Simulation of Complex Systems
Homework 4: Network models

Time and place:

- Assistance: November 30, 2016, 08:00-09:45, MT14
- Assessment: December 9, 2016, 08:00-09:45, F-T7204

Computer lab: The computer lab serves two purposes: To assist with homework and assess homework. The time we have available for computer lab is limited, so it is important that you ***come prepared to the lab***, with specific questions you need to have answered. Everyone involved will appreciate the reduced queue times. You are also ***strongly encouraged*** to team up and collaborate on the problems, to ask your classmates if you are stuck at some point, and to assist classmates in need of advice. But you must write your own implementation and generate your own results!

Examination: During lab hour, presumably in groups, you should demonstrate your homework to a tutor. ***Be prepared for the assessment:*** Generate all figures and/or videos up front. There will be no time for running simulations during assessment. Check that you have answered all questions carefully. Writing down your answers beforehand will many times be helpful. ***No late assignments will be accepted.***

In this homework set we look at the models of networks covered in the lectures and their defining features. For background material, see the lecture plan page for the reviews by Mark Newman and Albért and Barabási, but make sure you use the definitions provided below.

Visualizing networks is an interesting problem in itself. If you want you can write your own method (for some pointers, see this page), but otherwise I suggest you use `gplot` in MATLAB or **GraphPlot** in Mathematica. If you are using some other common language, you should be able to find a package that does the trick.

In most of the exercises it makes most sense to use the adjacency matrix representation for the networks. Make sure to have zeros on the diagonal if you do not intend there to be self-edges (and twos for self-edges, as most of the math assumes each edge has two ends in the matrix). For large networks it really pays off to use a sparse representation, such as those provided by `sparse` in MATLAB and **SparseArray** in Mathematica.

The Erdõs-Rényi random graph: This model has two parameters, n and p. It consists of n nodes and each of the $n(n-1)/2$ possible edges is present independently with a probability p.

The Watts-Strogatz small world model: This model has three parameters, n, c, and p. c should be even. It consists of n nodes situated on a circle, each connected to its c nearest neighbors (so that $c = 2$ gives an ordinary circle). To this graph is then added (we do not use the version with rewiring) random shortcuts through the following procedure: for each edge in the original graph, pick two random nodes and add an edge between these with probability p; or replace if the edge is already there.

The Albért-Barabási preferential growth model: This is a model of network formation with one important parameter, m. It starts with some configuration of $n_0 \geq m$ connected nodes. Then, at each time step for some given number of steps, we add a new node with m new connections. These connections are made with nodes chosen proportionally to their degree, so the probability of choosing node j is $\Pi(k_j) \propto k_j$.[1]

Clustering coefficient: The clustering coefficient for a network is defined as[2]

$$\frac{(\text{number of triangles}) \times 3}{\text{number of connected triples}}.$$

A triangle is a triplet of nodes i, j, k such that $A_{i,j} = A_{i,k} = A_{j,k} = 1$. The number of triangles is straightforward to calculate, for example by noting that all paths of lengths tree can easily be spotted in the cubed adjacency matrix, A^3. The number of connected triples is simply $\sum_i k_i(k_i - 1)/2$ (can

[1] To simulate this efficiently, it is best to keep a list of the ends of edges and choose uniformly from this list, rather than use a non-uniform selection from the list of nodes.

[2] There is a related quantity often seen in the literature, the average clustering coefficient. It is calculated by averaging C_i = (number of pairs of neighbors of i that are connected)/(number of pairs of neighbors of i). We will not use this measure here.

you see why?). Depending on how you perform your calculation, you might need the factor of three or not.

Average path length and diameter: The shortest path l_{ij} between nodes i and j is defined as the least number of edges one has to traverse to get from one to the other. The average path length l of a network is then the average of all such distances, $l = \frac{1}{n(n-1)} \sum_{i \neq j} l_{ij}$. The diameter d of a network is the maximal such distance, $d = \max l_{ij}$.

Exercises:

1. Generate an Erdõs-Rényi random graph. Use as large a network as is practical when it comes to run time for the program. Plot its degree distribution together with the theoretical prediction $P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$. **To demonstrate**: A visualisation of the network (a graph plot) and the degree distribution plot. **(3p)**

2. Generate a small world network and visualise it before and after adding the shortcuts. Use parameters such that the circle structure is clearly visible in both cases. **To demonstrate**: A visualisation of the two networks. **(3p)**

3. Implement the preferential growth model and generate a network with a power-law distribution. Again, use as large a network as is practical. Plot the inverse cumulative degree distribution on log-log scales together with the theoretical prediction $F(k) = 2m^2 k^{-\gamma+1}$ with $\gamma = 3$.[3] This distribution is much less susceptible to noise than the pure degree distribution. **To demonstrate**: a visualisation of the network and the power-law plot. **(4p)**

4. Write a routine for calculating the clustering coefficients of your graphs (the point is to increase your understanding of the concept, so using ready-made code will net no points). Check your algorithm on a few small world networks with $p = 0$ and different n and c, comparing with the exact formula $C = \frac{3}{4}\frac{c-2}{c-1}$. Calculate the clustering coefficient of the graph in the file smallWorldExample.txt on the homepage. Hint: the clustering coefficient is $0.xxx280$, the average clustering coefficient

[3] The inverse cumulative distribution function is $F_X(x) = \Pr[X \geq x]$ and for a power-law distribution is also a power-law with an exponent of one less.

(which you shouldn't calculate) is 0.619846. **To demonstrate**: a graph plot of the example network, the calculated clustering coefficient for the example graph, and a code snippet showing your algorithm. **(5p)**

5. Write a routine for calculating the average path lengths and diameters of your graphs (see above regarding purpose). Calculate the average path length and diameter of the graph in the file smallWorldExample.txt on the homepage. Hint: the average path length is $x.xx232$. **To demonstrate**: The calculated average path length of the example graph, and a code snippet showing your algorithm. **(5p)**

6. The files network1.txt, network2.txt, and network3.txt on the homepage contains real world data from three networks (in sparse format, pairs of connected nodes). One is a social network of email exchanges at a Spanish university [1], one is the Western States power grid [2], and one is the (largest cluster of the) protein interaction network in yeast [3]. Your task is to use the tools you've learned about and constructed so far to identify which of these is which (tools: degree distribution, clustering coefficient, diameter, average path length). The networks are quite large (though not compared to most datasets used in the field) and depending on the speed of your algorithms, you might not be able to calculate e.g. the full average distance (but from experience most do). In that case, think of how you can receive partial information on the properties (and make sure your ideas work on your models). **To demonstrate**: Your calculated results and identification of each network, explained in terms of your results. What are the key properties of each network that sets it apart from the rest? **(5p)**

References

[1] R. Guimer, L. Danon, A. Daz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68(6):065103, December 2003.

[2] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[3] H. Jeong, S. P. Mason, A.-L. Barabasi, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, May 2001.