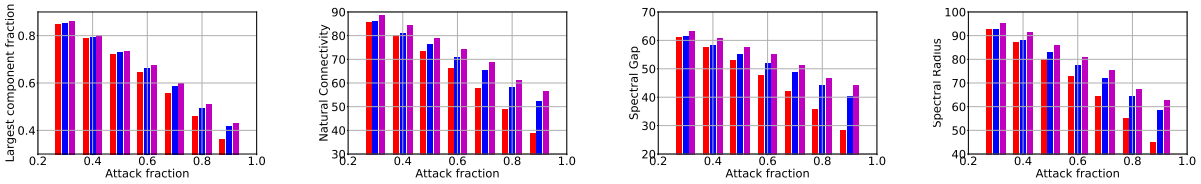


(e) Largest component fraction of network B.
(f) Natural connectivity of network B.
(g) Spectral gap of network B.
(h) Spectral radius of network B.

Figure 7: The IAS-PG network.



(a) Largest component fraction of network A.
(b) Natural connectivity of network A.
(c) Spectral gap of network A.
(d) Spectral radius of network A.



(e) Largest component fraction of network B.
(f) Natural connectivity of network B.
(g) Spectral gap of network B.
(h) Spectral radius of network B.

Figure 8: The IAS-Air network with backup links.

WIRE achieved the best performance in terms of the largest component fraction for most of the attack fractions, which makes sense since the largest connected component size is directly related to the objective of our MDP problem.

In reality, there exist circumstances, in which we cannot add a link between any two nodes in the network. This means we may not be able to rewire the links according to our algorithm. Thus, we considered another scenario, in which we assume each node has some backup links we can activate during the attack. In this experiment, for each node we randomly generated a number of backup links equal to 20% multiplying by its degree. So during the attack, for each link we cut, we can activate a backup link of one of its endpoints. The other parameters are the same as the rewiring case. From Figure 8, we can see our algorithm still outperforms the others.

V. Conclusion

In this paper, we studied the problem of improving robustness of interdependent networks against the localized attack. We proposed a novel algorithm, named ***REALWIRE***, to improve the robustness of the interdependent networks and to limit the impact of the attack by rewiring the links of the networks in real-time. We formulated the problem as an MDP problem, which has been proved to be NP-hard, and then proposed a greedy algorithm. The simulation results showed the performance of ***REALWIRE*** is close to the exact solution of the MDP problem in a small network and ***REALWIRE*** outperforms the others in different networks when the attack fraction is high.

ACKNOWLEDGMENT

This work was supported in part by NSF (IIS-1715385 and IIS-1651203), and HDTRA1-16-0017.

REFERENCES

[1] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.

[2] M. Di Muro, C. La Rocca, H. Stanley, S. Havlin, and L. Braunstein, "Recovery of interdependent networks," *Scientific reports*, vol. 6, 2016.

[3] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, no. 7291, pp. 1025–1028, 2010.

[4] X. Yuan, S. Shao, H. E. Stanley, and S. Havlin, "How breadth of degree distribution influences network robustness: Comparing localized and random attacks," *Phys. Rev. E*, vol. 92, no. 3, p. 032122, 2015.

[5] A. Vespignani, "Complex networks: The fragility of interdependency," *Nature*, vol. 464, no. 7291, pp. 984–985, 2010.

[6] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin, "Networks formed from interdependent networks," *Nature physics*, vol. 8, no. 1, pp. 40–48, 2012.

[7] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[8] R. Albert, H. Jeong, and A.-L. Barabasi, "Attack and error tolerance of complex networks," *Nature*, vol. 406, pp. 378–382, 2000.

[9] D. S. Callaway, M. E. J. Newmann, S. H. Strogatz, and D. J.Watts, "Network robustness and fragility: Percolation on random graphs," *Phys. Rev. Lett.*, vol. 85, pp. 5468–5471, 2000.

[10] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, pp. 130–131, Sept 1999.

[11] M. Newman, *Networks: An Introduction*. Oxford University Press, Inc., 2010.

[12] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," *Proc. the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.

[13] A. Zeng and W. Liu, "Enhancing network robustness against malicious attacks," *Phys. Rev. E*, vol. 85, no. 6, p. 066130, 2012.

[14] S. Shao, X. Huang, H. E. Stanley, and S. Havlin, "Percolation of localized attack on complex networks," *New Journal of Phys.*, vol. 17, no. 2, p. 023049, 2015.

[15] H. Chan, L. Akoglu, and H. Tong, "Make it or break it: Manipulating robustness in large networks," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 325–333.

[16] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, "Gelling, and melting, large graphs by edge manipulation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 245–254.

[17] Y. Zhang, A. Adiga, S. Saha, A. Vullikanti, and B. A. Prakash, "Near-optimal algorithms for controlling propagation at group scale on networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3339–3352, 2016.

[18] H. Chan, S. Han, and L. Akoglu, "Where graph topology matters: the robust subgraph problem," in *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 2015, pp. 10–18.

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

[20] C. Chen, J. He, N. Bliss, and H. Tong, "Towards optimal connectivity on multi-layered networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 2332–2346, 2017.

[21] A. Bakshi, A. Velayutham, S. Srivastava, K. Agrawal, R. Nayak, S. Soonee, and B. Singh, "Report of the enquiry committee on grid disturbance in northern region on 30th july 2012 and in northern, eastern & north-eastern region on 31st july 2012," *New Delhi, India*, 2012.

[22] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[23] Z. Chen, H. Tong, and L. Ying, "Realtime robustification of interdependent networks under cascading attacks," Arizona State University, Tech. Rep., 2018, available at http://www.public.asu.edu/~zchen113/Publications/TechnicalReport.pdf.

[24] W. Jun, M. Barahona, T. Yue-Jin, and D. Hong-Zhong, "Natural connectivity of complex networks," *Chinese Phys. Lett.*, vol. 27, no. 7, p. 078902, 2010.

[25] F. D. Malliaros, V. Megalooikonomou, and C. Faloutsos, "Fast robustness estimation in large social graphs: Communities and anomaly detection," in *Proc. of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012, pp. 942–953.

[26] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proc. of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 177–187.

[27] H. Chan and L. Akoglu, "Optimizing network robustness by edge rewiring: a general framework," *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1395–1425, 2016.