Network under Joint Node and Link Attacks: Vulnerability Assessment Methods and Analysis

Thang N. Dinh, *Member, IEEE* and My T. Thai, *Member, IEEE*,

***Abstract*—Critical infrastructures such as communication networks, electrical grids, and transportation systems are highly vulnerable to natural disasters and malicious attacks. Even failures of few nodes or links may have a profound impact on large parts of the system. Traditionally, network vulnerability assessment methods separate the studies of node vulnerability and link vulnerability, and thus ignore joint node and link attack schemes that may cause grave damage to the network.**

To this end, we introduce a new assessment method, called β-disruptor, that unifies both link and node vulnerability assessment. The new assessment method is formulated as an optimization problem in which we aim to identify a minimum cost *set of mixed links and nodes* that removal would severely disrupt the network connectivity. We prove the NP-completeness of the problem and propose an $O(\sqrt{\log n})$ bicriteria approximation algorithm for the β-disruptor problem. This new theoretical guarantee improves the best approximation results for both link and node vulnerability assessment in literature. We further enhance the proposed algorithm by embedding it into a special combination of simulated annealing and variable neighborhood search method. The results of our extensive simulation-based experiments on synthetic and real networks show the feasibility and efficiency of our proposed vulnerability assessment methods.

***Index Terms*—Approximation algorithm; Joint node and link attacks; Vulnerability assessment;**

I. Introduction

Disruptive events, ranging from natural disasters to malicious attacks, can drastically compromise the network's ability to meet its quality-of-service(QoS) requirements, if not cause widespread service outages and potentially total network breakdown [1], [2], [3], [4]. Moreover, there is a significant concern over critical infrastructures in electrical power grids and highway systems as targets for terrorist attacks [5]. To mitigate the risk and develop proactive responses, it is essential to assess network vulnerability to identify the most destructive attack scenarios.

Although there has been a significant amount of work on assessing network vulnerability, most previous works focus mainly on using centrality measurements e.g. degree, betweeness, and closeness centralities [6], [7] to identify critical links or nodes. Unfortunately, these approaches only determine the relative importance of *a small number* of nodes or links and cannot reveal the enormous damage potential caused under *simultaneous* attacks. Other set of works studies links and nodes removal problems that optimize several global graph measures, such as clustering coefficient, network diameter, etc. However, these measures do not cast well for particular kinds of network vulnerability, when the network connectivity is of high priority. To this end, *pairwise connectivity*, the number of node pairs that remain connected, has been recently used as an effective measure to account for the effect of the attacks [2], [8], [9], [10], [11].

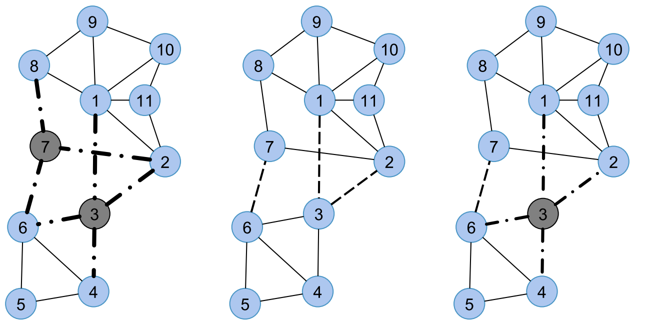


Fig. 1: In **a.**, removing nodes 3 and 7 effectively breaks the network into two parts, disrupting 67% connectivity. This causes more damage than removing the highest degree nodes 1 and 2 which only disrupts 35% connectivity. Figures **a.**, **b.**, and **c.** also show minimum cost solutions to reduce 50% of the connectivity assuming links have cost 2 and nodes have cost 3. The minimum cost is 6 if we remove only nodes (**a.**) or only links (**b.**), and is 5 if we remove both links and nodes (**c.**). Thus, it is insufficient to study node and link attacks separately.

The advantage of the pairwise connectivity metric over the node centrality measures is illustrated in Fig. 1a. Assume two nodes are to be removed from our simple example. If the two nodes are selected according to their degree centrality, nodes 1 and 2 will be removed and the residual network remains connected. However, if we remove nodes to minimize the pairwise connectivity, nodes 3 and 7 are going to be targeted, and the network is effectively broken into two smaller components; and the fraction of connected pairs reduces drastically from 55 to 18, a 67% reduction.

Fig. 1 also illustrates a fundamental shortcoming of existing work: the ability to assess network vulnerability under *joint node and link attacks*. The three sub-figures show the minimum cost attack strategies to reduce $\beta = 50\%$ pairwise connectivity, assuming each link has cost 2 and each node has cost 3. While the minimum costs for both node-attack (Fig. 1a) and link-attack (Fig. 1b) are 6, the minimum cost

This work is partially supported by the DTRA YIP grant number HDTRA1-09-1-0061 and the NSF CAREER Award number 0953284.

T. N. Dinh, M. T. Thai are with the Dept. of Comp. & Info. Sci. & Eng., University of Florida, Gainesville, FL, 32611.
E-mail: {tdinh, mythai}@cise.ufl.edu.
T. N. Dinh is currently with the Dept. of Comp. Sci., Virginia Commonwealth University, Richmond, VA, 23284.
E-mail: tndinh@vcu.edu.

for node-link attacks (node 3 and link $(6,7)$) (Fig. 1c) is only 5. Thus, it is insufficient to assess link vulnerability and node vulnerability separately when both links and nodes in the network can be targeted. To make matters worse, assume node 3 and link $(6,7)$ have the same cost $\epsilon > 0$, the minimum costs for node, link, and node-link attacks will be $3+\epsilon, 4+\epsilon$, and 2ϵ, respectively. As the ratios $(3+\epsilon)/(2\epsilon)$ and $(4+\epsilon)/(2\epsilon)$ go unbounded, the existing methods can seriously misjudge the network vulnerability.

To address the shortcoming, we study the effect of *joint node and link attacks* in term of connectivity. We introduce a new problem, called β-disruptor, that finds a minimum cost set of *nodes and links* whose removal degrades the pairwise connectivity to a great extent (a fraction β). The β-disruptor problem aims to provide a more comprehensive assessment on network vulnerability. It generalizes both the β-vertex disruptor and the β-edge disruptor problems proposed in our previous work [9]. To our best knowledge, this is the first work to address the effect of simultaneous attacks on both links and nodes on network pairwise connectivity.

Our contributions are summarized as follows

- Providing an underlying framework toward assessing vulnerability under *joint node and link attacks* and formulating it as an optimization problem β-disruptor. Other performance measures such as the maximum flow between a given source-destination pair [11], [12], the average maximum flow between pairs of nodes [11], etc. can also be used in place of pairwise connectivity to define new problems.
- Our major result is an $O\left(\sqrt{\log n}\right)$ bicriteria approximation algorithm for both *undirected and directed* networks. The algorithm finds a β-disruptor with the cost at most $O\left(\sqrt{\log n}\right)$ times that of an optimal β'-disruptor, with β' slightly less than β.
- We propose an efficient meta-heuristic which combines simulated annealing, variable neighborhood search, and spectral clustering. The efficacy and scalability of our proposed algorithms is shown through extensive experiments on both synthetic and real-world datasets.

Related work. Many existing works on network vulnerability assessment mainly focus on the local centrality measurements to differentiate between critical links and nodes and the others, see [13], [9]. Other global graph measures have also been proposed to assess network vulnerability. These measures are mainly functions of graph properties, such as the diameter, global clustering coefficient, etc. [1], [2].

Matisziw and Murray [13] first proposed the pairwise connectivity as an effective measurement and use mathematical programming to solve for exact solutions. Arulselvan et al. later define the Critical Node/Edge problems, which the main objective is to identify top k nodes/links that removal minimize the pairwise connectivity in the residual network, and provide NP-completeness proofs and integer programming formulations. However, the run-time for exact solutions scale exponentially with the network size. Di Summa et al. [14] proved that the critical node detection problem (CNP) is also NP-complete on trees for the total weighted pairwise connectivity metric. Shen et al. [15] proved that the CNP is polynomially solvable in trees and series-parallel graphs for the cases when the nodes have uniform costs and and the objective is either minimizing the size of the largest component or maximizing the number of residual components.

We first proposed the assessing vulnerability methods in form of optimization problems β-edge/vertex disruptor in [9], [10]. The paper presents NP-hardness of β-edge/vertex disruptor problems, an $O\left(\log^{1.5} n\right)$ bicriteria approximation algorithm for β-edge disruptor, and an $O\left(\log n \log\log n\right)$ bicriteria approximation algorithm for β-vertex disruptor. These works, however, consider node failures and link failures separately.

Several works consider multiple attacks that happen at both links and nodes at the same time [16], [17], [18] with other measurements of network connectivity. The most often used measures are two-terminal-reliability (whether or not two specific nodes s and t are connected) or all-terminal-reliability (whether the network is connected). Those measures can only capture whether or not the network is disconnected but cannot reveal the level of disconnectivity/fragmentation as in the case of pairwise connectivity.

Organization. We briefly present terminologies and problem definitions in Section II. Then we propose the $O(\sqrt{\log n})$ bicriteria approximation algorithm for β-disruptor in Section III. Section IV presents the efficient heuristic to find β-disruptor. We obtain numerical results in Section V. The conclusion is presented in Section VI.

II. Preliminaries

A. Model and Definitions

We abstract our general network model as a graph $G = (V, E)$, where V refers to a set of nodes and E refers to a set of links. Each vertex $u \in V$ is associated with a cost $c(u) \geq 0$ and each edge $(u,v) \in E$ has a cost $c(u,v) \geq 0$. For convenience, we also denote the number of nodes and links by n and m, respectively.

In an undirected graph, a vertex pair $(u,v) \in V \times V$ is connected iff there exists a path between u and v. In a directed graph, a vertex pair (u,v) is said to be connected if there exist paths between u and v in *both directions*. We denote the pairwise connectivity of a graph G by $\mathcal{P}(G)$. Apparently, the pairwise connectivity is maximized at $\binom{n}{2}$ when G is a (strongly) connected graph. For convenience, we use the word component to refer to connected component in undirected graphs and *strongly connected component* (SCC) in directed graphs whenever the context is clear.

β-disruptor. Given $0 \leq \beta \leq 1$, a β-disruptor D_β is a pair of subsets

$$D_\beta = (V_\beta \subseteq V, E_\beta \subseteq E)$$

that removal from G will make the pairwise connectivity in the residual graph $G' = (V \setminus V_\beta, E \setminus (E_\beta \cup V_\beta \times V_\beta))$ to be at most $\beta\binom{n}{2}$. The *β-disruptor problem* asks for a β-disruptor with the minimum total cost

$$c(D_\beta) = \sum_{u \in V_\beta} c(u) + \sum_{e \in E_\beta} c(e).$$

There are two special types of β-disruptor: if $V_\beta = \emptyset$, then D_β is a *β-edge disruptor*; and if $E_\beta = \emptyset$, then D_β is a *β-vertex disruptor*. The uniform-cost versions of *β-edge disruptor problem* and the β-vertex disruptor problem are previously studied in [19].

By definition, β-vertex disruptor is a special case of β-disruptor when all edges have infinity costs and β-edge disruptor is a special case of β-disruptor when all vertices have infinity costs. Since both vertex and edge disruptor are NP-hard, the β-disruptor problem is also NP-hard for $0 < \beta < 1$.

B. Nodes and Edges with Excess Costs

We give simple criteria to identify quickly "safe" nodes and edges that are not the vulnerabilities of the network due to their excess costs. This helps narrowing down the search space for vulnerabilities and suggests the protection priorities and resource allocation to other network elements.

Since removing either u or v causes more disruption than removing the edge (u, v), edges with excess costs can be identified based on the following lemma.

Lemma 1: An edge $(u, v) \in E$ with $c(u, v) > \min\{c(u), c(v)\}$ will not appear in any optimal β-disruptor for any $\beta \geq 0$.
The lemma also reflects the fact that nodes' costs are often higher than the costs of the incident links.

Similarly, removing a node has the same effect as removing all the incident edges; and a node u is an excess cost node if $c(u)$ is higher than the total costs of the incident edges.

Lemma 2: A vertex u with $c(u) > \sum_{(u,v)\in E} c(u, v)$ will not appear in any optimal β-disruptor for any $\beta \geq 0$.

For non-excess nodes and edges, Lemmas 1 and 2 provide the relative caps for how much extra resource we should allocate to those network elements.

III. Bicriteria Approximation Algorithm for Joint Link & Node Attacks

In this subsection, we present an $O(\sqrt{\log n})$ bicriteria approximation algorithm for the β-disruptor problem. Since both β-vertex disruptor and β-edge disruptor are special cases of β-disruptor, our new algorithm improve the best results for β-vertex disruptor, the $O(\log n \log\log n)$ bicriteria approximation algorithm, and β-edge disruptor, the $O(\log^{3/2} n)$ bicriteria approximation algorithm [9].

TABLE I: Table of Symbols

Notation	Meaning
n	Number of vertices/nodes
m	Number of edges/links
G'	The auxiliary graph of G
$\mathcal{P}(G)$	Pairwise connectivity of G
$\alpha(G)$	Minimum ratio cut in G
$\alpha_{\min}(G)$	$\min\{\alpha(C) \mid C$ is a SCC of $G\}$
$G[E \setminus E_\beta]$	Residual graph after removing edges in E_β
$G[-D_\beta]$	Residual graph after removing D_β
$\mathrm{OPT}_\beta(G)$	Cost of optimal β-disruptor in G
$\mathrm{OPT}^E_\beta(G)$	Cost of optimal β-edge disruptor in G

A. Algorithm Description

We will refer to the input network as the *original network*. We first reduce the β-disruptor problem in the original network to an instance of the β-edge disruptor problem in an *auxiliary directed graph*. The reduction maps each undirected edge to two *alternating directed edges* and each node to a *surrogate edge*. More importantly, we show that the reduction 'preserves' relative performance guarantees. We then apply a recursive cut procedure to find a near-optimal set of both alternating edges and surrogate edges that correspond to a β-disruptor in the original network.

Our algorithm JLNA(G) to find a β-disruptor in directed graph G is summarized in Algorithm 1. For clarity, we provide a list of symbols in Table I. In the first phase, the algorithm constructs an auxiliary graph G' by splitting each vertex $v \in V$ into two new vertices v^+ and v^-. Formally, the set of vertices and edges in G' are defined as

$$V' = \{\ v^-, v^+ \mid v \in V\}$$
$$E' = \{(v^-, v^+) \mid v \in V\} \cup\ \{(u^+, v^-) \mid (u, v) \in E\}$$

In addition, we assign costs $c'(.)$ for edges in G' as follows: $c'(v^-, v^+) = c(v)$ for the surrogate edge (v^-, v^+) and $c'(u^+, v^-) = c(v^+, u^-) = c(u, v)$ for alternating edges (u^+, v^-) and (v^+, u^-). In the case, E is a mix of both undirected and directed edges, we can also convert each directed edge $(p, q) \in E$ into an alternating edge $(p^+, q^-) \in E'$ with a cost $c'(p^+, q^-) = c(p, q)$.

In the second phase, the recursive cut procedure, shown in lines 4 to 11, construct a $\tilde{\beta}$-edge disruptor of G', denoted by $E_{\tilde{\beta}}$. Here for a given $\beta' < \beta$, $\tilde{\beta} = \frac{1}{2}(\beta + \beta')$. We show later in the proof of Theorem 1 that when $\mathcal{P}(G') > \tilde{\beta}\binom{n}{2}$ (line 6), we indeed obtain a β-disruptor in G. The $\tilde{\beta}$-edge disruptor is found by iteratively applying a subroutine SPARSE_CUT on the strongly connected components in G'. The subroutine SPARSE_CUT cut the components into smaller ones and the edges in a subset of the cuts are added to $E_{\tilde{\beta}}$. The process continues until the pairwise connectivity in the graph reduces to $\beta\binom{n}{2}$ or smaller. By the end of the second phase, $E_{\tilde{\beta}}$ is mapped back to edges and nodes in G to give a β-disruptor.

Algorithm 1 JLNA(G)

```
1.  Construct the auxiliary graph $G' = (V', E')$
2.  $\tilde{\beta} \leftarrow \frac{1}{2}(\beta + \beta')$
3.  $E_{\tilde{\beta}} \leftarrow \emptyset$
4.  for each SCC $C$ in $G'$
5.      $(C_E, C_\alpha) \leftarrow$ SPARSE_CUT$(C)$
6.  while $\mathcal{P}(G') > \tilde{\beta}\binom{n}{2}$
7.      Find a SCC $C^*$ of $G'$ with minimum cut ratio $C^*_\alpha$
8.      $E_{\tilde{\beta}} \leftarrow E_{\tilde{\beta}} \cup C^*_E$
9.      Remove edges in $C^*_E$ from $G'$
10.     for each new component $C'$ in $G'$
11.         $(C'_E, C'_\alpha) \leftarrow$ SPARSE_CUT$(C')$
12. $V_\beta \leftarrow \{v \mid (v^-, v^+) \in E_{\tilde{\beta}}\}$
13. $E_\beta \leftarrow \{(u, v) \mid (u^-, v^+) \in E_{\tilde{\beta}}\}$
14. return $D_\beta = (V_\beta, E_\beta)$
```

As shown in lines 4 and 5, the subroutine SPARSE_CUT is applied to each strongly connected component C to find a *minimum ratio cut* $\langle S', \overline{S'}\rangle$ in C. The cut ratio for a cut is defined as follows.

Definition 1: Let $G' = (V', E')$ be a directed graph. The ratio of a cut $\langle S', \overline{S'}\rangle$ is $\alpha(S') = \frac{c_{\text{out}}(S')}{|S'||\overline{S'}|}$, where $c_{\text{out}}(S')$ is the total cost of edges coming out from S'. In addition, a cut with the minimum cut ratio is called a *minimum ratio cut* and denoted by

$$\alpha(G') = \min_{S' \subsetneq V'} \alpha(S')$$

The output of SPARSE_CUT is a pair (C_E, C_α), where $C_E = \langle S', \bar{S'}\rangle$ and $C_\alpha = \alpha(S')$. To obtain the best theoretical performance guarantee, we use in place of SPARSE_CUT the sparsest cut algorithm in [20]. For completeness, we summarize the algorithm in the appendix.

Within the main loop of JLNA (lines 6 to 11) we select in each round a SCC C^* in G that has the smallest cut ratio. Let C^*_E and C^*_α be the cut set and the cut ratio of the cut found by SPARSE_CUT in C^*. We add C^*_E to $E_{\tilde{\beta}}$ and remove C^*_E from G. Removing $E_{\tilde{\beta}}$ breaks C^* into two or more strongly connected components. We again apply SPARSE_CUT on those components to find the minimum ratio cuts. The reason not to apply SPARSE_CUT on G' is that G' is likely disconnected after removing edges. Thus SPARSE_CUT will likely return SCCs in G' with minimum cut ratio zero.

The main loop terminates when the pairwise connectivity in G is no more than $\beta\binom{n}{2}$. Then we construct the final solution by mapping each surrogate edge $(v^-, v^+) \in E_{\tilde{\beta}}$ to the node v in G, and each alternating edge $(u^-, v^+) \in E_{\tilde{\beta}}$ to the edge (u, v) in G.

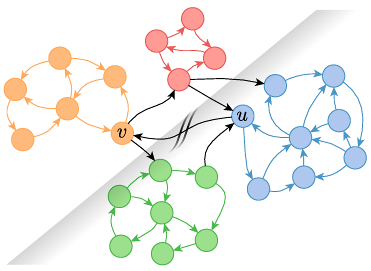


Fig. 2: High interdependence of networks' elements. Removing the marked link (u, v) breaks the (strongly) connected network into four components. Notice that the red and green components are separated from the others, even when none of the incoming links to or outgoing links from those components are removed.

B. Analysis of Approximation Ratio

We show that the JLNA algorithm is an $O(\sqrt{\log n})$ bicriteria approximation algorithm for the β-disruptor problem. We first show in Lemma 3 the connection between the cost of an optimal β-disruptor and the minimum cut ratio. This is the key lemma that show the relation between (bipartite) sparsest cuts to general (multi-way) cuts.

Cuts in directed networks have different characteristics in comparison to their counterpart in undirected networks. First, the cut ratios of $\langle S, \bar{S}\rangle$ and $\langle \bar{S}, S\rangle$ are different in general. In addition, different cuts may associate with the same set of links. For example, the cuts defined by $S = \{\text{blue nodes}\}$, and $S = \{\text{blue and green nodes}\}$ associates to the same set of links $\{(u, v)\}$. To treat these differences, we use a randomized argument in the following lemma. Second, components in directed networks are highly interdependent. As illustrated in Fig. 2, the failure of link (u, v) effectively breaks the network into four disconnected components. Red and green components loose the communication to other parts of the network, even none of their incoming and outgoing edges, colored in black, fail. In contrast, the only way to separate a component from the rest in undirected networks is to remove all links incident to the component.

To link the average cost to disrupt connected pairs in an optimal β-disruptor to the minimum cut ratio, we consider the random partitions of SCCs in the residual graph, subject to their topological order.

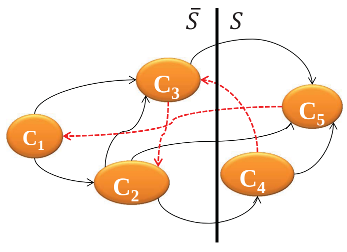


Fig. 3: Construct a (bipartite) cut from SCCs (represented by a node) in the residual graph $G[E \setminus M]$. The edges in M (removed edges) are colored in red. The cut $\langle S, \bar{S}\rangle$ consists of a subset of M: edges from C_4 to C_3 and from C_5 to C_2.

Lemma 3: Given a directed graph $G = (V, E)$ and a subset of edges $M \subseteq E$, if $\omega_M = \mathcal{P}(G) - \mathcal{P}(G[E \setminus M]) > 0$, then

$$\frac{c(M)}{\omega_M} \geq \frac{1}{3}\ \alpha_{\min}(G),$$

where $\alpha_{\min}(G) = \min\{\alpha(C) \mid C \text{ is a SCC of } G\}$.

Proof: Firstly, we consider the case G is strongly connected. Then $\alpha_{\min}(G) = \alpha(G)$ and $\mathcal{P}(G) = \binom{n}{2}$.

Observe that if we contract each SCC into a single node, we obtain the graph of SCCs which is a directed acyclic graph. Thus, there is a topological order for SCCs and we follow the convention that vertices with no incoming edges will have the smallest orders. Thus, w.l.o.g, we assume that the removed edges always come from SCCs with higher orders to SCCs with lower orders. Let $C_1, C_2, \ldots, C_k$ be SCCs in $G[E \setminus M]$ in that topological order and let $C_i(V)$ denote the set of vertices in component C_i. We have $\omega_M = \sum_{i<j} |C_i(V)||C_j(V)|$.

We can always find a cut that contains only edges in M and separates at least $\frac{1}{3}\omega_M$ pairs. Construct a cut $\langle S, \bar{S}\rangle$ of G as follows. We select into $\bar{S}$ vertices in $C_1, C_2, \ldots, C_t$ where $t \geq 1$ satisfies $|\bigcup_{i=1}^t C_i(V)| \leq \frac{n}{2}$ and $|\bigcup_{i=1}^{t+1} C_i(V)| > \frac{n}{2}$. An example for such a cut $\langle S, \bar{S}\rangle$ is given in Fig. 3.

+ If $|S| = |\bigcup_{i=1}^t C_i(V)| \geq \frac{1}{4}n$, the constructed cut will separate at least $\frac{1}{4}n \times \frac{3}{4}n > \frac{1}{3}\binom{n}{2} \geq \frac{1}{3}\omega_M$.

+ If $|S| = |\bigcup_{i=1}^t C_i(V)| < \frac{1}{4}n$, we consider two subcases. a) If $|\bigcup_{i=t+2}^n C_i| \geq \frac{1}{4}n$. Replace $\langle S, \bar{S}\rangle$ with $\langle S = \bigcup_{i=t+2}^n C_i, \bar{S} = V \setminus S\rangle$ that separates $\geq \frac{1}{3}\omega_M$ pairs.
b) Otherwise we must have $a = |C_{t+1}| \geq \frac{1}{2}n$. Replace $\langle S, \bar{S}\rangle$ with $\langle S = C_{t+1}, \bar{S} = V \setminus C_{t+1}\rangle$ that separates $a(n-a)$ pairs. Since $a \geq \frac{1}{2}n$, we have

$$\frac{1}{3}\omega_M = \frac{1}{3}\sum_{i<j}|C_i(V)||C_j(V)| \leq \frac{1}{3}\left(a(n-a) + \binom{n-a}{2}\right)$$
$$= (n-a)\frac{1}{3}\left(\frac{1}{2}n + \frac{1}{2}a - \frac{1}{2}\right) < a(n-a).$$

Thus we can always find a cut $\langle S, \bar{S}\rangle$ that separates at least

$\frac{1}{3}\omega_M$ pairs and the cost of that cut is at most $c(M)$ since $\langle S, \bar{S}\rangle \subseteq M$. It follows that $\frac{c(M)}{\omega_M} \geq \frac{1}{3}\ \alpha_{\min}(G)$.

Secondly, we consider the case that G is not strongly connected. Let $T_1, T_2, \ldots, T_l$ be SCCs of G, and let $M^{(j)}$ be the intersection of M and the edges in T_j, and T'_j be the subgraphs obtained from T_j after removing $M^{(j)}$. Apply the above result for the case the graph is connected on each connected component, we have

$$\begin{aligned} c(M) &= \sum_j c(M^{(j)}) \geq 3\sum_j \alpha(T_j)\left(\mathcal{P}(T_j) - \mathcal{P}(T'_j)\right) \\ &\geq 3\alpha_{\min}(G)\sum_j \left(\mathcal{P}(T_j) - \mathcal{P}(T'_j)\right) \\ &= 3\alpha_{\min}(G)\left(\mathcal{P}(G) - \mathcal{P}(G[E\setminus M])\right) \end{aligned}$$

Thus, the lemma holds for every directed graph G. ■

The quality and performance JLNA depend on the selection of SPARSE_CUT. For example, an exact algorithm to find minimum ratio cut will lead to a constant factor bicriteria approximation algorithm for β-disruptor. Unfortunately, finding the min ratio cut is an NP-hard problem [21]. Thus we have to rely on approximation algorithms to find good ratio cut in the graph.

Theorem 1: For any fixed $0 \leq \beta' < \beta$, algorithm JLNA finds a β-disruptor of cost at most $O(\sqrt{\log n})\mathrm{OPT}_{\beta'}$, where $\mathrm{OPT}_{\beta'}$ is the cost of a minimum β'-disruptor.

Proof: The proof consists of two steps. In the first step, we prove that $D_\beta = (V_\beta, E_\beta)$ is a β-disruptor of G. In the second step, we prove that the cost of D_β is at most $O(\sqrt{\log n})$ times the cost of a minimum β'-disruptor, denoted by $\mathrm{OPT}_{\beta'}$.

In order to prove that D_β is a β-disruptor of G, we show that the pairwise connectivity in G after removing edges in $G[-D_\beta] = (V\setminus V_\beta, E\setminus(E_\beta \cup V_\beta \times V_\beta))$ is at most $\beta\binom{n}{2}$. First, observe that vertices v^- and v^+ are either in the same SCC or they both are isolated. Here, we say a vertex is isolated if it belongs to a SCC of size one. Assume that $G'[E'\setminus E_{\tilde{\beta}}]$ can be decomposed into SCCs $C'_1, C'_2, \ldots, C'_l$ and $2t$ isolated vertices $w_1^-, w_1^+, \ldots, w_t^-, w_t^+$. Based on the construction of G', we can verify that there are l corresponding SCCs $C_1, C_2, \ldots, C_l$ and t isolated vertices $w_1, w_2, \ldots, w_t$ in $G[-D_\beta]$. Moreover, $|C'_i| = 2|C_i|$ for $i = 1..l$.
Therefore, we have

$$\begin{aligned} \tilde{\beta}\tbinom{2n}{2} &\geq \mathcal{P}(G'[E'\setminus E_{\tilde{\beta}}]) = \sum_i \tbinom{|C'_i|}{2} \\ &= 4\sum_i \tbinom{|C_i|}{2} + \sum_i |C_i| = 4\mathcal{P}(G[-D_\beta]) + (n-t) \end{aligned}$$

Since $\tilde{\beta} < \beta$, we have

$$\mathcal{P}(G[-D_\beta]) \leq \frac{1}{4}\left(\tilde{\beta}\tbinom{2n}{2} - (n-t)\right) \leq \beta\tbinom{n}{2}$$

Thus, we have completed the first step. We prove the second step as follows.

Let $D^*_{\beta'} = (V_{\beta'}, E_{\beta'})$ be a minimum β'-disruptor i.e. $c(D^*_{\beta'}) = \mathrm{OPT}_{\beta'}$. Define

$$E'_{\beta'} = \{(v^-, v^+) \mid v \in V_{\beta'}\} \cup \{(u^+, v^-) \mid (u,v) \in E_{\beta'}\}.$$

By mapping SCCs of $G[-D^*_{\beta'}]$ to those of $G'[E'\setminus E'_{\beta'}]$ as in the first step, we can show that $E'_{\beta'}$ is a β'-edge disruptor of G'. Thus,

$$\mathrm{OPT}_{\beta'}(G) \leq \mathrm{OPT}^E_{\beta'}(G').$$

Since $\beta' < \tilde{\beta}$, by Lemma 3 if removing a set of edges $M_\omega \subseteq E$ disrupts ω pairs of vertices, then $\frac{c(M_\omega)}{\omega} \geq 1/3\alpha_{\min}(G)$. At any round in the while loop of JLNA, since a set of edges $E^*_{\beta'}$ in a minimum β'-edge disruptor, for some $0 < \beta' < \beta$, can disrupt at least $(\beta - \beta')\binom{n}{2}$ additional pairs in G, we have

$$\mathrm{OPT}^E_{\beta'} / \left((\beta-\beta')\tbinom{n}{2}\right) \geq 1/3\alpha_{\min}(G). \tag{1}$$

In addition, our cut procedure is an $O(\sqrt{\log n})$ factor approximation algorithm for the min cut ratio problem, the average cost to disrupt a pair by removing C^*_E is upper bounded by $O(\sqrt{\log n})\alpha_{\min}(G)$. By (1), the average cost to disrupt pairs in the graph at any step is at most $O(\sqrt{\log n})(\mathrm{OPT}^E_{\beta'}) / \left((\beta-\beta')\binom{n}{2}\right)$. Therefore, even when E_β disrupt all $\binom{n}{2}$ pairs in G, the total cost is no more than

$$O(\sqrt{\log n}) \times \frac{\mathrm{OPT}^E_{\beta'}}{(\beta-\beta')\binom{n}{2}} \times \tbinom{n}{2} \leq \frac{O(\sqrt{\log n})}{(\beta-\beta')} \times \mathrm{OPT}^E_{\beta'}.$$

Thus we have

$$c(E_{\tilde{\beta}}) \leq O(\sqrt{\log 2n})\mathrm{OPT}^E_{\beta'}(G') \leq O(\sqrt{\log n})\mathrm{OPT}_{\beta'}(G).$$

That yields the proof. ■

Remarks. While the JLNA algorithm can provide a performance guarantee on the produced solution, it can be further improved. First, JNLA often disrupts more than a fraction β of the connected pairs and result in a higher cost solution. Second, the SPARSE_CUT procedure in JLNA has a high time complexity of $O(n^{9.5})$ (it involves solving a large size semidefinite programming as shown in the appendix). We address these issues of JLNA to provide an improved algorithm in the next section.

IV. Hybrid Meta-heuristic

We propose in Algorithm 2 a hybrid meta-heuristic (HMH) that improves over JLNA w.r.t. the following two aspects:

1) HMH avoids disrupting more pairs than necessary by controlling the difference between the connectivity in the residual graph and the target connectivity. The pairwise connectivity is kept to be within $(\beta \pm \tau)\binom{n}{2}$ where τ is a positive parameter and is iteratively reduced by half. HMH returns the minimum cost β-disruptor encountered during the search as the solution.
2) HMH improves the running time by replacing the sparse cut algorithm in [20] with an efficient spectral partitioning method in subsection IV-B. Further, the solution is refined in each step with lightweight local search methods in subsection IV-C.

A. Controlling the pairwise connectivity.

We use a parameter τ, similar to the heating condition in Simulated Annealing [22], to control how far the pairwise connectivity in the graph can diverge from the target connectivity $\beta\binom{n}{2}$. After each round τ is reduced by half until it is negligibly small.

First, HMH recursively separates the SCCs in the graph with a spectral partitioning method (described in the next part)

Algorithm 2 HMH(G)

```
Construct the auxiliary graph $G' = (V', E')$
$E_\beta \leftarrow \emptyset, \tau \leftarrow \min\{\beta, 1-\beta\}$
while $\tau > 1/\binom{n}{2}$
    $\tau \leftarrow \frac{1}{2}\tau$
    repeat
        Partition SCCs of $G'$ using the spectral method
        Add the edges into $E_\beta$
    until $E_\beta$ is a $(\beta-\tau)$-edge disruptor
    for $k = 1$ to 3 /* Phase 1: Condensation */
        repeat
            Consider all type $k$ neighbors $E'_\beta$ that
            $c(E'_\beta) \le c(E_\beta)$ and $\mathcal{P}(G[E \setminus E'_\beta]) \le \beta\binom{n}{2}$
            Find among them $E'_\beta$ with the smallest cut ratio
            $E_\beta \leftarrow E'_\beta$
        until no change in $E_\beta$
    for $k = 1$ to 4 /* Phase 2: Exploration */
        repeat
            Consider all type $k$ neighbors $E'_\beta$ that
            $(\beta-\tau)\binom{n}{2} \le \mathcal{P}(G[E \setminus E'_\beta]) \le (\beta+\tau)\binom{n}{2}$
            Find among them $E'_\beta$ with the smallest cut ratio
            $E_\beta \leftarrow E'_\beta$
        until no change in $E_\beta$
return the best solution so far
```

until the pairwise connectivity is at most $(\beta - \tau)\binom{n}{2}$ (line). If multiple SCCs can be cut at the same time, the algorithm select the SCC with the minimum ratio cut as in the JLNA algorithm. The algorithm follows by two phases: *condensation* and *exploration*, in which we improve the solution in terms of cost and cut ratio with local search methods in subsection IV-C. For simplicity, we use the term neighbor to refer to a candidate solution that can be obtained from the current solution E_β by applying one of the local changes in IV-C.

In the condensation phase, we move from the current solution E_β to a smaller cost neighbor. And among the possible neighbors, we move to the one which results in the *smallest cut ratio*. In the exploration phase, we emphasize on improving the cut ratio to find potential good partition of the network. Moving to neighbors with higher costs is possible during this phase as long as the pairwise connectivity differs at most $\tau\binom{n}{2}$ from the target connectivity level $\beta\binom{n}{2}$.

B. Spectral Partition

The algorithm to find sparse cuts in [20] has a high time complexity $O(n^{9.5})$ as it requires solving a large semidefinite program. We replace that algorithm with a more efficient spectral partitioning method. Spectral algorithms often give high quality solutions and can be implemented efficiently by standard linear algebra packages [23], [24].

Let $A = \{c_{ij}\}$ be the cost matrix of $G = (V, E)$ where $c_{ij} = c(v_i, v_j)$ is the cost of edge (v_i, v_j) and $c_{ij} = 0$ if $(v_i, v_j) \notin E$. The unnormalized graph Laplacian matrix [25] is defined as $L = D - A$, where D is a diagonal matrix with the weighted degrees of vertices on the diagonal.

If G is an undirected graph: For $x \in \mathbb{R}^n$ we have

$$x^T L x = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(x_i - x_j)^2 \ge 0. \tag{2}$$

The matrix L is symmetric and positive semi-definite. L has n non-negative, real-valued eigenvalues $\lambda_1 = 0 \le \lambda_2 \le \ldots \le \lambda_n$. W.l.o.g, we assume that G is connected. Then the second smallest eigenvector of L, λ_2, is known as the algebraic connectivity of the graph and can be used to describe many properties of graphs [25]. We shall use the eigenvector corresponding to λ_2 to derive the bisection of vertices in G.

Recall that SPARSE_CUT aims to find the min ratio cut

$$\min_{S \subsetneq V} \frac{c(S, \bar{S})}{|S||\bar{S}|} \tag{3}$$

Consider a vector $x \in \{0,1\}^n$ represent a set of vertices in S i.e. $x_i = 1$ if $v_i \in S$ and $x_i = 0$ otherwise. We rewrite the min ratio cut problem as

$$\min_{x \in \{0,1\}^n, x \ne 0, \mathbb{1}} \frac{\sum_{(v_i,v_j)\in E} c_{ij}(x_i - x_j)^2}{\sum_i \sum_j (x_i - x_j)^2} \tag{4}$$

Since the problem is NP-hard, we relax the condition $x_i \in \{0,1\}$ to $x_i \in [0,1]$. Substitute x with vector $y = x - \frac{\|x\|_1}{n}$. After some algebra, we obtain an equivalent problem of (4)

$$\min_{y \ne 0, y \perp \mathbb{1}} \frac{1}{n}\frac{y^T L y}{y^T y} \tag{5}$$

By Courant-Fisher theorem [25], the solution of the above minimizing problem is exactly the eigenvector corresponding to the second smallest eigenvalue of λ_2. So we can approximate the optimal solution of the min ratio cut problem with the second eigenvector of L by transforming the real-valued x into a zero-one vector. Specifically, we sort the values of x_i to give a linear ordering of the vertices then determine the splitting index p that yields the best cut ratio.

If G is a directed graph: We perform one of the symmetrization methods such as $(A + A^T)/2$ or AA^T [26] to transform the matrix A into a symmetric matrix before applying the spectral partitioning.

C. Variable Neighborhood Search

For β-edge disruptor problem, multiple neighborhood structure is essential to obtain high quality solutions. They help in both minimizing the cut ratio and the total cut cost. It is essential that HMM allows both "downhill" moves that reduce the total cut cost and "uphill" moves that increase the cost of the solution but reduce the cut ratio.

We consider four different neighborhood structures. From a solution or a partial solution $E_\beta \subset E$, the set of neighbors in each neighborhood structure can be obtained as follows

- Type 1: Merge two connected components in $G[E \setminus E_\beta]$ i.e. remove the edges between them from E_β.
- Type 2: Move a vertex from one component to an adjacent component in $G[E \setminus E_\beta]$.
- Type 3: Swap places of two adjacent vertices (u, v) which belong to two different components.
- Type 4: Partition a component in $G[E \setminus E_\beta]$ with the spectral partitioning method in subsection IV-B.

Time Complexity. Since the algorithm has at most $\log \binom{n}{2} = O(\log n)$ phases, and it spends at most $O(n^3)$ times to improve the solution within each phase, the HMH algorithm has a time complexity $O(n^3 \log n)$. If we assume that the eigenvalues can be found within a constant number of iterations [27], HMM will have a time complexity $O(n^2 \log n)$.

TABLE II: Sizes of studied networks

Network	Nodes	Links
US Backbone network[28]	71	98
Synthesis networks	100	200
CAIDA AS [29]	8, 020	36, 406
Oregon AS [29]	11,174	23, 410

V. Experimental Studies

We illustrate through our experiments the need to assess network vulnerability under joint node and link attacks and the efficiency of our proposed algorithms.

A. Experiment Setups

Datasets. We perform experiments on four synthesis networks of the same size and three real communication networks, namely US Backbone network [28], CAIDA AS [29], and Oregon AS [29]. The network sizes are given in Table II.

The *synthesis networks* are generated with the following network models.

- *Erdos-Reyni*: A random graph of 100 vertices and 200 edges following the Erdos-Reyni model [30].
- *Barabsi-Albert*: A power-law model using preferential attachment mechanism [31].
- *Watts–Strogatz*: A random graph which exhibit small-world phenomenon following model [32] with the dimension of the lattice 2 and the rewiring probability 0.3[32].
- *Forest fire*: A random power-law graph following Forest fire model by Leskovec et al. [29] with the forward and backward burning probabilities 0.3 and 0.9, respectively.

Real-world traces networks include the following three.

- *US Backbone network*: The backbone cabling network of XO company [28].
- *CAIDA AS*: The CAIDA AS Relationships Dataset from Sep. 17, 2007 [29].
- *Oregon AS*: AS peering information inferred from Oregon route-views between March 31 and May 26, 2001 [29]. Only the largest connected component with 11,174 nodes and 23,410 links is considered.

Assigning costs for nodes and edges. Assigning meaningful costs for edges and vertices is a challenging task which usually depends on the availability of the data. For simplicity, we assume that all edges has uniform removal costs $c(e) = 1 \; \forall e \in E$. Note that we can always multiply simultaneously edge and vertex costs with a constant, then all optimal disruptors stay optimal (with the costs multiplied by the same constant). We assign the cost of removing a vertex u to be $c(u) = b + \alpha d(u)$, where b and α are non-negative constants. In other words, attacking a node requires paying a base cost b and an extra cost that is proportional to the degree centrality. Other centrality measurements e.g. PageRank, Betweeness centrality can also be used in place of $d(u)$ to weight the u's importance.

Solving for the second eigenvector. The major time of HMH (Algorithm 2) spends on finding the second smallest eigenvector of the Laplacian matrix. The eigenvectors are found using the *Implicitly Restarted Arnoldi Method*, implemented in ARPACK [27]. We use SuperLU [33] as the linear systems solver.

We use the Shift and Invert spectral transformation to enhance the convergence rate. We select a scalar $\sigma = 0.01$, called the *shift*, and transform the original problem $Lx = \lambda x$ into the shift-and-invert problem $(L - \sigma I)^{-1} x = \mu x$ where $\mu = 1/(\lambda - \sigma)$. Note that setting $\sigma = 0$ will crash ARPACK since L is non-invertible (sum of rows equal zero) [1] . In addition, spectral partitioning is performed on the symmetrized matrix $A' + A'^T$, where A' is the adjacency matrix of the auxiliary graph G', constructed in Algorithm 1.

Enviroment. All algorithms are implemented in C++ and compiled with GCC 4.4 compiler on a 64 bit Linux machine with a Quad-core AMD Opteron 2350 2.0 Ghz processor and 32 GB memory. Only a single core is used during the experiments. The mathematical optimization package to solve linear programming formulation is GUROBI 4.5.

B. Comparison of the three disruptor types

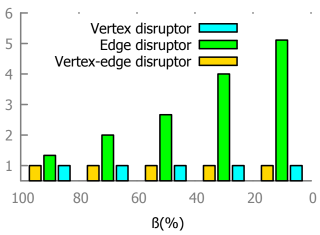
In this section, we experiment with different cost schemes for vertices and edges to highlight the connections among the three different disruptor types: edge, vertex, and general (vertex-edge). We find the optimal solutions for each type of disruptor by solving Mixed Integer Linear Programming (MILP) formulations in the appendix and in [10].

First, the cost of optimal β-disruptor is always less than the costs of both β-edge disruptor and β-vertex disruptor. This suggests that while many networks are vulnerable to only node attacks (e.g. scale-free networks) or only link attacks, all networks exhibit higher level of vulnerability to the joint attacks on both nodes and links. Thus it is essential to assess the network for such grave attack schemes.

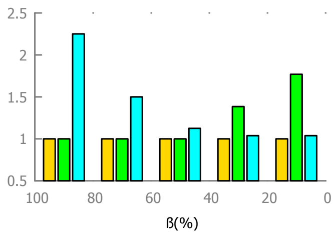
Second, when we apply the cost schemes $c(u) = b + \alpha d(u)$, the cost of minimum β-disruptor can be strictly less than or equal the minimum of those of β-edge disruptor and β-vertex disruptor, depending on the values of b and α. To distinguish between these two cases, we note that if $c(u) < c(u, v)$ for some $(u, v) \in E$, then the edge (u, v) should not be removed (as we can remove u instead). Similarly, a node u with $c(u) > \sum_{(u,v)\in E} c(u, v)$ will not be removed since we can always remove all of its incident edges. Therefore, we obtain the following cases (with further discussion in the appendix).

- $\alpha = 0,\; b \leq 1$: $\mathrm{OPT}_\beta = \mathrm{OPT}^V_\beta \leq \mathrm{OPT}^E_\beta$ i.e. the optimal β-disruptor contains *no edges*.
- $\alpha = 0,\; b > 1$: the optimal solutions contain no u with $d(u) < b$.
- $0 < \alpha < 1$: the optimal β-disruptor contains only vertices of degree at least $\frac{b}{1-\alpha}$.

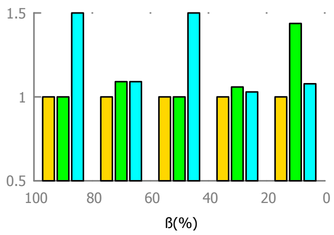
[1]The 'eigs' function to find eigenvalues in MATLAB crashes for this reason.



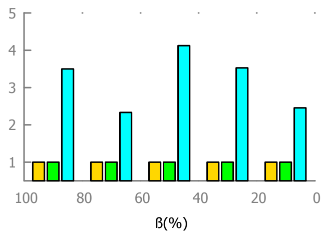
(a) $\alpha = 0$ and $b = 0.25$



(b) $\alpha = 0$ and $b = 2.25$



(c) $\alpha = 0.25$ and $b = 2.25$



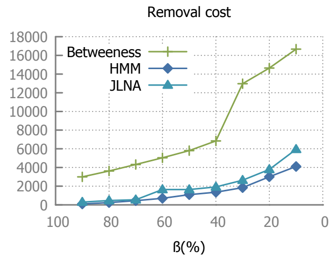
(d) $\alpha = 1.25$ and $b = 1$

Fig. 4: The normalized optimal costs of three different disruptor types on the US Backbone network.

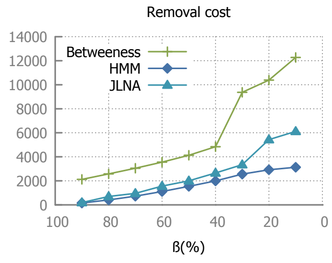
- $1 \leq \alpha$: $\text{OPT}_\beta = \text{OPT}^E_\beta \leq \text{OPT}^V_\beta$ i.e. the optimal β-disruptor contains *no vertices*.

We test four different settings of α and b that correspond to the above four cases on the fiber backbone operated by a major U.S. network provider [28]. The optimal costs of three disruptor types are shown in Fig. 4. In Fig. 7a, the costs of β-disruptor equal exactly the cost of β-vertex disruptor; in Fig. 4d, the costs of β-disruptor equal exactly the cost of β-edge disruptor; and in Figs. 4b and 4c, the costs of β-disruptor are *strictly less than* the minimum of both edge-disruptor and vertex-disruptor. These agree with the above four mentioned cases.

Another observation is that for small β the cost of β-edge disruptor is less than that of β-vertex disruptor, while for large β the vertex-disruptor has substantially smaller cost. This suggests that small scale attacks should target links, while large scale attacks should pay more attention to nodes to reduce the attack cost. Nevertheless, a combination of both node and link attacks would result in a more cost-effective strategy to break the network.



(a) Oregon AS



(b) CAIDA AS

Fig. 6: Disruptor costs in AS networks

C. Performance Analysis

We analyze the solution quality and running time of the proposed algorithms. Specifically, we compare the following algorithms:

- *HMH*, the hybrid meta-heuristic presented in Section IV.
- *JNLA*, the bicriteria algorithm in Section III;
- *Betweeness*, a greedy algorithm that iteratively removes the edge with the highest betweeness centrality [34]; and
- *Opt. β-dis.*, optimal β-disruptor found with MILP in the appendix.

The costs of nodes and links are assigned according to the above linear scale $c(u) = b + \alpha d(u)$ with $\alpha = b = 0.25$.

Solution Quality. The disruptor costs (the smaller the better) for synthesis networks and AS relationship networks are shown in Figs. 5 and 6, respectively. As shown in the figures, there is a consistent order in terms of solution quality: Opt. β-dis., HMM, JLNA, and Betweeness (from the highest to the lowest). We note the absence of Opt. β-dis. in AS routing networks due to the limit in solving MILP.

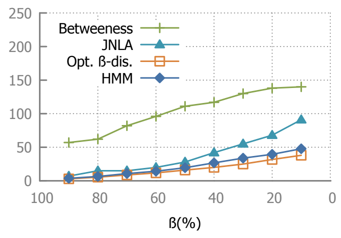
In average, HMM, the runner-up, is only about 25% away from the optimal solutions. It performs far better than the naive Betweeness method and slightly better than JNLA. The improvement of HMM over JLNA is due to the local search procedures. These procedures lower the cost by avoiding separating more than neceessary number of connected pairs, one of the major disadvantages of JNLA. The gaps between HMM and JNLA are quite visible in several cases e.g. in Barabasi network when $\beta = 0.8$ or in Watts-Strogatz network when $\beta = 0.7$. Thus employing local search is essential to get closer to the optimal solutions.

By comparing the disruptor cost of the same algorithm (e.g. Opt β-disruptor) across different network topologies, it shows that the networks in decreasing order of their ‘robustness’ are Erdos-Reyni, Watts-Strogatz, Barabasi, and Forest fire. For example, with a cost equivalent to removing 10% of the links in the network, the Opt β-disruptor method can disrupt 50% connectivity in the Erdos-Reyni network, 60% connectivity in the Barabasi and Watts-Strogatz network, and up to 80% in the Forest fire network. Also, the gaps between Opt β-disruptor and the other algorithms suggest that the ‘complexity’, i.e. how hard it is to approximate the disruptor, tends to follow the same order (i.e. Erdos-Reyni networks are the hardest to find disruptor and Forest fire networks are the easiest instances).

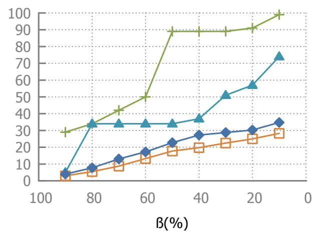
TABLE III: Average running time in seconds

Network	Betwn.	JNLA	Opt β.	HMM
Erdos-Reyni	6.2e-3	5.7e-2	2.3e+4	1.5e-1
Barabsi-Albert	6.5e-3	3.6e-2	7.6e+3	9.0e-2
Watts-Strogatz	8.0e-3	3.6e-2	2.4e+4	1.1e-1
Forest fire	8.8e-3	4.5e-2	3.1e+2	1.2e-1
Oregon AS	1.2e+2	1.3e+2	-	1.9e+2
CAIDA AS	5.0e+1	1.7e+2	-	2.3e+2

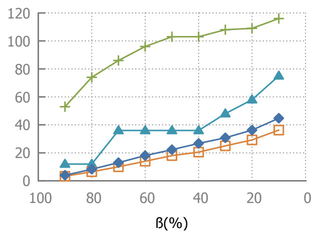
Running Time. We show the average running time in Table III. Consistently, Opt β-dis. takes the longest time, followed by HMM, JNLA, and Betweenness takes the least time. In general, when the network has more than 500 nodes, the optimal solution cannot be found after several weeks. And adding more computing resources (CPU and memory)



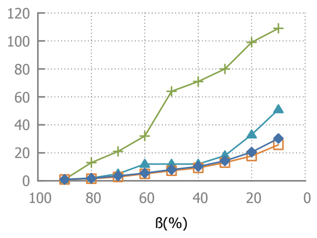
(a) Erdos-Reyni (random) network



(b) Barabasi (power-law) network



(c) Watts-Strogatz network



(d) Forest fire network

Fig. 5: Costs of disruptor algorithms on the synthesis networks (the lower the better)

won't likely to help due to the exponentially increase in the complexity of the problem. Moreover, finding Opt β-dis. via solving the MILP is not only time-consuming but also memory-consuming. When when the network has few thousands nodes, the MILP formulation does not fit in the 32GB memory, let alone solving it.

While Betweeeness is the fastest of all, its quality is far from satisfactory. This leaves us with either JLNA or HMM for larger networks (more than few hundred of nodes). Also HMM takes about 50% additional time comparing to JLNA to produce (consistently) better solutions.

Overall, when the network has less than 200 nodes, we found that optimal solutions are often achievable by solving the MILP. For networks with 200 nodes and 2000 nodes, we can first attempt to solve MILP and if it fails, we can use HMM algorithm. When network has more than 2000 nodes, due to the memory limit, it is advisable to deploy HMM to approximate the optimal disruptor.

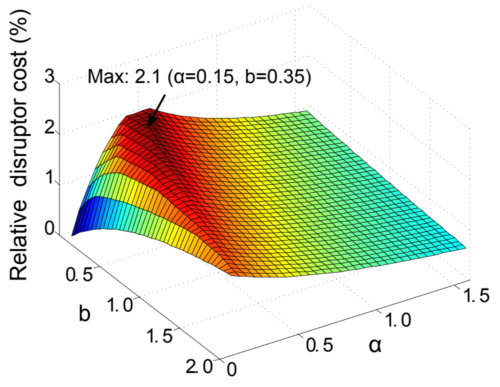


Fig. 7: Cost of Optimal β-disruptor in US backbone network when $\beta = 50\%$. The cost is measured in percentage of the total costs of all nodes and links.

D. Application: Optimal Resource Allocation for Network Protection

We demonstrate that our solutions to assess joint nodes and links failures can be applied to find optimal strategy to allocate resource for maximum network protection. That is given a fixed budget B, we determine the amount of resource that each network element receives to make the network robust against the attacker. In other words, we aim to maximize the cost that the attacker has to pay in order to reduce the fraction of pairwise connectivity to β.

For simplicity, we limit our attention to the cost allocation scheme in Subsection V-A, i.e., all edges have the same cost $c(e) = 1$ and the cost of removing a vertex u is $c(u) = b + \alpha d(u)$. We will need an additional step to scale simultaneously all the costs so that they sum up to B. Among all possible values of b and α, we look for pairs that maximize the cost of β-disruptor.

We show in Fig. 7 the cost of optimal β-disruptor for US backbone network when $\beta = 50\%$, $b \in [0..2.0]$ and $\alpha \in [0..1.5]$. The cost of optimal β-disruptor is measured in percentage of B, the total cost of all nodes and links. As shown in the figure, the highest cost is achieved at $\alpha = 0.15$ and $b = 0.35$.

Despite the simplicity of the allocation scheme, the results offers several insights into how to allocate resource to protect network elements. First, neither focusing on protecting only links (e.g. setting $\alpha = b = 0$) or only nodes (e.g. assigning large values for b, α) result in optimal protection plans. There is certain balance between the effort to spend on the nodes and those on the links to maintain. This balance, when b is small, is reflected through the value of α. Second, the strategy to protect a node u with an amount of resource proportional to its degree $d(u)$ (i.e. setting $b = 0$) is not necessary optimal.

Last but not least, given a method to compute the optimal β-disruptor, the problem of finding the optimal values of b and α to maximize the disruptor cost has the form of a convex optimization problem, which can be solved efficiently [35]. We conjecture that this convex form also holds true for more complicated formulations of the resource allocation problem. Thus, our proposed methods can act as a key component to identify optimal resource allocation strategies.

VI. Concluding Remarks

Joint node and link attacks pose a serious threat to the network. In addition to network connectivity, it is also important to assess the vulnerability of the network under joint node and links networks in terms of other performance metrics such as network throughput, maximum network flow between source-destination pairs, and so on. Furthermore, the problem of allocating resource to protect the network under the joint attacks is of great importance and is the topic of our future study.

References

[1] T. H. Grubesic, T. C. Matisziw, A. T. Murray, and D. Snediker, "Comparative approaches for assessing network vulnerability," *Inter. Regional Sci. Review*, 2008.

[2] A. Murray, T. Matisziw, and T. Grubesic, "Multimethodological approaches to network vulnerability analysis," *Growth Change*, 2008.

[3] A. Sen, S. Murthy, and S. Banerjee, "Region-based connectivity - a new paradigm for design of fault-tolerant networks," in *HPSR*, 2009.

[4] S. Banerjee, S. Shirazipourazad, and A. Sen, "Design and analysis of networks with large components in presence of region-based faults," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1–6.

[5] A. Pinar, J. Meza, V. Donde, and B. Lesieutre, "Optimization strategies for the vulnerability analysis of the electric power grid," *SIAM J. on Optimization*, vol. 20, 2010.

[6] R. Albert, H. Jeong, and A. Barabasi, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, p. 14, 2000.

[7] R. Albert, I. Albert, and G. L. Nakarado, "Structural vulnerability of the north american power grid," *Phys. Rev. E*, vol. 69, no. 2, p. 10, 2004.

[8] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos, "Detecting critical nodes in sparse graphs," *Computers and Operations Research*, vol. 36, no. 7, 2009.

[9] T. N. Dinh, Y. Xuan, M. T. Thai, P. M. Pardalos, and T. Znati, "On new approaches of assessing network vulnerability: Hardness and approximation," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 609 –619, 2012.

[10] T. N. Dinh and M. T. Thai, "Precise structural vulnerability assessment via mathematical programming," in *Proc. of IEEE MILCOM*, 2011.

[11] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters," *IEEE/ACM Trans. Netw.*, pp. 1610–1623, 2011.

[12] P. K. Agarwal, A. Efrat, S. K. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of wdm networks to probabilistic geographical failures," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[13] T. C. Matisziw and A. T. Murray, "Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure," *Comput. Oper. Res.*, vol. 36, pp. 16–26, January 2009.

[14] M. Di Summa, A. Grosso, and M. Locatelli, "Complexity of the critical node problem over trees," *Comput. Oper. Res.*, vol. 38, no. 12, pp. 1766–1774, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.cor.2011.02.016

[15] S. Shen and J. C. Smith, "Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs," *Netw.*, vol. 60, no. 2, pp. 103–119, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1002/net.20464

[16] K. Aggarwal, J. S. Gupta, and K. Misra, "A simple method for reliability evaluation of a communication system," *Communications, IEEE Transactions on*, vol. 23, no. 5, pp. 563–566, 1975.

[17] C. Colbourn, *The combinatorics of network reliability*, ser. International Series of Monographs on Computer Science Series. Oxford University Press, Incorporated, 1987.

[18] N. S. Fard and T.-H. Lee, "Cutset enumeration of network systems with link and node failures," *Reliability Engineering & System Safety*, vol. 65, no. 2, pp. 141 – 146, 1999.

[19] T. N. Dinh, Y. X., M. T. Thai, E. Park, and T. Znati, "On approximation of new optimization methods for assessing network vulnerability," in *Proc. of IEEE INFOCOM*, 2010.

[20] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev, "O($\sqrt{\log n}$) approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems," in *STOC*. ACM, 2005.

[21] S. Arora, E. Hazan, and S. Kale, "O($\sqrt{\log n}$) approximation to sparsest cut in $\tilde{O}(n^2)$ time," *SIAM J. Comput.*, vol. 39, no. 5, 2010.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[23] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888 –905, aug 2000.

[24] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *NIPS*, vol. 14, no. 14, p. 849856, 2001.

[25] Mohar and Poljak, "Eigenvalue in combinatorial optimization," *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, 1992.

[26] M. Meila and W. Pentney, "Clustering by weighted cuts in directed graphs," in *Proceedings of the SIAM Conference on Data Mining*, 2007.

[27] R. B. Lehoucq, D. C. Sorensen, and C. Yang, "Arpack users guide: Solution of large scale eigenvalue problems by implicitly restarted arnoldi methods." 1997.

[28] "US IP Backbone network XO company," http://www.xo.com/about/network/Pages/overview.aspx.

[29] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *KDD*. ACM, 2005, pp. 177–187.

[30] P. Erdos and A. Renyi, "On the evolution of random graphs," *Publ. Math. Inst. Hungary. Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[31] A. Barabasi, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Phy. A*, 2000.

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

[33] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, "A supernodal approach to sparse partial pivoting," *SIAM J. Matrix Analysis and Applications*, vol. 20, no. 3, pp. 720–755, 1999.

[34] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, p. 3541, 1977.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*, ser. Berichte über verteilte messysteme. Cambridge University Press, 2004.

[36] S. Arora, S. Rao, and U. Vazirani, "Expander flows, geometric embeddings and graph partitioning," in *STOC*. ACM, 2004.

Appendix

Mixed Integer Linear Programming

We formulate the β-disruptor as an Mixed Integer Linear Programming (MILP) problem. For each node $u \in V$, we use an integral variable to indicate whether u is removed. $s_u = 1$ if node u is removed and $s_u = 0$, otherwise. Similarly, we associate each edge $(u, v) \in E$ with an integral variable x_{uv}, where $x_{uv} = 1$ iff edge (u, v) is removed and $x_{uv} = 0$, otherwise. Finally, we use real variables $d_{uv} = d_{vu}$ to represent the distance (or disconnectivity) between all pairs $(u, v) \in V \times V$ in the residual network, i.e., $d_{uv} = 0$ if u and v are connected and $d_{uv} > 0$ otherwise. The formulation is as follows.

$$
\begin{aligned}
\text{minimize } & \sum_{u\in V}^{n} c(u)s_u + \sum_{e\in E} c(e)x_e && && (6)\\
\text{subject to } & d_{uv} \le s_u + s_v + x_{uv}, && (u,v)\in E, && (7)\\
& d_{uv} + d_{vw} \ge d_{uw}, && (u,v)\in E && (8)\\
& \sum_{u\neq v} d_{uv} \ge (1-\beta)\tbinom{n}{2}, && && (9)\\
& 0 \le s_u \le d_{uv} \le x_{uv} \le 1, && u, v && (10)\\
& s_u, x_{uv} \in \{0,1\}, && (u,v)\in E && (11)
\end{aligned}
$$

The objective minimizes the total cost of the removed vertices and edges, subjecting to constraint (9) that the pairwise connectivity in G is at most $\beta\binom{n}{2}$. Constraint (8), known as *triangle inequality*, implies that if u and v are connected, and v and w are connected, then u and w must be connected. Constraint (7) guarantees that for each edge $(u, v) \in E$, the distance $d_{uv} > 0$ only if either u, v, or edge (u, v) is removed.

Our MILP formulation offers two special improvements over the direct Integer programming formulations [8]. Firstly, it has only $m+n$ integral variables (s_u and x_e) as we can prove that $\binom{n}{2}$ variables d_{uv} are not required to be integers [10]. Secondly, it has only $O(m \times n)$ constraints while formulations for the similar problems [8] has at least $\Omega(n^3)$ constraints. For many real networks where the average degree is bounded by a constant, the number of constraints is substantially reduced to $O(n^2)$, leading to a huge reduction in memory and running time. The following lemma states the correctness of our formulation.

Lemma 4: The optimal solution of ILP (6-11) induces a minimum cost β-disruptor $D_\beta = (V_\beta, E_\beta)$ of G, where $V_\beta = \{u \mid s_u = 1\}$ and $E_\beta = \{(u, v) \mid x_{uv} = 1\}$.

The proof is similar to the case of the MILP for the β-vertex disruptor problem in [10], and is omitted here.

Approximation for Sparse Cut in Directed Graph [20]

We summarize the $O(\sqrt{\log n})$ approximation algorithm for the sparsest cut problem in directed graphs [20].

Definition 2: A unit-l_2^2 representation of a graph $G = (V, E)$ is an assignment of vector v_i to each vertex $i \in V$ such that

1) All vectors v_i lie on the unit sphere:
$$\forall i \in V ||v_i|| = 1$$
2) The l_2^2 triangle inequality holds:
$$\forall i, j, k \in V ||v_i - v_j|| \leq ||v_i - v_k||^2 + ||v_k - v_j||^2.$$
3) A unit-l_2^2 is c-spread if
$$\sum_{i<j} ||v_i - v_j||^2 \geq 4c(1-c).n^2.$$

Given a directed graph G, the algorithm first finds a unit-l_2^2 representation of G by solving the following Semidefinite Programming relaxation of the directed sparsest cut problem.

$$\text{minimize } \frac{1}{8} \sum_{e=(i,j)\in E} c(e)d(i,j)$$
subject to
$$||v_i - v_j|| \leq ||v_i - v_k||^2 + ||v_k - v_j||^2 \quad \forall i, j, k \in V \cup \{0\}$$
$$\sum_{i<j} ||v_i - v_j||^2 = 1, v_i \in \mathbb{R}^n \quad \forall i \in V \cup \{0\}$$
where $d(i, j) = ||v_i - v_j||^2 - ||v_0 - v_i||^2 + ||v_0 - v_j||^2$.

Then we consider the following two cases.

Case 1: If there is a vertex k such that the ball of squared-radius $\frac{1}{8n^2}$ around v_k contains at least $n/2$ vectors (other than v_0). Let $X = \{i \in V \mid ||v_i - v_k||^2 \leq \frac{1}{8n^2}\}$. We grow X gradually by adding to X in each step the closest vertex to X. During the process, we compute the cut ratio of the cut $(X, V \setminus X)$ and finally return the cut with the minimum ratio.

Case 2: There is no vertices k that satisfies the condition in Case 1. Perform the following steps.

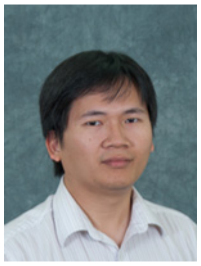
1) Scale all vectors by $2\sqrt{2}n$. Pick a vertex k as the center so that there is a constant fraction of all vertices lie in a spherical annulus of inner radius 1 and outer radius 160.
2) Apply the ARV separation algorithm in [36] for the vertices in the spherical annulus and vector v_0 to find a Δ-separated (w.r.t. the l_2^2 distance) sets S and T s.t. each of them contains at least $2c'$ fraction of vertices, for some constant $c' > 0$.
3) Find radius r s.t. at least half of the vectors in S lie inside the ball of radius r, centered at v_0, and at least half of the vectors lie outside the ball.
4) Let $S^+ = \{i \in S | ||v_0 - v_i||^2 \leq r^2\}$. Let $S^- = \{i \in S | ||v_0 - v_i||^2 \geq r^2\}$.
5) Let $T^+ = \{i \in T | ||v_0 - v_i||^2 \leq r^2\}$. Let $T^- = \{i \in T | ||v_0 - v_i||^2 \geq r^2\}$.
6) If $|T^+| \geq |T^-|$, then $S^* = T^+; T^* = S^-$; else $S^* = S^+; T^* = T^-$.
7) Find the minimum cut $(A, V \setminus A)$ between S^* and T^*. Output $(A, V \setminus A)$.

Theorem 2: [20] Given a directed graph G, the above algorithm find a sparsest cut with a cut ratio at most $O(\sqrt{\log n})$ times the minimum cut ratio.

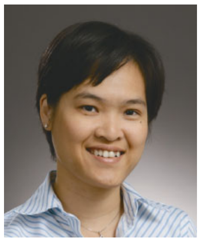
Proof of the observations in Section V-B

We present the observations in Section V-B and discussion.

- $\alpha = 0,\ b \leq 1$: The cost to remove a node u is $b + 0 \times d(u) = b \leq 1$. Thus the cost of removing any edge (one) is no smaller than the cost of removing either one of its end. Thus any solutions that contain edge(s) can be transformed into one with only nodes and without increasing the cost. Thus $\text{OPT}_\beta = \text{OPT}_\beta^V \leq \text{OPT}_\beta^E$ i.e. the optimal β-disruptor contains *no edges*.
- $\alpha = 0,\ b > 1$: The cost of removing u is $b > 1$. If we remove a node u with $d(u) < b$, a better solution is to remove all edges incident to u and pay a cost $d(u) < b$. Thus the optimal solutions contain no u with $d(u) < b$.
- $0 < \alpha < 1$: Using the same argument of removing all incident edges instead of removing a node, we obtain that the optimal β-disruptor contains only vertices of degree at least $\frac{b}{1-\alpha}$.
- $1 \leq \alpha$: Again, if $1 \leq \alpha$, then removing incidents edges is better than removing the node itself. Thus $\text{OPT}_\beta = \text{OPT}_\beta^E \leq \text{OPT}_\beta^V$ i.e. the optimal β-disruptor contains *no vertices*.



Thang N. Dinh (S'11-M'14) received the Ph.D. degree in computer engineering from the University of Florida in 2013. He is an Assistant Professor at the Department of Computer Science, Virginia Commonwealth University. His research focuses on security and optimization challenges in complex systems, especially social networks, wireless and cyber-physical systems. He serves on TPC of several conferences including GLOBECOM, SOCIALCOM, and was the publicity chair of CSoNet 13. He also serves on editorial board of Computational Social Networks journal.



My T. Thai (M'06) received the Ph.D. degree in Computer Science from the University of Minnesota, in 2005. She is an Associate Professor at the Computer and Information Science and Engineering Department, University of Florida. Her current research interests include algorithms and optimization on network science and engineering, with a focus on security. She has engaged in many professional activities, such as being the PC chair of EEE IWCMC 12, IEEE ISSPIT 12, and COCOON 2010. She is a founding EiC of Computational Social Networks journal, an Associate Editor of JOCO, IEEE Transactions on Parallel and Distributed Systems, and a series editor of Springer Briefs in Optimization. She has received many research awards including a UF Provosts Excellence Award for Assistant Professors, a DoD YIP, and an NSF CAREER Award.