Network Analysis
Network robustness

Samuele Crea

June 2024

1 The Study

In this last assignment, the purpose is to simulate failures of nodes within a graph to measure its robustness.

To do this, different types of attacks will be performed: some less effective such as random failures and others more targeted, such as eliminating nodes based on their betweenness, degree and pagerank value.

Once this is done on smaller graphs I will attempt to apply the script also on the graph used for the first two assignments.

At this point, the second part of the assignment is to improve the robustness of the graph by figuring out the most effective way to do this.

2 Early tests on small graph

As mentioned above, the 4 types of failure were tested on small graphs created with the help of the networkX library.
The first graph used was the one I started from for many tests in previous assignments: the karate club graph offered by networkX.

This graph has very few nodes (34) but it may be interesting to see how these types of attacks behave on a small graph.

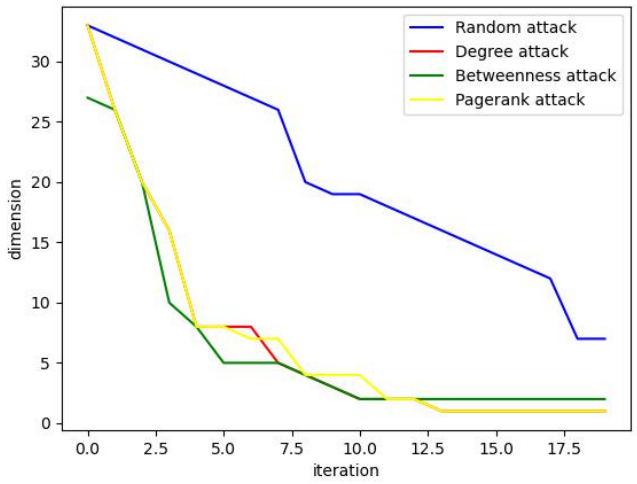

Figure 1: dimension of giant component in a small graph

As we can see, the graph in Figure 1 represents the size of the giant component at each iteration of the failure simulation process.
As we can see after 20 failures the giant component almost always tends to 0. This is also clearly due to the fact that more than 60 percent of the graph has been removed.

As was imaginable random failures are the worst method of quickly and significantly reducing the giant component although in this case even this type of approach seems to be quite effective.

All other types of targeted failures for such a small graph have the same behavior more or less.

The second graph I decided to take into analysis is a random barabasi albert type graph, also generated with 2000 nodes with the help of the networkX library.

This type of graph is scale-free so what we expect is to see a fairly pronounced resistance to random failures and instead a rather weak response to targeted attacks.

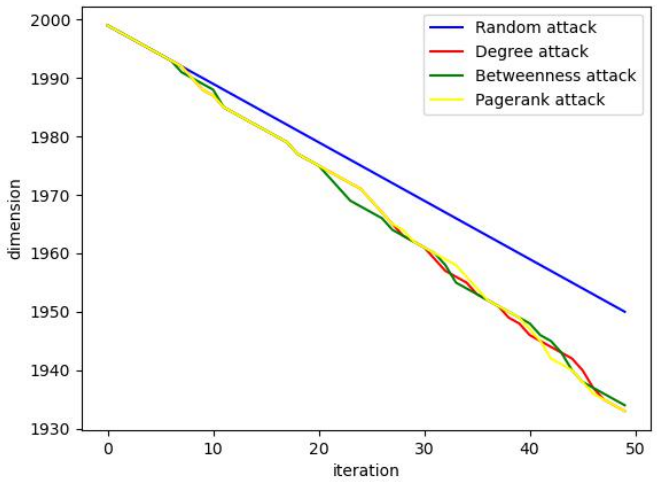

Figure 2: dimension of giant component in a barabasi albert graph

As imaginable, the graph reacts better to random failure than to targeted attacks after 50 rounds in which one node is removed at a time.
The presence of only 2000 nodes perhaps does not make the experiment extremely meaningful but nevertheless a substantial difference between the two types of failure is already visible.

3 Test on a real graph

After testing the script on small graphs we move on to the study on a real graph, the one used in the first two assignments.

Also for this experiment, as in the previous one,I used a reduced version of the GPlus social network graph.
This decision was again made because of the high computational time of the script, which made it impossible for me to test my graph.

The graph of GPlus, as was already mentioned in the first assignment after careful analysis of its properties, is a graph that has some extremely connected nodes with a central role (hub) and others with few connections.

It is clear how this is an important point in analyzing the robustness of a graph, and that removing nodes such as 2300 leads to a drastic reduction in the giant component.

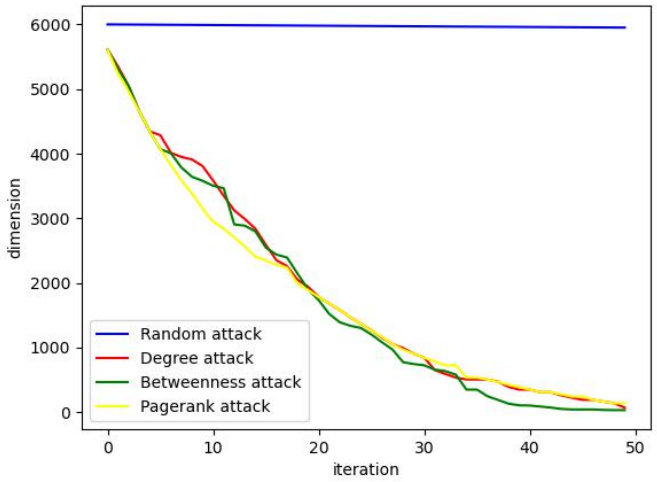

Figure 3: dimension of giant component of Gplus graph

As we can see from Figure 3, the graph reacts really badly to targeted attacks while random ones have almost no effect on it.
The weakness of this graph lies in the fact that it has very few important nodes that when removed drastically reduce the size of the giant component until it even disappears.

The graph therefore is not absolutely robust since, with the removal of only 50 nodes out of 6000 available, the giant component has almost disappeared.

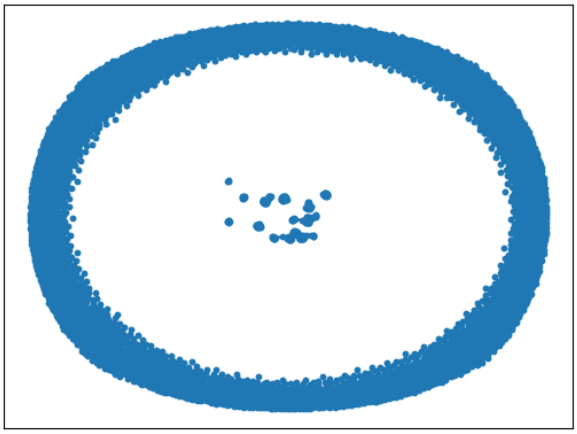

Figure 4: the graph after a degree failure of 50 nodes

Figure 4 shows the graph after an attack simulation was completed.
In this case, 50 nodes were removed based on their degree, and in the end we can see how the giant component remained the size of 72 nodes with 243 total edges in the graph.
The outer ring of nodes has no connections at all, while the central nodes only what remains of a now extremely small giant component.

4 Building robustness

It is possible to increase the robustness of a graph by adding edges between existing nodes.
The metric i will use to calculate the robustness of the graph is the critical threshold f_c.

This represents the fraction of nodes that must be removed from a network for the giant component to fragment into many smaller components.

$$f_c = 1 - \frac{1}{\frac{\langle k^2 \rangle}{\langle k \rangle} - 1}$$

The critical threshold was calculated through the formula above.

For the analysis, I decided to add nodes through the add edges function of networkX with three different methods:

- addition of edges between nodes with higher degree
- addition of edges between nodes with lower degree
- addition of edges between nodes with higher betweenness

I then ran the script with a test graph: the karate club graph.

Threshold	before	after-degree	after-peripheral	after-betweenness
f_c	0.5414	0.8651	0.5414	0.8650

Table 1: critical threshold in a small graph

As we can see from the table the best method is to add edges between "important" nodes in the graph, whether by degree or betweenness.
Enhancing peripheral nodes does not seem to have significant effects.

A total of 50 edges were added in Table 1, and in the image below we can see the structure of the graph before and after the addition.

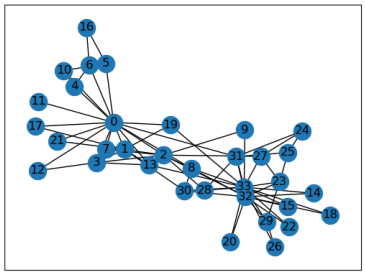

Figure 5: Before

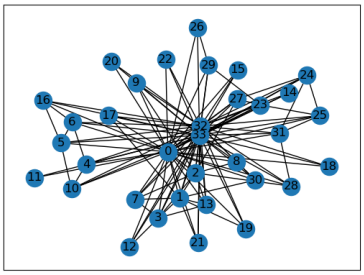

Figure 6: After

In conclusion, I applied the same algorithm on the real GPlus graph to see how the addition of nodes impacts a significantly larger graph. In this case the added edges were 1000 edges.

Threshold	before	after-degree
f_c	0.994654	0.995776

Table 2: critical threshold in a small graph

In this case the critical threshold was already very high, and through the addition of additional edges it is further improved even if only slightly.