Where Graph Topology Matters: The Robust Subgraph Problem

Hau Chan Shuchu Han Leman Akoglu
Stony Brook University
{hauchan, shhan, leman}@cs.stonybrook.edu

Abstract

Robustness is a critical measure of the resilience of large networked systems, such as transportation and communication networks. Most prior works focus on the global robustness of a given graph at large, e.g., by measuring its overall vulnerability to external attacks or random failures. In this paper, we turn attention to local robustness and pose a novel problem in the lines of subgraph mining: given a large graph, how can we find its most robust local subgraph (RLS)?

We define a robust subgraph as a subset of nodes with high communicability [15] among them, and formulate the RLS-PROBLEM of finding a subgraph of given size with maximum robustness in the host graph. Our formulation is related to the recently proposed general framework [39] for the densest subgraph problem, however differs from it substantially in that besides the number of edges in the subgraph, robustness also concerns with the *placement* of edges, i.e., the subgraph topology. We show that the RLS-PROBLEM is **NP**-hard and propose two heuristic algorithms based on top-down and bottom-up search strategies. Further, we present modifications of our algorithms to handle three practical variants of the RLS-PROBLEM. Experiments on synthetic and real-world graphs demonstrate that we find subgraphs with larger robustness than the densest subgraphs [9, 39] even at lower densities, suggesting that the existing approaches are not suitable for the new problem setting.

1 Introduction

Complex networked systems, such as the Internet, road networks, communication networks, the power grid, etc., are a major part of our modern world. The performance and reliable functioning of complex networks depend on their structural robustness, e.g., their ability to retain functionality in the face of damage to parts of the network [40].

Robustness has been studied in many fields including physics, biology, mathematics, and networking. The research areas include quantifying robustness of a network [12, 24, 27, 41], studying the response of networks to various attack strategies [1, 6, 11, 14, 24, 36], manipulating a network to improve its overall robustness [5, 7, 35, 42], and designing optimally robust networks from scratch [18, 21, 29, 31].

A vast majority of prior work has focused on the global robustness of graphs at large. On the other hand, research on local robustness is limited to a few works, e.g., on finding robust subgraphs with large spectral radius [2] and identifying critical regions [37]. In this paper, we turn attention to local robustness and pose a novel subgraph mining problem: given a large graph, how can we find its most robust local subgraph of a given size?

Our measure of robustness is the natural connectivity which is based on the reachability of the nodes, also phrased as their "communicability" [41]. As we introduced in prior work [7], it exhibits several desirable properties; e.g., it captures redundancy by quantifying the count and length of alternative/back-up paths between the nodes. As such, robust subgraphs are intuitively sets of nodes with high communicability among each other. From the practical point of view, they may form the cores of larger communities or constitute the central backbones in large networks, maintaining connectivity and communication at large [15].

While the robust subgraph problem has not been studied before, similar problems have been addressed (§6). Probably the most similar to ours is the densest subgraph problem, aiming to find subgraphs with highest average degree [4, 9, 22] or edge density [28, 39]. However, density is different from robustness; while the former concerns with the number of edges in the subgraph, the topology is also of concern for the latter (§2.2). We offer the following contributions.

- We formulate a new problem of finding the most robust local subgraph (RLS) in a given graph. While in the line of subgraph mining problems, it has not been studied theoretically before (§3.1).
- We show that RLS-PROBLEM is **NP**-hard, and further study its heredity and monotonicity properties (§3.2).
- We propose two fast heuristic algorithms to solve the RLS-PROBLEM for large graphs: a top-down greedy algorithm that iteratively removes nodes, and a bottom-up approach based on the greedy randomized adaptive search procedure (GRASP) [17] (§4).
- We introduce three practical variants of the RLS-PROBLEM (§3.3); and show how to modify our algorithms to address these problem variants (§4).

We extensively evaluate our methods on both synthetic and real-world graphs. As our RLS-PROBLEM is a new one, we compare to three algorithms (one in [9], two in [39]) for the densest subgraph problem. We find subgraphs with higher robustness than the densest subgraphs even at lower densities, demonstrating that the existing algorithms are not applicable for the new problem setting (§5).

2 Background and Preliminaries

2.1 Graph Robustness Robustness is a critical property of large-scale networks, and thus has been studied in physics, mathematics, computer science, and biology. As a result, there exists a diverse set of robustness measures, e.g., mean shortest paths, efficiency, pairwise connectivity, etc. [12].

In this paper, we adopt a spectral measure of robustness called *natural connectivity* [41], written as

$$\bar{\lambda}(G) = \log(\frac{1}{n}\sum_{i=1}^{n} e^{\lambda_i}) \ , \tag{2.1}$$

which can be thought of as the "average eigenvalue" of graph G, where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ denote a non-increasing ordering of the eigenvalues of its adjacency matrix $\mathbf{A}$.

Among other desirable properties [7], natural connectivity is interpretable; it is directly related to the subgraph centralities (SC) in the graph. The $SC(i)$ of a node i is known as its communicability [15], and is based on the "weighted" sum of the number of closed walks that it participates in:

$$S(G) = \sum_{i=1}^{n} SC(i) = \sum_{i=1}^{n}\sum_{k=0}^{\infty} \frac{(\mathbf{A}^k)_{ii}}{k!} \ ,$$

where $(A^k)_{ii}$ is the number of closed walks of length k of node i. The $k!$ scaling ensures that the weighted sum does not diverge, and longer walks count less. $S(G)$ is also referred to as the Estrada index [15] which strongly correlates with the folding degree of proteins [13].

Noting that $\sum_{i=1}^{n}(\mathbf{A}^k)_{ii} = \text{trace}(\mathbf{A}^k) = \sum_{i=1}^{n}\lambda_i^k$ and by Taylor series of the exponential function we can write

$$S(G) = \sum_{k=0}^{\infty}\sum_{i=1}^{n} \frac{(\mathbf{A}^k)_{ii}}{k!} = \sum_{i=1}^{n}\sum_{k=0}^{\infty} \frac{\lambda_i^k}{k!} = \sum_{i=1}^{n} e^{\lambda_i} \ .$$

As such, natural connectivity is the normalized Estrada index and quantifies the "average communicability" in G.

2.2 Robustness vs. Density Graph robustness appears to be related to graph density; however as we show here, there exist key distinctions between them.

Firstly, while density directly uses the number of edges e, such as $\frac{2e(G)}{|V|}$ as in average degree [4, 9, 22] or $\frac{2e(G)}{|V|(|V|-1)}$ as in edge density [28, 39], robustness follows an indirect route; it quantifies the count and length of paths and uses the graph spectrum. Thus, the objectives of robust and dense subgraph mining problems are distinct.

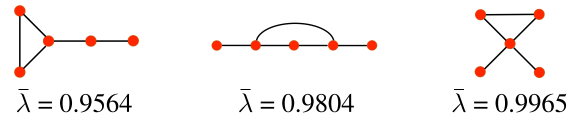


Figure 1: Example graphs with the same density but different robustness, due to their distinct graph topology.

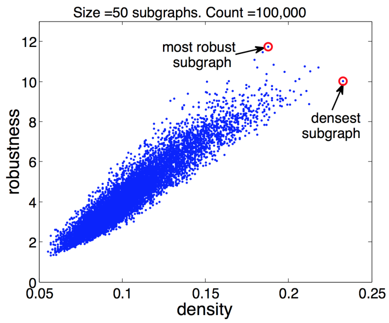


Figure 2: Robustness vs. Density of 100,000 connected subgraphs (blue dots) from a real-world email network.

More notably, density concerns with the number of edges in the graph and not with the topology. On the other hand, for robustness the *placement* of edges (i.e., topology) is as much, if not more important. In fact, graphs with the same number of nodes and edges but different topologies are indistinguishable from the density point of view (Figure 1).

To illustrate further, we show in Figure 2 the robustness vs. density of example subgraphs, each of size 50, sampled[1] from a real-world email network (§5, Table 1). While the two properties are correlated, subgraphs with the same density can have a range of different robustness. In fact, among the samples, the densest and the most robust subgraphs are distinct, indicating that one does not always imply the other.

3 Robust Local Subgraphs

3.1 Problem Definition In their inspiring work [39], Tsourakakis *et al.* recently defined a general framework for subgraph density functions, which is written as

$$f_\alpha(S) = g(e[S]) - \alpha h(|S|) \ ,$$

where $S \subseteq V$ is a set of nodes, $S \neq \emptyset$, $e[S]$ is the number of edges in the subgraph induced by S, $\alpha > 0$, and g and h are any two strictly increasing functions.

Under this framework, maximizing the *average degree* of a subgraph [4, 9, 22] corresponds to $g(x) = h(x) = \log x$ and $\alpha = 1$ such that

$$f(S) = \log\frac{e[S]}{|S|} \ .$$

In order to define our problem, we can relate the objective of our setting to this general framework. Specifically, our objective can be written as

$$f(S) = \log\frac{\sum_{i=1}^{|S|} e^{\lambda_i}}{|S|} \ ,$$

which is to maximize the *average eigenvalue* of a subgraph. Therefore, the objectives of the two problems are distinct, although they both fall under a more general framework [39].

[1] We create subgraphs by snowball sampling: pick a random node and progressively add its neighbors with probability p, and iterate in a depth-first fashion. Connectivity is guaranteed by adding at least one neighbor of each node. We use varying $p \in (0, 1)$ to control the tree-likeness, and obtain subgraphs with various densities.

In the following we formally define our robust local subgraph mining problem, which is to find the highest robustness subgraph of a certain size (hence the locality) in a given graph, which we call the RLS-PROBLEM.

PROBLEM 1. (RLS-PROBLEM) *Given a graph $G = (V, E)$ and an integer s, find a subgraph with nodes $S^* \subseteq V$ of size $|S^*| = s$ such that*

$$f(S^*) = \log \sum_{i=1}^{s} e^{\lambda_i([S^*])} - \log s \geq f(S), \quad \forall S \subseteq V, |S| = s.$$

S^* is referred as the most robust s-subgraph.

One can interpret a robust subgraph as containing a set of nodes having large communicability within the subgraph.

THEOREM 3.1. *The optimal* RLS-PROBLEM *is* **NP**-*Hard.*

Proof. See [8]. Omitted due to space limit.

3.2 Problem Properties

Certain characteristics of hard combinatorial problems sometimes guide the development of approximation algorithms for those problems. In this work, we study two such characteristics, namely semi-heredity and subgraph monotonicity, for the RLS-PROBLEM.

Problems that exhibit the (semi-)heredity or monotonicity properties often enjoy algorithms that explore the search space in a smart and efficient way. For example cliques exhibit heredity, i.e., all induced subgraphs are also cliques. This is a key property used in successful algorithms for the maximum clique problem, e.g., checking maximality by inclusion is a trivial task and effective pruning strategies can be employed within a branch-and-bound framework. Other algorithms exploit monotonicity to employ "smart node ordering" strategies to find iteratively improving solutions. Such orderings help starting with a promising node and sequentially adding the next node in the order such that the resulting subgraphs all satisfy some desired criteria, like a minimum density, which enables finding large solutions quickly.

THEOREM 3.2. *Robustness $\bar{\lambda}$ is not semi-hereditary. That is, a graph with $\bar{\lambda} = \alpha$ and $s > 1$ nodes is not always a strict superset of* some *graph with $s - 1$ nodes and $\bar{\lambda} \geq \alpha$.*

THEOREM 3.3. *Robustness $\bar{\lambda}$ is not subgraph monotonic.*

Proof. See [8] for definitions and proofs.

Alas, robust subgraphs do not exhibit any of these properties. This suggests that our RLS-PROBLEM is likely harder than the maximum clique and densest subgraph problems as, unlike robust subgraphs, (quasi-)cliques are shown to exhibit e.g., the (semi-)heredity property [28].

3.3 Problem Variants

In [8], we introduce three practical variants of our RLS-PROBLEM: finding (i) the most robust subgraph (no size constraint), (ii) top-k most robust s-subgraphs, and (iii) the most robust s-subgraph including a given set of seed nodes. We also show how to adapt our algorithms for the RLS-PROBLEM to these variants (§4).

4 Finding Robust Local Subgraphs

Given the hardness of the RLS-PROBLEM, we design two heuristic solutions. The first is GREEDYRLS, a top-down approach that iteratively removes nodes to obtain a subgraph of desired size. This greedy strategy serves as a simple baseline. Our second and proposed solution GRASP-RLS is a bottom-up randomized approach in which we iteratively add nodes to build up our subgraphs. Both solutions order the nodes by their contributions to the robustness.

4.1 Greedy Top-down Search Approach

This approach iteratively and greedily removes the nodes one by one from the given graph $G = (V, E)$, $|V| = n, |E| = m$, until a subgraph with the desired size s is reached. At each iteration, the node whose removal results in the maximum robustness of the residual graph is selected for removal.[2]

The removal of a node involves removing the node itself and the edges attached to it from the graph, where the residual graph becomes $G[V \backslash \{i\}]$. Let i denote a node to be removed. Let us then write the updated robustness $\bar{\lambda}_\Delta$ as

$$(4.2) \qquad \bar{\lambda}_\Delta = \log \left(\frac{1}{n-1} \sum_{j=1}^{n-1} e^{\lambda_j + \Delta\lambda_j} \right).$$

As such, we are interested in identifying the node that maximizes $\bar{\lambda}_\Delta$, or equivalently

(4.3)

$$\begin{aligned} \mathbf{max}. \;\; & e^{\lambda_1+\Delta\lambda_1} + e^{\lambda_2+\Delta\lambda_2} + \ldots + e^{\lambda_{n-1}+\Delta\lambda_{n-1}} \\ & e^{\lambda_1}(e^{\Delta\lambda_1} + e^{(\lambda_2-\lambda_1)}e^{\Delta\lambda_2} + \ldots + e^{(\lambda_{n-1}-\lambda_1)}e^{\Delta\lambda_{n-1}}) \\ & e^{\lambda_1}(e^{\Delta\lambda_1} + c_2 e^{\Delta\lambda_2} + \ldots + c_{n-1}e^{\Delta\lambda_{n-1}}) \end{aligned}$$

where c_j's denote $e^{\lambda_j - \lambda_1}$ $\forall j \geq 2$ and $c_j \leq 1$.

4.1.1 Updating the eigen-pairs

When a node is removed from the graph, its spectrum (i.e., the eigen-pairs $(\lambda_j, \mathbf{u_j})$) also changes. Recomputing the eigen-values to compute robustness $\bar{\lambda}_\Delta$ every time a node is removed is computationally challenging. Therefore, we employ fast update schemes based on the first order matrix perturbation theory [33].

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 each eigen-pair $(\tilde{\lambda}_j, \tilde{u}_j)$ is written as

$$\tilde{\lambda}_j = \lambda_j + \Delta\lambda_j \qquad \text{and} \qquad \tilde{\mathbf{u}}_\mathbf{j} = \mathbf{u_j} + \Delta\mathbf{u_j}$$

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

$$(4.4) \qquad \Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j}.$$

Proof. See [8].

[2] Robustness of the residual graph can be lower or higher; $S(G)$ decreases due to monotonicity, but the denominator also shrinks to $(s - 1)$ at every step.

Since updating the eigenvalues involves using the eigenvectors, which also change with node removals, we use the following to update the eigenvectors as well.

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

$$\Delta\mathbf{u_j} = \sum_{i=1, i\neq j}^{n} \left(\frac{\mathbf{u_i}'\Delta\mathbf{A}\mathbf{u_j}}{\lambda_j - \lambda_i} \mathbf{u_i} \right). \tag{4.5}$$

Proof. See [8].

4.1.2 Node selection for removal By using LEMMA 4.1, we can write the effect of perturbing $\mathbf{A}$ with the removal of a node i on the eigenvalues as

$$\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j} = -2\mathbf{u_{ij}} \sum_{v\in\mathcal{N}(i)} \mathbf{u_{vj}} \tag{4.6}$$

where $\Delta\mathbf{A}(i,v) = \Delta\mathbf{A}(v,i) = -1$, for $v \in \mathcal{N}(i)$, and 0 elsewhere, and $\mathcal{N}(i)$ denotes the set of neighbors of i. Thus, at each step we choose the node $i \in V$ that maximizes

$$e^{\lambda_1}\left(e^{-2\mathbf{u_{i1}}\sum_{v\in\mathcal{N}(i)}\mathbf{u_{v1}}} + \ldots + c_{n-1}e^{-2\mathbf{u_{in-1}}\sum_{v\in\mathcal{N}(i)}\mathbf{u_{vn-1}}} \right) \tag{4.7}$$

We remark that it is infeasible to compute all the n eigenvalues of a graph with n nodes, for very large n. Thanks to the skewed spectrum of real-world graphs [16], we can rely on the observation that only the top few eigenvalues have large magnitudes. This implies that the c_j terms in Equ. (4.3) and also Equ. (4.7) become much smaller for increasing j and can be ignored. Therefore, we use the top t eigenvalues to approximate the robustness of a graph. In the past, the skewed property of the spectrum has also been exploited to approximate triangle counts in large graphs [38].

The outline of the GREEDYRLS algorithm, its complexity analysis, and its adaptations for the RLS-PROBLEM variants (§3.3) can be found in [8].

4.2 Greedy Randomized Adaptive Search Procedure (GRASP) Approach The top-down approach makes a greedy decision at every step. If the desired subgraphs are small, however, this incurs many greedy decisions, especially on large graphs where the number of greedy steps $(n - s)$ would be excessive. Since the RLS-PROBLEM does not exhibit monotonicity or semi-heredity properties (§3.2), taking large number of greedy steps can yield poor performance. Therefore, we propose a bottom-up approach that performs local operations to build up solutions from scratch.

Our local approach is based on a meta-heuristic called GRASP [17] for solving combinatorial optimization problems. A GRASP, or greedy randomized adaptive search procedure, is a multi-start or iterative process, in which each iteration consists of two phases: (i) a construction phase, in which an initial feasible solution is produced, and (ii) a local search phase, in which a better solution with higher objective value in the neighborhood of the constructed solution is sought. The best overall solution becomes the final result.

The pseudo-code in Algorithm 1 shows the general GRASP for maximization, where $T_{\max}$ iterations are done. For maximizing our objective, we use $f : S \rightarrow \mathbb{R} \equiv \bar{\lambda}$, i.e., the robustness function as given in Equ. (2.1). We next describe the details of our two GRASP phases.

Algorithm 1 GRASP-RLS

```
Input: Graph $G = (V, E)$, $T_{\max}$, $f(\cdot)$, $g(\cdot)$, integer $s$
Output: Subset of nodes $S^* \subseteq V$, $|S^*| = s$
$f^* = -\infty$, $S^* = \emptyset$
for $z = 1, 2, \ldots, T_{\max}$ do
    $S \leftarrow$ GRASP-RLS-CONSTRUCTION($G$, $g(\cdot)$, $s$)
    $S' \leftarrow$ GRASP-RLS-LOCALSEARCH($G$, $S$, $f(\cdot)$, $s$)
    if $f(S') > f^*$ then $S^* \leftarrow S$, $f^* = f(S)$
end for
return $S^*$
```

4.2.1 Construction In the construction phase, a feasible seed solution is iteratively constructed, one node at a time. At each iteration, the choice of the next node to be added is determined by ordering all candidate nodes C in a restricted candidate list, called RCL, with respect to a greedy function $g : C \rightarrow \mathbb{R}$, and randomly choosing one of the candidates in the list. Candidate set in the first iteration is set to V and in later iterations it contains the nodes in the neighborhood $\mathcal{N}(S)$ of the current solution S. The size of RCL is determined by a real parameter $\beta \in [0, 1]$, which controls the amount of greediness and randomness. $\beta = 1$ corresponds to a purely greedy construction, while $\beta = 0$ produces a purely random one. Algorithm 2 describes our construction phase.

Algorithm 2 GRASP-RLS-CONSTRUCTION

```
Input: Graph $G = (V, E)$, $g(\cdot)$, integer $s$
Output: Subset of nodes $S \subseteq V$
$S \leftarrow \emptyset, C \leftarrow V$
while $|S| < s$ do
    Evaluate $g(v)$ for all $v \in C$
    $\bar{c} \leftarrow \max_{v\in C} g(v)$, $\underline{c} \leftarrow \min_{v\in C} g(v)$
    Select $\beta \in [0, 1]$ using a strategy
    $RCL \leftarrow \{v \in C | g(v) \geq \underline{c} + \beta(\bar{c} - \underline{c})\}$
    Select a vertex $r$ from $RCL$ at random
    $S := S \cup \{r\}$, $C \leftarrow \mathcal{N}(S) \backslash S$
end while
return $S$
```

Selecting $\mathbf{g}(\cdot)$: We aim to include locally dense nodes in our seed solutions. Therefore, in the first iteration of the construction we use $g(v) = \frac{t(v)}{d(v)}$, where $t(v)$ denotes the number of local triangles of v, and $d(v)$ is its degree. Initially the candidate set C is equal to the node set V, thus we approximate the local triangle counts for speed [38]. In later iterations we use $g(v) = \Delta\bar{\lambda}_v$; the difference in robustness when a candidate node is added to the current subgraph.

Selecting $\boldsymbol{\beta}$: Setting $\beta = 1$ is purely greedy and produces the same seed subgraph in every GRASP iteration. To incorporate randomness while staying close to the greedy best-first selection, we choose $\beta \in [0.8, 1]$ uniformly at random at every step. This produces high quality solutions in the presence of large variance in constructed solutions [17].

4.2.2 Local Search A solution generated by GRASP-RLS-CONSTRUCTION is a preliminary one and may not necessarily have the best robustness. Thus, it is almost always beneficial to apply a local refinement procedure to each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution with a better one in the neighborhood of the current solution. It terminates when no better solution can be found. We describe our local search phase in Algorithm 3.

As the RLS-PROBLEM asks for a subgraph of size s, the local search takes as input an s-subgraph generated by construction and searches for a better solution around it by "swapping" nodes in and out. Ultimately it finds a locally optimal subgraph of size upper bounded by (s+1). As an answer, it returns the best s-subgraph with the highest robustness found over the iterations. As such, GRASP-RLS-LOCALSEARCH is an adaptation of a general local search procedure to yield subgraphs of desired size.

Algorithm 3 GRASP-RLS-LOCALSEARCH

```
Input: Graph $G = (V, E)$, $S$, integer $s$
Output: Subset of nodes $S' \subseteq V$, $|S'| = s$
$more \leftarrow$ TRUE, $S' \leftarrow S$
while $more$ do
    if $\exists v \in S$ such that $\bar{\lambda}(S \backslash \{v\}) \geq \bar{\lambda}(S)$ then
        $S := S \backslash \{v^*\}$ s.t. $v^* := \max_{v \in \mathcal{N}(S) \backslash S} \bar{\lambda}(S \backslash \{v\})$
        if $|S| = s$ then $S' \leftarrow S$ end if
    else
        $more \leftarrow$ FALSE
    end if
    $add \leftarrow$ TRUE
    while $add$ and $|S| \leq s$ do
        if $\exists v \in \mathcal{N}(S) \backslash S$ s.t. $\bar{\lambda}(S \cup \{v\}) > \bar{\lambda}(S)$ then
            $S := S \cup \{v^*\}$, $v^* := \max_{v \in \mathcal{N}(S) \backslash S} \bar{\lambda}(S \cup \{v\})$
            $more \leftarrow$ TRUE
            if $|S| = s$ then $S' \leftarrow S$ end if
        else
            $add \leftarrow$ FALSE
        end if
    end while
end while
return $S'$
```

The local search is guaranteed to terminate, as the objective value (i.e., subgraph robustness) improves with every iteration and it is upper-bounded by the robustness of the $(s + 1)$-clique. We provide the complexity analysis and the GRASP-RLS algorithm variants in [8].

Table 1: Real-world graphs. δ: edge density, $\bar{\lambda}$: robustness

Dataset	$n = \lvert V \rvert$	$m = \lvert E \rvert$	δ	$\bar{\lambda}$
Jazz	198	2742	0.1406	34.74
Celegans N.	297	2148	0.0489	21.32
Email	1133	5451	0.0085	13.74
Oregon-A	7352	15665	0.0005	42.29
Oregon-B	10860	23409	0.0004	47.54
Oregon-C	13947	30584	0.0003	52.10
P2P-GnutellaA	6301	20777	0.0010	19.62
P2P-GnutellaB	8114	26013	0.0008	19.45
P2P-GnutellaC	8717	31525	0.0008	13.35
P2P-GnutellaD	8846	31839	0.0008	14.46
P2P-GnutellaE	10876	39994	0.0007	7.83
DBLP	317080	1049866	2.09×10^{-5}	103.18
Web-Google	875713	4322051	1.13×10^{-5}	99.36

5 Evaluation

We evaluate our methods extensively on numerous synthetic and real-world graphs. Our real graphs, as in Table 1, come from various domains, including biological, email, Internet AS backbone, P2P, collaboration, and the Web.

Our work is in the general lines of subgraph mining, however with a new objective based on robustness. The closest to our setting is the densest subgraph mining. Therefore, we compare our results to dense subgraphs found by Charikar's algorithm [9] (which we refer to as Charikar), as well as by Tsourakakis *et al.*'s two algorithms [39] (which we refer to as Greedy and LS for local search following the convention in their work). We remark that the objectives used in those works are distinct; namely, average degree and edge-surplus, respectively, and are also different from ours.

We first evaluate the accuracy of the algorithms against ground truth. To do so, we create synthetic graphs and inject a clique in each graph. Note that a clique is both the densest and the most robust subgraph of a certain size. Therefore, the algorithms are compared on the same grounds.

Table 2 provides precision, recall, and subgraph size $|S|$ averaged over ten Erdős-Rényi random graphs, with $n = 3000$ nodes and $p = \{0.5, 0.1, 0.008\}$, in which we inject a clique of size 30. p's are selected to capture very dense, medium-dense, and sparse graphs. We notice that while all methods perform sufficiently well for sparse graphs with $p = 0.008$, accuracy of GRASP-RLS is superior to the competing methods at all densities.

We also compare the algorithms on the Chung-Lu random power-law graphs [10], with $n = 3000$ and power-law exponent varying from 2.2 to 3.1 as observed in real graphs (larger exponent implies a sparser graph), in which we inject a clique of 15 nodes. We run the LS algorithm seeded with one of the nodes of the clique as previously done in [39], while GRASP-RLS is not favored by such a selection. Precision and recall curves averaged over ten graphs are given in Figure 3. We note that while the accuracies of all methods improve by the increasing exponent as the task becomes eas-

Table 2: $\mathcal{P}$recision& $\mathcal{R}$ecall (avg.) for our GRASP-RLS& GREEDYRLS, Charikar [9], Greedy & LS [39] on ten ER graphs.

ER parameters		GRASP-RLS		GREEDYRLS		Charikar [9]			Greedy [39]			Local Search [39]		
n	p	$\lvert S\rvert$	$P=R$	$\lvert S\rvert$	$P=R$	$\lvert S\rvert$	P	R	$\lvert S\rvert$	P	R	$\lvert S\rvert$	P	R
3000	0.5	30	0.97	30	0.02	3000	0.01	1	3000	0.01	1	3000	0.01	1
3000	0.1	30	1	30	0.95	3000	0.01	1	29.60	0.99	0.97	20.63	0.37	0.35
3000	0.008	30	1	30	0.99	30	1	1	30	1	1	28.23	0.94	0.93

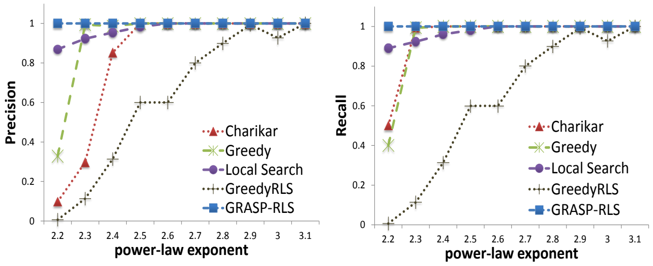


Figure 3: Precision & Recall for our GRASP-RLS, Charikar [9], Greedy & Local Search [39] vs. exponent of power-law graphs.

ier, GRASP-RLS remains superior with robust performance at all host graph densities.

Cliques are both the densest and the most robust subgraphs, however, it is expected that the algorithms will find different subgraphs in general due to their distinct objectives. To understand their differences, we turn to real world graphs and compare the robust and dense subgraphs based on three main criteria: (a) robustness $\bar{\lambda}$ as in Equ. (2.1), (b) triangle density $t[S]/\binom{|S|}{3}$, and (c) edge density $e[S]/\binom{|S|}{2}$.

Table 3 shows results on our largest graphs from each category. Note that the three algorithms we compare to try to find the densest subgraph without a size restriction. Thus, each one obtains a subgraph of a different size. To make the robust subgraphs (RS) comparable to the densest subgraphs (DS), we find subgraphs of size s equal to the ones found by Charikar, Greedy, and LS, respectively noted as s_{Ch}, s_{Gr}, and s_{Ls}. As such, we compare to the *best* results achieved by each of the densest subgraph algorithms.

We notice that densest subgraphs found by Greedy and LS are often substantially smaller than those found by Charikar, and also have higher edge density, which is the same observation as in [39]. On the other hand, robust subgraphs have higher robustness than densest subgraphs, *even* at lower densities. This shows that high density does not always imply high robustness, and vice versa, illustrating the differences in the two problem settings.

Thus far, we also note that GRASP-RLS consistently outperforms GREEDYRLS, suggesting that the proposed bottom-up search is superior to the greedy top-down search.

Figure 4 shows the relative difference in robustness of GRASP-RLS subgraphs over again, the best results obtained by Charikar, Greedy, and LS on all of our real graphs. We achieve a wide range of improvements depending on the graph, where the difference is always positive. The improvements with respect to the LS results are the most pronounced.

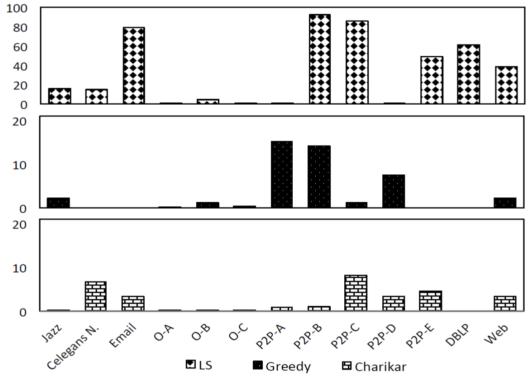


Figure 4: Robustness improvement (%) of GRASP-RLS over (top to bottom) best LS, Greedy, and Charikar results.

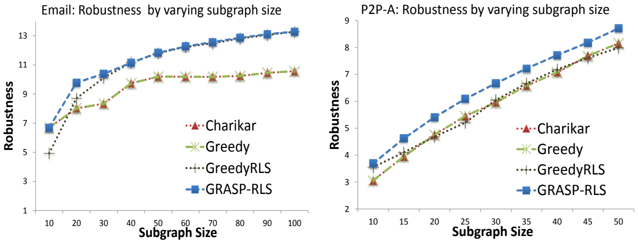


Figure 5: Subgraph robustness at varying subgraph sizes s.

Comparisons in Table 3 and Figure 4 are for subgraphs at sizes where best results are obtained for each of the three densest subgraph algorithms. Our algorithms, on the other hand, accept a subgraph size input s. Thus, we next compare the competing methods at varying output sizes. Charikar and Greedy are both top-down methods, in which the lowest degree node is removed at each step and the best subgraph (best average degree or edge surplus, respectively) is output among all graphs created along the way. We modify these so that we pull out the subgraphs when size s is reached during the course of their run.[3] Figure 5 shows that our GRASP-RLS produces subgraphs with higher robustness at varying sizes on two example graphs (similar results on others). This also shows that the densest subgraph approaches are not directly applicable to our problem.

Experiments thus far illustrate that we find subgraphs with robustness higher than the densest subgraphs. These are relative results. To show that the subgraphs we find are in fact robust, we next quantify the magnitude of the robustness values we achieve through significance tests.

[3] Local search by [39] finds locally optimal subgraphs, which are not guaranteed to grow to a given size s. Thus, we omit comparison to LS subgraphs at varying sizes. Figure 4 shows that improvements over LS subgraphs are already substantially large.

Table 3: Comparison of robust and densest subgraphs. Ch: Charikar [9], Gr: Greedy [39], Ls: Local search [39].

Data	Method	robustness $\lambda[S]$			triangle density $\Delta[S]$			edge density $\delta[S]$		
	(s_{Ch}, s_{Gr}, s_{Ls})	Ch	Gr	Ls	Ch	Gr	Ls	Ch	Gr	Ls
Email	DS (271, 12, 13)	13.58	8.51	4.96	**0.0009**	**1.0000**	0.2237	**0.0600**	**1.0000**	0.5897
	RS-Greedy	13.94	5.96	6.27	0.0001	0.0696	0.0606	0.0523	0.7576	0.7179
	RS-GRASP	**14.04**	**8.52**	**8.91**	0.0007	**1.0000**	**0.8671**	0.0508	**1.0000**	**0.9487**
Oreg-C	DS (87, 61, 52)	34.44	30.01	27.69	0.0868	0.1768	0.2327	**0.3892**	**0.5311**	0.5927
	RS-Greedy	34.31	24.70	21.75	0.0857	0.1022	0.1193	0.3855	0.4131	0.4367
	RS-GRASP	**34.47**	**30.14**	**28.01**	**0.0870**	**0.1775**	**0.2375**	0.3884	0.5301	**0.5943**
P2P-E	DS (386, 22, 4)	8.81	6.40	0.86	**9.77E-06**	0.0	0.0	**0.0306**	**0.4372**	0.6667
	RS-Greedy	9.10	5.22	0.86	6.83E-06	0.0	0.0	0.0267	0.3593	0.6667
	RS-GRASP	**9.22**	**6.41**	**1.29**	6.93E-06	0.0	**0.5**	0.0270	**0.4372**	**0.8333**
Web	DS (240, 105, 18)	52.15	47.62	10.20	**0.0266**	**0.2160**	0.4178	**0.2274**	**0.4759**	0.7254
	RS-Greedy	41.57	22.56	8.69	0.0027	0.0082	0.2525	0.0710	0.1225	0.6144
	RS-GRASP	**53.96**	**48.68**	**14.11**	0.0153	0.1246	**1.0000**	0.1296	0.3996	**1.0000**

Given a subgraph that GRASP-RLS finds, we bootstrap $B = 1000$ new subgraphs by rewiring its edges at random. We compute an empirical p-value for each subgraph by dividing the number of randomly rewired subgraphs that have larger robustness by B. The p-value essentially captures the probability that we would be able to obtain a subgraph with robustness greater than what we find by chance if we were to create a topology with the same number of nodes and edges at random (note that all such subgraphs would have the same edge density). Thus a low p-value implies that, among the same density topologies, the one we find is in fact robust with high probability.

Figure 6 shows that the subgraphs we find on almost all real graphs are significantly robust at 0.05. For cases with large p-values, it is possible to obtain higher robustness subgraphs with rewiring. For example, *P2P-E* is a graph where all the robust subgraphs (also the dense subgraphs) found contain very few or no triangles (see Table 3). Therefore, rewiring edges that short-cut longer cycles they contain help improve their robustness. We remark that large p-values indicate that the found subgraphs are not significantly robust, but does not imply our algorithms are unable to find robust subgraphs. That is because the rewired more robust subgraphs do not necessarily exist in the original graph G, and it is likely that G does not contain any subgraph with robustness that is statistically significant.

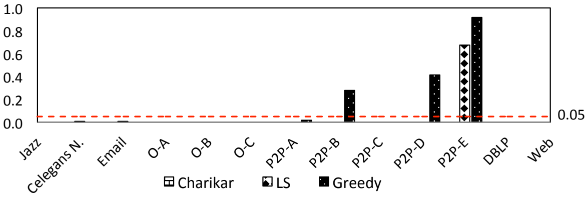


Figure 6: p-values of significance tests indicate that GRASP-RLS subgraphs have significantly large robustness.

Next we analyze the performance of our GRASP-RLS approach in more detail. Recall that GRASP-RLS-Construction quickly builds a subgraph which GRASP-RLS-LocalSearch uses to improve over to obtain a better result. In Figure 7 we show the robustness of subgraphs obtained at construction and after local search on two example graphs for $s = 50$ and $T_{\max} = 300$. We notice that most of the GRASP-RLS iterations find a high robustness subgraph right at construction. In most other cases, local search is able to improve over construction results significantly. In fact, the most robust outcome on *Oregon-A* (Figure 7 left) is obtained when construction builds a subgraph with robustness around $\bar{\lambda} = 6$, which the local search improves over $\bar{\lambda} = 20$.

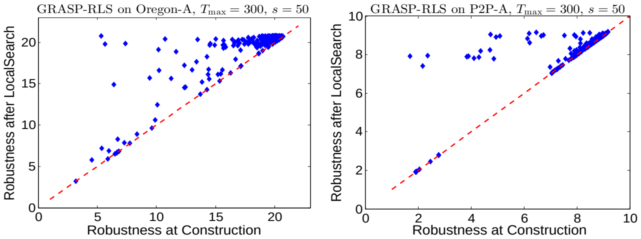
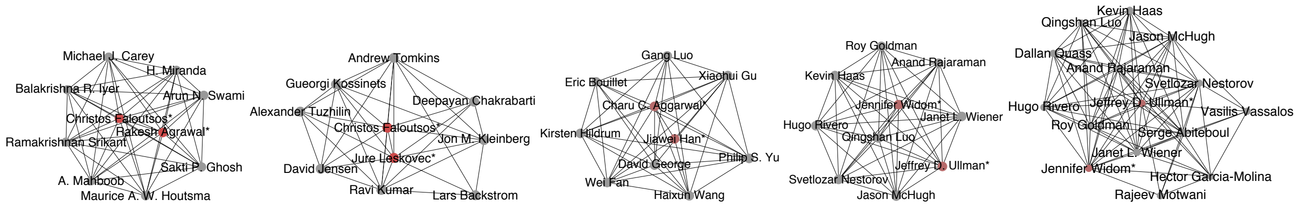


Figure 7: $\bar{\lambda}$ achieved at GRASP-RLS-Construction versus after GRASP-RLS-LocalSearch.

We next perform several case studies on the DBLP co-authorship network to qualitatively analyze our subgraphs. Here, we use the seeded variant of our problem (variant (iii) in §3.3). Christos Faloutsos is a prolific researcher with various interests. In Figure 8 (a), we invoke his interest in databases when used with Rakesh Agrawal as seeds, as Agrawal is an expert in this field. Later in (b), we invoke his interest in data mining when we use Jure Leskovec as the second seed, who is a rising star in the field. Likewise in (c) and (d) we find robust subgraphs around other selected prominent researchers in data mining and databases. In (d) we show how our subgraphs change with varying size. Specifically, we find a clique that the seeds J. Widom and J. Ullman belong to, for s=10. The subgraph of s=15, while no longer a clique, remains stable in which other researchers like R. Motwani and H. Garcia-Molina are included.



(a) {C. Faloutsos, R. Agrawal} δ=1, Δ=1, $\bar{\lambda}$=6.7

(b) {C. Faloutsos, J. Leskovec} δ=0.78, Δ=0.51, $\bar{\lambda}$=5.0

(c) {J. Han, C. C. Aggarwal} δ=0.91, Δ=0.78, $\bar{\lambda}$=6.0

(d) {J. Widom, J. D. Ullman} δ=1, Δ=1, $\bar{\lambda}$=6.7 δ=0.8, Δ=0.58, $\bar{\lambda}$=9.0

Figure 8: Robust DBLP subgraphs returned by our GRASP-RLS algorithm when seeded with authors indicated in (a)-(d).

Given the local search characteristics of GRASP-RLS, its complexity is linear in host graph size, as we theoretically show in [8]. Figure 9 also illustrates the linear scalability w.r.t. input graph size empirically.[4]

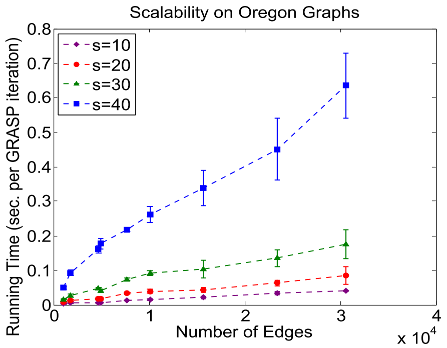


Figure 9: Scalability of GRASP-RLS by graph size m and subgraph size s (run time avg'ed over 10 runs, bars: 25%-75%).

6 Related Work

The work by Albert *et al.* showed that scale-free graphs are robust to random failures but vulnerable to intentional carefully-planned attacks [1]. This observation has stimulated studies on the response of networks to various attack strategies [6, 7, 11, 14, 24, 36]. Other works look at how to design networks that are optimal with respect to some survivability criteria [18, 21, 31, 29]. A vast body of these works focuses on global robustness of graphs at large.

With respect to research on local robustness, Trajanovski *et al.* aim to spot critical regions in a graph the destruction of which would cause the biggest harm to the network [37]. Similar works aim to identify the critical nodes and links of a network [19, 25, 32, 34]. These works try to spot vulnerability points in the network, whereas our objective is somewhat orthogonal: identify robust regions. Closest to ours, Andersen *et al.* consider spectral radius as an objective criterion and propose algorithms for identifying small robust subgraphs with large spectral radius [2].

While having major distinctions as we illustrated in this work, robust subgraphs are related to dense subgraphs, which have been studied extensively. Finding the largest clique in a graph, well-known to be **NP**-complete [20], is also shown to be hard to approximate [23].

A relaxation of the clique problem is the densest subgraph problem. Goldberg [22] and Charikar [9] designed exact poly-time and $\frac{1}{2}$-approximate linear-time solutions to this problem, respectively, where density is defined as the average degree. This problem is shown to become **NP**-hard when the size of the subgraph is restricted [3]. Most recently, Tsourakakis *et al.* [39] also proposed fast heuristic solutions, where they define density as edge surplus; the difference between number of edges and α fraction of maximum edges, for user-specified constant $\alpha > 0$. Likewise, Pei *et al.* study detecting quasi-cliques in multi-graphs [30]. Other definitions include k-cores, k-plexes, and k-clubs, etc. [26].

Dense subgraph discovery is related to finding clusters in graphs, however with major distinctions. Most importantly, dense subgraph discovery has to do with absolute density where there exists a preset threshold for what is sufficiently dense. On the other hand, graph clustering concerns with relative density measures where density of one region is compared to another. Moreover, not all clustering objectives are based on density and not all types of dense subgraphs can be found by clustering algorithms [26].

In summary, while similarities among them exist, discovery of critical regions, robust subgraphs, cliques, densest subgraphs, and clusters are substantially distinct graph mining problems, for which different algorithms can be applied. To the best of our knowledge, our work is the first to consider identifying robust local subgraphs in large graphs.

7 Conclusion

We introduced the RLS-Problem of finding the most robust local subgraph of a given size in large graphs, as well as its three practical variants. While our work bears similarity to densest subgraph mining, it differs from it in its objective; robustness emphasizes subgraph topology more than edge density. We showed that our problem is **NP**-hard and that it does not exhibit semi-heredity or subgraph monotonicity properties. We designed two heuristic algorithms based on top-down and bottom-up search strategies, and showed how we can adapt them to address the problem variants. We found that our bottom-up strategy provides consistently superior results, scales linearly with input graph size, and finds subgraphs with significant robustness. Experiments on

[4] We use nine *Oregon* graphs with various sizes [7], the largest three of which are listed in Table 1. Running time is averaged over $T_{\max}$ iterations as one can run each pair of construction followed by local search phases completely in parallel.

synthetic and real graphs showed that our subgraphs are of higher robustness than densest subgraphs even at lower densities, which illustrates the novelty of our problem setting.

Our research sets off several future directions, including the hardness analysis for the RGS-PROBLEM, exploration of new robustness measures with desirable properties, and the design of efficient algorithms for those new objectives.

Acknowledgments This material is based upon work supported by the ARO Young Investigator Program with Contract No. W911NF-14-1-0029, NSF CAREER 1452425, NSF IIS-1017181, and a gift from Northrop Grumman. Any findings and conclusions expressed in this material are of the authors' 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] R. Andersen and S. M. Cioaba. Spectral densest subgraph and independence number of a graph. *J. UCS*, 13(11), 2007.

[3] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Disc. Appl. Math.*, 121(1-3):15–26, 2002.

[4] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2), 2000.

[5] 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.

[6] 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.

[7] H. Chan, L. Akoglu, and H. Tong. Make it or break it: Manipulating robustness in large networks. In *SDM*, 2014.

[8] H. Chan, S. Han, and L. Akoglu. Where graph topology matters: The robust subgraph problem. *CoRR*, abs/1501.01939, 2015. http://arxiv.org/abs/1501.01939.

[9] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.

[10] F. R. K. Chung and L. Lu. The average distance in a random graph with given expected degrees. 1(1), 2003.

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

[12] W. Ellens and R. E. Kooij. Graph measures and network robustness. *CoRR*, abs/1311.5064, 2013.

[13] E. Estrada. Characterization of the folding degree of proteins. *Bioinformatics*, 18(5):697–704, 2002.

[14] E. Estrada. Network robustness to targeted attacks: The interplay of expansibility and degree distribution. *The Euro. Phys. J. B*, 52(4):563–574, 2006.

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

[16] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, 1999.

[17] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *J. of Optimization*, 6:109–133, 1995.

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

[19] T. Fujimura and H. Miwa. Critical links detection to maintain small diameter against link failures. In *INCoS*, 2010.

[20] M. Garey and D. Johnson. *Computers and Intractability - A guide to the Theory of NP-Completeness*. Freeman, 1979.

[21] W. K. Ghamry and K. M. F. Elsayed. Network design methods for mitigation of intentional attacks in scale-free networks. *Telecom. Systems*, 49(3):313–327, 2012.

[22] A. V. Goldberg. Finding a maximum density subgraph. Technical Report CSD-84-171, UC Berkeley, 1984.

[23] J. Hastad. Clique is hard to approximate within $n^{(1-\epsilon)}$. In *FOCS*, pages 627–636. IEEE Computer Society, 1996.

[24] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Phy. R. E*, 65(5), 2002.

[25] V. Latora and M. Marchiori. Vulnerability and protection of infrastructure networks. *Phys. Rev. E*, 71:015103, 2005.

[26] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Springer, 2010.

[27] 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.

[28] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1-2):244–257, 2013.

[29] G. Paul, T. Tanizawa, S. Havlin, and H. Stanley. Optimization of robustness of complex networks. *The Eur. Phys. J. B*, 38(2):187–191, 2004.

[30] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *KDD*, pages 228–238, 2005.

[31] 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.

[32] Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Trans. Netw.*, 21(3), 2013.

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

[34] M. D. Summa, A. Grosso, and M. Locatelli. Branch and cut algorithms for detecting critical nodes in undirected graphs. *Comp. Opt. and Appl.*, 53(3):649–680, 2012.

[35] A. Sydney, C. M. Scoglio, and D. Gruenbacher. Optimizing algebraic connectivity by edge rewiring. *Applied Mathematics and Computation*, 219(10), 2013.

[36] 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, 2012.

[37] S. Trajanovski, F. A. Kuipers, and P. V. Mieghem. Finding critical regions in a network. In *INFOCOM*, 2013.

[38] C. E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *ICDM*, pages 608–617, 2008.

[39] C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. A. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *KDD*, 2013.

[40] D. Watts. Security and vulnerability in electric power systems. *35th N. American Power Symp.*, pages 559–566, 2003.

[41] J. Wu, B. Mauricio, Y.-J. Tan, and H.-Z. Deng. Natural connectivity of complex networks. *Ch. Phy. Let.*, 27(7), 2010.

[42] A. Zeng and W. Liu. Enhancing network robustness for malicious attacks. *CoRR*, abs/1203.2982, 2012.