Make It or Break It: Manipulating Robustness in Large Networks

Hau Chan
Stony Brook University
hauchan@cs.stonybrook.edu

Leman Akoglu
Stony Brook University
leman@cs.stonybrook.edu

Hanghang Tong
The City College of New York
tong@cs.ccny.cuny.edu

Abstract

The function and performance of networks rely on their robustness, defined as their ability to continue functioning in the face of damage (targeted attacks or random failures) to parts of the network. Prior research has proposed a variety of measures to quantify robustness and various manipulation strategies to alter it. In this paper, our contributions are two-fold. First, we critically analyze various robustness measures and identify their strengths and weaknesses. Our analysis suggests natural connectivity, based on the weighted count of loops in a network, to be a reliable measure. Second, we propose the *first* principled manipulation algorithms that directly optimize this robustness measure, which lead to significant performance improvement over existing, ad-hoc heuristic solutions. Extensive experiments on real-world datasets demonstrate the effectiveness and scalability of our methods against a long list of competitor strategies.

1 Introduction

Robustness, which generally speaking, measures the resilience of a network in response to the external perturbations (e.g., intentional attacks or random failures), is a fundamental property for a variety of networks, such as social, information, communication, biological networks and so on. Networks that sustain their functionality and responsiveness under such changes (targeted or random) are considered to be more robust than others that fail to do so.

In the past few decades, research on robustness has been concerned with *measuring* the robustness of a given network [1, 11, 12], *tracking* its dynamics when the network evolves over time [21], *manipulating* its structure (e.g., by removing nodes) to alter its robustness [1, 3, 15], and *comparing* the robustness of different networks under a certain type of perturbation [1, 4, 7].

In particular, a vast majority of prior works have focused on quantifying robustness and thanks to those efforts we now have a variety of different robustness measures, e.g., size of largest connected component, inverse shortest distances, algebraic connectivity, etc. In principle, the common ingredient/component of these measures is the level of network *connectivity*. While each of these robustness measures has its own emphasis and rationality, as we discuss in the next section, many measures have several shortcomings in capturing the desired connectivity and resilience properties of networks. For example shortest distances are quite sensitive to small alterations, algebraic connectivity does not change monotonically, etc. (see §2.1 for more details).

Ideally, a fully connected network is the most robust; however it is not feasible to design such real-world networks due to constraints in physical space, budget, etc. Alternatively, building redundant (i.e., alternative) paths among the network agents (i.e., nodes) helps improve the resilience against damage in the network. The more and shorter these alternative paths are, the better the resilience would be. Thus, several other measures [19, 27] are built on the so-called *subgraph centrality*, which measures the total (weighted) count of loops in the network.

While most prior research focused on robustness measures, little has been done on *how to manipulate the robustness of a given network by modifying its underlying link structure*. For instance, how can we *enhance* the robustness of a power grid network by carefully introducing a few new power lines? How can we *break down* a disease network by removing (say by an operation) some of its most important cells? How can we maximally break down an adversary network (e.g., a terrorist network) by cutting out some of its most important communication channels? To date, little, except a few ad-hoc solutions has been proposed to answer these kinds of questions (see §2.2 for more details).

In this paper, we focus on two problems; (1) quantifying, and (2) manipulating the robustness in large networks. In particular, we address the following questions: (Q1) *Robustness measure*: While there exist many different robustness measures, it is unclear for the practitioner which measure should s/he choose. What is a good robustness measure that captures the desired resilience properties of a graph? (Q2) *Manipulation algorithms*: Given such a desired measure, how can we design effective and scalable algorithms that directly optimize it for manipulating robustness?

We start by carefully choosing *natural connectivity* [27] as a reliable robustness measure. Next we propose a novel framework called MIoBI (for Make It or Break It) for controlling the robustness of a given graph by modifying its topology. First, we focus on the problem of maximally

decreasing the robustness of a given network by deleting nodes or links. Second, we study the problem of maximally increasing the robustness by carefully introducing a set of new links. A unique feature of our methods is that they aim to *directly* optimize the corresponding robustness measure, which leads to significant performance improvement over the existing, ad-hoc solutions. The proposed methods scale to large graphs, with near-linear complexity in time and space. We summarize our main contributions as follows:

- *Assessment of robustness measures*: We analyze several measures in the literature for their capabilities of capturing desired resilience properties of graphs. We conclude that natural connectivity, that accounts for alternative paths, proves to be a reliable measure.
- *Principled robustness manipulation algorithms*: We formulate manipulation problems to degrade graph robustness via edge/node removal (resp. called MIoBI-BreakEdge/MIoBI-BreakNode) or to improve it via edge addition (MIoBI-MakeEdge) operations. We propose effective and scalable algorithms to identify the best operations for a given budget. Our algorithms are based on theoretical bases and provide the *first* principled, rather than ad hoc, solutions. Prior research has discrepancy between robustness measures considered and manipulation algorithms used (e.g., largest connected component size as measure vs. degree-based removal). We bridge this gap by *directly* optimizing our chosen robustness measure under manipulation.
- *Extensive experiments*: We evaluate our methods on several real-world datasets across different domains for effectiveness and scalability. We show that i) proposed methods outperform a long list of ad hoc strategies, and ii) successfully scale to large graphs with the empirical running time growing near-linearly in graph size.

2 Related Work

The function and performance of networks rely on their robustness, defined as their ability to continue functioning in face of damage to parts of the network. We organize related work on robustness into two sections: (1) work on proposing measures to quantify robustness, and (2) work on studying the effects of network manipulation on robustness.

2.1 Measuring Robustness Simple and effective measures of robustness are essential in the areas of network design and monitoring. In graph theory, robustness can comprise of properties ranging from redundancy and diversity, to concepts such as the ability to operate under perturbation or the efficiency of feedback mechanisms. In principle, the graph *connectivity* is a fundamental measure of robustness.

In [1] network robustness is defined as the critical removal fraction of nodes (or edges) from the network that causes its sudden disintegration. To monitor disintegration, they propose to track the diameter, relative size of the largest connected component (LCC), and average size of isolated clusters. Intuitively, as the fraction of removed nodes/edges increases, the performance of the network eventually collapses at a critical fraction f that corresponds to the network robustness; the larger f is, the more robust the network is. However, while can be computed analytically for special network structures [7, 4], f in general needs to be computed via simulations. Moreover, component sizes do not fully reflect the *level* of connectedness of a given network.

Other prior works have proposed mathematically compact representations to quantify robustness. These measures include connectivity based on minimum node/edge cut [12], average inverse shortest path (geodesic) distances of connected components [1, 3, 15], and algebraic connectivity based on the second smallest (or first non-zero) eigenvalue of the Laplacian [11]. However, these measures only partly reflect the ability of graphs to retain connectedness after manipulation, and fail to exhibit the variation of robustness sensitively [27]. In particular, shortest paths are prone to change drastically with simple alterations and do not capture redundancies (i.e., alternative paths). Moreover, algebraic connectivity takes the value of zero for all disconnected graphs which makes it a measure that is too coarse for complex networks, and it does not change monotonically with the addition/deletion of more edges [27].

Related to geodesic distance, [21] propose a modified measure for time-varying graphs called shortest temporal distance. [19] incorporate the spectral gap related to spectral expansion properties [9] as well as subgraph centralities of the nodes in the network, to quantify network robustness. Other measures include toughness [6], scattering number [16], tenacity [18], integrity [2], fault diameter [17], and isoperimetric number (related to node/edge expansion) [20]. These take into account the cost as well as the magnitude of damage to a network. However, they are combinatorial measures for general graphs.

In summary, while all these and several other measures capture graph connectivity one way or another, they have one or more of the following shortcomings: i) prone to drastic changes by small graph alterations, ii) partially capturing connectivity or alternative paths, iii) combinatorial to compute efficiently, iv) meaningful only for connected graphs, and v) non-monotonically changing by more modifications. Thus, we adopt *natural connectivity* [27] as our measure, which successfully avoids these pitfalls (details in §3).

2.2 Manipulating Robustness One can study the change in robustness under i) few and random node/edge failures, or ii) many or targeted failures (or attacks). In their study, [1] showed that scale-free graphs are resilient to random failures but sensitive to targeted attacks, while for random networks there is smaller difference between the two. As such, researchers proposed and studied different manipulation strategies for targeted attack scenarios for real-world networks.

The most frequently studied strategy to *degrade* robustness has been the removal of most connected (i.e., highest degree) nodes [1, 3, 15]. Further, [15] compared this strategy to node/edge removals based on betweenness centrality and showed that betweenness yields better results, especially for removal of edges. Different from most, and similar to our MIoBI-MakeEdge, [3] investigated modification schemes to *improve* network robustness. In particular, they studied (1) edge rewiring, and (2) edge addition strategies based on i) random, or ii) preferential schemes. They concluded that in general preferential edge additions, i.e. connecting lowest degree nodes, yield the best result.

Finally, designing networks that are optimal with respect to some survivability criteria [12, 23] is a related but different research topic. These consider building a network from scratch, whereas we aim at modifying an existing network as effectively as possible under a given budget, without causing substantial changes to its existing structure.

We remark that all prior research revolves around ad-hoc manipulation techniques. They either use simulations, assume special network models/structures (e.g., random graphs), or develop heuristic edge/node elimination strategies, and compare them across each other (e.g., betweenness versus degree based removals). In contrast, we propose a *principled* framework for network manipulation.

3 Network Robustness

3.1 Notation We consider undirected unipartite irreducible graphs $G(V, E)$ with a vertex/node set V of size n and an edge set E of size m. An (undirected) edge of G between nodes p and r in V is written as $(p, r) \in E$. The set of neighbors and degree of a node $i \in V$ are denoted by $\mathcal{N}(i)$ and d_i, respectively. We use uppercase bold letters for matrices (e.g., $\mathbf{A}$) and lowercase bold for vectors (e.g., $\mathbf{a}$). Given a matrix $\mathbf{A}$, $\mathbf{A}(i, j)$ corresponds to the element at i^{th} row and j^{th} column of $\mathbf{A}$. Moreover, $\mathbf{A}(i, :)$ and $\mathbf{A}(:, j)$ represent the i^{th} row and j^{th} column of matrix $\mathbf{A}$, respectively. We denote the transpose with a prime (e.g., $\mathbf{A}'$, $\mathbf{a}'$). The eigenvalue and associated eigenvector pairs of the adjacency matrix $\mathbf{A}$ are denoted by $(\lambda_j, \mathbf{u_j})$. The i^{th} element of an eigenvector $\mathbf{u_j}$ is represented by $\mathbf{u_{ij}}$.

3.2 Our Robustness Measure In this paper, we adopt a spectral measure of robustness in complex networks, called *natural connectivity* [27], which can be written as follows.

$$\bar{\lambda} = \ln(\frac{1}{n}\sum_{j=1}^{n} e^{\lambda_j}) \tag{3.1}$$

which corresponds to an "average" eigenvalue of $G(V, E)$, where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ denote a non-increasing ordering of the eigenvalues of its adjacency matrix $\mathbf{A}$.

Natural connectivity not only has a simple mathematical formulation that can be interpreted as the average eigenvalue of the graph, but it also has clear physical and structural meaning that can be tied to several connectivity properties of networks. In particular, it explicitly characterizes the redundancy of alternative paths in the network by quantifying the weighted number of closed walks of all lengths.

A walk in G is an alternating sequence of nodes and edges $v_0e_1v_1e_2v_2 \dots e_kv_k$ where $v_i \in V$ and $e_i(v_{i-1}, v_i) \in E$. The walk is closed if $v_0 = v_k$. The number of walks is an important measure for network robustness. Intuitively, it captures the redundancy of routes between the nodes and redundant routes ensure that connections between nodes remain possible in face of damage to the network. Ideally, robustness could consider the number of alternative routes of different lengths for all pairs of nodes, however this measure becomes intractable for very large graphs. Therefore, natural connectivity focuses on the closed walks of the graph.

Closed walks can be directly related to the subgraphs of a graph and derived from the *sum* of the subgraph centralities of all the nodes in the graph. The subgraph centrality $SC(i)$ of a node i is determined based on the "weighted" sum of the number of closed walks that it participates in. Therefore,

$$S(G) = \sum_{i=1}^{n} SC(i) = \sum_{i=1}^{n}\sum_{k=0}^{\infty} \frac{(\mathbf{A}^k)_{ii}}{k!} = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{u_{ij}}^2 e^{\lambda_j}$$
$$= \sum_{j=1}^{n} e^{\lambda_j} \sum_{i=1}^{n} \mathbf{u_{ij}}^2 = \sum_{j=1}^{n} e^{\lambda_j}$$

where $(A^k)_{ii}$ is the number of closed walks of length k of node i. The $k!$ scaling ensures that (i) the weighted sum does not diverge, and (ii) longer walks count less. We note that $S(G)$ is also known as the Estrada index of the graph [8]. As such, we can write

$$\bar{\lambda} = \ln(\frac{1}{n}\sum_{j=1}^{n} e^{\lambda_j}) = \ln(\frac{1}{n}S(G))$$

Moreover, natural connectivity is closely related to *self-communicability* [10] of nodes in the network. The general communicability function between nodes p, q is written as

$$C_{pq} = \sum_{k=0}^{\infty} c_k(\mathbf{A}^k)_{pq}$$

and thus, subgraph centrality can be thought of as self-communicability with *factorial penalization* of walk lengths. The general communicability between any pair of nodes p, q (again with factorial penalty) can be written as (using Taylor series and the spectral decomposition of $\mathbf{A}$)

$$C_{pq} = \sum_{j=1}^{n} \mathbf{u_{pj}}\mathbf{u_{qj}} e^{\lambda_j}.$$

The above arguments show that (1) natural connectivity exhibits characteristics about the communicability in the network through alternative paths, which closely relate to robustness. It associates the robustness to network topology,

graph spectra, and dynamical properties. Moreover, it was shown [22] that (2) natural connectivity has strong discrimination in quantifying the robustness of complex networks and can exhibit the variation of robustness sensitively even for disconnected networks (unlike e.g., algebraic connectivity). Finally, (3) natural connectivity can be shown to change strictly monotonically with the addition/deletion of more and more nodes/edges [27], which agrees with intuition (unlike e.g., node/edge connectivity, algebraic connectivity). These indicate that the natural connectivity can measure the robustness of complex networks stably even for very small sized and disconnected networks. For these reasons, we choose natural connectivity as our network robustness measure.

4 Manipulating Network Robustness

4.1 Problem Definitions We start by introducing the problems we address to manipulate network robustness. Due to space considerations, we briefly (but formally) describe our problems below.[1]

PROBLEM 1. **MIOBI-BREAKEDGE (Edge Deletion)**

Given: A large network G (with $n \times n$ adjacency matrix $\mathbf{A}$) and an integer (budget) k;

Output: A set of k *edges* from $\mathbf{A}$, the deletion of which creates the largest *drop* of the network robustness (as given in Equ. (3.1)) of G.

PROBLEM 2. **MIOBI-BREAKNODE (Node Deletion)**

Given: A large network G (with $n \times n$ adjacency matrix $\mathbf{A}$) and an integer (budget) k;

Output: A set of k *nodes* from $\mathbf{A}$, the deletion of which creates the largest *drop* of the network robustness (as given in Equ. (3.1)) of G.

PROBLEM 3. **MIOBI-MAKEEDGE (Edge Addition)**

Given: A large network G (with $n \times n$ adjacency matrix $\mathbf{A}$) and an integer (budget) k;

Output: A set of k *non-edges* of $\mathbf{A}$, the addition of which creates the largest *increase* in the network robustness (as given in Equ. (3.1)) of G.

A naive way to solve these problems is to enumerate all possible subsets of size k and select the one that yields the best result. However, this strategy is quite infeasible; with search space size $\binom{m}{k}$ for MIOBI-BREAKEDGE, $\binom{n}{k}$ for MIOBI-BREAKNODE, and $\binom{\binom{n}{2}-m}{k}$ for MIOBI-MAKEEDGE (we refer to [5] for formal proofs related to the hardness of these problems).

[1] We remark that we focus on node/edge deletion and edge addition operations to manipulate graph robustness. Another possible graph operation is edge rewiring [3], where existing edges are rewired to connect different pairs of nodes. A principled rewiring can also be done using similar methods to ours; in particular by removing an existing edge via *reverse* MIOBI-BREAKEDGE, and adding it back via MIOBI-MAKEEDGE.

Next we describe our proposed approximation algorithms to solve the problems formulated in this section. Note that the first two problems aim at maximally decreasing (i.e., "breaking") network robustness, whereas the third problem aims at improving (i.e., "making") the robustness.

4.2 "Breaking" Network Robustness

Problem 1: Edge Deletion First, we address the edge deletion problem MIOBI-BREAKEDGE, which aims to find the set of k edges to delete from the graph so that the robustness is shrunk the most.

Let S denote the selected set of k edges to be removed. Let us then write the new, updated robustness $\bar{\lambda}_\Delta$ as

$$\bar{\lambda}_\Delta = \ln(\frac{1}{n}\sum_{j=1}^{n} e^{\lambda_j + \Delta\lambda_j}) \tag{4.2}$$

where $\Delta\lambda_j$ is the difference in λ_j after the adjustment to the graph. To quantify the updated robustness of the graph in face of change, we need to be able to efficiently update the eigenvalues of $\mathbf{A}$ when the graph changes.

Updating the eigenvalues. Using the first order matrix perturbation theory [25], we can compute changes to the eigenvalues $\Delta\lambda_j$ efficiently.

Let $(\lambda_j, \mathbf{u_j})$ be the j^{th} (eigenvalue, eigenvector) pair of the graph G with adjacency matrix $\mathbf{A}$. Let $\Delta\mathbf{A}$ and $(\Delta\lambda_j, \Delta\mathbf{u_j})$ denote the change in $\mathbf{A}$ and $(\lambda_j, \mathbf{u_j})$ $\forall j$, respectively (where $\Delta\mathbf{A}$ is symmetric). Suppose after the adjustment $\mathbf{A}$ becomes

$$\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$$

where $(\tilde{\lambda_j}, \tilde{u_j})$ is written as

$$\tilde{\lambda_j} = \lambda_j + \Delta\lambda_j \qquad \text{and} \qquad \tilde{\mathbf{u_j}} = \mathbf{u_j} + \Delta\mathbf{u_j}$$

LEMMA 4.1. *Given a perturbation $\Delta\mathbf{A}$ to a matrix $\mathbf{A}$, its eigenvalues can be updated by*

$$\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j}. \tag{4.3}$$

Proof. Omitted for brevity. See [5]. □

Using *Lemma 4.1*, perturbing $\mathbf{A}$ with any given edge (p, r) affects the eigenvalues as

$$\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j} = -2\mathbf{u_{pj}}\mathbf{u_{rj}} \tag{4.4}$$

where $\Delta\mathbf{A}(p, r) = \Delta\mathbf{A}(r, p) = -1$ and 0 elsewhere.

As such, for *Problem 1* we are interested in k edges that will minimize $\bar{\lambda}_\Delta$ in Equ. (4.2), or equivalently

$$\begin{aligned} \textbf{min.}\ & e^{\lambda_1+\Delta\lambda_1} + e^{\lambda_2+\Delta\lambda_2} + \ldots + e^{\lambda_n+\Delta\lambda_n} \\ & e^{\lambda_1}(e^{\Delta\lambda_1} + e^{(\lambda_2-\lambda_1)}e^{\Delta\lambda_2} + \ldots + e^{(\lambda_n-\lambda_1)}e^{\Delta\lambda_n}) \\ & c_1(e^{\Delta\lambda_1} + c_2 e^{\Delta\lambda_2} + \ldots + c_n e^{\Delta\lambda_n}) \end{aligned} \tag{4.5}$$

where c_j's denote constant terms and $c_j \leq 1, \forall j \geq 2$.

To find the k most effective edges, we follow a cautious strategy which iteratively finds the best single edge to remove, for k steps. That is, every time an edge is removed,

the criterion/score that is used to find an edge to remove will be updated for the remaining edges, since each removed edge changes this score as it changes the robustness.

This cautious edge deletion strategy has been used in prior research [15] where edge betweenness is used as criterion to remove edges to decrease robustness. Instead of choosing the top-k edges with the highest edge betweenness in one shot, the idea is to remove a single edge in each step after which the betweenness is updated for the remaining, where this procedure is repeated for the next k steps. This is often referred as the "re-calculated" strategy and has been shown to perform better (i.e., affect robustness more) compared to its top-k counterpart.

Therefore, following the re-calculated strategy and by using Equ. (4.4) and (4.5) we will choose the edge (p, r) that minimizes the following:

(4.6)
$$\min_{(p,r)\in E} \quad c_1\left(e^{-2\mathbf{u_{p1}u_{r1}}}+c_2e^{-2\mathbf{u_{p2}u_{r2}}}+\ldots+c_ne^{-2\mathbf{u_{pn}u_{rn}}}\right)$$

Our criterion/score to select edges to remove as given in Equ. (4.6) changes whenever an edge is removed, as the graph structure and thus eigenvectors $\mathbf{u_j}$ change. Thus, after every step we also need to update the eigenvectors. The key question is how to compute changes $\Delta\mathbf{u_j}$ efficiently. For that, we again resort to matrix perturbation theory [25].

Updating the eigenvectors.

LEMMA 4.2. *Given a perturbation $\Delta\mathbf{A}$ to a matrix $\mathbf{A}$, its eigenvectors can be updated by*

$$\Delta\mathbf{u_j} = \sum_{i=1,i\neq j}^{n}\left(\frac{\mathbf{u_i}'\Delta\mathbf{A}\mathbf{u_j}}{\lambda_j-\lambda_i}\mathbf{u_i}\right). \tag{4.7}$$

Proof. Omitted for brevity. See [5]. □

Finally, we remark that it is infeasible to compute all the n eigenvalues of graphs with n nodes, for very large n. Luckily, given the skewed spectrum of real-world graphs, only the top few eigenvalues have large magnitudes which implies that the c_j terms in Equ.s (4.5, 4.6) become much smaller for increasing j. Thus, compute the top t eigenvalues to approximate the robustness of a graph in our experiments.

The pseudo-code of our algorithm for the edge deletion problem MIOBI-BREAKEDGE is given in Algorithm 1. For a fixed budget k, MIOBI-BREAKEDGE is linear w.r.t the size of the graph for both time and space cost.

LEMMA 4.3. Complexity of MIOBI-BREAKEDGE. *The time cost of Alg. 1 is $O(kmt + knt^2)$. The space cost of Alg. 1 is $O(m + nt + k)$.*

Proof. Omitted for brevity. See [5]. □

Problem 2: Node Deletion Next, we address the node deletion problem MIOBI-BREAKNODE, which aims to find the set S of k nodes to delete from the graph so that the robustness is reduced the most. Deletion of a node involves deletion of the node as well as all its incident edges, i.e. edges attached to it.

Algorithm 1 MIOBI-BREAKEDGE

Input: Graph $G(V, E)$, its adj. matrix $\mathbf{A}$, and int. budget k
Output: Set S of k edges to be removed

1: $S = \emptyset$
2: Compute the top t (eigenvalue, eigenvector) pairs $(\lambda_j, \mathbf{u_j})$ of $\mathbf{A}$, $1 \leq j \leq t$
3: **for** $step = 1$ to k **do**
4: Select the edge $(\bar{p}, \bar{r})$ out of $\forall(p, r) \in E$ that minimizes Equ. (4.6) for top t eigenvectors, i.e.

$$\min_{(p,r)\in E} c_1\left(e^{-2\mathbf{u_{p1}u_{r1}}}+c_2e^{-2\mathbf{u_{p2}u_{r2}}}+\ldots+c_te^{-2\mathbf{u_{pt}u_{rt}}}\right)$$

where $c_1 = e^{\lambda_1}$ and $c_j = e^{(\lambda_j-\lambda_1)}$ for $2 \leq j \leq t$
5: $S := S \cup (\bar{p}, \bar{r})$, $E := E\backslash(\bar{p}, \bar{r})$
6: Update $\mathbf{A}$; $\mathbf{A}(\bar{p}, \bar{r}) = 0$ and $\mathbf{A}(\bar{r}, \bar{p}) = 0$
7: Update top t eigenvalues of $\mathbf{A}$ by Equ. (4.3)
8: Update top t eigenvectors of $\mathbf{A}$ by Equ. (4.7)
9: **end for**
10: Return S

Similar to edge deletion, we can delete nodes from the graph one by one (i.e., re-calculated strategy). To do so, we need to find a score similar to Equ. (4.6) for each node to quantify its effect of change on the graph spectrum. Again, using $\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j}$ from *Lemma 4.1*, we will write down a score for each node i where only the i^{th} row and i^{th} column of $\Delta\mathbf{A}$ contain non-zero entries; $(i, v) = (v, i) = -1$, $v \in \mathcal{N}(i)$, for neighbors $\mathcal{N}(i)$ of i.

We can illustrate the node scoring with a toy example, where say we are to remove a node i with 3 neighbors indexed by n_1, n_2, n_3. Let $\mathbf{w_j} = \mathbf{u_j}'\Delta\mathbf{A}$. We can see that $\mathbf{w_{n_1j}} = \mathbf{w_{n_2j}} = \mathbf{w_{n_3j}} = -\mathbf{u_{ij}}$, and $\mathbf{w_{ij}} = -\sum_{v\in N(i)}\mathbf{u_{vj}}$. As such, $\Delta\lambda_j = \mathbf{w_j}\mathbf{u_j} = -\mathbf{u_{ij}}\mathbf{u_{n_1j}} - \mathbf{u_{ij}}\mathbf{u_{n_2j}} - \mathbf{u_{ij}}\mathbf{u_{n_3j}} - \sum_{v\in\mathcal{N}(i)}\mathbf{u_{vj}}\mathbf{u_{ij}}$, equivalently $\Delta\lambda_j = -\mathbf{u_{ij}}(\mathbf{u_{n_1j}} + \mathbf{u_{n_2j}} + \mathbf{u_{n_3j}} + \sum_{v\in\mathcal{N}(i)}\mathbf{u_{vj}}) = -2\mathbf{u_{ij}}\sum_{v\in\mathcal{N}(i)}\mathbf{u_{vj}}$.

Thus, in general $\Delta\lambda_j$ for a removal of node i is given as

$$\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j} = -2\mathbf{u_{ij}}\sum_{v\in\mathcal{N}(i)}\mathbf{u_{vj}} \tag{4.8}$$

For *Problem 2* we are interested in selecting k nodes that will minimize $\bar{\lambda}_\Delta$ in Equ. (4.2). As we will select the nodes iteratively one by one, we will pick the node i that minimizes the following at every step.

(4.9)
$$\min_{i\in V} \quad c_1\left(e^{-2\mathbf{u_{i1}}\sum_{v\in\mathcal{N}(i)}\mathbf{u_{v1}}}+\ldots+c_ne^{-2\mathbf{u_{in}}\sum_{v\in\mathcal{N}(i)}\mathbf{u_{vn}}}\right)$$

where c_j's denote the constants as before. Note that we will also consider only the top t eigenvectors to compute the node selection scores in the experiments.

The algorithm for the node deletion problem MIoBI-BreakNode follows similar lines as of the algorithm for MIoBI-BreakEdge, where we use Equ. (4.9) instead of Equ. (4.6) in Line 4 of Algorithm 1 (omitted for brevity).

Lemma 4.4. Complexity of MIoBI-BreakNode. *The time cost of Alg. for* MIoBI-BreakNode *is* $O(kmt + knt^2)$. *The space cost is* $O(m + nt + k)$.

Proof. Omitted for brevity. See [5]. □

4.3 "Making" Network Robustness

Problem 3: Edge Addition Finally we address the edge addition problem MIoBI-MakeEdge; find k edges to place to the graph so as to improve the robustness the most.

MIoBI-MakeEdge is a harder and computationally more demanding problem than MIoBI-BreakEdge, since there are $O(n^2)$ potential edges to add to a given graph (compared to $O(m)$ edges to remove). As for large graphs quadratic operations are not desirable, we need to design an algorithm that is fast and that scales well.

Similar to deletion, we will adopt the re-calculated strategy for edge additions and find the k edges one by one iteratively. As such, at every iteration, we are interested in finding the edge that maximizes the following.

(4.10)
$$\max_{\substack{(p,r)\notin E \\ p\in V, r\in V}} \quad c_1\left(e^{2\mathbf{u_{p1}}\mathbf{u_{r1}}} + c_2 e^{2\mathbf{u_{p2}}\mathbf{u_{r2}}} + \ldots + c_n e^{2\mathbf{u_{pn}}\mathbf{u_{rn}}}\right)$$

According to the Perron-Frobenius theorem [13], the principal eigenvector associated with the largest eigenvalue of non-negative irreducible matrices has all positive entries. As $G(V, E)$ is a connected undirected graph, $\mathbf{A}$ is irreducible and $\mathbf{u_1}$ is a positive vector. On the other hand other $\mathbf{u_j}$'s, $j > 1$, might potentially have negative entries. This makes finding the edge that maximizes Equ. (4.10) without enlisting all $O(n^2)$ edges challenging.

Next we introduce a fast approximation strategy to pick edges to add without enlisting all possible edges. In particular, we note that the second and onwards terms in Equ. (4.10) keep getting smaller and smaller, due to the skewed spectrum of large real-world graphs [19]. Therefore, we focus on the first term, i.e. $e^{2\mathbf{u_{p1}}\mathbf{u_{r1}}}$. We create a set $\mathcal{C} \subset V$ of size $d_{\max}$, where $d_{\max}$ denotes the maximum node degree in G, that consists of the nodes with highest $\mathbf{u_1}$ entries. For all non-edges (p, r) of G, $p \in \mathcal{C},\ r \in \mathcal{C},\ p \neq r$, we compute Equ. (4.10) considering the top t eigenvectors, and we add the edge $(\bar{p}, \bar{r})$ with the maximum value. We repeat this procedure k times. Algorithm 2 gives the steps of our proposed edge addition algorithm in detail.

Lemma 4.5. Complexity of MIoBI-MakeEdge. *The time cost of Alg. 2 is* $O(mt + kd_{\max}^2 t + knt^2)$. *The space cost is* $O(m + nt + k)$.

Proof. Omitted for brevity. See [5]. □

Algorithm 2 MIoBI-MakeEdge

Input: Graph $G(V, E)$, its adj. matrix $\mathbf{A}$, and int. budget k
Output: Set S of k edges to be added

1: $S = \emptyset$
2: Compute the top t (eigenvalue, eigenvector) pairs $(\lambda_j, \mathbf{u_j})$ of $\mathbf{A}$, $1 \leq j \leq t$
3: **for** $step = 1$ to k **do**
4: Compute the largest degree $d_{\max}$ of $\mathbf{A}$
5: Find the candidate subset $\mathcal{C}$ of $d_{\max}$ nodes with the highest $\mathbf{u_1}$ eigen-scores
6: Select the edge $(\bar{p}, \bar{r})$ out of $\forall (p, r) \notin E, p \in \mathcal{C},\ r \in \mathcal{C},\ p \neq r$, that maximizes Equ. (4.10) for top t eigenvectors, i.e.
$$\max_{\substack{(p,r)\notin E \\ p\in\mathcal{C}, r\in\mathcal{C}}} \quad c_1\left(e^{2\mathbf{u_{p1}}\mathbf{u_{r1}}} + c_2 e^{2\mathbf{u_{p2}}\mathbf{u_{r2}}} + \ldots + c_t e^{2\mathbf{u_{pt}}\mathbf{u_{rt}}}\right)$$
7: $S := S \cup (\bar{p}, \bar{r})$, $E := E \cup (\bar{p}, \bar{r})$
8: Update $\mathbf{A}$; $\mathbf{A}(\bar{p}, \bar{r}) = 1$ and $\mathbf{A}(\bar{r}, \bar{p}) = 1$
9: Update top t eigenvalues of $\mathbf{A}$ by Equ. (4.3)
10: Update top t eigenvectors of $\mathbf{A}$ by Equ. (4.7)
11: **end for**
12: Return S

5 Experimental Evaluation

We evaluate our algorithms with respect to (1) effectiveness in manipulating graph robustness, and (2) running time and scalability, on several real-world graphs. For each kind of graph manipulation, i.e. problem setting, we compare to several ad-hoc heuristic strategies that we compiled.

Datasets. We use the datasets shown in Table 1 (available at `http://snap.stanford.edu/data/`) to evaluate our methods. The Oregon Autonomous System (AS) graphs are AS-level router networks inferred from Oregon route-views, and were collected once a week, for 9 consecutive weeks. Gnutella graphs are the peer-to-peer (P2P) connectivity networks collected daily, over 5 consecutive days.

Evaluation criteria. For effectiveness, we report the relative % change of robustness, i.e. $100|R - \bar{R}|/R$, where R and $\bar{R}$ respectively denote the initial and the final robustness after k operations (the larger the change, the better). For computational cost, we report the wall-clock time in seconds.

Set up. We use top $t = 50$ eigen-pairs for our methods. For large perturbations to the graph (e.g., high degree nodes removed for node deletions), the accumulated error for updating eigen-pairs (using Equ.s (4.3)&(4.7)) might increase rapidly and the performance could degrade. To overcome this issue, we recompute the exact eigen-pairs of the perturbed graph every 50 operations[2]. We call our always-update methods 'Naive' and recomputed ones 'RC@50'.

[2]Sensitivity experiments showed that for a large set of recompute intervals in $[1, 1000]$, the results remained stable (omitted for brevity).

Table 1: Dataset summary.

Dataset	n	m	$density$
Oregon-A	633	1,086	0.0054
Oregon-B	1,503	2,810	0.0024
Oregon-C	2,504	4,723	0.0015
Oregon-D	2,854	4,932	0.0012
Oregon-E	3,995	7,710	0.0009
Oregon-F	5,296	10,097	0.0007
Oregon-G	7,352	15,665	0.0005
Oregon-H	10,860	23,409	0.0004
Oregon-I	13,947	30,584	0.0003
P2P-GnutellaA	6,301	20,777	0.0010
P2P-GnutellaB	8,114	26,013	0.0008
P2P-GnutellaC	8,717	31,525	0.0008
P2P-GnutellaD	8,846	31,839	0.0008
P2P-GnutellaE	10,876	39,994	0.0007

5.1 Effectiveness of Proposed MIOBI Framework We first describe the heuristic strategies we compared our methods to, for each problem setting. These heuristics are not tied to natural connectivity, in fact to any measure, hence they select nodes/edges irrespective of a robustness measure.

MIOBI-BREAKEDGE *competitor strategies* (11). (1) 'rand': randomly picked k edges (avg.'ed over 10 runs); edges (p, r) with (2) 'rich-rich': highest $d_p d_r$; (3) 'poor-poor': smallest $d_p d_r$; (4) 'rich-poor': highest $|d_p - d_r|$; (5) 'betw': highest edge-betweenness; (6) 'embed': highest embeddedness [14]; (7) 'resist': highest effective resistance [24]; (8) 'netmelt': highest $\mathbf{u_{p1}} \mathbf{u_{r1}}$ [26]; (9) 'line-deg': highest degree in the line graph;[3] (10) 'line-eig': highest eigen-centrality in the line graph; and (11) 'line-page': highest Pagerank score in the line graph.

MIOBI-BREAKNODE *competitor strategies* (5). (1) 'rand': randomly picked k nodes (avg.'ed over 10 runs); nodes i with (2) 'max-deg': highest degree; (3) 'eig': highest eigen-centrality; (4) 'page': highest Pagerank score; and (5) 'cluster': highest local clustering coefficient.

MIOBI-MAKEEDGE *competitor strategies* (5). (1) 'rand': edges between randomly picked nodes (avg.'ed over 10 runs); (2) 'rich-rich': edges between nodes with highest degrees; (3) 'poor-poor': edges between nodes with lowest degrees (same as 'preferential addition' in [3]); (4) 'rich-poor': edges $(p, r) \notin E$ with highest $|d_p - d_r|$; and (5) 'netgel': edges $(p, r) \notin E$ with highest $\mathbf{u_{p1}} \mathbf{u_{r1}}$ [26];

Results. Fig. 1 shows the performance results; % robustness change vs. k for all methods on two selected datasets (See [5] for plots on other datasets). Using RC@50, our methods outperform all other heuristics for all datasets we considered. We further notice that 'rich-rich' performs well for edge deletions, and 'max-deg' and 'page' achieve quite close performance to our method for node deletions. For edge additions, 'rich-rich' performs the next-best and our method outperforms all competitors with a large margin.

To demonstrate performance on all datasets (fixed k), we give Tables 2, 3, 4, resp. for all three problem settings. We can see that our methods achieve the best performance across almost all datasets, and all manipulation settings.

5.2 Scalability of Proposed MIOBI Framework We used the *Oregon A-I* datasets, sorted by m, to evaluate the scalability of the proposed algorithms. The run time results are presented in Fig. 2 for various k.[4] We can see that the proposed methods empirically scale near-linearly wrt m, which means that they are suitable for large graphs.

6 Conclusion

In this paper we studied graph robustness; in particular problems related to its definition and manipulation in large graphs. We first analyzed various definitions and measures of robustness, and enlisted their capabilities of capturing desired resilience properties and shortcomings. We identified natural connectivity as a reliable measure, as it effectively quantifies the existence of alternative paths in a network. We formulated two new robustness manipulation problems, one of which is to maximally decrease (or "break") the robustness with edge or node deletions, and another is to maximally improve (or "make") the robustness with edge additions. We studied the hardness associated with these problems, and proposed effective, scalable, and adaptive algorithms to solve them, which are founded on a principled framework based on theoretical foundations. Finally, our experiments showed the superiority of our methods compared to a long list of heuristic, ad-hoc strategies.

Acknowledgments

This material is based upon work supported by the Army Research Office under Contract No. W911NF-14-1-0029 and W911NF-09-2-0053, Stony Brook University Office of Vice President for Research, National Science Foundation under Grant No. IIS1017415 and an NSF Graduate Research Fellowship. Any findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

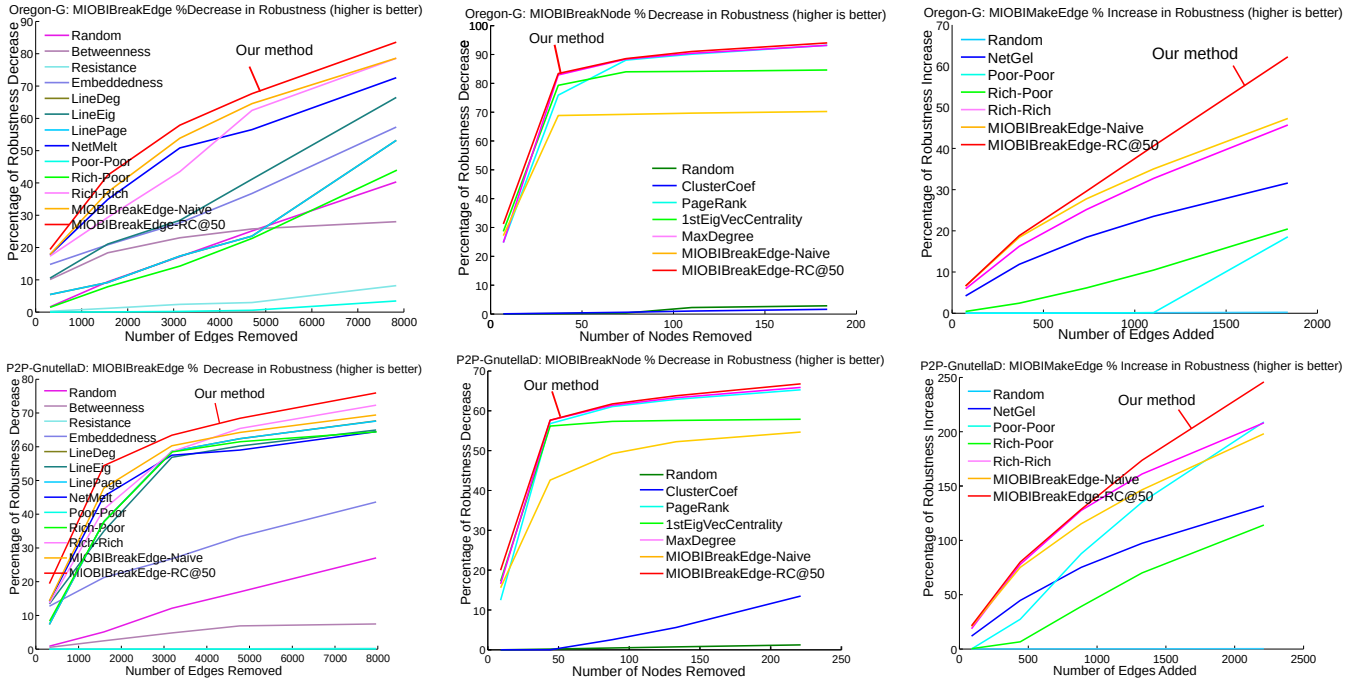
References

[1] R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(6794), 2000.

[2] C. A. Barefoot, R. Entringer, and H. Swart. Integrity of trees and powers of cycles. *SEICCGTC*, 58:103–114, 1987.

[3] The line graph $L(G)$ of a graph G is one where each edge in G becomes a node in $L(G)$, and there is an edge from one node to the other in $L(G)$ if the target of the former edge is the same as the source of the latter edge in the original graph G [26].

[4] All reported times are on a 64-bit machine, Intel Core i5-3570K CPU @3.40GHz and 8GB memory, running Ubuntu 12.10.



(a) MIOBI-BREAKEDGE (b) MIOBI-BREAKNODE (c) MIOBI-MAKEEDGE

Figure 1: % robustness change (higher is better) vs. budget k for proposed MIOBI and various competing heuristics on two datasets, (top) *Oregon-G* and (bottom) *P2P-GnutellaD*, for (a) edge deletions, (b) node deletions, and (c) edge additions. Notice that our methods outperform all the heuristics at all ranges of k. (figures best viewed in color)

Table 2: Edge deletion performances: % robustness change (higher is better) for all graphs when $k = 0.25m$ edges removed.

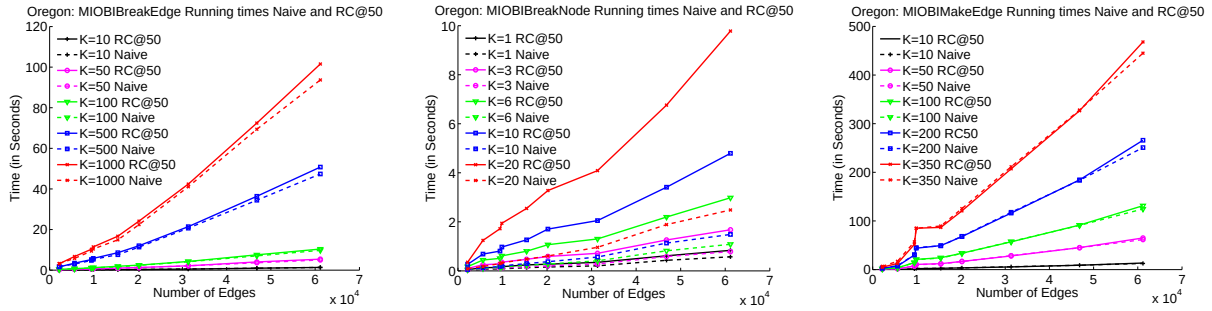
Methods	*O-A*	*O-B*	*O-C*	*O-D*	*O-E*	*O-F*	*O-G*	*O-H*	*O-I*	*G-A*	*G-B*	*G-C*	*G-D*	*G-E*
#Edges removed	543	1405	2362	2466	3855	5049	7833	11705	15292	5194	6503	7881	7960	9999
Random	41.16	42.02	37.48	37.18	39.31	38.45	40.32	40.38	39.65	24.71	24.86	28.00	27.05	27.79
Betweenness	29.49	24.53	24.41	23.56	24.74	25.53	27.99	28.44	29.12	4.51	3.62	13.82	7.46	11.00
Resistance	13.63	11.99	11.40	10.53	10.15	8.47	8.21	7.75	6.71	0.10	0.08	0.35	0.22	0.89
Embeddedness	54.76	56.04	52.92	55.03	56.63	55.82	57.31	60.50	61.07	67.90	62.60	45.24	43.62	26.82
LineDeg	41.10	42.52	43.67	57.95	54.63	55.33	53.18	64.58	67.91	73.94	74.24	63.57	67.67	50.29
LineEig	60.44	61.01	60.45	63.05	63.62	63.20	66.46	67.20	68.54	72.48	72.10	62.31	64.97	49.45
LinePage	41.10	42.52	43.67	57.95	53.77	55.33	53.18	64.58	66.17	73.91	74.15	63.71	67.60	50.00
NetMelt	60.99	61.22	61.38	73.17	68.33	68.86	72.58	75.36	72.47	71.58	71.51	62.36	64.47	48.78
Poor-Poor	13.97	10.67	8.61	2.66	4.80	3.27	3.53	2.55	2.09	0.02	0.02	0.20	0.09	0.59
Rich-Poor	35.78	38.93	40.93	48.52	49.99	46.81	43.98	57.95	64.05	73.26	72.75	62.25	64.45	45.85
Rich-Rich	63.50	64.35	64.30	74.48	74.95	70.12	75.16	79.07	76.01	75.84	76.11	68.95	70.89	55.80
MIOBI-Naive	57.26	64.84	65.00	66.86	70.59	74.88	78.62	79.59	81.66	75.19	74.60	66.88	69.40	50.01
MIOBI-RC@50	**66.11**	**71.10**	**72.78**	**79.66**	**79.10**	**82.05**	**83.57**	**85.97**	**87.04**	**79.73**	**80.34**	**74.59**	**75.96**	**64.68**

Table 3: Node deletion performances: % robustness change (higher is better) for all graphs when k=0.025n nodes removed.

Methods	*O-A*	*O-B*	*O-C*	*O-D*	*O-E*	*O-F*	*O-G*	*O-H*	*O-I*	*G-A*	*G-B*	*G-C*	*G-D*	*G-E*
#Nodes removed	16	38	63	71	100	132	184	272	349	158	203	218	221	272
Random	4.21	2.58	3.70	1.92	1.71	1.07	2.85	2.50	1.54	0.43	1.74	5.74	1.24	1.35
ClusterCoef	2.28	2.47	2.31	3.05	2.44	2.60	1.62	1.40	0.92	13.66	11.64	12.16	13.49	0.89
PageRank	**93.06**	91.47	93.69	92.20	92.93	92.22	93.19	93.21	**93.63**	75.26	74.58	62.96	65.29	45.91
1stEigVecCentrality	89.89	86.96	88.59	82.54	81.45	85.57	84.63	85.20	81.93	70.53	68.44	54.23	57.89	34.41
MaxDegree	92.25	90.80	93.44	92.36	92.81	92.35	93.18	93.06	93.55	75.56	74.85	63.52	65.86	46.68
MIOBI-Naive	92.38	82.55	89.82	82.20	83.92	83.30	70.22	71.72	69.03	36.60	51.39	59.65	54.67	38.48
MIOBI-RC@50	92.50	**91.51**	**93.92**	**92.47**	**93.49**	**93.15**	**94.04**	**94.24**	92.08	**76.19**	**75.69**	**64.19**	**66.75**	**47.24**

Table 4: Edge addition performances: % robustness change (higher is better) for all graphs when $k = 0.01n$ edges added.

Methods	*O-A*	*O-B*	*O-C*	*O-D*	*O-E*	*O-F*	*O-G*	*O-H*	*O-I*	*G-A*	*G-B*	*G-C*	*G-D*	*G-E*
#Edges added	6	15	25	29	40	53	74	109	139	63	81	87	88	109
Random	0.03	0.01	0.03	0.01	0.15	0.01	0.01	0.01	0.01	0.02	0.00	0.01	0.04	0.05
NetGel	1.79	2.70	2.42	2.40	2.58	3.57	4.20	4.73	4.98	5.48	6.84	12.29	11.84	18.85
Poor-Poor	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.38
Rich-Poor	0.55	0.47	0.58	0.77	0.64	0.55	0.42	0.44	0.45	0.26	0.22	0.08	0.31	1.58
Rich-Rich	3.38	3.97	3.58	3.86	3.66	5.24	5.94	6.76	6.86	9.58	12.52	14.94	18.70	24.29
MIoBI-Naive	**3.49**	**4.37**	**4.10**	**4.05**	**4.14**	5.60	6.59	7.36	7.75	10.38	13.13	22.62	20.74	34.25
MIoBI-RC@50	**3.49**	**4.37**	**4.10**	**4.05**	**4.14**	**5.62**	**6.61**	**7.41**	**7.81**	**10.49**	**13.20**	**23.16**	**21.40**	**35.84**



(a) MIoBI-BREAKEDGE (b) MIoBI-BREAKNODE (c) MIoBI-MAKEEDGE

Figure 2: Scalability of proposed methods: all three algorithms scale near-linearly wrt graph size. (figures best in color)

[3] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish. Improving network robustness by edge modification. *Physica A: Stat. Mech. and its Appl.*, 357(3-4):593–612, 2005.

[4] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network Robustness and Fragility: Percolation on Random Graphs. *Phys. Rev. Let.*, 85(25):5468–5471, 2000.

[5] H. Chan, L. Akoglu, and H. Tong. Manipulating the robustness of large networks by node and edge alterations. *Technical Report SBU-CSE-14-1*, 2014.

[6] V. Chvátal. Tough graphs and hamiltonian circuits. *Discrete Mathematics*, 306(10-11):910–917, 2006.

[7] R. Cohen, K. Erez, D. B. Avraham, and S. Havlin. Breakdown of the Internet under Intentional Attack. *Physical Review Letters*, 86(16):3682–3685, 2001.

[8] E. Estrada. Characterization of 3D molecular structure. *Chemical Physics Letters*, 319(5-6):713–718, Mar. 2000.

[9] E. Estrada. Network robustness to targeted attacks. the interplay of expansibility and degree distribution. *Physical Journal B - Complex Systems*, 52(4):563–574, 2006.

[10] E. Estrada, N. Hatano, and M. Benzi. The physics of communicability in complex networks. abs/1109.2950, 2011.

[11] M. Fiedler. Algebraic Connectivity of Graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[12] H. Frank and I. Frisch. Analysis and Design of Survivable Networks. *IEEE Trans. Comm. Tech.*, 18(5):501–519, 1970.

[13] F. Gantmacher. *The Theory of Matrices*. Number Bd. 2 in Chelsea Publishing. American Mathematical Society, 1960.

[14] M. S. Granovetter. Economic action and social structure: Problem of embeddedness. *A. J. of Soc.*, 91:481–510, 1985.

[15] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Physical Review E*, 65(5):056109, 2002.

[16] H. A. Jung. On a class of posets and the corresponding comparability graphs. *J. Comb. Theory*, 24(2):125–133, 1978.

[17] M. Krishnamoorthy and B. Krishnamurthy. Fault diameter of interconnection networks. *Computers & Mathematics with Applications*, 13(5–6):577 – 582, 1987.

[18] D. M. M. Cozzens and S. Stueckle. The tenacity of a graph. *ICTAG*, 24:1111–1122, 1995.

[19] F. D. Malliaros, V. Megalooikonomou, and C. Faloutsos. Fast robustness estimation in large social graphs: Communities and anomaly detection. In *SDM*, pages 942–953, 2012.

[20] B. Mohar. Isoperimetric numbers of graphs. *J. Comb. Theory, Ser. B*, 47(3):274–291, 1989.

[21] S. Scellato, I. Leontiadis, C. Mascolo, P. Basu, and M. Zafer. Evaluating temporal robustness of mobile networks. *IEEE Trans. Mob. Comput.*, 12(1):105–117, 2013.

[22] Y.-L. Shang. Local natural connectivity in complex networks. *Chinese Physics Letters*, 28(6), 2011.

[23] B. Shargel, H. Sayama, I. R. Epstein, and Y. Bar-Yam. Optimization of robustness and connectivity in complex networks. *Phys Rev Lett*, 90(6):068701, 2003.

[24] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comp.*, 40(6):1913–1926, 2011.

[25] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[26] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, pages 245–254. ACM, 2012.

[27] J. Wu, B. Mauricio, Y.-J. Tan, and H.-Z. Deng. Natural connectivity of complex networks. *Chinese Physics Letters*, 27(7):78902, 2010.