Interactive Visualization of Robustness Enhancement in Scale-free Networks with Limited Edge Addition (RENEA)

Armita Abedijaberi[1], Nathan Eloe[2], and Jennifer Leopold[1]

[1]Department of Computer Science, Missouri University of Science and Technology, Rolla, MO USA
Email: {*aan87, leopoldj*}*@mst.com*
[2]School of Computer Science and Information Systems, Northwest Missouri State University, Maryville, MO USA
Email: *nathane@nwmissouri.edu*

Abstract

Error tolerance and attack vulnerability of scale-free networks are usually used to evaluate the robustness of these networks. While new forms of attacks are developed everyday to compromise infrastructures, service providers are expected to develop strategies to mitigate the risk of extreme failures. Recently, much work has been devoted to design networks with optimal robustness, whereas little attention has been paid to improve the robustness of existing ones. Herein we present RENEA, a method to improve the robustness of a scale-free network by adding a limited number of edges. While adding an edge to a network is an expensive task, our system, during each iteration, allows the user to select the best option based on the cost, amongst all proposed ones. The edge-addition interactions are performed through a visual user interface while the algorithm is running. RENEA is designed based on the evolution of the network's largest component during a sequence of targeted attacks. Through experiments on synthetic and real-life data sets, we conclude that applying RENEA on a scale-free network while interacting with the user can significantly improve its attack survivability at the lowest cost.

1 Introduction

One of the most important features of large networks is their degree distribution, $P(k)$, or the probability that an arbitrary node is connected to exactly k other nodes. Many real-life networks display a power-law degree distribution with heavy-tailed statistics, which are called scale-free networks. In a scale-free network the probability that a node has k links follows $P(k) \sim k^{(-\lambda)}$, where λ is called the degree exponent and its value is typically in the range between $2 < \lambda < 3$ [1]. Scale-free networks are created by preferential attachment [2], which means newly introduced nodes prefer to connect to existing high-degree nodes. Starting with a small number of nodes, when a new node is added to the network, considering the preferential linking, it will connect to other nodes with the probability proportional to their degree. Coupled with the expanding nature of many networks this explains the occurrence of hubs, which hold a much higher number of links than most of the nodes in the network. Scale-free topology is widely observed in many communication and transportation systems, such as the Internet, World Wide Web, airline networks, wireless sensor networks, and power supply networks, all of which are essential to modern society. One of the most important properties in scale-free networks is the fact that while these networks are strongly tolerant against random failures, they are fragile under intentional attacks on the hubs. Intuition tells us that disabling a substantial number of nodes/edges will result in an inevitable functional disintegration of a network by breaking the network into tiny, non-communicating islands of nodes. However, scale-free networks can be amazingly resilient against accidental failures; even if 80% of randomly selected nodes fail, the remaining 20% still form a compact cluster with a path connecting any two nodes [2]. In fact, the fragileness of scale-free networks

DOI reference number: 10.18293/DMSVLSS2017-015

under intentional attack comes from their heavy-tailed property, causing loss of a large number of links when a hub node is crashed. Hence, the heavy loss of network links quickly makes the network sparsely connected and subsequently fragmented. However, random failures affect mainly the numerous small degree nodes, the absence of which doesn't disrupt the network's integrity.

Considering error tolerance and attack vulnerability, which are two common and important properties of scale-free networks, extensive research efforts have been made to study the robustness of such networks which is defined as the ability of the surviving nodes to remain, as much as possible, interconnected. On that account it is important to understand how to design networks which are optimally robust against malicious attacks, with examples of terrorist attacks on physical networks or attacks by hackers on computer networks. However, it is not possible to abandon the existing networks, which are the result of years of evolution, and rebuild them from the beginning. Hence, it is significantly important to study the optimizing guidelines to enhance the robustness of existing networks in an interactive environment and incorporate the user's input in order to acquire the optimal result at a lower cost.

In this paper, we study the problem of how to improve the robustness of an existing scale-free network and show that its attack survivability can be significantly improved by adding a limited number of edges to it at the lowest cost. This process is carried out by visualizing the process and interacting with the user. There will be no impact on the error tolerance, keeping the degree distribution of the network as much as possible intact.

The organization of this paper is as follows; Section II provides an overview of related work about robustness enhancement in scale-free networks. In Section III we discuss RENEA and its graphical user interface. An example of running RENEA is presented in Section IV. Section V focuses on experiments and results. Finally, conclusions and our plans for future work are presented in Section VI.

2 Related Work

The main approaches to improve robustness of scale-free networks, through topology reconfiguration, can be generally classified into two main categories. The first category involves rewiring edges of the network and the second category suggests addition of edges to the network. Both approaches are carried out in order to obtain a network structure with better robustness. In this section we summarize existing works regarding these two approaches as well as network visualization tools.

2.1 Reconfiguration with Edge Rewiring

In the method proposed in [3], edges are selected randomly to be removed from the graph or to be added to the graph. However, this reconfiguration will change the degree distribution of the graph as well as its diameter, which is not desirable for the network. Another edge rewiring method is proposed in [4], in which two edges A-B and C-D are selected. If A-C and B-D do not already exist in the network, the rewiring operation replaces A-B and C-D with A-C and B-D as long as such reconfiguration does not generate a loop. This rewiring operation obviously does not change any nodal degree since it converts a random network into another one with the same degree distribution. However, for real networks with economic constraints, the nodal degree conservation is not enough due to the cost. In fact, the total length of links cannot be exceedingly large and the number of changes in the network should remain small. In order to minimize the cost, any swap is accepted, only if the increase of the robustness is greater than or equal to a threshold. This procedure is repeated with another randomly chosen pair of edges until no further substantial improvement is achieved. This method results in a network with an onion-like structure [5].

Even though swapping the edges can increase the network robustness, there are some spatially limited real-life networks where edges are hard to re-configure, e.g. power grid networks and the Internet router network. On the other hand, however, there exist the spatial unlimited networks such as airline networks and the Internet switcher network. For spatially unlimited networks whose edges can be easily re-linked, in the Switch Link (SL) method [6], the top P_c fraction of the large-degree nodes are defined as hubs. For each hub, the SL finds two non-hub nodes connecting it. The edge connected to the first non-hub node is kept and the edge connecting the second non-hub node to the first non-hub node is switched. This process will be repeated until all the links connected to the hub nodes are addressed. For the spatially limited networks, the SL method is not economic and feasible since nodes are usually far from each other. In this case, the split hub (SH) method is proposed [6]. This method also starts with defining the top P_c fraction of nodes as hubs and replaces them by a 3-clique which is a complete graph in which every two distinct nodes are adjacent. Then it connects the non-hub nodes, that were connected to the original hub node, to the nodes of the clique randomly.

Edge reconfiguration can change community structure of a network [7]. The community structure refers to the functional modules in the network that play an important role in regards to cascading failures. A network with a strong community structure has few edges between its communities. Hence, its structure is more fragile in terms

of attacks on those edges in comparison with networks with a weak community structure. A method [7] is proposed to improve the robustness of a network while preserving its community structure. In this 3-step method, the importance of nodes is calculated based on their degree. Step 1 reconfigures each community to have an onion-like structure [5]. Step 2 swaps edges in such a way that important nodes only connect to the nodes within the same community. Step 3 swaps edges to increase the number of edges between communities. These 3 steps are recursively applied on the network until its robustness hardly increases.

2.2 Reconfiguration with Edge Addition

The number of possible ways of adding an edge to a graph with N nodes and L edges is equal to $\binom{N}{2} - L$. For large real-life (sparse) networks, it is almost infeasible to compare all these possibilities and find the optimal edges to add. Some techniques have been introduced in literature to add edges to an existing graph based on different criteria. Three different enforcing strategies are practiced in [8]. The first method randomly selects a pair of nodes in the network and establishes a new edge between them. The second method selects a pair of nodes with the lowest degree in the network and establishes a new edge between them. Finally, the last method selects a pair of nodes with the highest degree in the network and establishes a new edge between them. According to experiments, the method that prefers low-degree nodes as candidates for adding edges reinforces the attack survivability of the network with a lower cost in comparison with the other methods.

It has been suggested [9] that assortative networks (i.e., high-degree nodes in the network that are more likely linked with other high-degree nodes), are more robust than their disassortative counterparts (i.e., high-degree nodes in the network that are more likely linked with low-degree nodes). Thereupon, a method is proposed [9] to enhance robustness of a network by increasing its assortativity. In this method, a layer index is assigned to each node based on the degree of that node. Accordingly, the layer index for nodes with the lowest degree is 0, for nodes with the second lowest degree is 1, and so on. Then, the probability of adding an edge between a random pair of nodes depends on their layer index difference (i.e., nodes within the same layer are connected with greater probabilities than nodes in different layers); this leads to higher assortativity in the network.

The way that nodes are arranged in a graph is considered as a key factor in overall robustness and efficiency of that graph [10]. The star structure is efficient (the average shortest path length is small), yet fragile in case of removal of the central node. On the contrary, the circle structure is robust with regard to removal of any single node, yet inefficient (the average shortest path length is large). The Node-protecting Cycle method [11] is proposed to combine the properties of both circle and star structures to improve robustness and efficiency of a network.

All of the aforementioned edge rewiring and edge addition algorithms are proposed to improve the robustness of an existing network. However, these works consider one aspect in a network and neglect the others. When it comes to a scale-free network, it is important to take every structural aspect of the network into consideration. In addition, feasibility of a proposed re-configuration of a network must be considered. Having a method to improve the robustness of a scale-free network against malicious attacks while keeping its resilience in case of random failures without disturbing its small-world property at a very low cost is still an unsolved problem. In a scale-free network with the small-world property, the distance between any pairs of nodes is relatively small [12].

2.3 Network Visualization

A number of software tools have been developed to model, analyze, and visualize data that can be represented as a graph or a network. Typically, these tools allow the user to annotate nodes and edges with metadata, and provide utilities such as random graph generation, calculation of analytical measures (e.g., centrality, network distances, PageRank, etc.), and filtering (i.e., viewing only a portion of the graph based on some criteria). Several of these viewers were designed for a particular problem domain, such as finding motifs in gene regulatory networks (e.g., GeneNetWeaver [13]). KDnuggets [14] provides a list of what it considers to be the top 30 network visualization tools that can be used for a wide variety of network applications (e.g., in domains such as biology, finance, and sociology). It is notable that many of these tools are open-source and can be customized to provide additional functionality. The software presented herein differs from other available network visualization tools in the analysis that it facilitates. To the best of our knowledge, there are no other visual network analysis tools available for the purpose of reinforcing robustness of a scale-free network in an interactive environment.

3 Robustness Enhancement Algorithm

The problem setting we consider in this work is to modify a given graph's structure under a given budget to improve its robustness. The budget is often defined in terms of the maximum number of edges that can be added to the network. In practice, the cost of establishing an edge between a pair of nodes is not zero. For real networks, with economical constraints, it is preferable to reduce the overall cost of adding extra edges, keeping the total length

(geographically), as low as possible while still achieving the same amount of robustness enhancement. The measure of robustness employed in our algorithm and the edge-addition strategy along with its graphical user interface are specified in the following sections.

3.1 Robustness Metric

The definition of network robustness might change according to a specific application. In this work, the removal of a node from a network is called a "node-knockout" or a "node-attack", and the robustness of a network is measured by the size of the Largest Connected Component (LCC) in the network after a node-attack [5]. To quantify this, we proceed with a series of node-attacks and subsequently measure the robustness after each node-removal. Hence, the robustness R is defined as:

$$R = \frac{1}{N} \sum_{Q=1}^{N} S(Q)$$

where N is the number of nodes in the network and $S(Q)$ is the fraction of nodes in the LCC after removing Q nodes. The normalization factor $1/N$, makes robustness of networks with different sizes comparable. The minimum value of R is equal to $1/N$, where the network is a star graph and the maximum value of R is equal to 0.5, where the network is a complete graph. A robust network corresponds to a large R value. This distinctive measure is not only simple, but also practical, due to the calculation of the size of the largest component during all possible system-wide failures or intentional attacks.

In our targeted attack scenario, we implicitly admit that the attacker perfectly knows the network's degree sequence and thus can cause maximum damage. An intentional attack is targeted to disrupt the network by removing the most important nodes; here, we find the most connected node, remove it along with all the edges incident with it, calculate $S(Q)$, update the degree sequence for the remaining nodes, and find the new most connected node to repeat the process until the network completely collapses. In case of two or more nodes having the same degree, we simply pick one of them randomly.

3.2 RENEA

Here we present an algorithm called RENEA to improve the robustness of a scale-free network by adding a limited number of edges to it, such that the optimized network, compared to the initial one, has a significantly higher value of robustness. We assume that the 'defender' knows about the intention of the 'attacker' to cause maximum damage to the network.

For scale-free graphs there is always a very large component which is a connected subgraph that contains a constant fraction of the entire graph's nodes. Resultantly, one can randomly remove more than 80% of the nodes in the graph without destroying that component. Hence, the network will still possess large-scale connectivity [15]. On the other hand, an attack that simultaneously eliminates as few as 10–20% of the hubs can cause that component to disappear suddenly and break the network into several isolated components. The main idea in our method is to increase the robustness of a network by adding a limited number of edges to it, and therefore to increase the lifetime of the largest component of the graph upon removing high-degree nodes.

Depending upon the nature of a network, adding an edge can be very costly and some networks can only afford a limited number of them. Moreover, in order to keep other structural properties of networks such as their degree distribution as much intact as possible, it requires addition of a limited number of edges to the network. Given that, a threshold parameter is needed for the algorithm to constrain the maximum number of edges that can be added to the network. Furthermore, our iterative algorithm provides the user with a number of edge-additional candidates during each iteration and lets them choose the one that causes the minimum cost in order to be added to the network.

The inputs to our iterative algorithm RENEA are an undirected, unweighted scale-free network represented as a graph $G(V, E)$ with $|V| = N$ nodes and $|E| = L$ edges, the budget δ (maximum number of edges that can be added to G), and θ (the initial percentage of hubs to remove in the attack simulation process). The output from RENEA is a more robust graph $G(V, E')$ with the same number of nodes yet more edges ($L \leq |E'| \leq L + \delta$). The steps of the algorithm are as follows:

0) $m = 0$ *// m is the number of edges added to G*
1) Simulate a targeted attack and remove θ percent of hubs from G.
2) $listComps$ = Sorted list of disconnected components based on their size, in non-increasing order.
3) $comp1$ = The largest component in $listComps$.
4) $comp2$ = The second largest component in $listComps$.
5) Select a node $x \in comp1$
6) Select a node $y \in comp2$
7) $E' = \{x\text{–}y\} \cup E$. *// add edge x–y to G*
8) $comp = comp1 + comp2$
9) Remove $comp1$ and $comp2$ from $listComps$.
10) Add $comp$ to the beginning of $listComps$.
11) $m = m + 1$
12) Repeat steps 3-11 until $|listComps| == 1$ or $m == \delta$.

RENEA starts off with simulating an attack on graph G (line 1). There are two common types of attacks that can be carried out on a network known as serial-attack and sudden-attack. In both of these attacks an initial θ percent of highest-degree nodes are to be removed. In the sudden-attack, θ percent of highest-degree hubs are identified and removed at once, whereas in the serial-attack, which is known to be more damaging, hub removal happens a bit differently. In the serial-attack, first the highest-degree hub is removed, then the degree of each remaining node is recalculated and among them again the highest-degree hub is removed until θ percent of hubs are discarded. In the graphical user interface, a user can choose either of these options to simulate an attack on the network.

Once G is fragmented, all disconnected components are found, where the number of nodes in each component determines the size of that component. In order to add the least number of edges yet gain the highest improvement in robustness, the single-node components will be ignored. $listComps$ contains a list of components sorted based on their size in a descending order (line 2). The first and second members of $listComps$, the largest and second largest components, are called $comp1$ and $comp2$, respectively (lines 3-4).

The algorithm adds an edge between node x in $comp1$ and node y in $comp2$. There exists a list of candidates whose size is equal to $|comp1| \times |comp2|$. Our user interface allows the user to pick one edge that can impose the least cost to the network (lines 5-7). Afterwards, $comp1$ and $comp2$ are combined into $comp$ and both are removed from $listComps$, and $comp$ is added to the beginning of $listComps$ (to keep it sorted) (lines 8-10). Variable m is used to keep track of the number of edges added to G (line 11). At this point one edge is already added to G ($E' = E \cup \{x\text{–}y\}$) and m=1. RENEA, as an iterative algorithm, repeats steps 3-11 until $listComps$ contains only one component (at the end of each iteration, the size of $listComps$ decreases by one) or the desired number of edges are added to G ($|E'| = L + \delta$) (line 12).

3.3 Edge Removal

Even though applying RENEA on a graph improves its robustness remarkably, adding more edges to a network comes with an extra cost. Hence, depending on the nature of the network, some users may or may not decide to mitigate the total cost by getting rid of some edges from the graph yet still have a considerable overall enhancement in the robustness of the network.

Upon request of the user, our algorithm nominates some edges to get removed from the graph based on the betweenness centrality value of them. The edge betweenness centrality is defined as the number of the shortest paths that pass through an edge of a network. An edge with a high edge betweenness centrality score represents a bridge-like connector between two parts of a network and the removal of such edges may affect the communication between many pairs of nodes through the shortest path between them.

To implement the edge-removal part, first each edge is associated with an edge centrality value. Then these values are sorted in increasing order. The user can request the maximum number of edges that s/he is willing to remove from the network. Removing edges with high betweenness, that occupy critical roles in the network, can force many pairs of nodes to be re-routed on a longer way in order to communicate with each other. It can also degrade the overall efficiency of the network in terms of communication. Hence, the algorithm starts removing edges with the lowest betweenness value, as long as that edge-removal does not make the network disconnected, until the desired number of edges are removed.

3.4 The Graphical User Interface

RENEA is implemented using the Python programming language using the powerful NetworkX [16] library to perform graph operations. The Graphical User Interface (GUI) uses elements of the networx_viewer to easily enable an interactive view of the graph that allows scrolling, zooming, and moving nodes. The Graph Viewer element is included in a Tkinter based GUI; Tkinter is the standard GUI library in Python and is included in a variety of distributions of the language. This helps to ensure that the application is as cross platform as possible.

The RENEA user interface [*Figure 1*] consists of the following controls: (i) a file chooser to allow the user to select a file that contains a list of nodes and edges of the graph, (ii) a text input field to specify the initial value of the threshold (θ), (iii) a text input field to specify the maximum number of edges to be added to the graph, (vi) a drop-down menu to select the desired type of attack on the graph, (vi) a control button to start the process of adding edges to the graph to enhance robustness, (v) a text input field to specify the maximum number of edges to be removed from the graph, and (vi) a control button to start the process of removing edges from the graph.

As shown in [*Figure 2*], a data set has been selected, and all of its specifications are presented in the GUI such as the number of nodes, edges, and initial robustness. The important nodes in terms of their degree (hubs) are presented using bigger circles. In this GUI the user can drag and relocate each edge and node, and zoom in/out on the graph for a closer look.

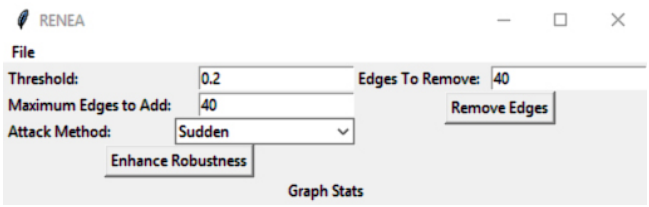

Figure 1: RENEA GUI

4 A Running Example In RENEA

To demonstrate the basic concepts behind RENEA, here we walk through a simple example. We start by uploading a graph in the GUI with 20 nodes, 21 edges, and the initial robustness of 0.1052. Once the graph is uploaded, all of its specifications are presented in the GUI [*Figure 3*]. For this particular graph, we set *'Threshold' = 20%*, *'Maximum Edges to Add' = 3*, and *'Attack Method' = Serial*. Once the user hits the *'Enhance Robustness'* button, the algorithm runs and suggests a list of edge IDs to be added. The user can accept or discard the proposed edges [*Figure 4*]. Once the user accepts these edges they will be added to graph (they are displayed with thicker lines in blue) [Figure 5]. If the user chooses to delete 3 edges, once they hit the *'Remove Edges'* button, the algorithm calculates the betweenness of the edges and nominates a maximum of three edges with the lowest betweenness whose removal does not disconnect the graph. These edges are displayed in red with thicker lines [*Figure 6*]. The GUI also shows the amount of decrease in robustness due to edge removal. Once the user accepts the changes those edges get removed and the updated robustness along with the final graph is displayed [*Figure 7*].

5 Experiments

5.1 Dataset

In this section, we experimentally examine the performance of RENEA, and for performance evaluation comparison, we have also implemented three other edge-addition algorithms: one to add edges between randomly selected pairs of nodes, one to add edges between nodes with high degrees only, and finally one to add edges between low-degree nodes. For the experiment, we used a real-life American Airlines dataset which is an unweighted, undirected, and connected scale-free graph. This dataset contains 332 airports (nodes) and 2126 flights between airports (edges) and contains no multi-edges (two or more edges that are incident to the same two nodes) or self-loops (an edge from a node to itself).

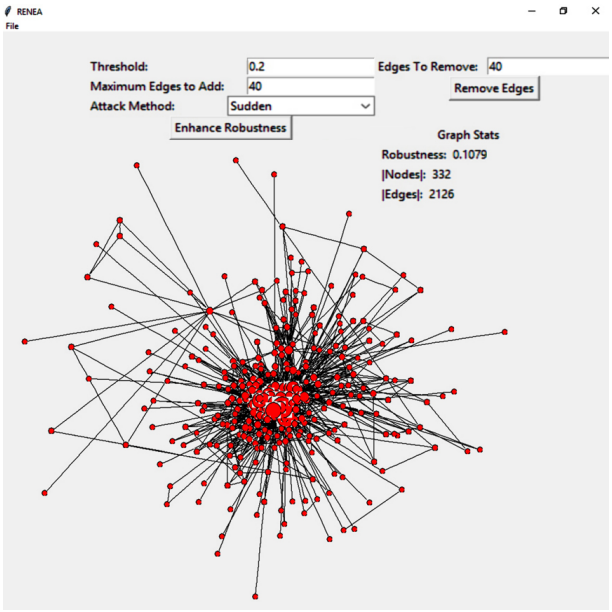

Figure 2: RENEA GUI after uploading a dataset containing USAir network; all the specifications of this graph (e.g., number of nodes, number of edges, and initial robustness) also are displayed.

5.2 Results

We tested RENEA and three other algorithms on the American Airlines network. We ran each algorithm 10 times and took the average of improvement acquired by each. For each experiment, we added 20, 30, 40, 50, and 60 edges to the original graph using different algorithms and computed the obtained robustness. As shown in [*Table 1*], RENEA significantly outperforms the other algorithms, accomplishing two times more improvement in comparison with the other methods. Results show that by adding less than 3% of edges to the graph, robustness improves up to 70%, whereas the random edge addition can only improve the robustness up to 35%. [*Figure 8*] shows how the size of the biggest component in the American Airlines network is changing while removing q hubs from the original network. The size of the largest component after applying RENEA causes the graph to hold its integrity for a longer time as opposed to using the other methods. [*Figure 9*] compares the degree distribution of the US Air network before and after adding 50 edges via applying RENEA. Because of adding a limited number of edges to the graph, the shape of its degree distribution remains almost the same. The reason that a scale-free network is robust in terms of random failure is because having a power-low degree distribution and adding a limited number of edges to the graph does not cause severe changes in it. The goal is to keep all the unique properties of the graph the same while enhancing its robustness against targeted attacks.

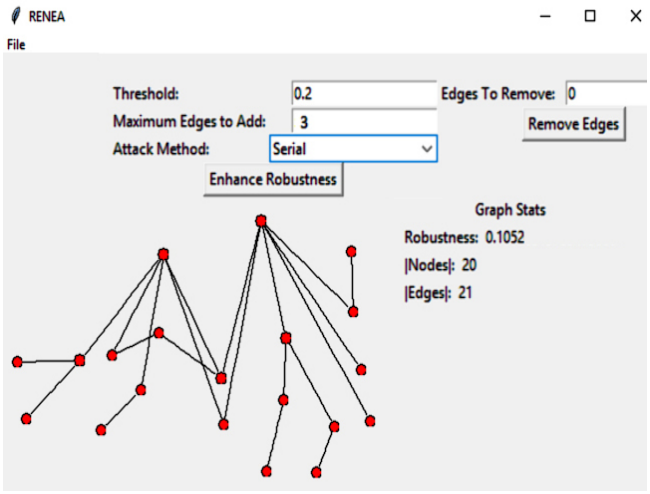

Figure 3: A graph with 20 nodes and 21 edges.

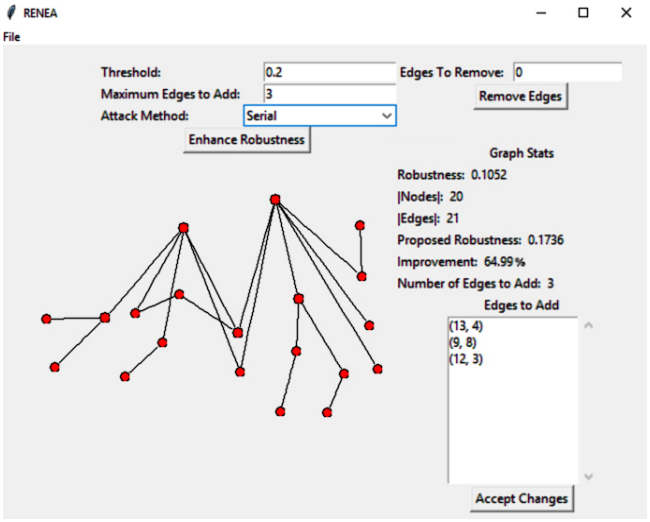

Figure 4: A list of suggested edges to add to enhance robustness.

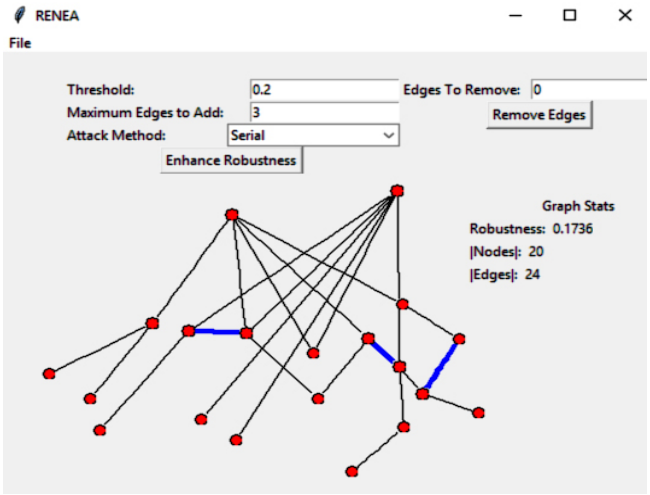

Figure 5: Graph after adding 3 edges.

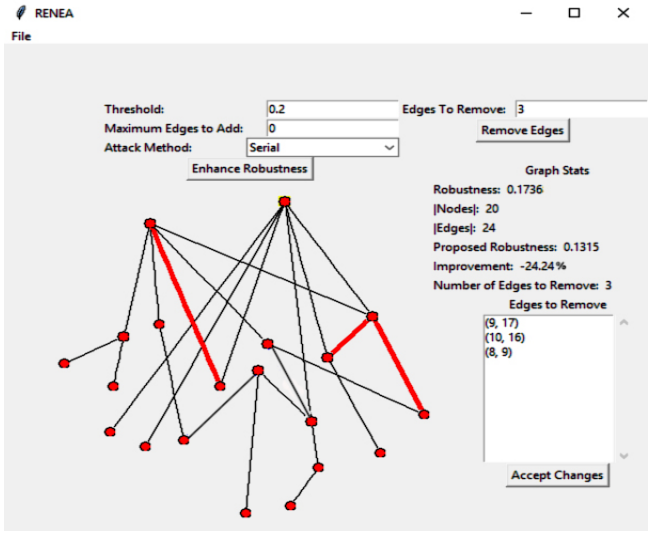

Figure 6: 3 edges are nominated to get removed.

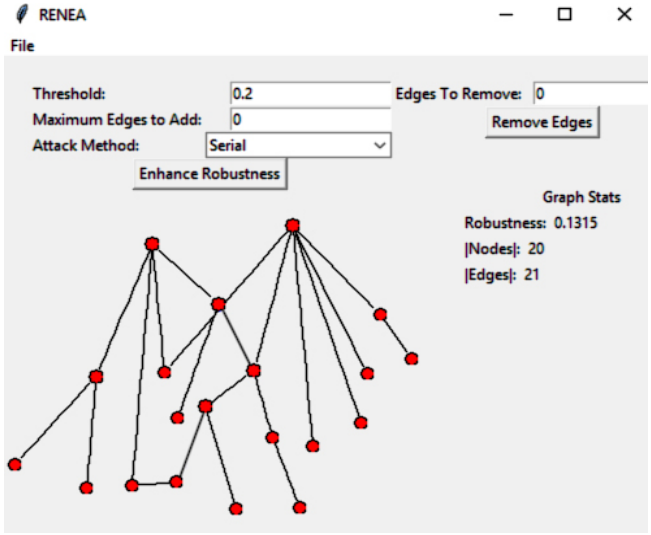

Figure 7: After adding 3 edges and removing 3 edges, the overall robustness improvement is 25%.

6 Conclusion and Future work

Real-life complex networks are known to be resilient in terms of random failures yet fragile in the terms of targeted attacks. To enhance their total robustness, one strategy is to add more edges to the network and the other is to rewire some qualified edges in the network. Based on the type of the network either of these methods could be used. Mostly these strategies accept a change only if it improves the robustness of the network by a threshold. Hence, it requires them to compute the robustness of the graph before and after applying a change which could be very time-consuming depending on the size of the graph. So they are not considered efficient in terms of time. Additionally,

Table 1: The performance of RENEA vs. three other algorithms Algo1, Algo2, and Algo3. Algo1 adds edges randomly, Algo2 adds edges among low-degree nodes, and Algo3 adds edges among high-degree nodes. The dataset is an American Airlines dataset with 332 nodes, 2126 edges and initial robustness of 0.1079. We added 20, 30, 40, 50, and 60 edges to the original graph and the averages of total robustness improvement acquired after running each algorithm are presented.

Initial R 0.1079	20 edges	30 edges	40 edges	50 edges	60 edges
RENEA	26.17%	38.16%	49.94%	55.25%	66.09%
Algo1	11.08%	18.19%	23.20%	30.44%	34.07%
Algo2	11.55%	13.17%	15.42%	21.80%	24.60%
Algo3	10.26%	12.14%	19.09%	21.51%	21.72%

none of these solutions take the nature of the network into consideration. Thus, their proposed solution is not always a practical one.

In this work, we presented RENEA, an iterative algorithm that is designed to enhance the overall robustness of a scale-free network against malicious attacks by adding a limited number of edges to it. Adding new connections to the nodes arbitrarily and without any constraint can change the nodal degree of the graph and disturb other structural properties of it. We also presented a user interface for RENEA that allows the user to see the effect of the changes to the network. In addition to the excellent performance of RENEA, the fact that it does not compute the robustness during each iteration, makes it work very fast to communicate with the user. During each iteration, RENEA suggests the best edges such that adding them can best increase the robustness of a graph, and the user, considering the cost, can accept or reject them. Finally, if the user also wants to remove some edges from the graph to mitigate the overall cost, RENEA nominates the ones with low betweenness centrality value whose removal does not make the graph disconnected. Our future work will be focused on performing formal studies of the usefulness and usability of the user interface as well as further improvement in the performance of RENEA.

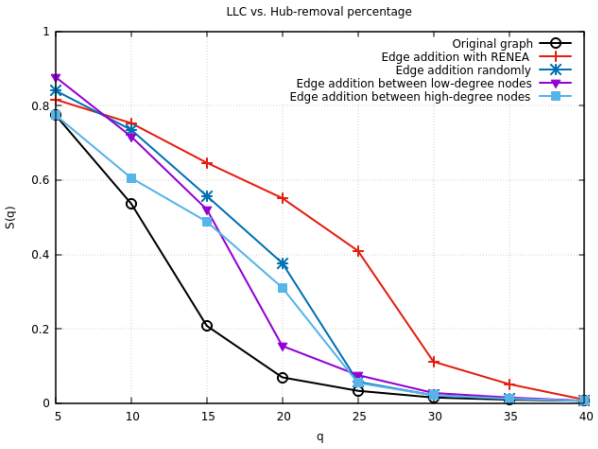

Figure 8: X-axis represents the number of removed nodes and Y-axis represents the size of LLC in the American Airlines network before and after reconfiguration. The line in red shows the performance of RENEA.

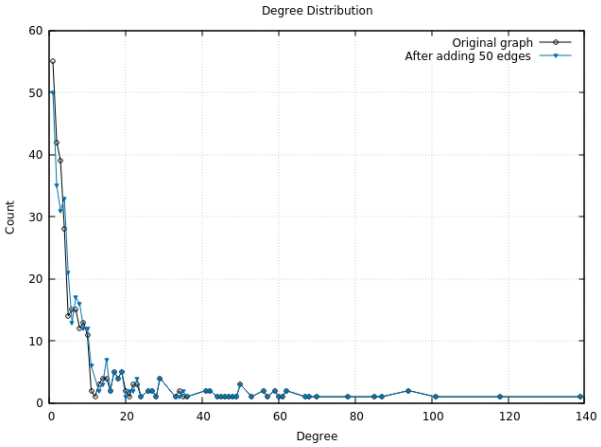

Figure 9: X-axis represents degrees of nodes and Y-axis represents how many nodes share the same degree. After adding 50 edges to the US Air network, its degree distribution remains almost the same.

References

[1] Cohen, Reuven, and Shlomo Havlin. "Scale-free networks are ultrasmall." Physical review letters 90.5 (2003): 058701.

[2] Barabsi, Albert-Lszl, and Rka Albert. "Emergence of scaling in random networks." science 286.5439 (1999): 509-512.

[3] Beygelzimer, Alina, et al. "Improving network robustness by edge modification." Physica A: Statistical Mechanics and its Applications 357.3 (2005): 593-612.

[4] Xiao, S., et al. "Robustness of scale-free networks under rewiring operations." EPL (Europhysics Letters) 89.3 (2010): 38002.

[5] Schneider, Christian M., et al. "Mitigation of malicious attacks on networks." Proceedings of the National Academy of Sciences 108.10 (2011): 3838-3841.

[6] Ze-Hui, Qu, et al. "Enhancement of scale-free network attack tolerance." Chinese Physics B 19.11 (2010): 110504.

[7]] Yang, Yang, et al. ”Improving the robustness of complex networks with preserving community structure.” PloS one 10.2 (2015): e0116551.

[8] Zhao, Jichang, and Ke Xu. ”Enhancing the robustness of scale-free networks.” Journal of Physics A: Mathematical and Theoretical 42.19 (2009): 195003.

[9] Wu, Zhi-Xi, and Petter Holme. ”Onion structure and network robustness.” Physical Review E 84.2 (2011): 026106.

[10] Chan, Hau, Shuchu Han, and Leman Akoglu. ”Where graph topology matters: the robust subgraph problem.” Proceedings of the 2015 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2015.

[11] Li, Li, et al. ”Enhancing the Robustness and Efficiency of Scale-free Network with Limited Link Addition.” TIIS 6.5 (2012): 1333-1353.

[12] Wang, Xiao Fan, and Guanrong Chen. ”Complex networks: small-world, scale-free and beyond.” IEEE circuits and systems magazine 3.1 (2003): 6-20.

[13] Schaffter, Thomas, Daniel Marbach, and Dario Floreano. ”GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods.” Bioinformatics 27.16 (2011): 2263-2270.

[14] KDNuggets. Top 30 Social Network Analysis and Visualization Tools, June 2015. http://www.kdnuggets.com/2015/06/top-30-social-network-analysis-visualization-tools.html. Accessed March 28, 2017.

[15] Callaway, Duncan S., et al. ”Network robustness and fragility: Percolation on random graphs.” Physical review letters 85.25 (2000): 5468.

[16] Schult, Daniel A., and P. Swart. ”Exploring network structure, dynamics, and function using NetworkX.” Proceedings of the 7th Python in Science Conferences (SciPy 2008). Vol. 2008. 2008.