COMP4641: Social Information Networks Analysis and Engineering

2020 Spring Semester Assignment 1

Date assigned:
Due time: 23:59pm, Apr 01 (Wed), 2020.

IMPORTANT NOTES

- Your grade will be based on the correctness, efficiency and clarity.
- Late submission: 25 marks will be deducted for every 24 hours after the deadline.
- ZERO-Tolerance on Plagiarism: All involved parties will get zero mark.

Problem 1 (40%)

Given the definition of following models:

- Static geographic model: Given N nodes randomly dispersed in the space, each node has a fixed position and each node is connected to its m nearest neighours, m is the same for all the nodes.
- Random growth model: Similar with preferential attachment growth model (Barabasi-Albert model), except that for each new added node, the probability pi that the new node is connected to node i is the same for all the existed nodes $p_i = \frac{1}{\sum_j 1} = \frac{1}{n}$, n is the total number of existed nodes when the new node is added.

In this question we will compare these two models with Erdos-Renyi model and preferential attachment model.

(a) For an Erdos-Renyi model with N nodes and link probability p, what is the expected degree $\bar{k}$?

(b) Consider a static geographic model with N nodes and $m \approx \bar{k}$, then compare it with the previous Erdos-Renyi model, whether the following statements are true or false, and give your explainations?

1) the static geographic model has stronger locality

2) the static geographic model has shorter average shortest path

(c) Compare random growth model with Erdos-Renyi model, explain why links are unevenly distributed in random growth model? Which model has more nodes with degree 1?

(d) Compare random growth model with preferential attachment model, for the following two pictures , each has 500 nodes, which model fits best to each picture and explain why?

(e) Suppose for random growth model, when each new node is added, it's connected to m other nodes, and at each time only one new node is added, then at time t, consider the degree $K^{t_0}(t)$ of the node which was added at time t_0, it satisfies

$$\frac{dK^{t_0}(t)}{dt} = \frac{m}{t}$$

with $K^{t_0}(t_0) = m$, give the solution for $K^{t_0}(t)$?

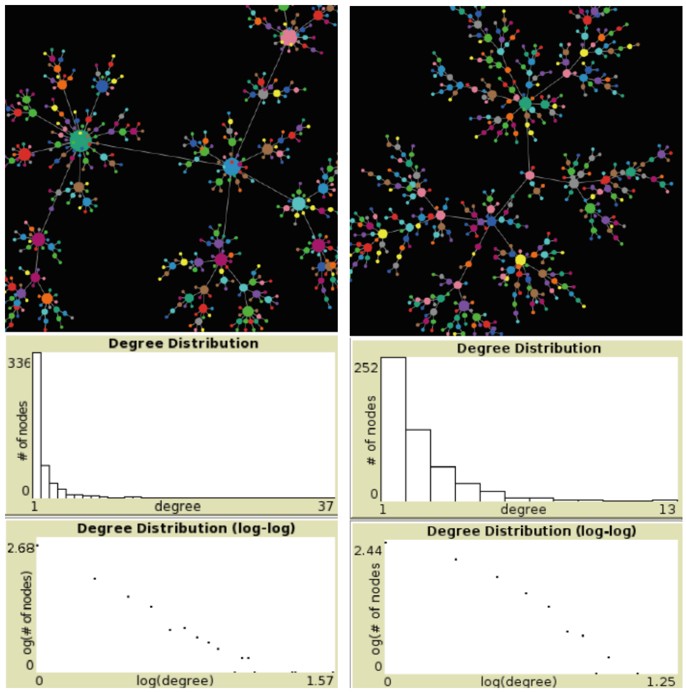
(f) For preferential attachment model,

$$K^{t_0}(t) = m\left(\frac{t}{t_0}\right)^{\frac{1}{2}}$$

when t_0 is small, $K^{t_0}(t)$ is large, which means old nodes have large degrees at time t. Now consider when t is large, the percentage of nodes with degree smaller than $K^{t_0}(t)$ is

$$P(k \leq K^{t_0}(t)) = \frac{t - t_0}{t} \tag{1}$$

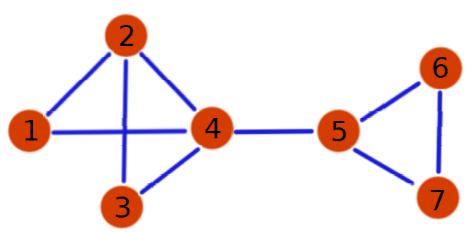
explain why and give the degree distribution $P(k)$?



(a) (b)

Figure 1: (a) Model 1; (b) Model 2.

Problem 2 (30%) Given the following network,



(a) What is the normalized degree centrality for node 2? What are the normalized betweenness centrality, normalized closeness centrality and clustering coefficient for node 4?

(b) What is the value of network constraint for node 5?

(c) What is the modularity matrix for this network? You can check with the provided notebook's results and also see how a brutal force search is used to find the optimum community structure with maximum modularity.

Problem 3 (Programming) (30%)

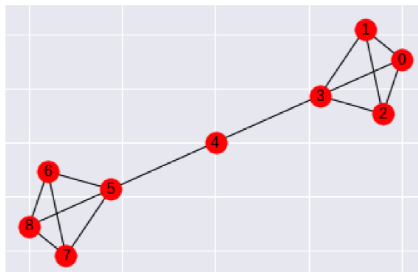
Given a adjacency matrix of some undirected graph, you are required to write a python program that performs the following two functions:

(a) Implements the Girvan-Newman algorithm and outputs the resultant hierarchical decomposition of the network. If there are multiple edges have the same highest betweenness score, remove all of them simultaneously.

(b) Implements the calculation of modularity, and outputs the corresponding cluster structure.

Implementation Details:

For example, given the following graph (**Note: A graph different from the example graph above will be used in grading. So do NOT hard-code this graph in your program**)



- Your program should first read the input file:

```
9
0 1 1 1 0 0 0 0 0
1 0 1 1 0 0 0 0 0
1 1 0 1 0 0 0 0 0
1 1 1 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0
0 0 0 0 1 0 1 1 1
0 0 0 0 0 1 0 1 1
0 0 0 0 0 1 1 0 1
0 0 0 0 0 1 1 1 0
```

  - the first line contains the number of nodes;
  - the remaining lines contain the adjacency matrix.

- Your program output should have two parts:

  1) The first part outputs the hierarchical decomposition produced by the Girvan-Newman algorithm, in the following format:

```
network decomposition:
([0, 1, 2, 3], [5, 6, 7, 8], [4])
([0], [1], [2], [3], [5], [6], [7], [8], [4])
```

  2) The second part outputs the modularity results, in the following format:

```
3 clusters: modularity 0.4209
9 clusters: modularity -0.1148
optimal structure: ([0, 1, 2, 3], [5, 6, 7, 8], [4])
```