Question 1: Ring Group Graphs

Task

- Question 1 presents the necessary steps and parameter in order to construct a Ring Group Graph.
- I am tasked with utilising these steps in order to fulfil the objective of investigating the degree distribution of Ring Group Graphs for $p + q = 0.5$, with $p > q$.
- Additionally, I was tasked with investigating the relationship between the diameter of Ring Group Graphs and p, in this case q is a fixed value and we have $p > q$.

Implementation

Constructing a Ring Group Graph

When constructing a Ring Group Graph, we are provided with four parameter m, k, p and q and the graph is constructed by initially creating mk vertices and paritioning them into m group each of size k - with the groups being labelled from 0 to $m - 1$. We then add edges between all pairs of vertices in the same group with probability p, and add edges between all pairs of vertices in adjacent groups with probability p.
In order to achieve this I created a Graph class to which I can add vertices. I also created a Vertex class - the vertex class stores the unique identifier, label and neighbours of the vertex. I iteratively created mk instances of the vertex class and add them to the graph. I then create an (undirected) edge between each vertex if the difference between their labels was 0, 1 or -1 providing a randomly generated value between 0 and 1 was less than the probability p, or if the randomly generated value was less than the probability q.
In order to calculate the degree distribution of a single graph, I simply navigate through each vertex, counting the number of neighbours the vertex has (as this is the degree). I then increase the frequencey of this degree (in our graph) by 1, and later normalise the results acquire so that the information presented in the graph are the degree distributions illustrated below.

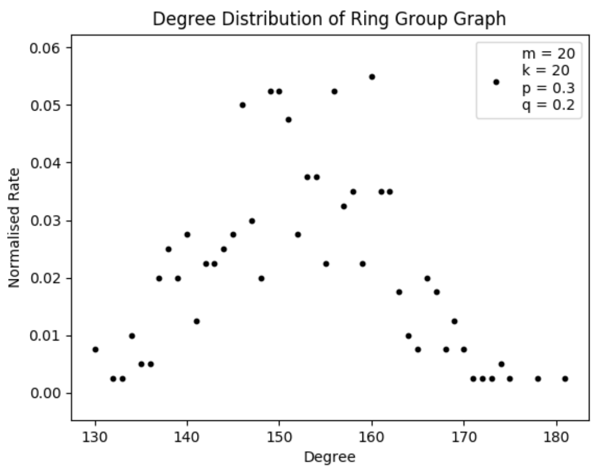

Figure 1: Degree Distribution of Ring Group Graph

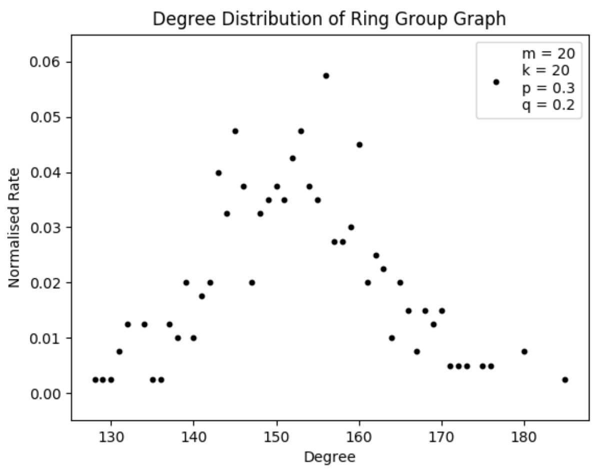

Figure 2: Degree Distribution of Ring Group Graph

Above we see Figure 1 and Figure 2, both of which are examples of degree distributions for a Ring Group Graph (with $m = k = 20$, $p = 0.3$ and $q = 0.2$) as we see despite these distributions having similar shapes they vary quite substantially considering they present degree distributions for graphs created with identical parameters. This is due to the element of randomness when creating edges between pairs of nodes. In order to acquire a more useful distribution I repeatedly produced degree distributions and plotted the average distribution, in all graph instances a much more representative degree distribution was acquirable in around 100 iterations. Below we see Figure 3 a degree distribution for the graphs above except in this case they have been averaged over 100 iterations (as described):

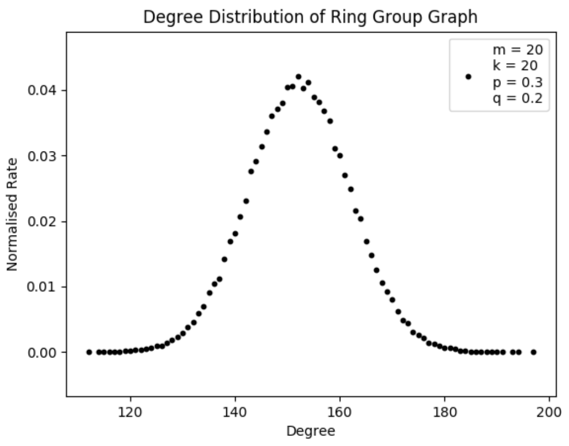

Figure 3: Degree Distribution of Ring Group Graph (averaged over 100 iterations).

Clearly Figure 3 is a much better representation of the degree distribution of the Ring Group Graph for the example parameters used.

The degree distribution of Ring Group Graphs

For task 1 when investigating the degree distribution with $p + q = 0.5$ and $p > q$, I initially chose to fix the values of m and k to 20 (displayed on each graph) and vary the values of p and q such that the constraints on p and q were still satisfied. Below I will illustrate degree distributions of the ring group graph $m = k = 20$, and see how the degree distribution is altered as we vary the values of p and q.

If we analyse Figure 4, Figure 5, Figure 6 and Figure 7 we again see that these are degree distributions of ring group graphs, with parameter values $m = 20$, $k = 20$ and p and q varying such that $p + q = 0.5$ and $p > q$.
Let us first discuss Figure 5, Figure 6 and Figure 7 as we see that each of these graphs display roughly the same distribution which may be modelled by a Gaussian (Normal) distribution. However, the value of the mean degree and degree variance clearly vary as we vary p and q. As q increases (and in turn) p decreases, the distribution is shifted further and further along the x-axis (in the positive direction) - this represents the mean degree increasing . This shows that in general for the same values of m and k we expect a larger q value to increase the degree of each vertex. This is to be expected as the higher q is the more chance there is that an edge is constructed between each pair of vertices (regardless of labels). Due to the constraint $p+q = 0.5$ it is a natural consequence that as we increase the value of q we must decrease the value of $p = 0.5 - q$. This graph shows that if q increases and p decreases then the average degree of each vertex is increased (for fixed m, k) - however as stated above this is a consequence of the constraints and were I to increase both p and q we would see that this would further shift the degree distribution along the x-axis (in the positive direction).

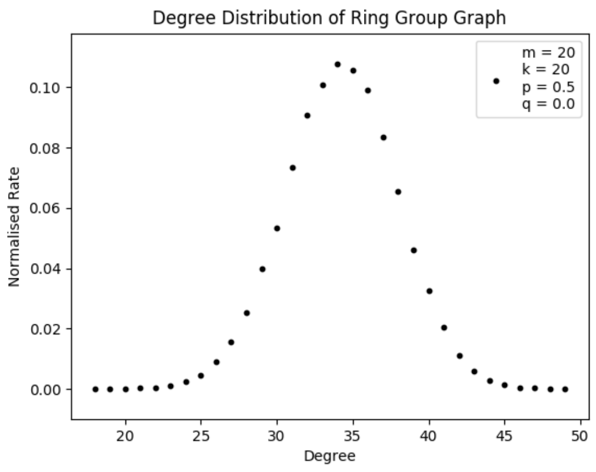

Figure 4: Degree Distribution of Ring Group Graph

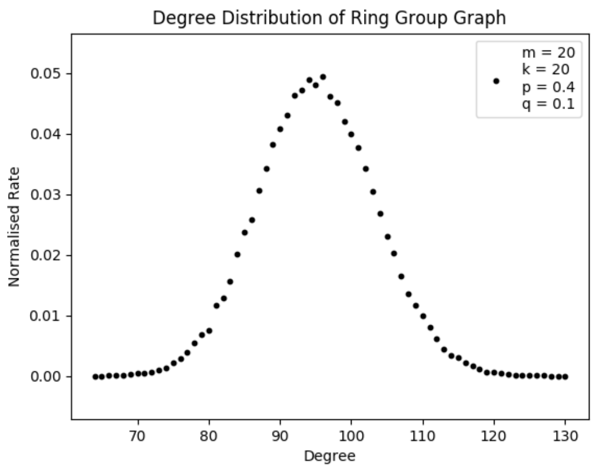

Figure 5: Degree Distribution of Ring Group Graph

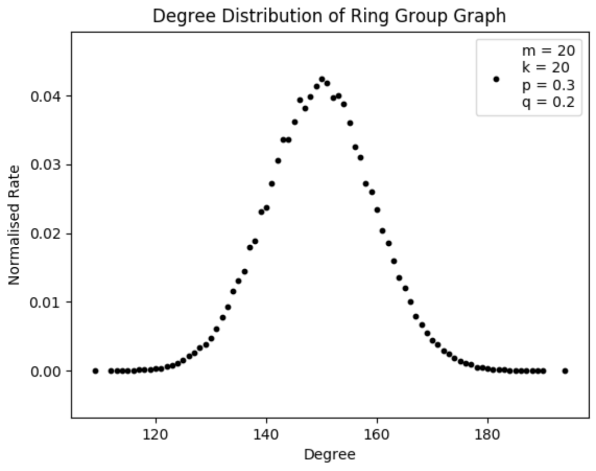

Figure 6: Degree Distribution of Ring Group Graph

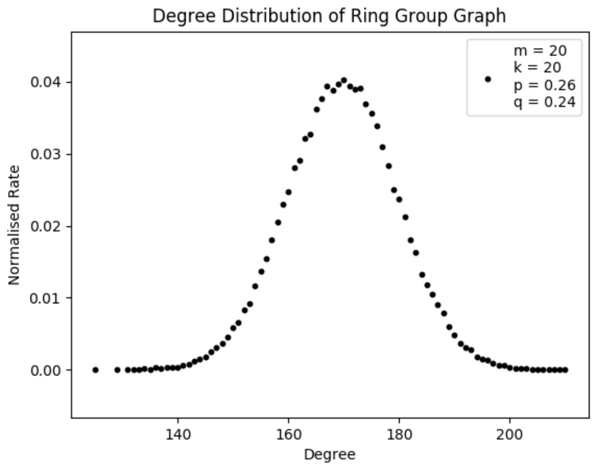

Figure 7: Degree Distribution of Ring Group Graph

If we now consider Figure 4 we see that the distribution it presents is similar to that of the other figures. In this case we have that $p = 0.5$ and $q = 0$, this means that if a vertex is in the same or an adjacent group to another vertex then there is a that an edge will be constructed between the two vertices with probability 0.5, otherwise an edge will not be constructed between the two vertices.
The next trend we notice amongst the graphs (most noticable by comparing Figure 4 and Figure 7) is that as q increases in value the range of the normalised rate seems to become far lower, and the variance of the degrees seems to increase. This is visible from the fact that Figure 7 has far more points than Figure 4. This is again due to the fact that the vertices represented in Figure 7 have a far greater chance of having neighbours that aren't in their group or an adjacent group. When there is no probability of this happening there is far less variance in degree and hence the normalised rate for a given degree is likely to be higher.

Now let us consider the degree distribution of a graph with probabilities $p = 0.5$ and $q = 0.05$ - in this case we shall alter the number of vertices by altering the values of m and k, and analyse how this effects our degree distribution.
The rationale behind our values of p and q is that whilst we desire an element of randomness (given by q) in order to properly exhibit the features of the ring group graph it makes sense that p has a greater value

of q as this means causes edge formation to be less random and more reliant on the properties of the ring group graph i.e. the label of each vertex.

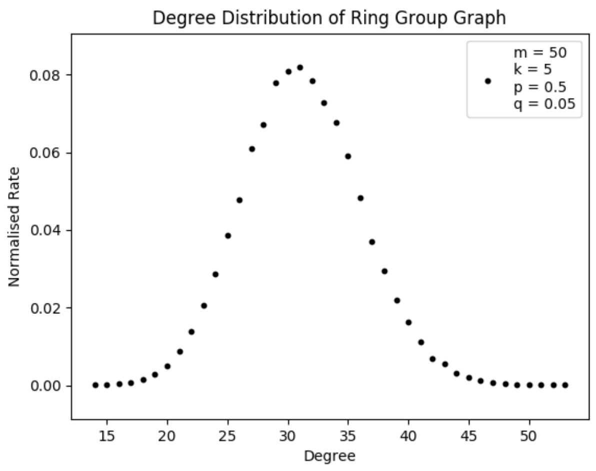

Figure 8: Degree Distribution of Ring Group Graph

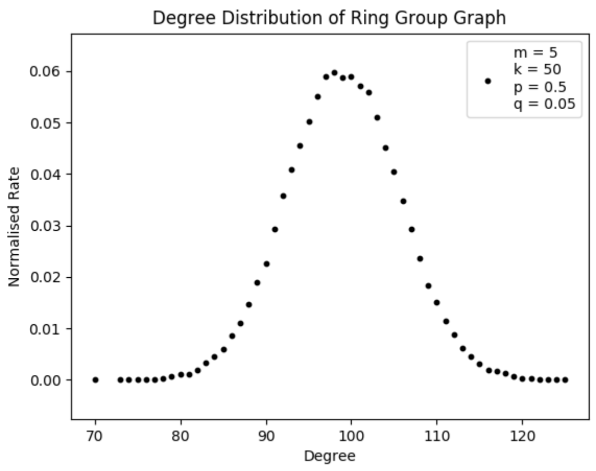

Figure 9: Degree Distribution of Ring Group Graph

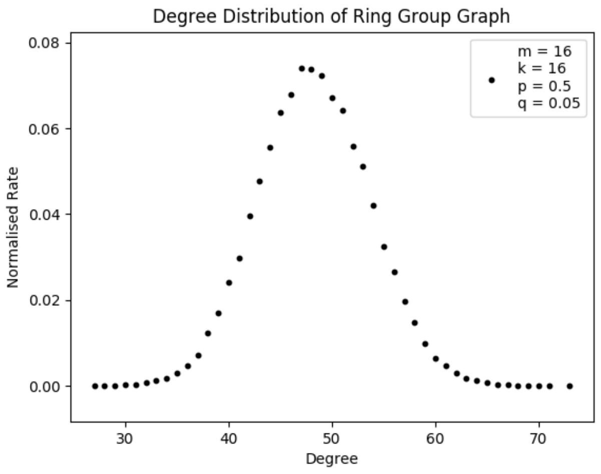

Figure 10: Degree Distribution of Ring Group Graph

Above we see 3 plots Figure 8, Figure 9 and Figure 10. Each of them have roughly 250 vertices and the same values $p = 0.5$ and $q = 0.05$ as parameters. However, for each graph the values of m and k differ. In Figure 8 we have $m = 50$ and $k = 5$ clearly the number of groups is much larger than the number of elements per group and as a result the degree on average is relatively low (as there are less vertices in adjacent groups or the same group to become neighbours).
In Figure 9 we see the opposite scenario $m = 5$ and $k = 50$ clearly we have a low number of large groups and as we expect this results in each vertex having many neighbours and thus higher degrees (as there are many vertices in adjacent groups or the same group to become neighbours with).

In Figure 10 we have $m = k = 16$ again this produces a similar number of vertices (256), and is the middle-ground between the two above approaches and this is reflected in its degree distribution.
Clearly from the above we see that if we wish to increase the degree of vertices in the Ring Group Graph in general, we should decrease m and increase k. If we wish to decrease the degree of vertices in general we should increase m and reduce k.

The relationship of the diameter of the Ring Group Graphs and p

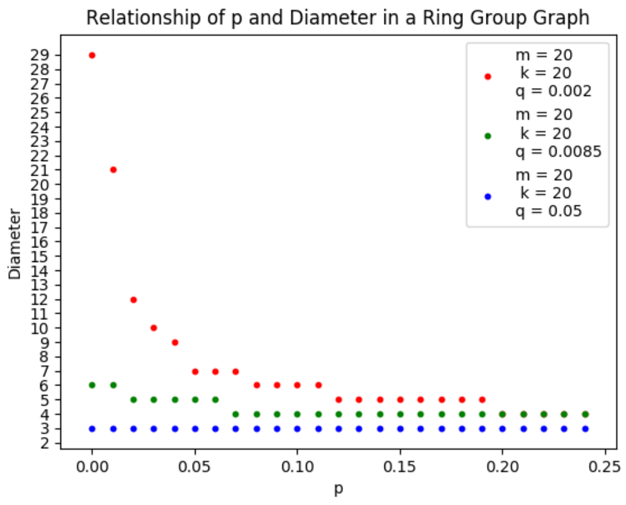

Figure 11: Relationship of diameter and p in the Ring Group Graph

Figure 11 illustrates the relationship of the diameter of a Ring Group Graph with 400 nodes and the probability p in the cases when $q = 0.002$, $q = 0.0085$ and $q = 0.05$. We can identify a slight negative correlation such that as p increases, the diameter of the graph becomes shorter. As p increases this results in more edges being added to the graph. Obviously, we cannot increase the diameter of a graph by adding edges to it - this can only reduce the diameter of the graph and thus it makes sense that as we increase p the diameter decreases. We can use the the same argument to justify why this relationship is only observable for very small q, this is because as we increase q the graph becomes more connected (as described before) and a more connected graph is likely to have a smaller diameter.

Question 2: Brilliance Distribution

Task

Investigate the distribution of vertex brilliance for the graph from coauthorship.txt, the Preferential Attachment Graph and the Ring Group Graph.

Implementation

The brilliance of a vertex v is the largest value k such that we have a set $\{v, n_1, \ldots, n_k\}$ such that v is joined to every vertex n_i and no other pair of vertices are adjacent. Essentially, it is the size of the largest set of neighbours of v such that none of the neighbours share an edge.

In order to model the brilliance distribution of the Co-authorship network we must first parse the text file 'coauthorship.txt' into the program such that we can create a graph out of it. Once we've done this we find that the coauthorship graph has 1559 vertices and 40016 edges.
We not only wish to model the brilliance distribution of the Co-authorship network but also that of the Preferential Attachment Graph and the Ring Group Graph. In order to produce graphs with similar numbers of vertices and edges to the Co-authorship network I had to experiment with parameters considerably and found the following to be true:

- For the Ring Group Graph parameters $m = 50$, $k = 31$, $p = 0.2$ and $q = 0.01$ provided a graph with 1550 vertices and roughly 41000 edges.
- For the Preferential Attachment Graph parameters $n = 1559$ and $step = 36$ provided a graph with 1550 vertices and roughly 40600 edges.

Preferential attachment in a graph is the property of new nodes that are added to the graph being more likely to connect with vertices that are already well-connected - i.e. vertices with higher degree. In order to facilitate this property in our graph model we initially create a complete graph with n nodes and utilise the PATrial class from lectures to iteratively add nodes to the graph. The function that utilises this class ensures that when we add an edge to a pair of vertices the edge is undirected. I chose to do this by adding each vertex to the other vertex's neighbours i.e. if vertex 1 is added as a neighbour to vertex 2 then vertex 2 is added as a neighbour to vertex 1.
Calculating the brilliance of each vertex in a graph is an NP-hard problem, and thus the following method I describe provides a very good approximation for the brilliance of each vertex but we may obtain slightly different approximations if we were to vary the order in which we consider vertices of the graph.
Method for calculating brilliance:

- For each vertex in the graph create an empty array to store neighbours contributing to the vertex's brilliance
- For each vertex in the graph consider all of the vertex's neighbours.
- For each neighbour of the vertex consider all of the neighbour's neighbours.
- If one of the neighbour's neighbours is already in the brilliance array then do not add the vertex's neighbour to the brilliance array - otherwise add it.
- The brilliance of the vertex is equal to the number of vertices in the brilliance array.

Analysis

Below we have the vertex-brilliance distribution of the Co-authorship network.

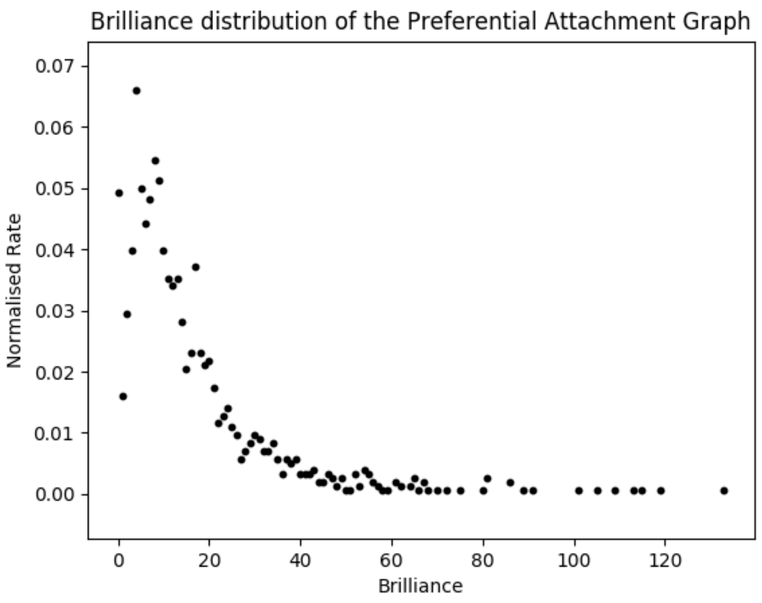

Figure 12: The vertex-brilliance distribution of the Co-authorship network.

The construction of the graph represented in Figure 12 is described at the top of this section - it has 1559 vertices and roughly 40016 edges. We notice that the average brilliance is roughly 20 which is substantially less than the average degree - roughly 52. This is to be expected due to connectedness of the graph, as many of the vertices are likely to share neighbours. If we consider this in terms of the 1559 scientists that the graph is modelling it would say that on average each scientist has coauthored papers with 54 other scientists and that on average only 20 of these 54 scientists have not co-authored papers with other members of the 54 scientists. This suggests that the network is very connected and that many of the same group of scienctist have co-authored papers with one anoth. The fact there is a brilliance of 20 suggests that there are a number of scientists whom rarely co-author papers, or do so frequently with people that other scientists are unlikely to have produced papers with other scietists.
Essentially Figure 12 shows that a large proportion of vertices have low brilliances in real-terms this means that a high proportion of the scientists are well-connected (and have produced papers with a lot of the other scientists). This makes sense as scientists will want to co-author papers with other reputable/experienced scientists.
The graph is an Inverse Gaussian (Inverse Normal) distribution which suggests whilst a normal distribution would present a graph where edges are constructed randomly this is the opposite and connections are formed based on preference.

The same approximator for Brilliance that was applied to the Co-authorship network can also be applied to the preferential attachment graph.

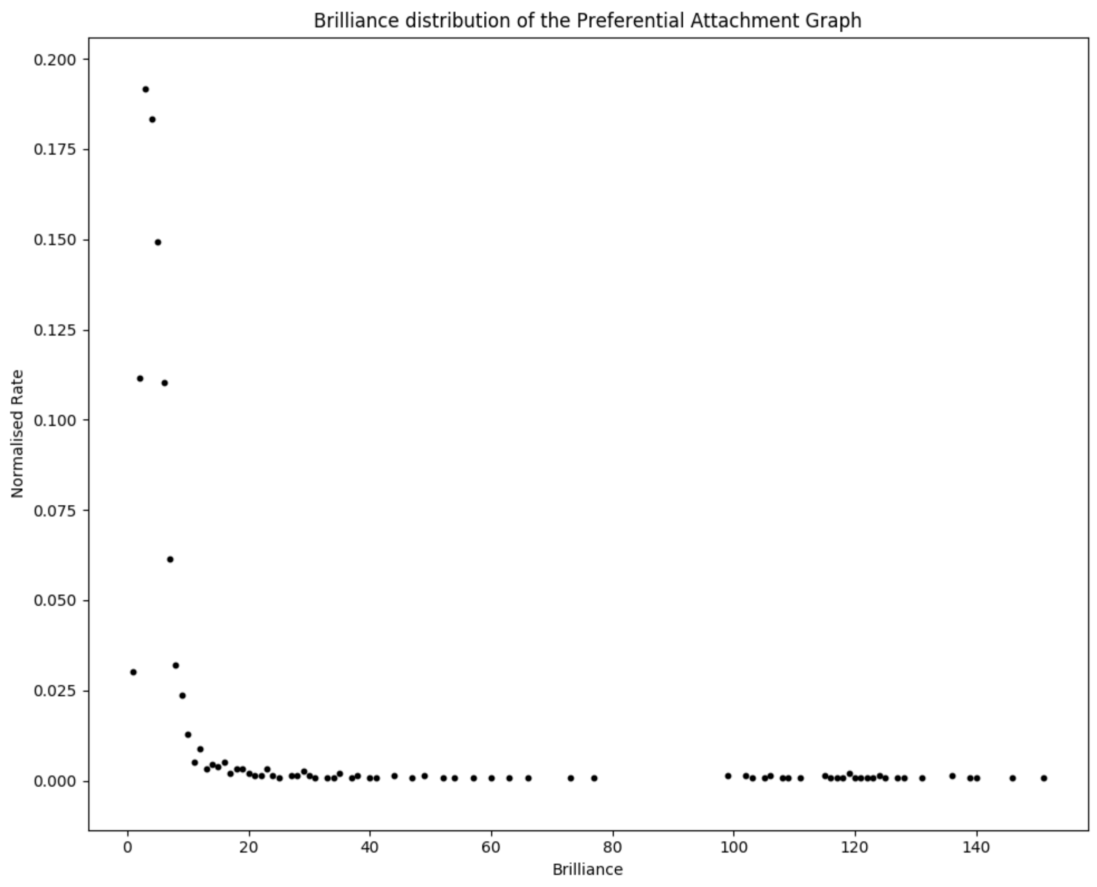

Figure 13: The vertex-brilliance distribution of the Preferential Attachment graph.

The construction of the graph represented in Figure 13 is described at the top of this section - it has 1559 vertices and roughly 40600 edges. Clearly Figure 13 is of a similar shape to Figure 12 - this should not be a surprise. When scientists are choosing another scientist to co-author a paper with they will not be doing this at random. The fact that the co-authorship graph is similar to the preferential attachment graph suggests that new scientists that are looking to produce papers would prefer to produce papers with reputable/experienced scientists that have already produced papers with numerous other scientists. When we consider this, it makes logical sense. Clearly the Co-authorship Network is one which exhibits preferential attachment.

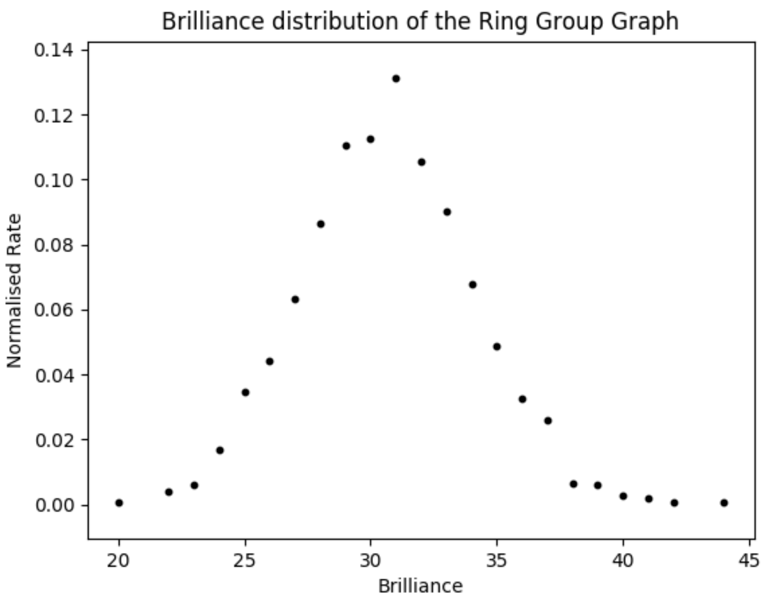

Figure 14: The vertex-brilliance distribution of the Ring Group graph.

Finally, let us investigate the vertex brilliance distribution of the Ring Group Graph (as investigated in Question 1). The construction of the graph represented in Figure 14 is described at the top of this section - it has 1550 vertices and roughly 41000 edges. The production of the Ring Group Graph relies on probabilities p and q to dictate when edges are formed between a pair of vertices. This use of probability ensures that the degree distribution of the Ring Group Graph is normally distributed around the mean/expected degree (as seen in Question 1). As we have a normal degree distribution we would expect a normal brillance distribution (as displayed).

Question 3: Searching in Graph

Task

The task we were presented with is the problem of searching a graph. This is when provided with both a start vertex and a target vertex, beginning at the start vertex traverse the graph (through querying neighbours) until the target vertex is found - with the intention of 'querying' the fewest number of nodes when doing so.

In the following explanation/description and analysis I will refer to 'Naive Search' and 'Random Search', I define them as the following:

- **Naive Search**: Query every neighbour of the current vertex (initially the start vertex), if the target vertex is not found make of the current vertex's neighbours the current vertex. Perform the same process on the neighbours of this vertex until the target vertex is queried.
- **Random Search**: Query a random neighbour of the current vertex and make it the new current vertex. If the current vertex is not the target vertex, repeat this process of querying a random neighbour until you query the target vertex.

Searching a Random Graph

Implementation

The graph that we will be performing our search algorithm on is a random graph with 500 vertices and 25000 edges are created between these vertices with probability 0.1.
It may be assumed in the following description of our implementation, if at any point the target vertex is queried the search time is returned and our algorithm terminates. It may also be assumed that if at any point we refer to querying a vertex, we are also incrementing the search time.

- We are provided with a random start vertex from which we will begin our search, and a random target vertex which we aim to arrive at with the lowest possible search time (using fewest queries). We set the start vertex to the current vertex.
- We begin our process by checking the number of neighbours that the current vertex has i.e. identifying the start vertex's degree. If this degree is large, in our case we consider large to be above $\frac{n}{2}$, we query each of the vertex's neighbours (as the target node is likely to be adjacent to the current vertex). If the degree is small (i.e. less than $\frac{n}{100}$) we query each of its neighbours to check if any of them are the target node - we then move to one of these neighbours.
- However, if the current vertex does not have a large or small degree we simply query a neighbour at random (due to the unordered nature of our vertex's neighbours).
- We repeat the above steps until the target vertex is queried.

Explanation and Rationale

We initially start at the start vertex knowing only the *id* of the vertex and the number of neighbours (degree) the vertex has. In a random graph, of these pieces of information it is only the number of neighbours that has any significance to us - thus it makes sense that we utilise this in order to make a more informed choice of which vertex to travel to.
In a random graph, graphs with larger degree are more likely to be connected to the target node.

Thus if we're at a vertex with large degree it makes sense to query it's neighbours - in our case we consider a large degree to be at least $\frac{n}{2}$. After querying the neighbours of the vertex if we have not queried the target vertex we simply move to one of the vertex's neighbours at random as there is no information about the neighbours that we can base our choice on.
If the vertex we're at has very small degree, we consider this to be $\frac{n}{100}$, then we check all of the neighbours of the vertex. We do this because it is very quick to do and through experimentation it seemed to positively impact the search time.
If the degree of the current vertex is neither significantly large or significantly small then we simply query the next vertex.

Effectiveness

The following is a comparison of the hybrid search algorithm I developed and described above as well as both the naive search algorithm and the random search algorithm:

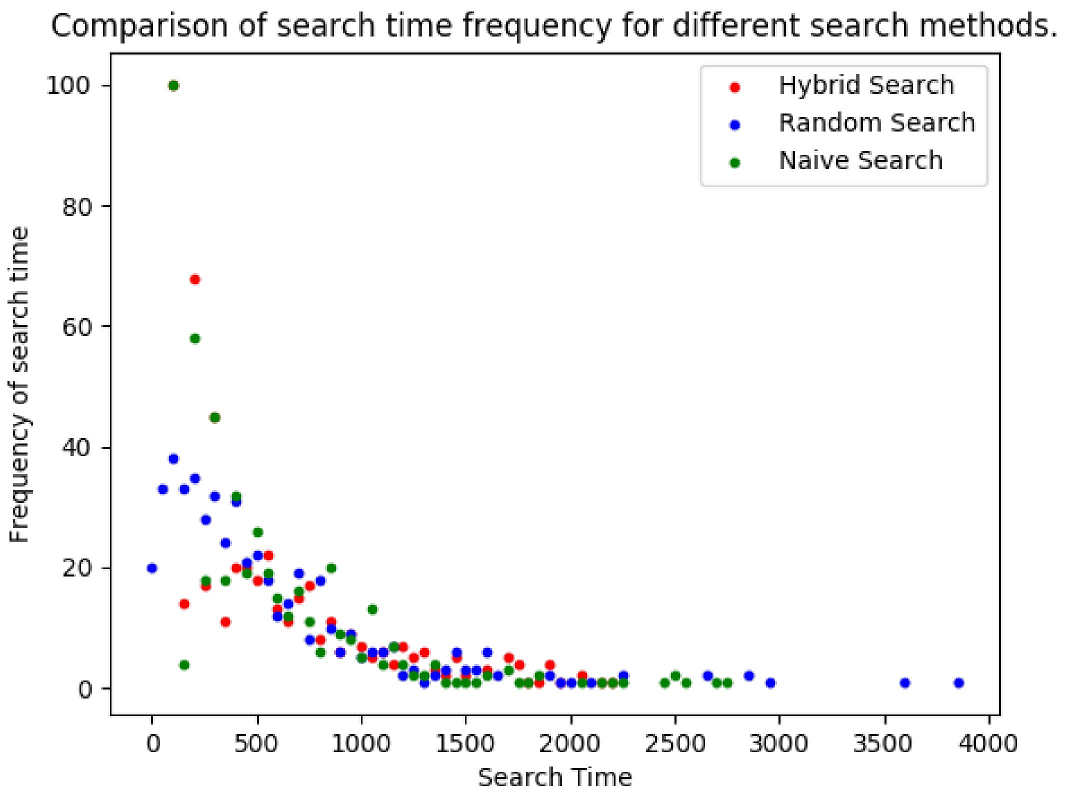

Figure 15: The frequency of search times achieved for search algorithms on a random graph.

The results in Figure 15 were attained on a random graph with 500 vertices and 23875 edges.

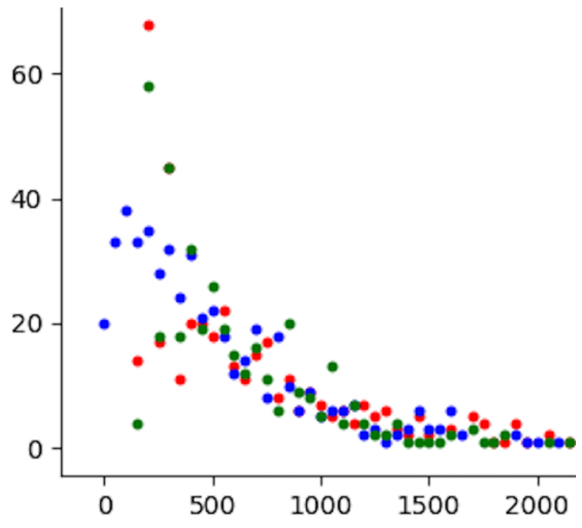

Figure 16: A segment of Figure 15, displaying all points produced by the Hybrid Algorithm.

We clearly see from Figure 15 and Figure 16 that whilst the Naive Search method has instances of nearly 3000 in search time and the Random Search has instances of nearly 4000 in search time the maximum search time resulting from the hybrid algorithm barely surpasses 2000.
With this in mind we also see that the Naive Search works relatively well in some cases, this occurs when the start vertex and goal vertex generated are nearby one another and is more attributable to chance than it's systematic approach.
The majority of the search times acquired are around $\leq$ number of nodes which suggests that whilst far from optimal the suggested method acquires more than reasonable search times.

Searching a Ring Group Graph

Task

The task we were presented with is the same problem of searching a graph as above, except in this case vertices are assigned labels dependent on which group they are in, and we can utilise the group a vertex is in to estimate how far the vertex is from the target vertex.

Implementation

The graph that we will be performing our search algorithm on is a ring group graph with 500 vertices and 17000 edges, with parameters $m = 25$, $k = 20$, $p = 0.3$ and $q = 0.05$. The reasoning behind this is that the higher the value of q, the more random edge construction becomes and the less representative our model is of a typical Ring Group Graph and the more representative it is of a Random Graph.
It may be assumed in the following description of our implementation, if at any point the target vertex is queried the search time is returned and our algorithm terminates. It may also be assumed that if at any point we refer to querying a vertex, we are also incrementing the search time.
All differences below are performed *modulom*, i.e. the difference between $mk - 1$ and 0 is 1.

- We have the target vertex and start vertex passed into our algorithm as parameters.
- We set the current vertex to the start vertex and while we haven't found our target vertex we do the following.
- We calculate the difference between the label of our current vertex and the label of our target vertex.
- If the result of the calculation is less than or equal to 1 then the target vertex is likely to be a neighbour thus we query each of the current vertex's neighbours and should we find the target vertex we terminate the algorithm. If we do not find the target vertex we set the current vertex to a random neighbour.
- If the result of the calculation is greater than 1:
- We aim to reduce the difference between our label and the target vertex's label. In order to do so we first query a random neighbour.
- If the 'difference between this neighbours label and the target's label' is less than the 'difference between the current vertex's label and the target's label' we make the neighbour the current vertex.
- If the 'difference between the neighbours label and the target's label' is less than the 'difference between the current vertex's label and the target's label $+\frac{m \cdot k}{10}$' we make the neighbour the current vertex with probability 0.2.
- Otherwise, we make the neighbour the current vertex with probability 0.05.
- We repeat the above process from bullet point 3 .

Explanation and Rationale

In the production of my algorithm I assumed that $p > q$, as if we have $q > p$ we essentially lose many of the properties you would expect in a Ring Group Graph.
The manner in which a Ring Group Graph is constructed means that the majority of neighbours for each vertex in the graph will be vertices with the same label or adjacent labels i.e. their labels will be different by 1, 0 or -1 $module\, m$.
Thus initially we check if the label of our start vertex is different by 1, 0 or -1 and if it is we search through our neighbours as we know if it is likely to be a neighbour.
If this is not the case our next aim is to reduce the difference $(modulo\, m)$ between the current vertex and the target vertex. In order to do so we query a random neighbour, if this neighbour reduces the difference in labels we make it the current vertex.
If the selected neighbour increases the difference in labels by $<= \frac{1}{mk}$ then we make it the current vertex with probability 0.2 - otherwise we make it the current vertex with probability 0.05. The idea behind this is that if we continually only move to vertices with closer labels, it can be easy to become caught in a $localminima$ and by changing vertex with a small element randomness it may help us to obtain our solution more quickly.
Once the current vertex is altered, if it is not the target vertex we repeat this process of assessing the difference in labels between the current vertex and the target vertex until the target vertex is found.

Effectiveness

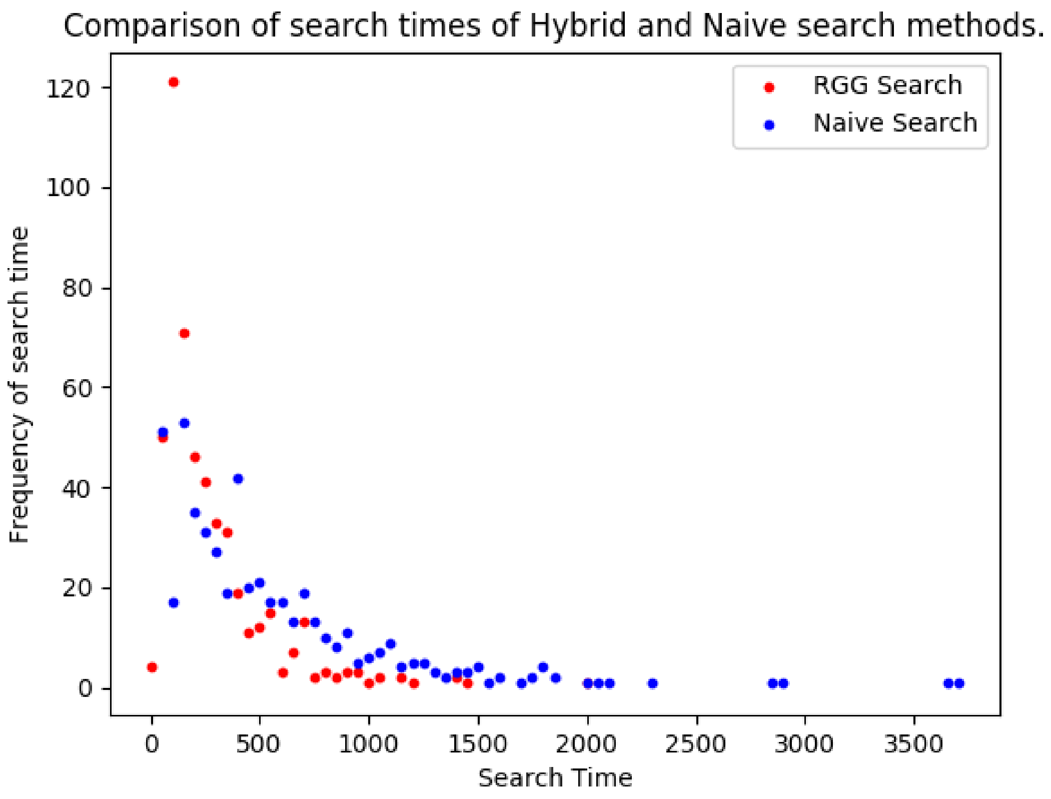

Figure 17: The frequency of search times achieved for search algorithms on a ring group graph.

In Figure 17 we see a comparison of the search method developed as part of this question and the

Naive Search I developed in the earlier question. This illustration was obtained on a ring group graph with 500 vertices, roughly 17000 edges - with parameters $m = 25$, $k = 20$, $p = 0.3$ and $q = 0.05$. As you can see the search method developed is far superior to the Naive Search in generl. The majority of search times acquired are in the region of $\frac{mk}{2}$ whilst the Naiver Search rarely acquires search times less than mk.