Exercise set #4 (16 pts)

- The deadline for handing in your solutions is October 15th 2018 23:55.
- Return your solutions (one `.pdf` file and one `.zip` file containing Python code) in MyCourses (Assignments tab). Additionally, submit your pdf file also to the Turnitin plagiarism checker in MyCourses.
- Check also the course practicalities page in MyCourses for more details on writing your report.

1. Percolation in Erdős-Rényi (ER) networks (8 pts)

Erdős-Rényi networks are random networks where N nodes are randomly connected such that the probability that a pair of nodes is linked is p. In network science, the ER random graphs are important because they provide the simplest reference to which one can compare real-world networks. Many interesting real networks are very large (in number of nodes) and sparse (in a sense that single nodes have very few connections as compared to the network size).

We will analyse large and sparse ER graphs, where the average degree $\langle k \rangle$ is some fixed (and small) number, and the size of the network N is very large. Theoretically we will be thinking of networks that are infinitely large, but where $\langle k \rangle$ is still fixed (i.e., $N \to \infty$ and $p \to 0$ such that $p(N-1) = \langle k \rangle$ stays constant). In terms of simulations we will use as large networks as is convenient from the computational point of view, with the idea that larger network sizes will give us better results.

In this exercise, we will analyze the percolation properties of ER graphs. We will especially focus on the *percolation threshold* which is the value of $\langle k \rangle$ where the giant connected component appears (theoretically, this happens when for the largest component of size $S_{\max}$ we don't have $S_{\max}/N \to 0$, but $S_{\max}/N \to s_{\max}$ where $s_{\max} > 0$).

a) (2 pts) Using the idea of branching processes (presented in the lectures) and assumption that large and sparse ER graphs are tree-like calculate the expected number of nodes at d steps away, n_d, from a randomly selected node in an ER network as a function of $\langle k \rangle$ and d. Using this result, justify that the giant component appears in large and sparse ER networks when $\langle k \rangle > 1$.

   Hints:

   - Remember that the degree distribution of an ER network is a Poisson distribution when $N \to \infty$ such that $\langle k \rangle$ is constant.

b) (2 pts) Verify your analytical calculations for n_d using numerical simulations. Calculate the n_d value for $d \in \{0 \dots 15\}$, $\langle k \rangle \in \{0.5, 1, 2\}$, and starting from enough randomly selected nodes to get a good estimate for the expected value. Try out two network sizes: $N = 10^4$ and $N = 10^5$ to see how the size affects the calculations. For this and the following tasks, you can use the Python template `percolation_in_er_networks.py`

c) (1 pts) Explore the range at which the assumption of tree-likeness of the network is valid. This can be done, for example, by calculating the number of edges that nodes at depth d

have that go back to some earlier level in addition to the single edge that connects each node to the level $d-1$, and reporting the average fraction of such edges to all edges that go from depth d to earlier levels/depths. In a perfect tree this fraction is exactly 0. Comment on the results, and their effect on our results to exercise b). What are the other things that make your analytical calculation of n_d to differ from your simulation results?

d) (2 pts) Calculate the component sizes of simulated ER networks, and use this data to (loosely) verify that the percolation threshold of ER networks is at the average degree of $\langle k \rangle = 1$. That is, for $\langle k \rangle < 1$ the largest connected component is small (size being measured as number of participating nodes), and for $\langle k \rangle > 1$ it quickly reaches the network size.

Do this by generating ER networks of size $N = 10^4$ with different average degrees: $\langle k \rangle = [0.00, 0.05, ..., 2.45, 2.50]$. For each of the ER networks, **compute** the size of the *largest* component and **plot** it against $\langle k \rangle$.

e) (1 pt) Another, a more elegant, way to find out when the percolation transition happens is to try to find the point at which the possibility for the largest component size growth is the largest when the control parameter (here $\langle k \rangle$ or p) is changed very little. Think about the situation where $\langle k \rangle$ is changed so slightly that a single link is added between the largest component and a randomly selected node that is not in the largest component. The expected change in the largest component size in this situation is some times called *susceptibility*, and it should get very large values at the percolation transition point. The susceptibility depends on the size distribution of all the other components, and it can be calculated with the following formula:

$$\chi = \frac{\sum_s s^2 C(s) - s_{max}^2}{\sum_s s C(s) - s_{max}}, \tag{1}$$

where $C(s)$ is the number of components with s nodes. Calculate the susceptibility χ for each network generated in exercise d), and again plot χ as a function of $\langle k \rangle$. Explain the shape of the curve, and its implications.

2. Error and attack tolerance of networks (8 pts)

Error and attack tolerance of networks are often characterized using percolation analysis, where links are removed from the network according to different rules. Typically this kind of analyses are performed on infrastructure networks, such as power-grids or road networks. In this exercise, we will apply this idea to a Facebook-like web-page[1], and focus on the role of strong and weak links in the network. In this network, each node corresponds to a user of the website and link weights describe the total number of messages exchanged between users.

In the file `OClinks_w_undir.edg`, the three entries of each row describe one link:
`(node_i node_j w_ij)`,
where the last entry `w_ij` is the weight of the link between nodes `node_i` and `node_j`.

Your task is now to remove links (one by one) from the network in the order of

(i) descending link weight (i.e. remove strong links first),

(ii) ascending link weight (i.e. remove weak links first),

(iii) random order

[1]Data originally from `http://toreopsahl.com/datasets/`

(iv) descending order of edge betweenness centrality (computed for the full network at the beginning).

While removing the links, monitor the *size of the largest component* S as a function of the fraction of removed links $f \in [0, 1]$.

a) (4 pts) **Visualize** S as a function of f in all four cases **in one plot**. There should be clear differences between all four curves.
Hints:
   - For this task, you can use the Python template `error_and_attack_tolerance.py`.
   - In the exercise, `networkx.connected_components(G)` may turn out handy. It returns a list of the components of the network, each of them presented as a list of nodes belonging to the component.
   - Let `components` be the outcome from `networkx.connected_components(G)`. For getting the largest component, you can use `max(componets, key=len)`.
   - Edges of the present network are tuples of three values. For sorting them based on their weight, `sorted` function with `key` parameter can be useful. For more information, check `https://wiki.python.org/moin/HowTo/Sorting`.
   - If you decide to use `networkx.edge_betweenness_centrality`, remember that it returns a dictionary.
   - For sorting the edges based on their betweenness, `np.argsort` might be your choice. Check documentation at `https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html`.
   - The overall running time of this simulation can take up to a couple of minutes but not orders of magnitudes more.

Based on the plots, **answer** following questions:

b) (2 pt) For which of the four approaches is the network most and least vulnerable? In other words, in which case does the giant component shrink fastest / slowest? Or is this even simple to define?

c) (1 pt) When comparing the removal of links in ascending and descending order strong and weak links first, which ones are more important for the integrity of the network? Why do you think this would be the case?

d) (1 pt) How would you explain the difference between the random removal strategy and the removal in descending order of edge betweenness strategy?

Feedback (1 pt)

To earn one bonus point, give feedback on this exercise set and the corresponding lecture latest two day after the report's submission deadline.
Link to the feedback form: `https://goo.gl/forms/f7oSl1HKfayFyHVi1`.

References