CS 165 Project 3 Report

Huy Minh Tran

June 1, 2020

1 Introduction

1.1 Purpose

The purpose of the project is to learn about the social network structure generated by the Erdős-Rényi model and analyze its diameter, degree distribution and clustering coefficient. A social network is a network in which vertices represent people and edges represent a connection between two people. For instance, a Facebook profile is a vertex and if that Facebook profile is a friend with another Facebook profile, then there is an edge between the two profiles. This is used in many studies to determine people who share some common attribute are clustered or grouped together.

One popular property of a social network is the small world property, or six degree of separation. This property states that the shortest path between any two arbitrary vertices of a social network is at most six. The small world property is known to apply to modern social networks.

1.2 Experimental setup

The method to generate random numbers is to use random_device, default_random_engine and uniform_real_distribution or uniform_int_distribution from the standard library of C++. Each algorithm is performed 5 different times and the sum of diameter, degree distribution and clustering coefficient are divided by 5 to get the average.

1.3 Generating plot

The plot for diameter and clustering coefficient are on a lin-log scale. The size of the graph is plotted as the x-axis and the diameter and clustering coefficient are plotted as the y-axis, respectively. The plot for degree distribution is on a lin-lin scale and a log-log scale. The size of the graph is plotted as the x-axis and the degree distribution is plotted as the y-axis.

1.4 Content

The next section will analyze the model generated by Erdős-Rényi algorithm. Each content is organized as follows:

1. Description: pseudocode and explanation of the algorithm.
2. Experimental result: plot of the algorithm.
3. Analysis: the analysis of the algorithm of the model based on the data and the plot.

1.5 Erdős-Rényi Model

Erdos-Renyi is a model for random graphs that has n nodes, every pair of nodes is connected independently with a probability p and each node has an average degree of d = (n-1)p. Below is the pseudocode for generating the Erdős-Rényi model:

```
Graph G(n, p){
   // n is number of vertices
   // p is probability with which to add the edge

   Initialize graph g with n vertices
   int v = 1, w = -1
   while v < n:
      draw r uniformly at random in range of [0,1)
      w = w + 1 + std::floor(log(1-r)/log(1-p))
   while w >= v and v < n:
      w = w - v
      v = v + 1
   if v < n:
      add v and w to g
   return g
}
```

2 Degree Distribution

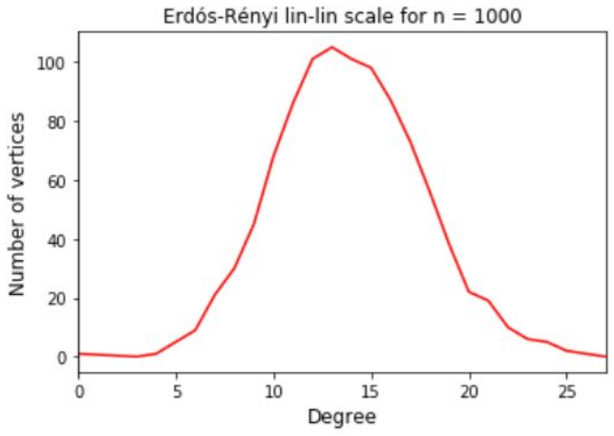
Degree distribution is the probability distribution of these degrees over the whole graph, meaning a degree can show how many vertices have that degree in the graph. The expected average degree of a graph is np = 2ln(n). P is the probability of an edge being independent from other edges and n is the number of vertices. Below is the pseudocode for getting the degree distribution of a graph:

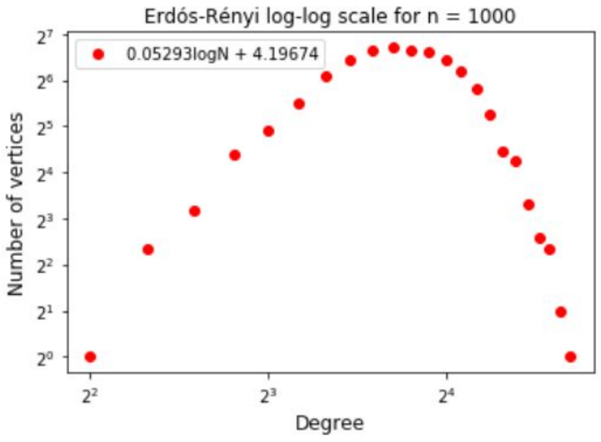
```
HashTable degree_distribution(g){
   // g is a graph
   HashTable dd
   for every vertex in g
      dd[degree of v] = dd[degree of v] + 1
   return dd
}
```

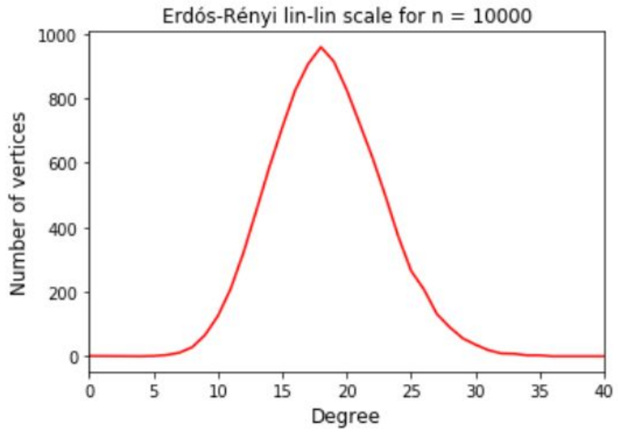
The table below shows the expected average degree of a graph with 1000, 10000 and 100000 vertices.

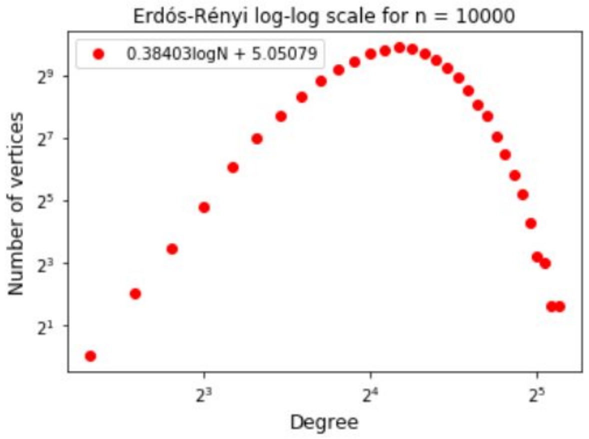
Size n	Expected Average Degree
1000	2ln1000 = 13.8
10000	2ln10000 = 18.4
100000	2ln100000 = 23.0

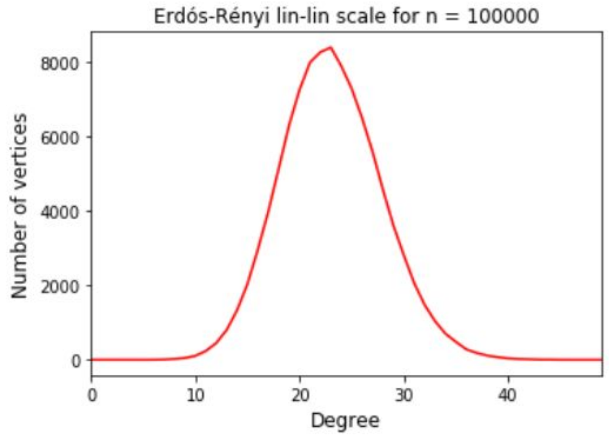
2.1 Experimental Result

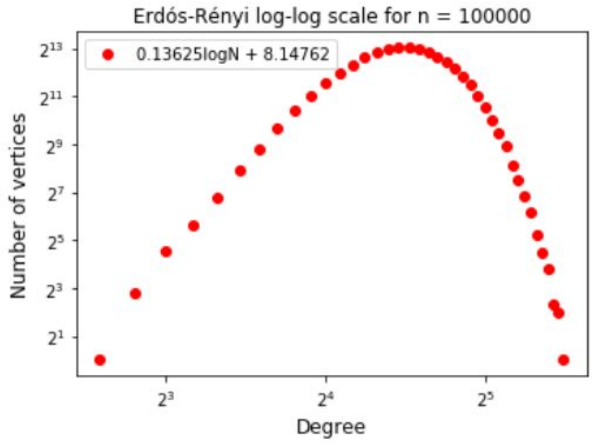












2.2 Analysis

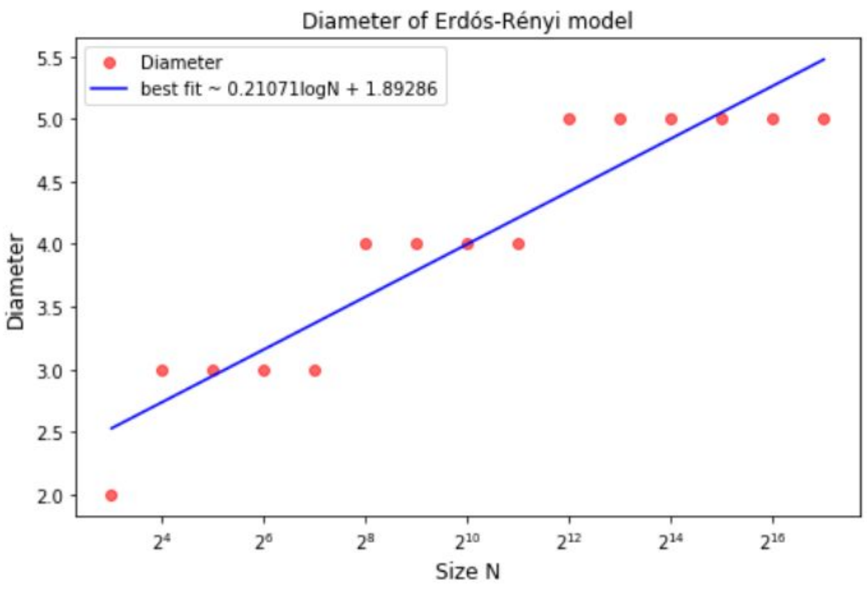
It can be seen that the graphs of degree distribution on the left follow a normal distribution. For n = 1000, 10000 and 100000, most vertices have the degree of 13, 18 and 23, respectively, which matches the expected average degree calculated above. However, the graphs of degree distribution plotted by a log-log scale have the shape of a parabola. Therefore, it is concluded that the degree distribution of Erdos-Renyi graphs does not follow the power law.

3 Diameter

The diameter is the greatest distance between any pair of vertices in the graph. Below is the pseudocode for getting the diameter of a graph:

```
int get_diameter(g){
   let n be a random node of g
   int d_max = 0
   bool changed = True
   while changed:
      let x be the furthest node away found by n using BFS
      if distance of x == d_max:
         changed = False
      else if d_max < distance of x:
         d_max = distance of x
         n = node
   return d_max
}
```

3.1 Experimental Result



3.2 Analysis

The first thing that can be noticed about the graph is the change in the diameter between size 2^7 and 2^8, 2^{11} and 2^{12}. This suggests that given a graph with a range of vertices, there is a specific diameter for that range. Therefore, it can be concluded that the diameter increases by some step function of size n. However, the best fit line shows that the logarithmic growth of the graph is closely approximate the actual diameter, so it can be said that diameter increases by a logarithmic function of size n.

4 Clustering Coefficient

Clustering coefficient is a measure of the degree to determine if the graph is clustered or not. It is calculated by three times the number of triangles in the graph over the number of paths of length 2. Below is the pseudocode on getting the clustering coefficient of a graph:

```
float get_clustering_coefficient(g){
   let float cc = 0.0
   int two_edge_path = get_two_edge_path(g)
   int triangles = get_triangle(g)
   return 3.0 * triangles / two_edge_path
}
```

```
int get_two_edge_path(g){
   let count = 0
   for every vertex in g:
      let d = degree of vertex
      count = count + d * (d - 1) / 2
   return count
}
```

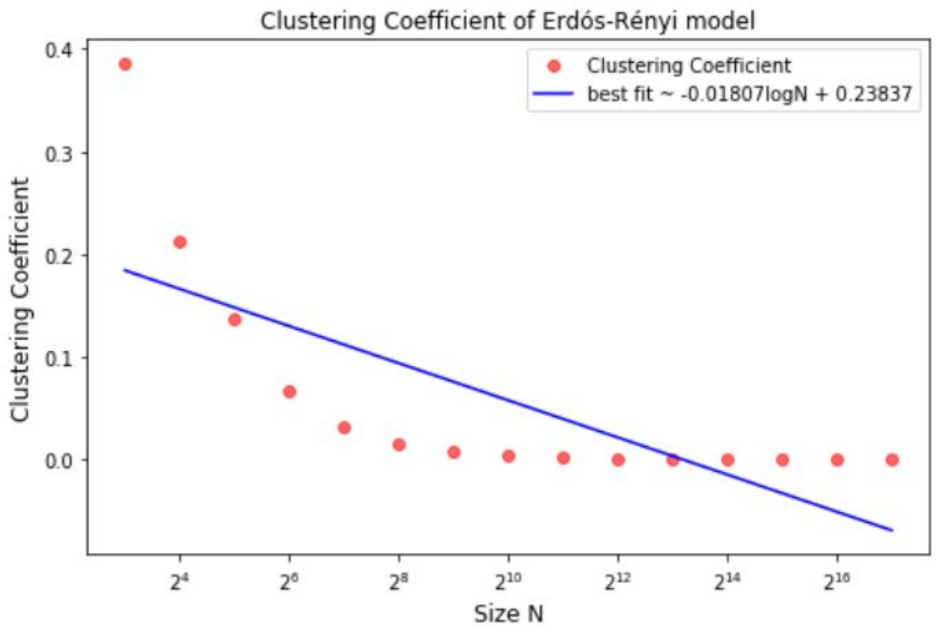
```
int get_triangle(g){
   initialize L: the degeneracy list
   initialize d: a hash table of every vertex in g and its degree
   initialize D: a hash table of degree in d and a list of the vertex
having that degree
   initialize N: the neighbor a vertex that come before that vertex in L
   initialize S: a set of vertex in L

   for every vertex in g:
      let i be the non-empty index of D
      let v = D[i].front()
      add v to front of L
      remove v from D[i]
      insert v to S
      initialize neighbors as a list of neighbor of v
      for neighbor n of v:
         if n is in S:
            d[n] = d[n] - 1
            move n to the appropriate D[degree of n]
            N[v].push_back(n)

      let count = 0
   for every vertex in L:
      let u and w be neighbor of vertex
      if g has edge(u,w):
         count = count + 1

return count
```

4.1 Experimental Result



4.2 Analysis

It can be concluded directly from the graph that the clustering coefficient of the graph decreases by a logarithmic function of n because the best fit line has a negative slope of -0.01807. It can be understood that as the size of the graph grows, the less triangles it produces but the graph generates more number of two edges path, which makes the result of the calculation very small. Therefore, the more the graph grows, the less cluster it is.

5 Conclusion

It is shown that the Erdős-Rényi model has the small world property, which is having the diameter less than 6. This also shows that the small world property actually exists in the social network. The degree distribution followed a normal distribution with the average degree of 13, 18 and 23 for a graph of 1000 vertices, 10000 vertices and 100000 vertices, respectively. Hence, in terms of social networks, it means a lot of people are very connected and not too many people know significantly more or less than the average. Finally, the clustering coefficient decreases as the size of the network grows. This indicates that the people in the network may know many other people, essentially, they are not very close to each other.