Lecture Notes: Social Networks: Models, Algorithms, and Applications
Lecture 10: February 16, 2012
Scribes: **Mauricio Monsalve** and Yamini Mule

1 Introduction: degree distributions

We will move onto studying different random graph models, which are special because of their degree distributions: the Configuration Model [5] and its variants and the Preferential Attachment models.

So far we have only studied the following random graph models:

- The Erdös-Renyi models ($G(n, m)$ and $G(n, p)$)
- The Small World models:
  - The Watts-Strogratz model
  - The Kleinberg model
  - The Liben-Nowell model

Of these models, we have studied their clustering coefficients and their Small World property.

Now, it could be nice if we could, at least, replicate the degree distributions observed in real world networks. Some people, as the physicist Newman, believe that the degree distribution of real world networks, which is often power law, is an important property to study.

Definition 1 *The* degree sequence *of a graph $G = (V, E)$ is the sequence of degrees of vertices V written in non-increasing order.*

For example, consider the graph illustrated in Figure 1. Its degree sequence is $(3, 3, 2, 2, 2)$.

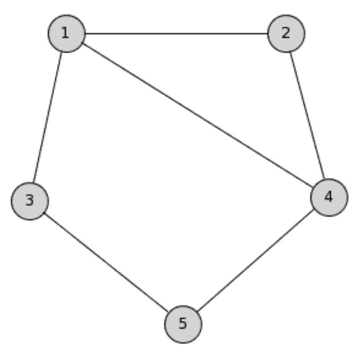

Figure 1: A graph consisting of five vertices and six edges.

Given a sequence of non increasing integers, how can we tell that there exists a *simple graph* that embodies that degree sequence? Not all non increasing sequences of non negative integers can be realized as degree sequences of simple graphs. These sequences that can are called *graphical sequences.*

The degree sequence of an existing graph is, of course, graphical. The problem whether the sequence is graphical or not arises when the degree sequence is synthetic. Such sequences can be generated after sampling the degrees from a degree distribution function, for example, fitted from a real world network.

Let us study degree sequences for a bit. Consider the degree sequence $(5, 3, 1, 1, 1)$. Is it graphical? No, because the highest degree vertex would have more edges than existing vertices in the graph. Also, observe that the sum of the degrees is odd, when it should be an even number. So, let us modify it by adding a 1 at the end: $(5, 3, 1, 1, 1, 1)$. The degree sequence is still not graphical. After connecting the highest degree vertex to the rest of the vertices, the second highest degree vertex does not have vertices to connect to.

Remark The Erdös-Gallai Theorem characterizes graphical sequences [3]. And the Havel-Hakimi algorithm is an efficient way to test whether the degree sequence is graphical [4].

2 The Configuration Model

We will now describe the basic Configuration Model for generating random graphs with specific degree distributions.

INPUT: A sequence $(d_1, d_2, ..., d_n)$ of integers such that $d_1 \geq d_2 \geq ... \geq d_n$ and $\sum_{i=1,n} d_i$ is even.

OUTPUT: A random graph (not necessarily simple) with degree sequence equal to $(d_1, d_2, ..., d_n)$.

The Configuration Model algorithm works as follows. Create vertices $V = \{1, 2, ..., n\}$ and assign them *stubs* or *half edges* according to the sequence $(d_1, d_2, ..., d_n)$, as in Figure 2. (These stubs or half edges are edges that are connected on one side while the other side remains free.) Then, to ellaborate the random graph, pick any two stubs uniformly at random, and connect their free ends. Then, these two stubs became one edge. Repeat this process until no free stubs are left.

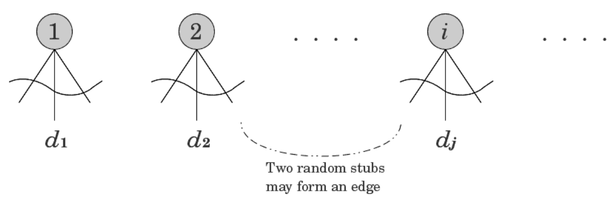

Figure 2: Vertices and stubs in the Configuration Model.

Observe that the algorithm allows loops (created when picking two stubs from the same vertex) and multiple edges (created when picking pairs of stubs from the same pairs of vertices). Thus, the generated graphs are not simple graphs but hypergraphs.

Also observe that the algorithm will stop as long as the sum of the degrees in the input degree sequence is even. The sequence does not need to be graphical.

Discussion In practice, we will be working most likely with networks described by simple graphs, yet this model generates hypergraphs. So, what should we do to deal with self loops and multiple edges?

1. One answer is to just allow them in the graph. All the mathematical analyses of the model follow when self loops and multiple edges are allowed.

2. Yet note that the degree sequence is not respected exactly when the hypergraph is reduced to a simple graph. But when the number of vertices is very large, the sampled graphs are closer to being simple graphs. A result states that, as the number n of vertices tends to infinite while the mean degree $c =< d_i >$ is left constant, then the number of self loops and multiple edges tend to zero.

3. Another answer consists in removing such special edges. This can be done through swaps, i.e. by rewiring the special edge to another randomly chosen vertex. This action preserves the degree sequence, and such small change should not perturb too much the results of the mathematical analyses. However, swapping does not guarantee that we obtain a simple graph. (For example, it cannot guarantee it when the degree sequence is non graphical.)

3 The Chung-Lu variant

Sometimes, we may be more interested in generating random graphs with an expected degree distribution rather than an exact degree distribution or sequence. We now introduce the Chung-Lu variant of the Configuration Model that generates graphs with loops but not multiple edges, which degree distributions are, in expectation, equal to the one supplied.

INPUT: A sequence $(d_1, d_2, ..., d_n)$ where $d_1 \geq d_2 \geq ... \geq d_n$ and $\sum_{i=1,n} d_i$ is even.

OUTPUT: A graph which expected degrees are given by the sequence $(d_1, d_2, ..., d_n)$.

The generation algorithm is as follows. Create vertices $V = \{1, 2, ..., n\}$, so that vertex i is related to d_i. Then, we visit each pair of vertices only once, and *flip a coin* to see whether adding a vertex or not. We add edge (i, j) with probability proportional to $d_i d_j$, i.e. $c d_i d_j$, where c is a constant. If we want vertex i to have expected degree d_i, then we solve c for:

$$deg(i) = \sum_{j=1,n} c d_i d_j = c d_i \sum_{j=1,n} d_j = d_i.$$

Then, using $c = 1/\sum_{i=1,n} d_i = \frac{1}{2m}$ (twice the number of edges) enables the sampling technique to generate graphs with expected degree sequence equal to the given one. (Observe that it is now possible to prevent loops, and a specific c_i constant should be used for each vertex instead of c.)

Note that this model relates to the original Configuration Model just as the ER model $G(n, p)$ relates to $G(n, m)$. This analogy also applied to the mathematical tractability of the model; the Chung-Lu model is much easier to analyze than the Configuration Model.

Let us try to compute the probability there is an edge between vertices i and j in the Configuration models. In the Chung-Lu model, this is simple: it is just $d_i/(2m - 1)$. However, the calculation is harder in the original Configuration Model.

Let us first compute the probability a stub of i is connected to some stub of j. This is just $j/(2m - 1)$. Let us now move onto the probability that any stub of i is connected to some stub of j. This is the same as the complement of the probability of not connecting to j. The probability of the first stub of i of not connecting to j is $1 - j/(2m - 1)$. The probability of the second stub of i of not connecting to j is $1 - j/(2m-2)$, and so on. Then, these probabilities are multiplied, and the complement is taken. Unfortunately, the formula is not easy to write because the probabilities that each stub is not connected to some stub in a vertex are not independent. This is the disadvantage of the original Configuration Model to the Chung-Lu variant.

4 Clustering coefficient

We now study the clustering coefficients of the configuration models. We start by computing the degree distribution of a neighbor. Suppose you start at a vertex (arbitrary) and walk along an incident edge (also arbitrary). What is the degree distribution of the vertex you have arrived to? In other words, what is the probability this vertex has degree k?

Let us assume we are in vertex i and arrive to vertex j, in the Chung-Lu model. The probability of getting to j from any stub is $d_j/(2m-1)$. Let $k = d_j$. Now, let us find the probability of arriving to any vertex of degree k. Note that the number of vertices of degree k is proportional to p_k. Then, the probability of getting to any vertex of degree k is equal to $np_k k/(2m-1) \approx np_k k/2m$. Since $2m = \sum_i d_i = n\left(\sum_i d_i/n\right) = n\langle k\rangle$, then we have that the probability is $np_k k/2m = p_k k/\langle k\rangle$.

Observe that the contact is more likely to have more edges than the average vertex! This is a paradox present in many social networks, and is called the *Friendship Paradox.* It can be expressed as *your friends are likely to have more friends than you.*

	vertices	average degree	average neighbor degree
biologist cell net	1,520,252	15.5	68.4

Table 1: Average degree versus average neighbor degree

Related, there are disease surveillance papers that make use of this paradox [2, 1]. In particular, a paper by Christakis and others stated that in order to accurately monitor and contain disease incidence (the number of people currently infected by the disease), instead of picking people at random and monitoring and immunizing them, it was better to to pick random people and then monitor and immunize their friends [1].

References

[1] Nicholas A. Christakis and James H. Fowler. Social network sensors for early detection of contagious outbreaks. *PLoS ONE*, 5(9):e12948, 09 2010.

[2] Reuven Cohen, Shlomo Havlin, and Daniel ben Avraham. Efficient immunization strategies for computer networks and populations. *Phys. Rev. Let.*, 91:247901, December 2003.

[3] Paul Erdös and Tibor Gallai. Graphs with prescribed degree of vertices.

[4] S.L Hakimi. *J. SIAM Appl. Math*, 10.

[5] Mark Newman. Random graphs with general degree distributions. In *Networks, an Introduction*, chapter 13. Oxford University Press, 2010.