Simulating Cascade-based attacks on complex networks

Esteves, Artur
arturesteves@tecnico.ulisboa.pt

Morais, Ricardo
ricardojhmorais@tecnico.ulisboa.pt

December 1, 2017

Abstract

In many realistic situations the flow of physical quantities in the network, as characterized by the loads on nodes, is important. This paper simulates the Motter-Lai model [1] which shows that intentional attacks can lead to a cascade of overload failures, and can cause serious damage to a network. This model shows that the heterogeneity of real-world networks makes them particularly vulnerable to attacks in that a large scale cascade may be triggered by disabling a single key node.

1 Introduction

This paper is the culmination of our efforts to replicate part of [1]. We live in a world surrounded by complex networks, like the Internet, power grids, etc. Heterogeneous networks due to the property of having only a few nodes with a lot of connections are vulnerable to cascading overload failures, where the removal of a critical node can trigger a domino effect that disables a great part of the network. In this paper we take a look at a variety of networks as we make attempts to find a network that highlights the results mentioned in [1]. To recreate the examples we resorted to the **python** programming language and **NetworkX**[2] as the graph library. The plots were made using **matplotlib**[3]. The networks were analyzed using **Gephi**[4].

2 Considerations

The Motter-Lai model is used for testing the cascading overload failure in networks. It gives us insights on the behavior of a network with different levels of tolerance. Also it shows how susceptible the network is to different types of attacks. The Motter-Lai model has the following properties:

- The load of a node in the graph is the number of shortest paths that pass through to it.
- The initial load of a node is the load of it before removing a node from the graph.
- Every node in the graph has a tolerance $\alpha \geq 0$.
- The capacity of a node given by $C(j) = (1 + \alpha)L_j, j = 1, 2, ...N$ where $C(j)$ is the capacity of the node j and L_j is the initial load of the node j.
- If the node load exceeds its capacity, the node fails and it is removed from the network permanently.
- $G = N\prime/N$, the relative size of the largest connected component after the simulation.

We used this model to elaborated the algorithm in Section 3.1

3 Replication

This sections describes the approach we took to try to replicate part of [1].

3.1 Algorithm

The objective of the algorithm is to simulate a cascading overload failure by deliberately removing a single node from the network according to the strategy. The removal causes the loads on the previously removed node to be distributed through the rest of the network. Some of the nodes will be burdened by the new load passing through them and might fail due to the exceeding of their capacity. When this nodes fail, they are removed from the network and the load is redistributed by the remaining nodes. The process continues until no node capacity is exceeded. This is what is referred by the cascading effect, a single removal of a node can iteratively cause the removal of a significantly large number of nodes in the network. This is the algorithm we devised: [1]

```
Data: graph, tolerance, strategy
Result: the size of the giant component
initialization;
calculate the capacity for each node;
use the strategy to remove one node from the graph;
initial_loads := number of shortest paths for each node;
current_loads := initial_loads;
stable = false;
while not stable do
    stable = true;
    current_loads := number of shortest paths for each node;
    calculate the capacity of each node using the initial_load and check if it was exceeded;
    for all nodes in the graph do
        if load exceeded the capacity of the node then
            remove the node from the graph using the removal strategy;
            stable = false;
        end
    end
end
calculate the number of nodes in the giant component;
```

Algorithm 1: The devised algorithm for the simulation of the cascading overload failure

3.1.1 Performance

The algorithm is very expensive in terms of performance. For every iteration, the algorithm has to calculate the number of shortest paths that passes through each node of the network in order to calculate its load. This is a very expensive operation, this operation alone has a time complexity of $O(n^3)$. In [1], networks with size of around 5000 were used. As we do not have enough resources to compute the operations in time due to the time complexity of the algorithm we considered large enough networks of 1000 and 2000 nodes to achieve similar results.

3.2 Removal Strategies

The node chosen to be removed in the network is crucial to have a devastating cascade overload failure on the network. Three removal strategies were used:

- Random removal
- Highest Degree removal
- Highest Load removal

[1] The respective implementation can be found at https://github.com/moraispgsi/redes-complexas/

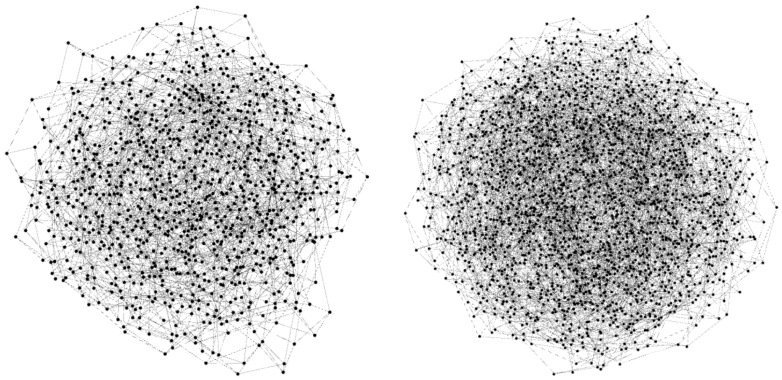
3.3 Networks

The selection of the networks was difficult, to replicate the results of the paper, the networks had to have specific characteristics, otherwise the results would be different from the results presents in [1]. Lets take a look at different scale-free networks and lets try to understand where the problem lies.

3.3.1 Homogeneous

In [1] the homogeneous network, as shown in Figure 1. This type of network configuration dictates that each node has the same degree, which means that the load is very well distributed.

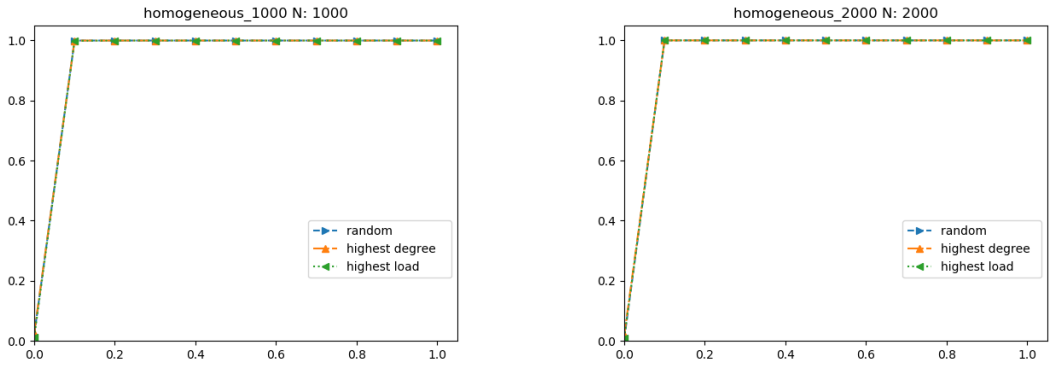
Figure 1: The figure shows two networks generated using the second experiment approach, one with 1000 nodes and another with 2000 nodes both with $< k >= 4$



(a) Network of the second experiment with 1000 nodes and 2000 edges

(b) Network of the second experiment with 2000 nodes and 2000 edges

Figure 2: The figure shows the results of the second experiment, each line represents a different removal strategy. The x axis represent the tolerance α and the y axis represent G



(a) Results obtained on the second experiment with a network of 1000 nodes with each removal strategy

(b) Results obtained on the second experiment with a network of 2000 nodes with each removal strategy

As we can see in Figure 2, the attack on a homogeneous network is only effective when the tolerance is at its minimum. All the strategies we used failed to demonstrate the cascade overload failure of the nodes. This happens because every node is equivalent within the network. Removing a node will not impact the network because each node has equal importance and the network will just readjust and distribute the load over several nodes, therefor the nodes are never overloaded.

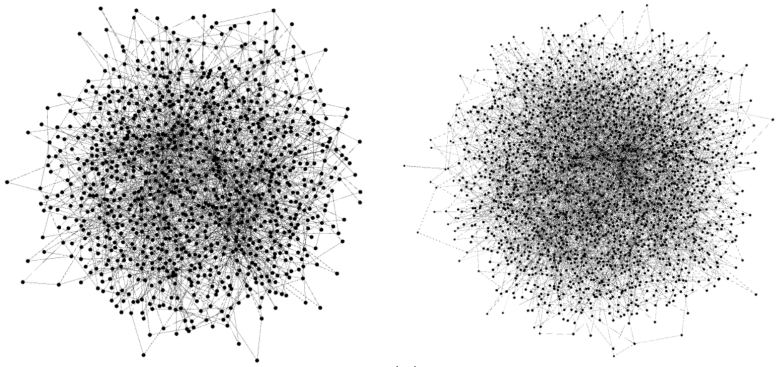
This is of course a very desirable configuration to resist attacks. It introduces a lot of redundancy to the network e.g. a lot of alternative good paths. In real world networks, it is very difficult to achieve this kind of network since each edge might signify an expensive cost[2].

[2]For example in a power grid network, more edges represent more expensive wire

3.3.2 First Experiment

Firstly we naively generated 2 scale-free networks using the library **NetworkX** and the function ***barabasi_albert_graph*** (uses preferential attachment) shown in Figure 3. The function returned 2 good looking networks with 1000 and 2000 nodes respectfully with an average degree of approximately 4. This networks seemed very promising at first, so we tried to apply the Algorithm 1.

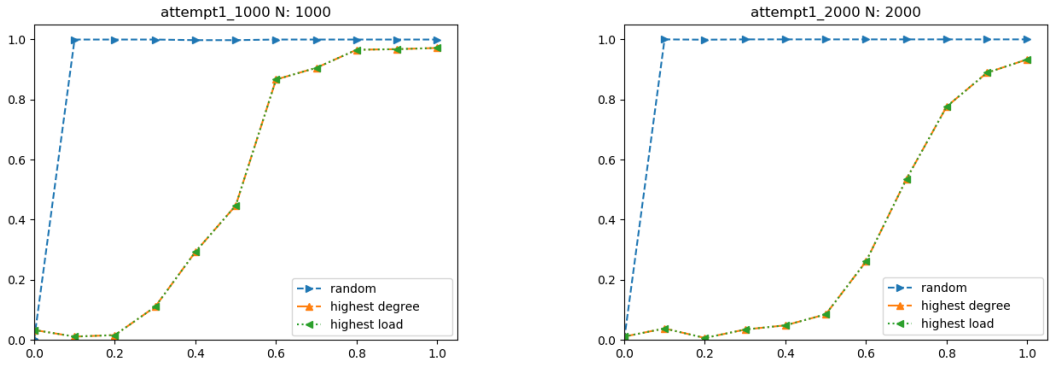
Figure 3: The figure shows two networks generated using the first experiment approach, one with 1000 nodes and another with 2000 nodes both with $< k > \approx 4$



(a) Network of the first experiment with 1000 nodes

(b) Network of the first experiment with 2000 nodes

Figure 4: The figure shows the results of the first experiment, each line represents a different removal strategy. The x axis represent the tolerance α and the y axis represent G



(a) Results obtained on the first experiment with a network of 1000 nodes with each removal strategy

(b) Results obtained on the first experiment with a network of 2000 nodes with each removal strategy

The results we obtained were somewhat different from the ones in [1]. The results for the highest degree and highest load were returning exactly the sames results[3] as you can see in the graphic in Figure 4, both the highest degree strategy line and the highest load strategy line are overlapping. This was a problem because in [1] it showed that the results were different. The logical reason for this is that in this networks, the node with the highest load is also the node with the highest degree and since the Algorithm 1 is deterministic, the result is always the same when removing the same node. The **random removal** strategy only affects the network when there is 0 tolerance and when it happens to remove a node with high degree or load. The **random removal** strategy is not very effective because in this type of networks it is highly probable that the random removal strategy will choose a node with a low degree and load. Looking at the result of the Figure 4a, we can state that to keep the network reasonably secure, each individual node in the network has to have at least a tolerance of 0.6. Also, the difference in the results of both networks hints

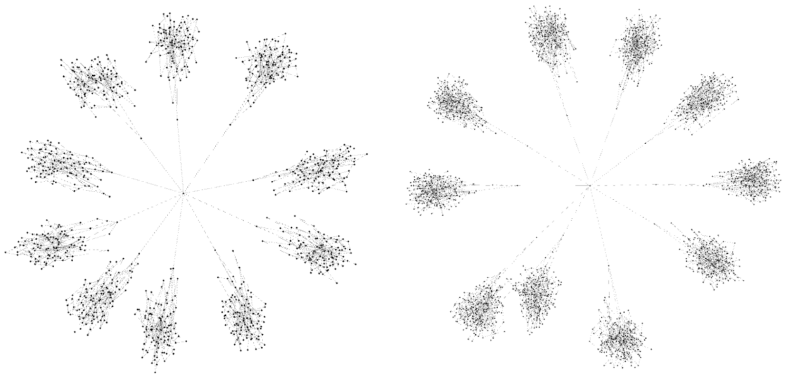
[3] We suspected the algorithm had some issue but it turned out it was because of the network arrangement

us that the bigger the network, the more tolerance each node has to have to prevent the cascading overload failure.

3.3.3 Second Experiment

We made another experiment to try and fix the problem from our previous experiment i.e the problem of having a network where the node with the highest degree is the same node that has the highest load. In this second experiment we tried making 10 scale-free subnetworks with $N/10$[4] nodes each, and connect them via a single central node. That solved that problem since the central node had the highest load but wasn't the one with the highest degree. This ensured that the removal strategies picked different nodes[5].

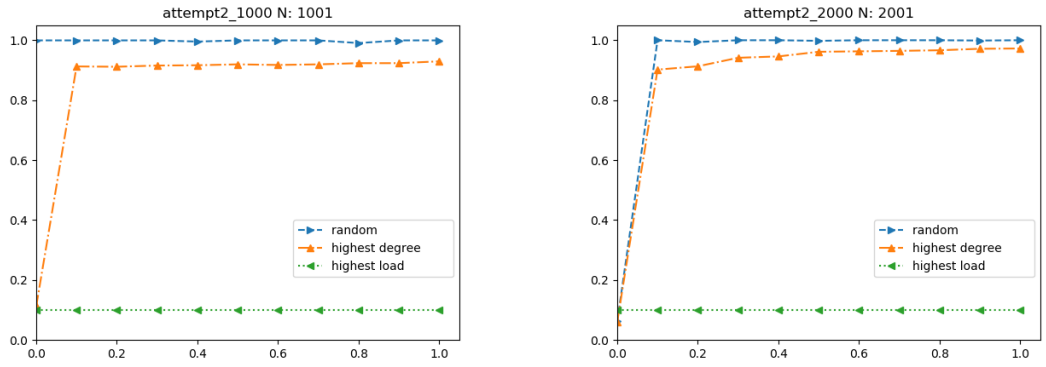
Figure 5: The figure shows two networks generated using the second experiment approach, one with 1001 nodes and another with 2001 nodes both with $< k > \approx 4$



(a) Network of the second experiment with 1001 nodes

(b) Network of the second experiment with 2001 nodes

Figure 6: The figure shows the results of the second experiment, each line represents a different removal strategy. The x axis represent the tolerance α and the y axis represent G



(a) Results obtained on the second experiment with a network of 1000 nodes with each removal strategy

(b) Results obtained on the second experiment with a network of 2000 nodes with each removal strategy

However this introduces another problem. When the central node is removed, all the subnetworks are disconnected from each other. This can be said to be a single point of failure. This disconnection stops the cascading overload entirely by reducing the number of shortest paths drastically and by reducing the load on the nodes of each subnetwork.

Obviously an attack by removing the central node to a network with this configuration will be very successful at tearing up the connectivity of the whole network, and it does so independently of the tolerance, as shown in Figure 6.

[4]Where N is the number of nodes of the entire network, in this case 1001 or 2001

[5]The Highest Degree removal strategy and the Highest Load removal strategy.

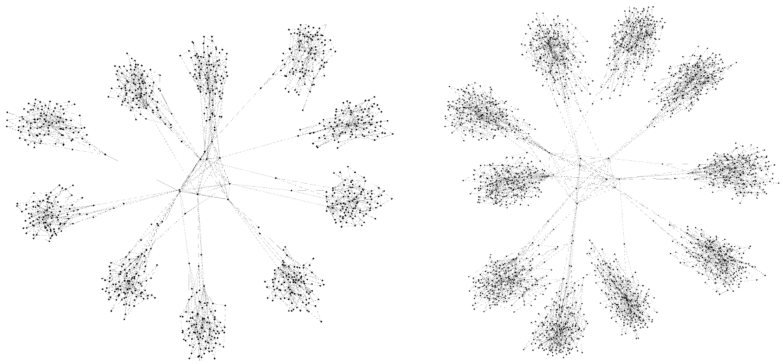
For the removal with the **highest degree strategy**, the results show that with this configuration, the network is safe enough with $\alpha \geq 0.1$. This is because the node picked for removal is within one of the subnetworks, and with $\alpha \geq 0.1$ the load does not spread to the other subnetworks, so the problem is contained withing the subnetwork.

Both networks sizes have nearly the same results. We can see that the scale of a network with this type of configuration is not important[6].

3.3.4 Third Experiment

With the knowledge we gathered from our previous two experiments, we tried to elaborate a third one. In this experiment our goal was to prevent the subnetworks to split up on the removal of a single node. So we again generated 10 scale-free subnetworks with size $N/10$ and we chose one of them to be in the center. The central subnetwork connects to each of the remaining subnetworks by one of its nodes as we can see in Figure 7.

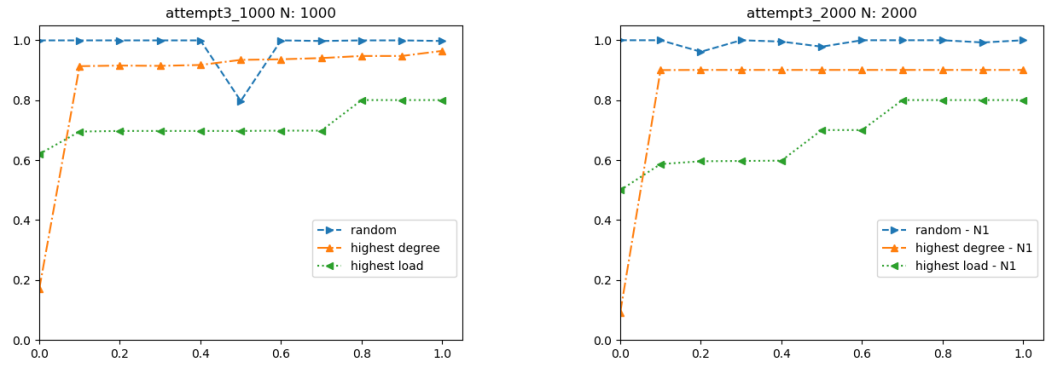
Figure 7: The figure shows two networks generated using the third experiment approach, one with 1000 nodes and another with 2000 nodes both with $< k > \approx 4$



(a) Network of the third experiment with 1000 nodes

(b) Network of the third experiment with 2000 nodes

Figure 8: The figure shows the results of the third experiment, each line represents a different removal strategy. The x axis represent the tolerance α and the y axis represent G



(a) Results obtained on the third experiment with a network of 1000 nodes with each removal strategy

(b) Results obtained on the third experiment with a network of 2000 nodes with each removal strategy

We made sure that the node with the highest degree was different from the node with the highest load. This way, the central nodes of the central subnetwork have the highest load but not the highest degree, and when one of the nodes in the center is removed, the connection between the subnetworks and the central subnetwork is not severed, and therefor the cascading overload does not end abruptly.

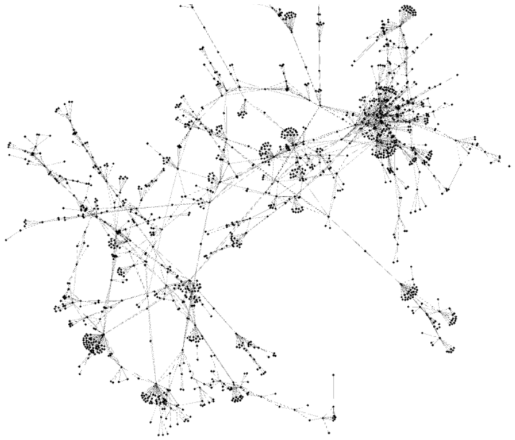
[6] In contrast with the results we achieved in the first experiment where the network size mattered

This results are similar to the ones in [1] where the removal of a node with the highest load is the most effective attack on the network. In this networks, and considering the tolerances 0.0 to 1.0 with increments of 0.1, using the highest load removal strategy tears 20% of the network.

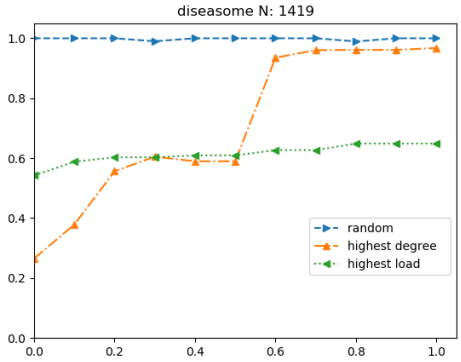
3.3.5 Diseasome

Another network we tried was the **diseasome** network[5] to see how it would react with each of the **removal strategies**. This network is hierarchical and has a lot of bridges.

Figure 9: The figure shows the **diseasome** hierarchical network. The figure also shows the results of the third experiment, each line represents a different removal strategy. The x axis represent the tolerance α and the y axis represent G



(a) The **diseasome** network.



(b) The figure shows the results of the **diseasome** network experiment, each line represents a different removal strategy. The x axis represent the tolerance α and the y axis represent G

As we can see from Figure 9b the network is very vulnerable to highest load attacks, this attack can disable 40% of the network in every tolerance level we tried. For tolerances lower than 0.2, the highest degree removal strategy is more effective, disabling more than 40% of the network.

4 Conclusions

As we can see from the previous examples, the networks that are closer to homogeneous are indeed more secure. We have reached the same conclusion presented in [1].

In [1] and in this paper we used several attack techniques for each network and from the results we concluded that heterogeneous networks are more susceptible to intelligent attacks. In every network we tested, the random strategy proved to not be very effective because it is most likely to select nodes that don't have a big role on the communication within the network. Homogeneous networks however proved to be very effective in counteracting any strategy of attack we used, so an attacker can't exploit the cascade overload failure in homogeneous networks. TODO: network perspective Homogeneous networks seems to be the best type of network configuration. Although this is not practical in real world scenarios as we explained in Section 3.3.1. In this case an analysis of the cascading failure can help to highlight the most critical nodes where security measures should take place.

References

[1] Adilson E. Motter and Ying-Cheng Lai. *Cascade-based attacks on complex networks*. Published 20 December 2002.

[2] NetworkX Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. http://networkx.github.io/.

[3] MathLab ploting library https://www.mathworks.com/help/matlab/ref/plot.html?requestedDomain=www.mathworks.com

[4] The Open Graph Viz Platform https://gephi.org/

[5] Human Disease Network, Goh K-I, Cusick ME, Valle D, Childs B, Vidal M, Barabási A-L (2007), Proc Natl Acad Sci USA 104:8685-8690