BLG 549E
October 28, 2021

Homework 1

Due date is **12th November 2021, 23:59!**

Homework Policy

- Use comments whenever necessary to explain your code.
- You will use Python as the programming language.
- IMPORTANT: This is an individual assignment! You are expected to act according to Student Code of Conduct, which forbids all ways of cheating and plagiarism. It is okay to discuss the homework with others, but it is strictly forbidden to use all or parts of code from other students' codes or online sources and let others do all or part of your homework.
- Only electronic submissions through Ninova will be accepted. You only need to submit your Jupyter Notebook file. Any comments and discussions should be included in the same file.
- If you have any questions, you can contact me at kamard@itu.edu.tr. Do NOT hesitate to send an e-mail if you are confused.
- Check out the homework solution of 2020. A neat example to follow. You can do better!

Jupyter Notebook Installation

You need to have Python and Pip installed in your computer to install Jupyter Notebook.

Windows

On command prompt (cmd.exe with admin mode):

```
C:\**path**> python -m pip install jupyter
```

After changing your current folder to the folder which you want to work on (see 'cd' command):

```
C:\**your_working_folder**> jupyter notebook
```

Then it will be launched on your default browser

Ubuntu/Linux/Unix/Mac

On Terminal:

```
$ pip install notebook
```

Then launch with:

```
$ jupyter notebook
```

For more information: https://jupyter.org/install

Homework 1

Introduction

You will carry out all the tasks below using **Ipython Notebook**. Simply add all your work to the provided template file **HW1_template.ipynb** using jupyter notebook.
In this homework, you will code up several experiments in Albert et al. paper[1] (You can click anywhere on this sentence instead of a small, hard-to-click word "here" to find the paper). To this aim, first simulate 60 networks: 30 **weighted undirected exponential networks** using Erdos-Renyi generative model [1, 2] and 30 **weighted free-scale networks** using an algorithm similar to that of described in [1]. You can make all weights **positive**. For this task, you can use ready-made pieces of code. But all needs to be well commented out and references added properly. The number of nodes in each network category (e.g., ER) should equal to 200.

Part A Simulate exponential and free-scale networks (35 Points, 5 points each question)

1. Briefly explain how the weighted Erdos-Renyi generative model works.
2. What are the key properties of weighted Erdos-Renyi (ER) graphs?
3. Briefly explain how weighted scale-free (SF) algorithm works.
4. What are the key properties of weighted SF graphs?
5. Visualize two random graphs you simulated (ER and SF).
6. Plot the overlaid distributions of the node strength centrality for all 30 ER graphs (transparent blue color) and the 30 SF graphs (transparent red color). What do you notice? Interpret your observation.
7. Plot the overlaid distributions of the eigenvector centrality for all 30 ER graphs (transparent blue color) and the 30 SF graphs (transparent red color). What do you notice? Interpret your observation.

Homework 1

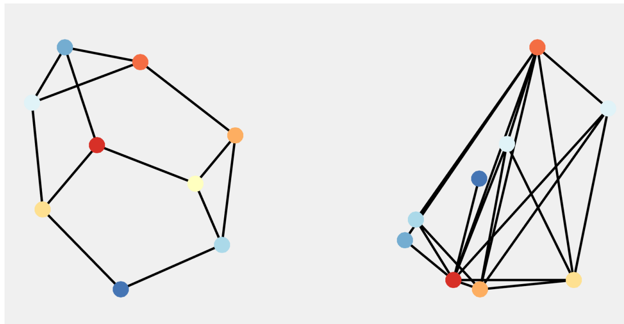

Figure 1: Example visualization of two graphs.

Part B: Analyzing Erdos-Renyi and Scale-Free graphs in both random error and target attack scenarios (65 Points)

1. **[10p]** Code up a function called 'findPercolationThreshold', where you input a positive weighted adjacency matrix and it automatically outputs its percolation threshold f_c.

2. **[5p]** Comment out each line of your 'findPercolationThreshold' function and add a text box explaining how your algorithm works.

3. **[10p]** Code up a function called 'largestConnectedComponent', where you input a positive weighted adjacency matrix and it automatically outputs (1) the largest connected component graph, (2) its node size S and (3) its diameter d defined as the average of the shortest paths between all pairs of nodes in the large connected component graph.

4. **[5p]** Comment out each line of your 'largestConnectedComponent' function and add a text box explaining how your algorithm works.

5. **[5p]** Random failure scenario. Code up a function called 'randomError', inputting (1) the graph adjacency matrix and (2) the fraction (between 0 and 1) of nodes to be randomly removed and returns the adjancy matrix graph excluding the removed nodes. Comment out the code and explain in a text box how it works.

6. **[5p]** Target attack scenario. Code up a function called 'targetAttack', inputting (1) the graph adjacency matrix and (2) the fraction (between 0 and 1) of nodes to be attacked removed and returns the adjancy matrix graph excluding the removed nodes. Note that the nodes to be removed are those with the highest strength. Comment out the code and explain in a text box how it works.

7. **[5p]** Given one random ER network and one random SF network (you select two from your 60 simulated graphs), plot the diameter d of the largest connected component as in **Fig.** 2 against the fraction of removed nodes in both random error and target attack scenarios. The nodes need to be removed progressively (e.g., remove 0.1, 0.2, until reaching 0.8).

   Important note: Watch out for the percolation threshold. You might need to remove a very small fraction of nodes before it breaks (< 0.1). Keep an eye on that. Adjust your fraction step accordingly.

   Produce 3 more similar plots for 6 randomly selected ER and SF graphs (3 ER and 3 SF). Totally, you will have 4 plots for pairs of ER and SF networks to compare.

8. **[5p]** What conclusions can you derive about SF and ER networks and their resilience to random errors and target attacks? Are these conclusions in line with the paper paper[1]?

9. **[5p]** Given one random ER network and one random SF network (you select two from your 60 simulated graphs), plot the size S of the LCC of the largest connected component as in **Fig.** 3 against the fraction of removed nodes in both random error and target attack scenarios.

   Produce 3 more similar plots for 6 randomly selected ER and SF graphs (3 ER and 3 SF). Totally, you will have 4 plots for pairs of ER and SF networks to compare.

10. **[5p]** What conclusions can you derive about SF and ER networks and their resilience to random errors and target attacks? Are these conclusions in line with the paper paper[1]?

11. **[5p]** Plot the percolation threshold for each of the 8 sampled graphs using your 'findPercolationThreshold' function. Are those thresholds concordents with what we observe in the plots you generated. Discuss and compare.

    Comparing ER against FS percolation thresholds for the 8 randomly selected networks, what conclusions can you derive?

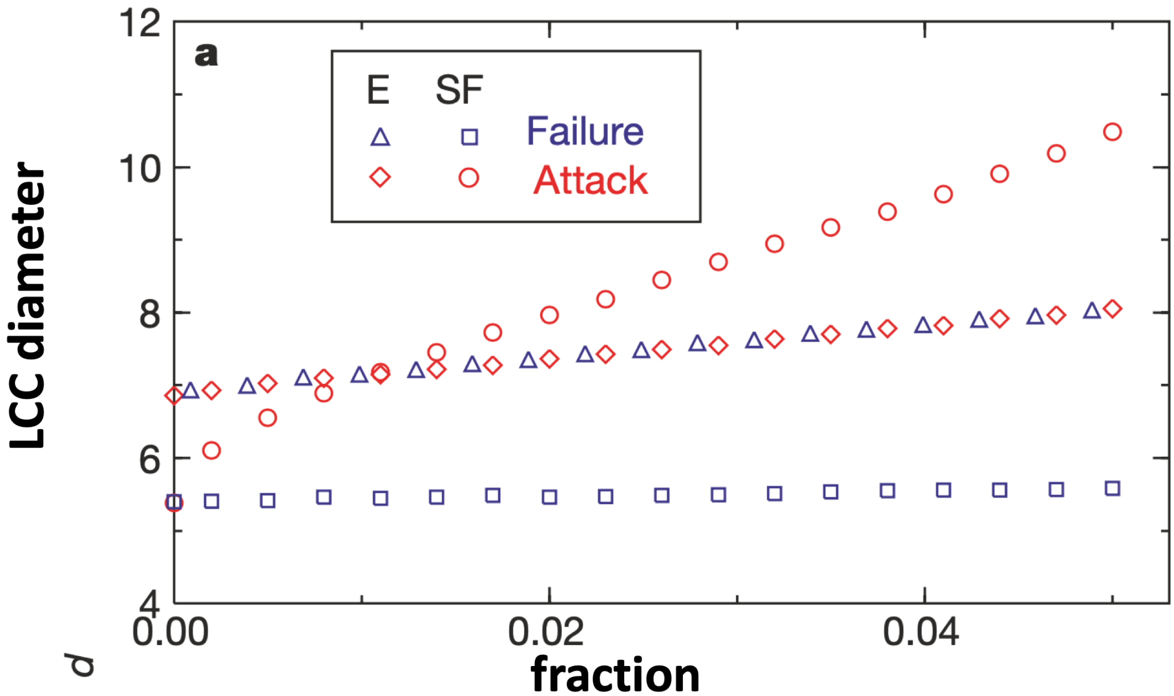

Figure 2: **Changes in the diameter d of the network as a function of the fraction f of the removed nodes. The figure and the title is from [1].** Here your x-axis will denote the LCC diameter d.

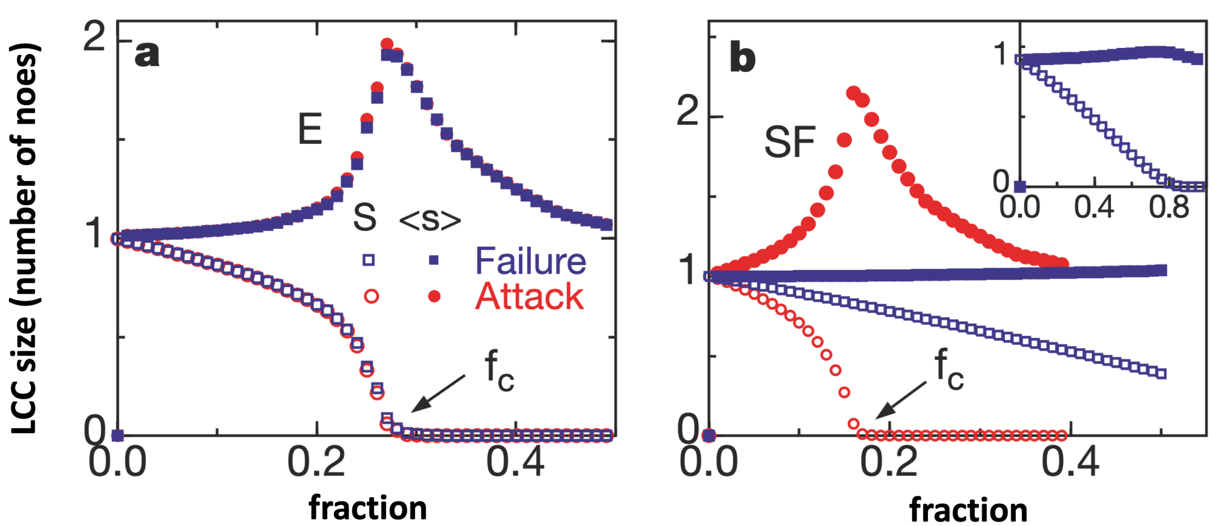

Figure 3: **Changes in the size S of the network as a function of the fraction f of the removed nodes. The figure and the title is from [1].** Here your x-axis will denote the LCC size S.

Penalty of [-10 points]: Your code should be generic. The number of networks to simulate should be defined in the beginning as a hyper-parameter to fix as well as the size of the networks. If the code cannot be executed when changing the number of networks to simulate or their size (i.e., number of nodes), there will be a penalty of 10 points.

References

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[2] Bela Bollobss. The evolution of random graphs. *Transactions of the American Mathematical Society*, 286(1):257–274, 1984.