CCS - Network Analysis Assignment

Fatema Alkhanaizi

November 6, 2018

Clearly document what you have done. For each question you should report on what you did and include, as needed, illustrative plots.

Question 1

Ring Group Graph Degree Distribution when $p + q = 0.5$, $p > q$

- Investigate the degree distribution of Ring Group Graphs for p + q = 0.5, p ¿ q.
- Decide which values of m, k, p and q to investigate.
- You should report on how the structure changes as p and q vary and whether the same effects are found for different values of m and k.
- Use plots to illustrate your observations.

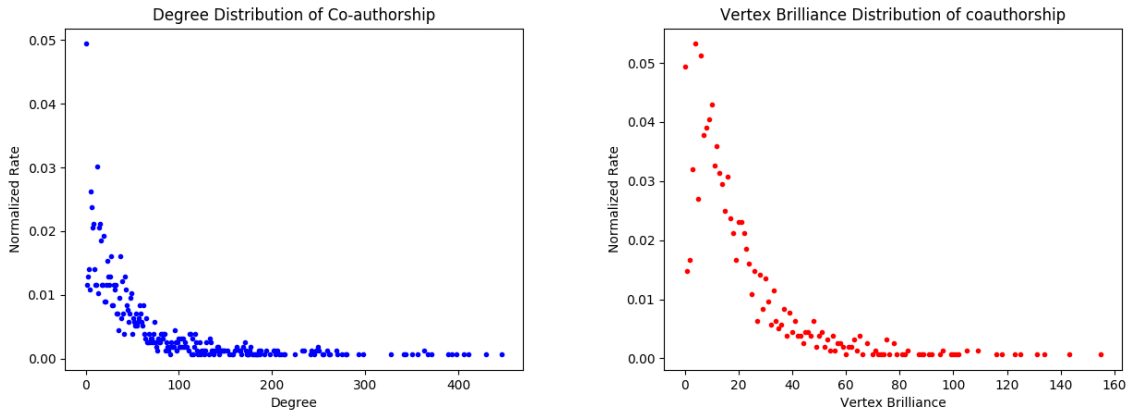

Figure 1: Loaded Co-authorship Graph including vertices with no edges

Diameter of Ring Group Graph and p (for a fixed q, $p > q$)

- Investigate the relationship between the diameter of Ring Group Graphs and p (for fixed q, p ¿ q).

Question 2

Distribution of Vertex Brilliance

Calculating Vertex Brilliance

- Construct the undirected graph defined in coauthorship.txt. Ignore edge weights.

- For the graph from coauthorship.txt, investigate the distribution of vertex brilliance.
- For each of the following types of graph
  - PA Graphs
  - Ring Group Graphs

  create examples with approximately the same number of vertices and edges as the graph from coauthorship.txt and investigate the distribution of vertex brilliance. Comment on what you find.

The vertex brilliance was caluculated based on a similar approach to calculate the diameter by using breadth-first search but by only checking the neighbours of the vertex's neighbours. The algorithm simplified: 1. create an empty dictionary ¡ this dictionary will hold the weight vertices for the k-star of the current vertex 2. populate the dictionary with v's neighbours and give each vertex in the dictionary a weight of 0 to begin with ¡ this weight will be used to determine which vertex to exclude to get the maximum k-star, the weight corresponds to how many v's neighbours the vertex is connected to 3. repeat until maximum k-star is found - for each vertex in the k-star dictionary go through its neighbours. If one of the vertex neighbour is a key in the k-star dictionary substract 1 from the weight of the vertex (I gave it negative value as just a notion that this is undesired vertex thus a negative affect but it can be calculated with the same manner by adding 1 but then the rest of the code will be modified so it get the maximum value from the k-star values instead of the minimum) - find the minimum weight (value) of the k-star dictionary - if the minimum value is 0 then stop as this indicates that non of the vertices in the dictionary are adjacent thus it is the maximum k-star set of vertices (ther can be more than one set but we only care about the length so the actual vertices of the maximum set don't have any actual value for us in this case) so stop the loop - else find the vertex with the minumum weight and remove it from the dictionary (this vertex is the most influential as it connects to the most number of vertices and by removing it the rest of vertices will be less adjacent, only the first found vertix will be removed) and reset all the other vertices values to 0 4. return the length of the dictionary This is essentially an algorithm to get the maximum independent set of sub-graph that only contains v and it's neighbours. I opted to using my own algorithm instead of using networkx's implementation.

Creating Vertex Brilliance Distribution

For each generated graphs (PA graph and Ring Group graph), the vertex brilliance distribution was averaged and normalized over 100 instances. The distribution was created following the same code as the degree distribution.

Vertex Brilliance Results

Co-authorship Graph

The co-authorship graph was loaded by following code provided in the lectures; it was just basic file-reading and making sure that the edges were undirected. The following figures are the degree distribution (in blue) and the vertex brilliance distribution (in red) of the graph:

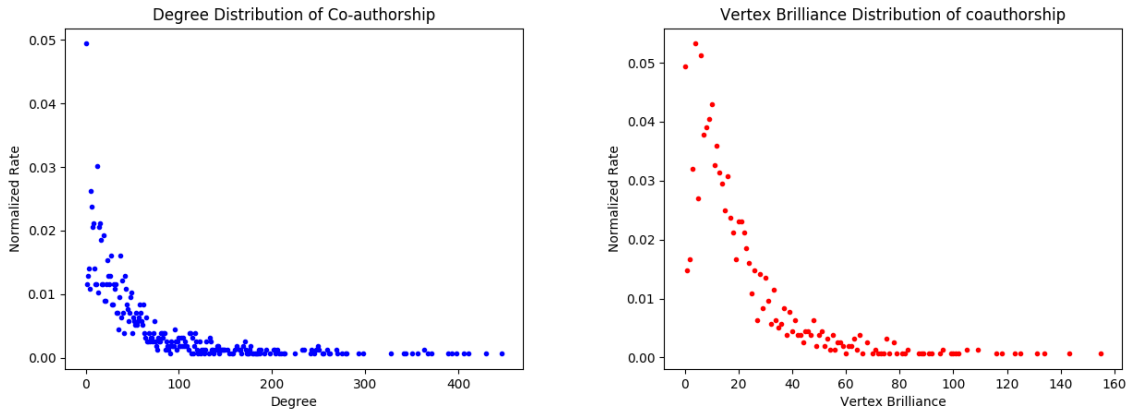

Figure 2: Loaded Co-authorship Graph including vertices with no edges

The general shape of both graphs is skewed to the right.

PA Graph

As the implementation of the PA Graph that we used in the lectures was directed, I modified networkx implementation and matched it to the PA Graph specifications provided in lecture:

- I made the first 1 to m vertices fully connected and repeated each vertex m times in a list for selecting the new edges for the remaining vertices
- For the remaining m + 1 to n vertices, I randomly picked a vertex from the previous list m times (created an edge). For each iteration until n is reached, I added the new vertex m times in the selection list along with its neighbours.

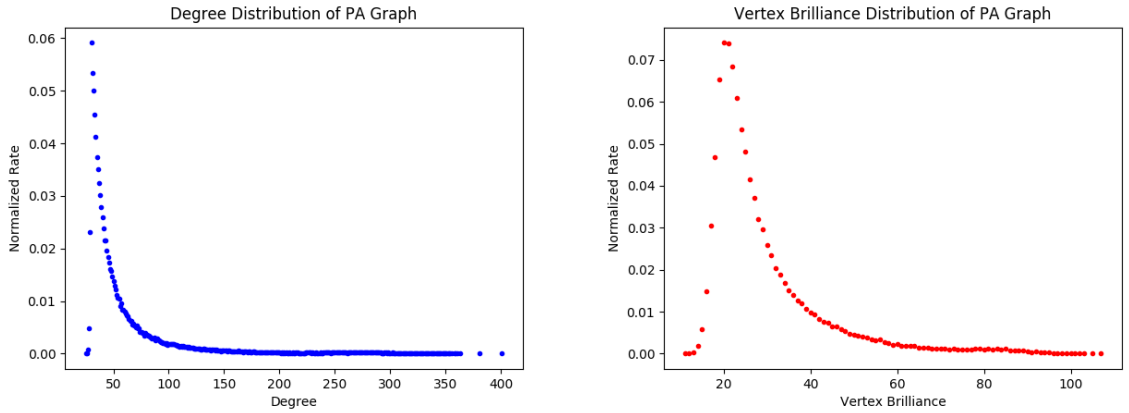

Figure 3: PA Graph parameters: m = 1559, m = 30

Both distributions has been averaged over 100 instances of the PA graph. The resulted graphs for both degree and vertex brilliance distributions are skewed to the right. The co-authorship graph strongly resembles the PA graph based on the distributions.

Ring Group Graph

To get approximately the same number of edges and nodes as the co-authorship graph, the following parameters were used: m = 60, k = 26, p = 0.22, q = 0.03; there are other possible

parameters but their distributions are similar to the distributions with the paramaters used here (in terms of shape).

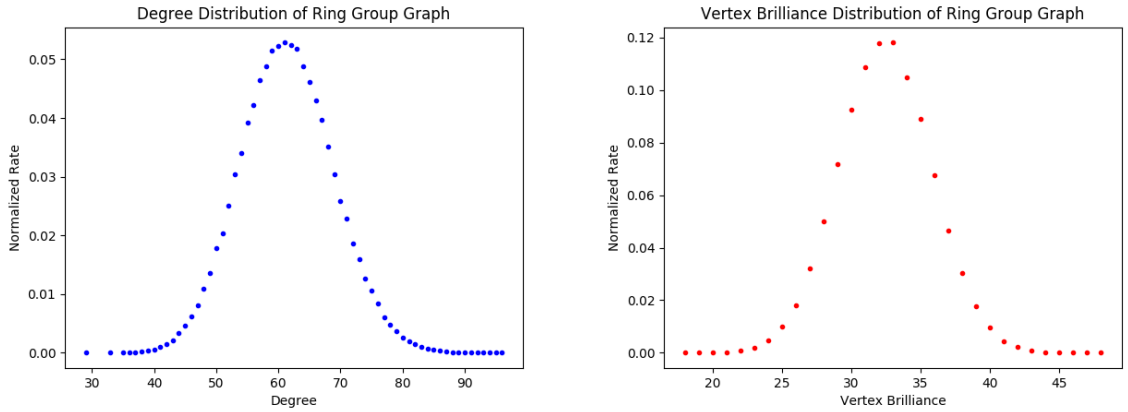

Figure 4: Ring Group Graph parameters: m = 60, k = 26, p = 0.22, q = 0.03

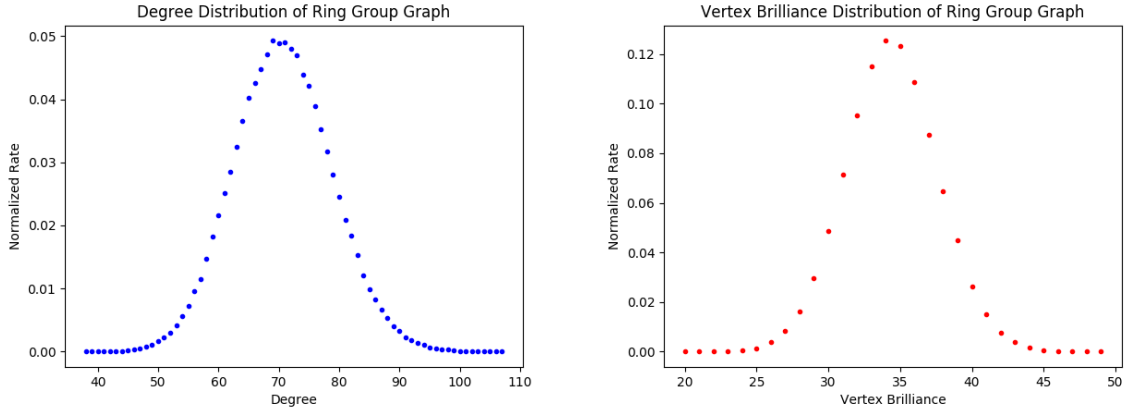

Figure 5: Ring Group Graph parameters: m = 120, k = 13, p = 0.26, q = 0.04

Both distributions are bell shaped (binomial distribution).

Based on the previous graphs, it can be concluded that the vertex distribution has the same shape as the degree distribution and the co-authorship graph closely resembles the PA graph.

Question 3

The search time for a given start and target is the total number of queries made in reaching the target from the start (the number of intermediate vertices is not counted). The search time for a graph is the average search time over all pairs of start and target vertices. Clearly these times depend on the algorithm used for deciding when to query, when to move to a new vertex and how to decide which vertex to move to. The aim is that the search time is as small as possible. Note that the array of neighbours of a vertex should not be considered ordered. If you create graphs in such a way that the neighbours are ordered, then either randomize the ordering or, when querying the array of neighbours, choose a random member of the array (from amongst those not already queried).

- describe an algorithm for searching in the graphs. (I expect that you will have different algorithms for the different types of graphs.)
- You should explain why you believe your strategy might be effective and implement and test it on many instances. You can choose the parameters yourself as long as you are not perverse for example, the groups in the Ring Group Graph should not be of size 1 or n and it is acceptable that all your testing for a particular type of graph is on instances generated with the same parameters.
- As searching on graphs that are not connected can be impossible, you should choose parameters so that the graphs are very likely to be connected.
- Plot search time against the number of instances that achieve that time. Comment on your plots. It is acceptable to estimate search time by looking at only a sample set of pairs of vertices.
- Credit will be given for the effectiveness of the algorithm you design, and also, independently, for your explanation of the rationale behind the design.

Search Random Graph

for Random Graphs, each vertex is labelled with a unique integer between 1 and n.

Algorithm

if $i \geq maxval$ **then**
  $i \leftarrow 0$
else
  if $i + k \leq maxval$ **then**
    $i \leftarrow i + k$
  end if
end if

Search Ring Group Graph

for Ring Group Graphs, each vertex is assigned a unique integer and the label of its group (see Question 1)