Complex Networks - Homework 9

Group 1: Iwan Pasveer, Flavia Leotta, Noah Frinking, Dylan Gavron

19th November 2024

1 Introduction

This document is supported by markdown file "CN_HW9_Group1.ipynb", refer to it for the code. For more information about the libraries used, see section "Softwares and libraries".

2 Homework 9.1

2.1 9.1.1

We are given a Networkx G:

```
G=nx.DiGraph()
G.add_edges_from([(2, 1), (1,2), (4,1), (4,3), (5,1), (11, 1),(11,4), (11,5), (5,11),
(6,11), (7,11), (8,11), (9,11), (10, 11), (10, 1), (9, 1), (8,1)])
```
(1)

to which we apply this function (the value of alpha is the standard one):

```
plot_smileygraph(G, alpha = 0.85)
```
(2)

to make it look like Figure 9.2 in Lecture Notes Chapter 9. The graph we obtain is shown in Figure 1. Colors were chosen by calculating percentiles of the page-rank values:

- Over the 90th percentile, the nodes are colored in orange;
- Between 75th and 89th percentile, they're light blue;
- Between the 65th and 74th, they're colored in red;
- The remaining are green.

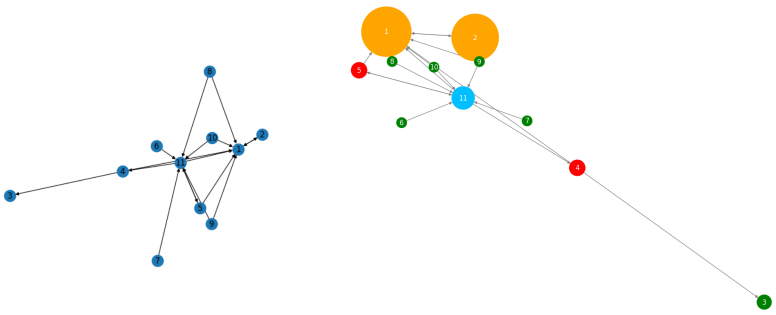

Figure 1: Graph G before and after using function (1).

We see a slight difference between our color coding and that of the provided image. When we run our algorithm, node 2 has the second-highest page-rank, but the figure 9.2 in the Lecture notes, shows that the node with the second highest score should be node 11. This is possibly due to slight differences in the definition of pagerank. The method of calculation in NetworkX may work differently than the method used to produce the image.

2.2 9.1.2

When we run the Page-rank algorithm allowing up to 20 iterations, the algorithm fails to converge: this is caused by the very low tolerance we set in the first part of the exercise ($tol = 1e-06$). Instead, if we raise the tolerance up to $1e-02$, the algorithm converges and gives the graph in Figure 2.

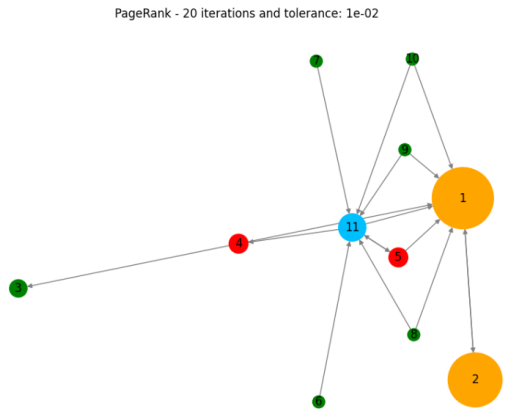

Figure 2: Graph G using Pagerank algorithm with 20 iterations and tolerance equal to $1e-02$.

In 20 iterations we see a greater variance in page-rank values than when we ran 100 iterations with a lower tolerance because the iterative process has not had ample time to settle. More time steps are needed to lower it, but since the exercises asks for a fixed value of iterations, we're only able to change the tolerance. The tolerance is the maximum allowed difference between the PageRank scores vector r at a certain time step k, and the vector obtained on the previous time step $k-1$.

$$tol \leq \|r^{(k)} - r^{(k-1)}\| \tag{3}$$

If the difference between two consecutive steps is smaller than the tolerance, the algorithm stops, and so we say that it has converged. With only 20 steps and a tolerance that is too low, our algorithm cannot converge so that's why we raise it, although it might result in a less accurate approximation.

2.3 9.1.3

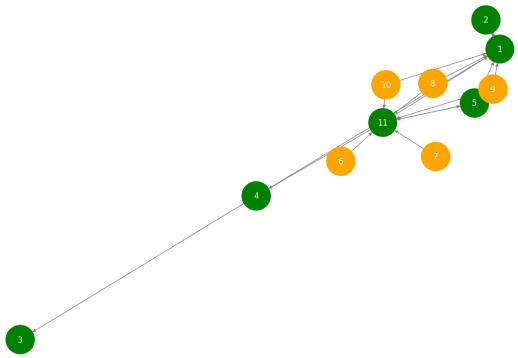

Figure 3: Graph G alpha equal to -0.01.

If a small negative value is chosen for α (in our example -0.01) then the probability of jumping between random nodes (1 - α) becomes greater than 1. This breaks the probability model as a probability cannot be less than 0 or greater than 1. This makes the pagerank values to be more

similar, as each node is nearly equally likely to be teleported to. This is reflected in Figure 3 which is the result of function:

```
pagerank_with_alpha_negative(G)
```
(4)

where each node has nearly the same size. We can see that node 1 and node 11 have slightly lower pagerank than initially shown in Figure 1. This is due to the now negative contribution of incoming links, and these nodes have the most in-edges.

2.4 9.1.4

α, being the likelihood that a random walker will follow a link, theoretically ranges from 0 to 1. We want to find the largest range for values of α where PageRank produces a result that reflects the relative importance of nodes. We consider a result meaningful when the variance of the PageRank scores is neither too high or too low and when, of course, the algorithm can converge. A variance that is too low might show that the algorithm didn't explore the relative importance of nodes (assigning the same one to each of them), while an high variance might over-estimate this importance.

We implement function:

```
find_alpha_range(G)
```
(5)

that explores value of alpha in a range of $[0.0, 1.0]$ with a step-size of 0.025. The range of variances that are considered meaningful goes from the 25th percentile to the 99th. The complete list of the elements that are returned by this function is reported on the code we provide as supporting material to this report. We can plot the values of alpha explored with function:

```
plot_range(G)
```
(6)

that plots Figure 4.

To speed up the search for the largest value of alpha possible, we implemented function:

```
find_largest_alpha(G)
```
(7)

that simply returns the second element of function (5).

Our function found that the largest range of alpha where the Pagerank algorithm produces meaningful results is $[0.225, 0.850]$.

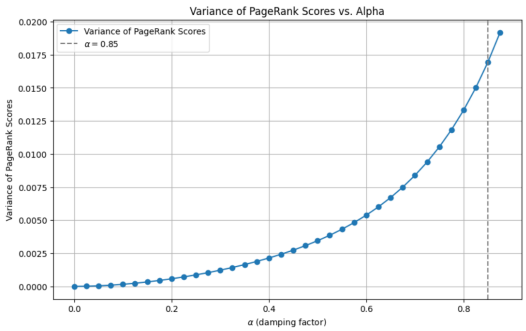

Figure 4: Variance observed in the Pagerank scores of graph G, depending on the value of alpha used.

- When α is lower than 0.225, all nodes start to have equal probability of being explored regardless of the network structure. Usually at these values the convergence is almost guaranteed, but doesn't take into consideration how the network is actually built.
- Instead, if α exceeds 0.850, the algorithm relies entirely on the link structure and never teleports. If the graph contains disconnected components or dangling nodes, not being able to teleport can make the algorithm diverge.

3 Homework 9.2

We implement the function:

```
compare_pagerank_on_BA_GNM(s, nodes_BA = 500, nodes_ER = 500, edges_BA = 4000,
edges_ER = 4000):
```

(8)

that, given a seed s, generates two graphs with the parameters given, and returns the two plots shown in the following sections. The first network is a Barabasi-Albert preferential attachment graph with 500 nodes (the default parameter) and an m-value of 8, computed to generate around 4000 edges. The second network is an Erdos-Renyi Random Graph with 500 nodes and exactly 4000 edges.

3.1 9.2.1

We show a logarithmic scatterplot of the Pagerank distribution for both graphs (Figure 5), computed using standard parameters.

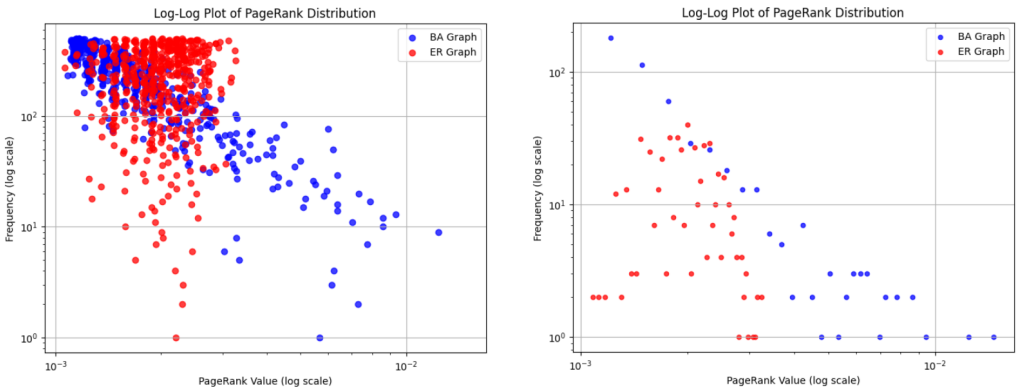

Figure 5: On the left, scatterplot in log-log scale of the PageRank values frequencies. On the right, the same data but grouped.

We can see that the Barabasi-Albert graph Pagerank values are distributed along a diagonal line: this suggests a power-law distribution. There are many nodes with low pagerank value, and few nodes with an high one. This is expected from the construction of the BA graph, which prefers to attach new edges to existing edges that have higher degree. This leads to the emergence of hubs, which have an higher Pagerank score than less connected nodes.

The Erdos-Renyi graph does not follow a power law, but peaks in frequency around a pagerank value of 0.07. There are little to no nodes with high pagerank, which we expect due to the method of construction of the network itself. Because each potential edge has the same probability of being realized, we don't expect a power law distribution to emerge: the smaller spread of pagerank values in comparison to the BA graph is expected because no nodes should become much more connected than others.

We reported 2 scatterplots that are achievable by using two versions of the same function. On the right side of Figure 5 we can appreciate that by grouping data (using hystogram grouping) the scatterplot appears less confusing. The trend is still visible and less influenced by noise, so this would be the ideal representation for graphs with an higher number of nodes.

3.2 9.2.2

We compared 100 trials of computing the pagerank performance (calculated in seconds) for each graph model and noticed that apparently the ER model often takes longer than the BA model (Figure 6).

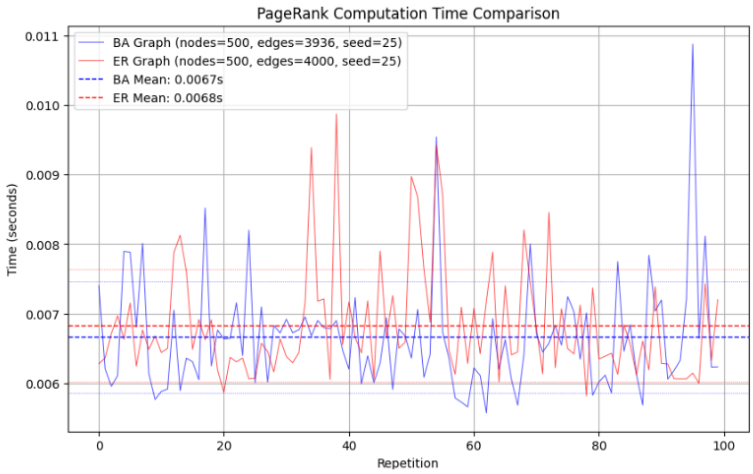

Figure 6: Performance of Pagerank algorithm over 100 repetitions, for a BA Network (blue) and a ER network (red).

The difference is, though, not big, so we decided to perform statistical testing on R-studio. We used the data reported in Table 1 and first tested our hypothesis:

$$H_0 : \text{PageRank algorithm performs the same for both Networks, } \mu_{BA} = \mu_{ER} \tag{9}$$

$$H_1 : \text{PageRank algorithm performs better in the BA Network, } \mu_{BA} < \mu_{ER} \tag{10}$$

by using function:

$$\text{t.test(ba\_time, er\_time, alternative = "less")} \tag{11}$$

Network type	**Mean ($\overline{x}$)**	**Standard deviation ($\sqrt{s^2}$)**
BA	0.00666 s	0.00081
ER	0.00683 s	0.00082

Table 1: Mean and standard deviation of the performance time of the Pagerank algorithm (100 repetitions) used to perform two t-tests, described in section 9.2.2.

The p-value of this test is 0.06846, so we can't reject the null hypothesis: there's no statistical proof that the Pagerank algorithm performs better on a BA network. Even if we check the alternative "two.sided" (which means, we're checking whether they're simply different) the p-value is still 0.1369, so we cannot conclude that the Pagerank algorithm performs differently in any of the two networks models.

4 Software and libraries

- Python version 3.12.7
- RStudio version 4.3.3 (2024-02-29 ucrt)
- Networkx version 3.3
- Matplotlib version 3.9.0
- Numpy version 1.26.4
- Timeit version 3.13.0