Network Analysis Assignment

1 Question 1: Ring Group Graphs

1.1 Degree Distribution

Initially, we analyse the degree distribution of Ring Group Graphs. Analysis will focus on how varying the values of p and q affect degree distribution for fixed values of m and k. By default, graphs depicting degree distribution are averaged over 100 samples.

The values p and q are constrained linearly. In order for clear comparison, when graphically displaying results, plots will be coloured according to figure 1. High values of p with lower q are coloured red and vice versa comparisons are yellow.

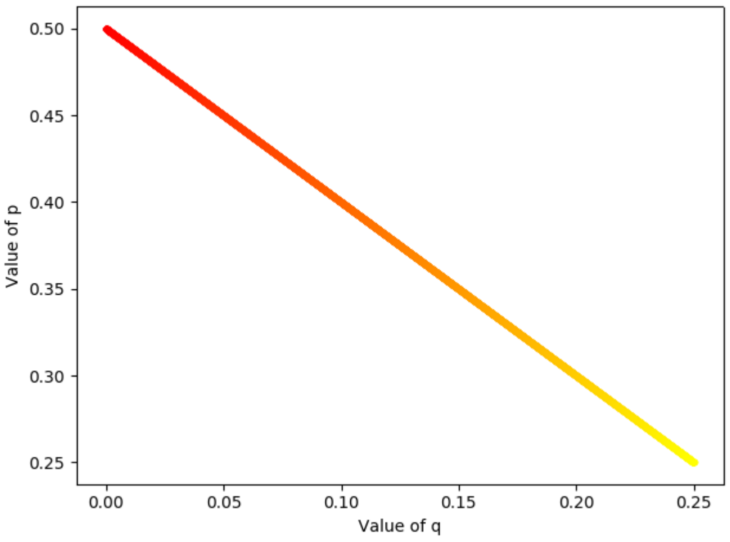
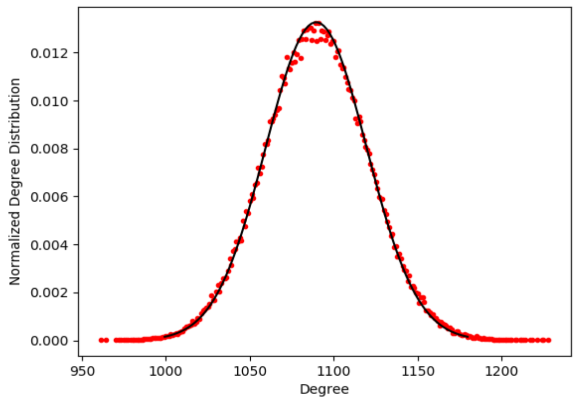


Figure 1: Colour indications of values for p and q under the constrains: p+q=0.5 p>q

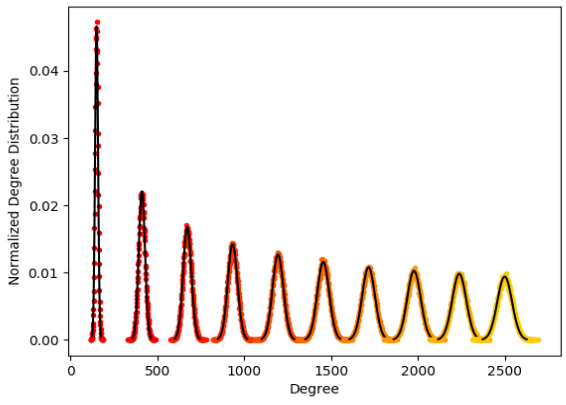
The Expected degree of each vertex in a ring group graph is as follows:

$$Expected(Degree) = p[(k-1)+2k] + q[k(m-3)] = p[3k-1] + q[mk-3k)]$$

For a vertex, v, there are (k-1) other vertices in its group followed by 2k vertices in adjacent groups. v has probability p in having an edge between each of these vertices and so will be connected to approximately p[3k-1] vertices from this group. There are (m-3) other groups to choose from each with k vertices, with an edge between v and each of these vertices forming with probability q, giving q[mk-3k)]. The degree distribution for a given graph should therefore average around this result.



(a) m=100 k=100



(b) m=100 k=100

Figure 2: Degree Distribution Plots for two ring group graphs, holding m and k constant

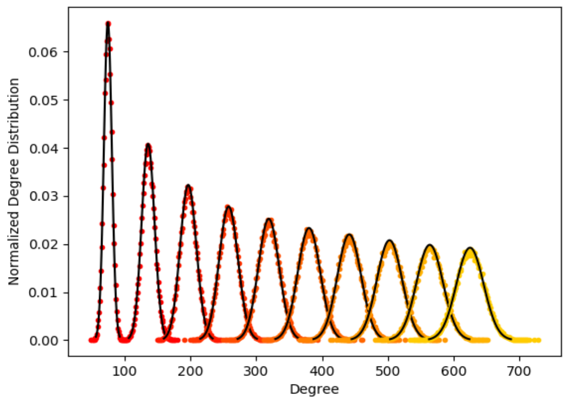
Table 1: m = 100, k = 100

p	q	Mean	Variance	Range	Number Of Edges
0.5	**0.0**	149.527	74.019	60	747673
0.472	**0.028**	40.487	330.025	118	2052458
0.444	**0.056**	671.833	563.938	166	3359116
0.417	**0.083**	932.701	787.185	184	4663562
0.389	**0.111**	1194.018	989.908	204	5970297
0.361	**0.139**	1455.129	1182.969	219	7275389
0.333	**0.167**	1716.392	1356.899	237	8582159
0.306	**0.194**	1977.462	1524.882	245	9887613
0278	**0.222**	2238.786	1646.596	254	11193790
0.25	**0.25**	2500	1784.87	268	12496676

Figure 2(a) approximates the degree distribution to that of a normal distribution, plotted from the mean and variance. The similarity between the plots shows that the degree distribution follows a normal trend, a property shared amongst all ring group graphs independent of the parameters. The distribution is technically binomial, as degree distribution is discrete data, indicated by an aggregated probability of a vertex connecting to any other vertex, averaged over all vertices. For the purposes of analysis, the binomial distribution will be approximated to normal, given that the sample size is sufficiently large. A black line is presented on all normalized results, indicating the normal approximation for each plot.

Figure 2(b) and table 1 shows how varying q from 0 to 0.25 and subsequently p respectively, impacts the degree distribution. The general trend following a normal distribution does not change, however the mean and variance do. Increasing q leads to larger degree variation per vertex as the height of each plot shrinks as the tails move further apart.

As q increases, so does the total number of edges and the variance, as can be seen in table 1. This is because as the p and q get closer together, the ring group graph is more similar to a random graph, doing so completely when p=q=0.25. When p is higher, the vertices dominant method for connecting edges to other vertices is doing so with probability p between those in the same and adjacent groups. The number of nodes then to connect to with probability p is less than that for probability q. Therefore, degrees tend to be distributed less at higher p values where connections with probability p dominate over few vertices. This explains why the plots become shorter and fatter


(a) m=50 k=50

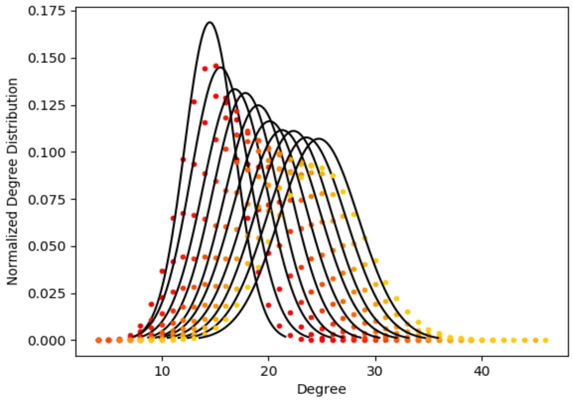
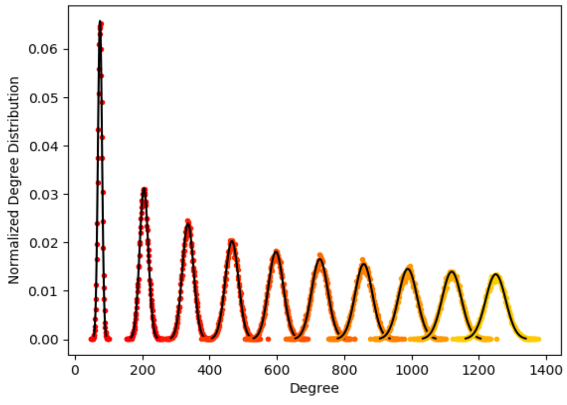

(b) m=10 k=10

Figure 3: Ring Group Graph for m=k


(a) m=100 k=50

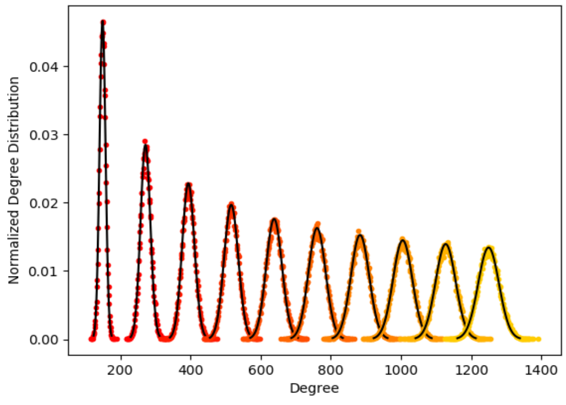
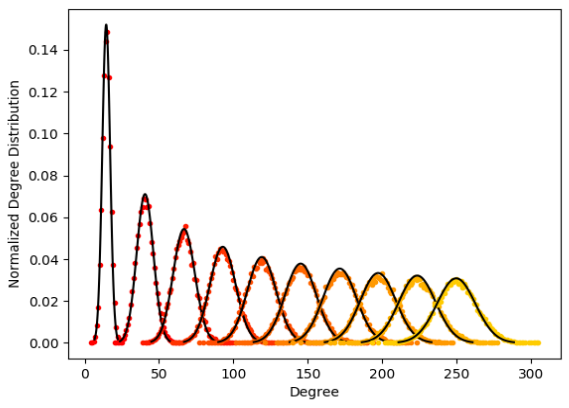

(b) m=50 k=100

Figure 4: Ring Group Graph for values 50 and 100 for m and k respectively

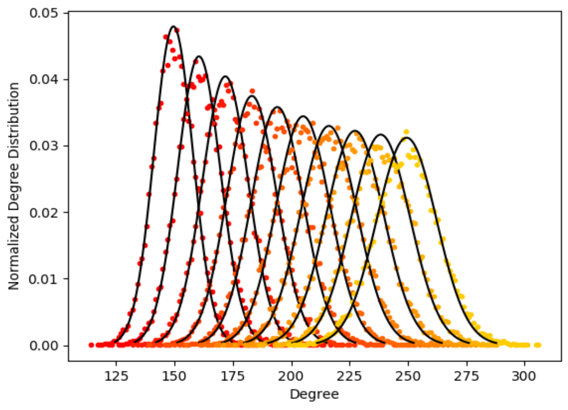
as p decreases relative to q. The table explains this well, showing the increasing number of edges are q increases with the growing variance and mean degree.

Figures 3 to 6 show how the relationship changes for different values of m and k. Figure 3 continues from figure 2, showing distributions over values m=k. For the others, the left-hand side indicates graphs when m > k and the right m < k.

All of the figures verify the conclusions drawn from figure 2(b) in the increasing variance as q increases, although at different rates. If m > k, it can be seen that the process happens at a more immediate rate in comparison to where m < k where the height reduction and variance increase is less apparent. This is because for low values of m and k, the ring group property of the graphs stops holding and the graph becomes a closer representation of a random graph. When m < k, more vertices are being connected with probability p within their own group and for m > k, vertices are connected outside to others with probability q, as fewer vertices exist in their group. Since p > q, the relative decline in each graph is a reflection of which probability the graph is depending on more.

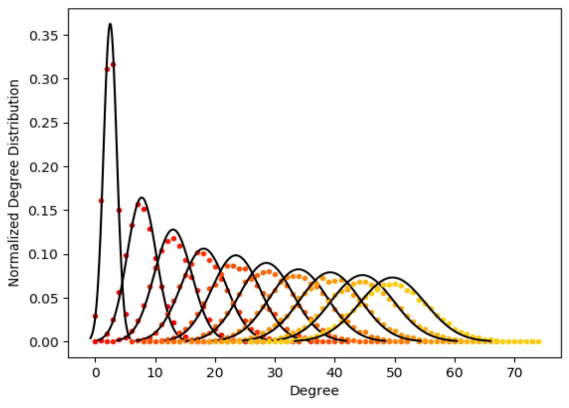


(a) m=100 k=10

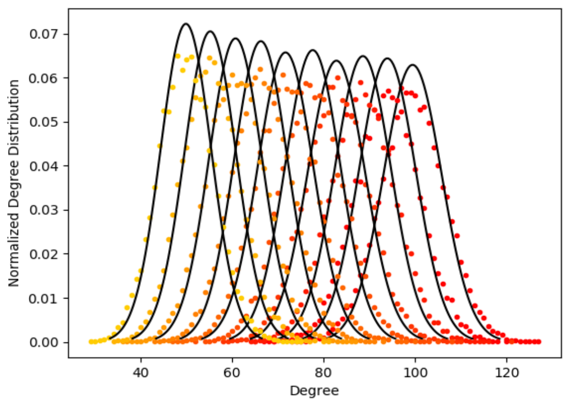


(b) m=10 k=100

Figure 5: Ring Group Graph for values 10 and 100 for m and k respectively



(a) m=100 k=2



(b) m=2 k=100

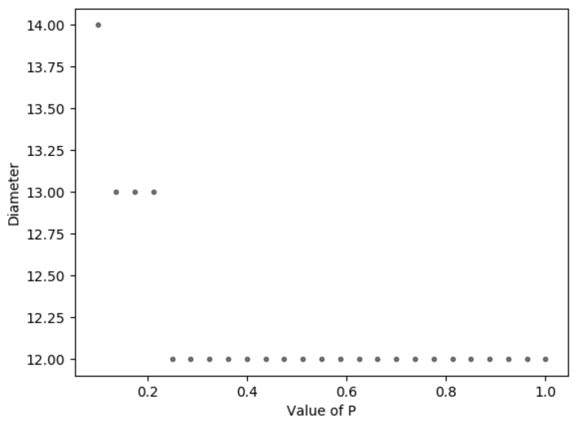
Figure 6: Ring Group Graph for values 2 and 100 for m and k respectively

1.2 Diameter

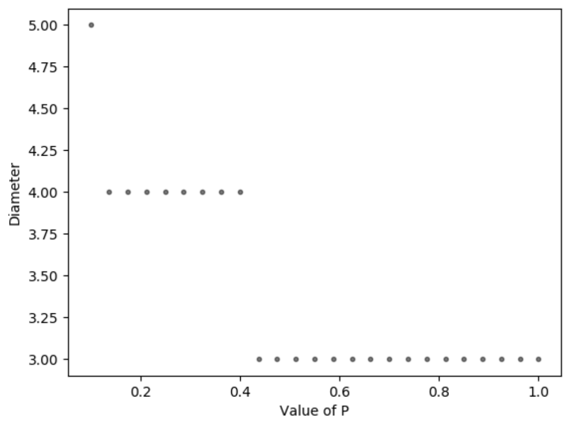
Analysing the relationship between the value of p and the diameter for RG graphs for a fixed value of q. Figures 7 and 8 shows the plots created under these conditions, averaged over 10 samples because calculating the diameter is computationally time consuming. Diameter is a measure of connectivity in an undirected graph. A graph with a higher degree distribution therefore is more likely to have a smaller diameter.

As can be seen, for values of q greater than approximately 0.15, the diameter of the graph is consistently 2. q is a measure of connectivity between groups in a ring group graph and so as it increases, movement between groups is better facilitated and as such the diameter is likely to fall. p measures connectivity inside of groups and is thus not as significant towards calculating the diameter. However, viewing it as non-deterministically pick a neighbour from a vertex, which is connected to a target vertex (or switching the vertices round) highlights why p still holds significance. Having a high value of p spreads out the value of q over more of the vertices in a group and so allows one vertex in a group to move to any other group by simply moving to a neighbour that is connected to it.

This can be seen effectively in figure 7 and 8, which shows the diameter reaching its lowest value when q is above 0.2 very any values of p, then slowly moving to a degree of 3 when q is 0.05 and then increasing exponentially as q decreases towards 0.

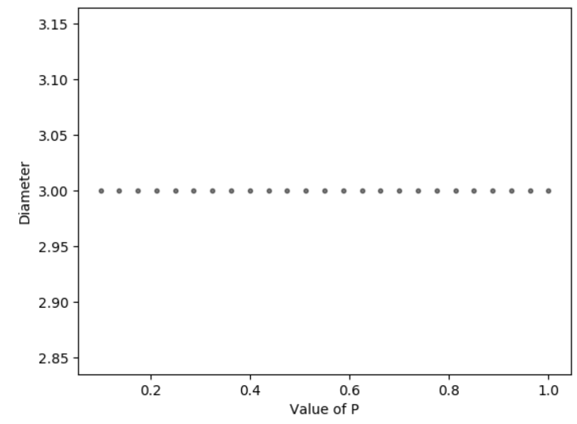


(a) q = 0.0

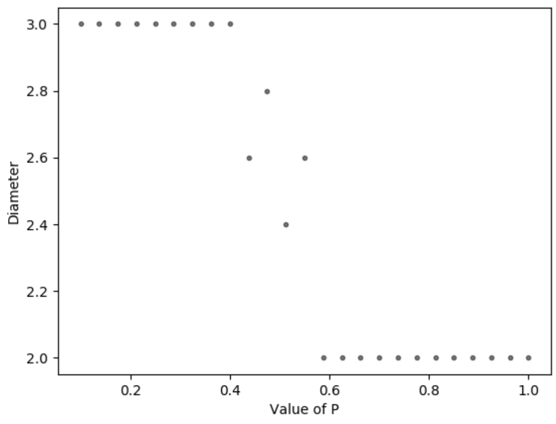


(b) q = 0.01

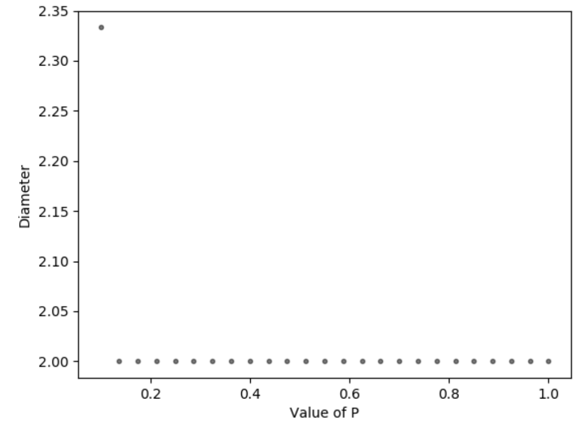
Figure 7: The Diameter when varying p in a Ring Group graph for fixed q: part1



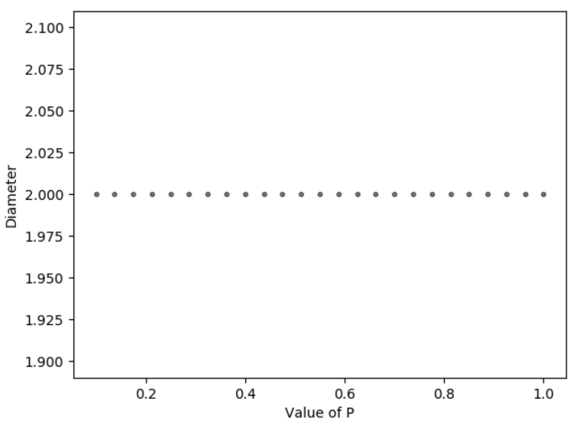
(a) q = 0.05



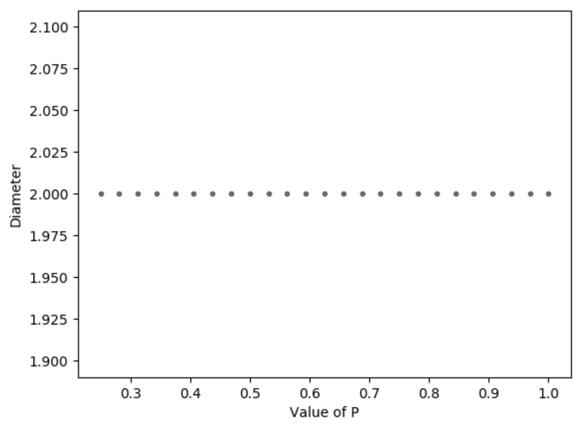
(b) q = 0.1



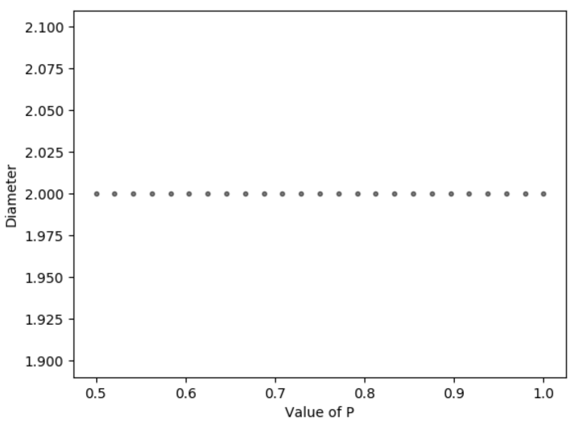
(c) q = 0.15



(d) q = 0.2



(e) q = 0.25



(f) q = 0.5

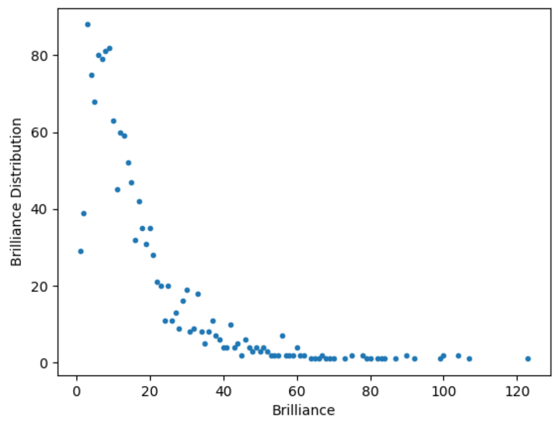
Figure 8: The Diameter when varying p in a Ring Group graph for fixed q: part2

2 Question 2: Vertex Brilliance

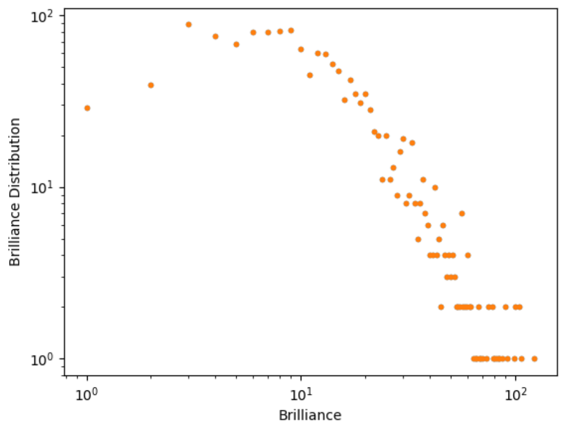
Parameters for Preferential Attachment (PA) Graphs and Ring Group (RG) Graphs were established by trial and error until similar numbers of vertices and edges were achieved to that of the co-authorship graph example, with 1559 vertices and 40016 edges respectively. For PA graphs, the number of vertices was set to 1559 to exactly match. PA Graphs are generated initially directed and is subsequently converted so edges are undirected. An approximate out degree of 35 for generation achieved on average 39610 over 100 samples compared with 41497 for out degree 36. Therefore, the out degree is valued at 35 to achieve similarity with co-authorship. For the plotting and analysis of PA graphs and RG graphs, data was averaged over 100 samples.

			Brilliance Distribution		
	Number of Nodes	**Number of Edges**	**Mean**	**Variance**	**Range**
Coauthorship Graph	1559	40016	16.260	239.508	122
PA Graph	1559	39622.0	29.762	12.027	22
Ring Group Graph	1550	39519.0	29.728	12.427	22

2.1 Co-authorship graph



(a) Distribution

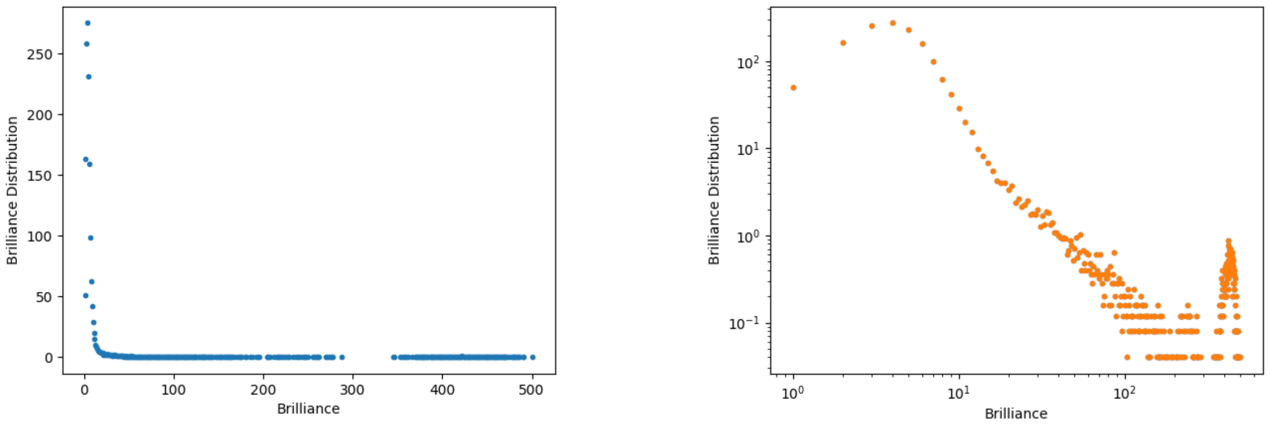


(b) logplot

Figure 9: Co-Authorship Brilliance distribution

The graph seems to resemble an inverse Gaussian distribution. A normal Gaussian distribution would be present if the data was random and the skewed results indicate a pattern. A peak at early brilliance can potentially be explained by suggesting academics co-author each others work in small groups, separated by topic/genre, co-authoring similar topics amongst devisable groups. As such academics are likely to cluster amongst themselves. Since work is usually very specific, academics will commonly only reference a select few other academics and vice versa, given that there is a limited amount of research in a particular area. This will lead to natural clusters forming in the data, grouping academics together where they work on similar topic areas. Graphically, this means that such groups with see high connectivity amongst themselves with infrequent adjacency to other parts of the graph. As a result, the brilliance of such groups will reflect their size, given that they are connected to each other and as such more vertices must be removed before a k-star is created. The peak at the start of the graph reflects this, showing a average size of these clusters reflected in the brilliance. The distribution then declines as we move away to more infrequent cluster sizes. Such conclusions though are limited by the breadth of the data, however the significant difference in the mean and significantly higher variance of plots when compared to the other graphs indicates clustering maybe taking place, therefore explaining the resulting brilliance distribution.

2.2 PA Graphs



(a) Distribution (b) logplot

Figure 10: PA Graph Brilliance distribution, numberOfNodes=1559, outDegree=35

The chosen method for generating PA graphs was to produce a directed graph using the out-degree and number of nodes as parameters and then converting the result into an undirected representation.

The resulting plot is heavily skewed, with smaller brilliance values frequently appearing and then less brilliance variation looking towards the end. When the first few vertices are allocated edges, they are done so to every other vertex in the graph. The number of such vertices is equal to the out degree and such nodes have a maximum vertex degree. This means a star graph with them at the centre is the same graph and as such, a maximal independent set and thus vertex brilliance created is very high due to the amount of options for removal available. As more vertices are established, the degree of each steadily decreases and with it the vertex brilliance. This yields the trend seen in figure 10(a) until a stability point is reached whereby the degree will not decrease any further. This explains the spike at the start of the graph.

2.3 Ring Group Graphs

The chosen graph size is m=50 k=31 (1550 nodes in total), p=0.08 and q=0.03. The brilliance distribution, like the degree distribution in section 1, seems to follow a binomial distribution. Approximating to the normal distribution and plotting the expected results shows that this connection. Because of the link between the degree distribution and vertex brilliance, with a higher degree lending its self towards a higher potential independent set and subsequently a higher brilliance value, the brilliance trend is pretty much identical to that of degree distribution. Figure 10(a) shows this, with a normal distribution plotted on normalized brilliance results to show the similarity.

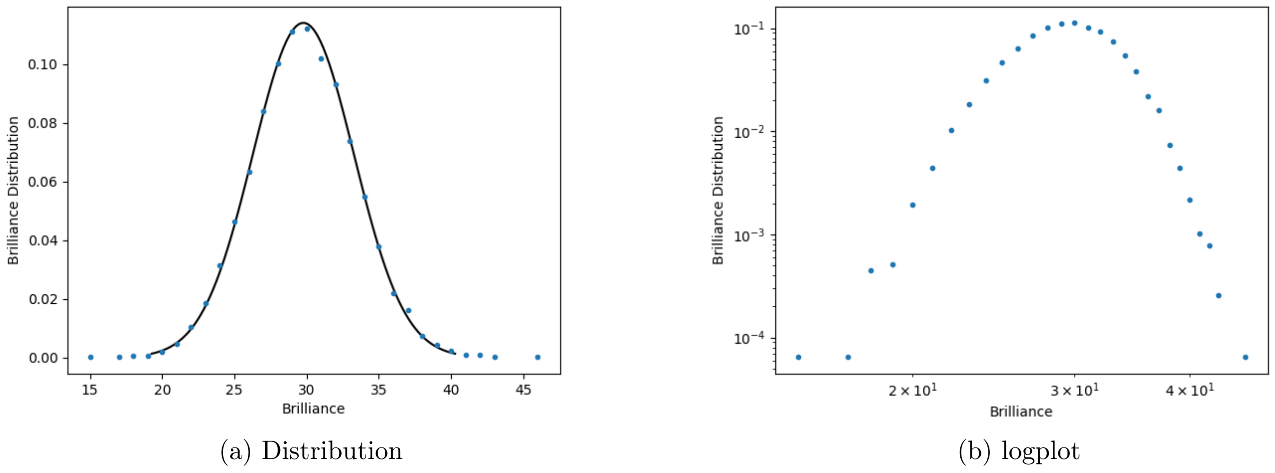


Figure 11: RG Graph Brilliance distribution m=50, k=31 p=0.08, q=0.03

3 Question 3: Graph Searching

During graph searching, the next node queried should have the highest probability of being the target vertex compared to every other vertex in the graph. The algorithm searching the graph should follow this procedure in order to minimise the amount of queries made and the resulting search time.

3.1 Random Graphs

A random graph is generated by specifying the number of nodes in the graph, n followed by a probability, p that each vertex is connected to another vertex. When searching a random graph, there are multiple potential strategies to consider:

1. Query all neighbours of a vertex searching for the target vertex. If not found move to a random new node
2. Partially query the neighbours of the current vertex, only analysing a selected few, decided upon by the algorithm, succeeding if the target vertex is found and if not moving to one of the queried nodes. The percentage queried is the equal to the number of neighbours times the probability used to generate the random graph.
3. Randomly pick a vertex from the list of a neighbours, finishing if it is the target vertex else moving to it and repeating. This is equivalent to a random walk.

Each of these strategies was attempted, however method 1 is optimal. As already stated, the query made should be on the vertex in the graph that currently has the highest probability of being the target vertex.

In a random graph, every vertex has an equal chance of being adjacent to the target vertex, with probability p. The total number of other vertices in the graph is equal to (n-1) and the expected degree therefore for the current vertex is p(n-1). Therefore:

$$\text{Probability that a given vertex is connected to the target} = \frac{p}{p(n-1)} = \frac{1}{(n-1)}$$

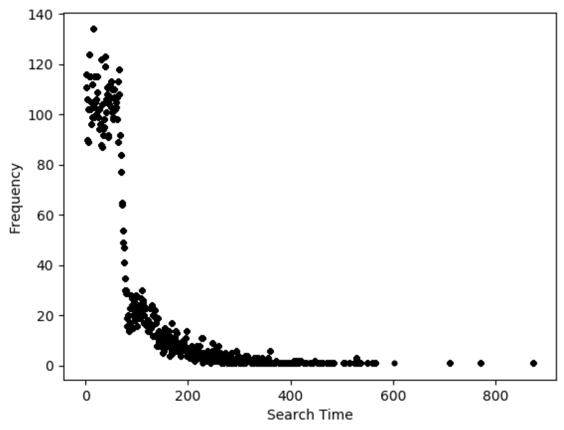
Algorithm 1 Query All: Random Graph Searching Algorithm

```
Require: $Graph, startVertex, targetVertex$
1: $searchTime \leftarrow 0$
2: $neighbours \leftarrow graph[currentVertex]$
3: while $TargetVertexNotFound$ do
4:     $currentVertex \leftarrow startVertex$
5:     for $neighbour$ in $neighbours$ do          ▷ Loop through all neighbours of the current vertex
6:         $searchTime \leftarrow searchTime + 1$
7:     if $neighbour == targetVertex$ then return $searchTime$
8:                                                 ▷ Return search time if target found
9:     $currentVertex \leftarrow random\ neighbour$        ▷ Else randomly move to a new vertex
```

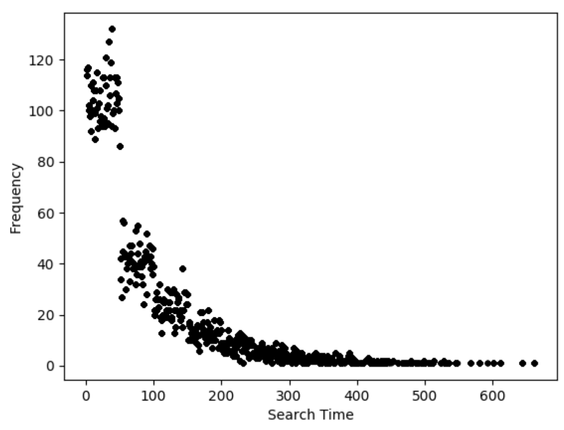
The initial query therefore has probability $\frac{1}{n-1}$ of being the target vertex. However, after a query has been made to a neighbour, that vertex is confirmed as not being the target, therefore the total sample space decreases in size by 1. The probability that a query to a second neighbour being the target vertex is $\frac{1}{n-2}$. The probability of finding the target therefore increases as you query more neighbours following the formula $\frac{1}{n-i}$ where i is the query number.

This means that for a random graph, independent of the probability p, used to generate the graph, querying all the neighbours will on average yield the best search time.

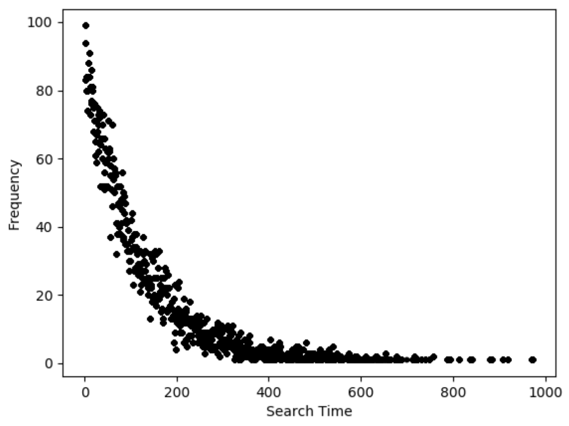
For data analysis, 100,000 graph samples were taken and averaged, with the search times measured and plotted. 100 vertices were used in the graph, in order to narrow down the possible search times and improve the resulting plot. Two probabilities were analysed, 0.5 and 0.25. The data shows that querying all the neighbours yields the lowest variance/spread in search times and on average will find the target vertex fastest according to the mean.



(a) Query All



(b) Partial Query

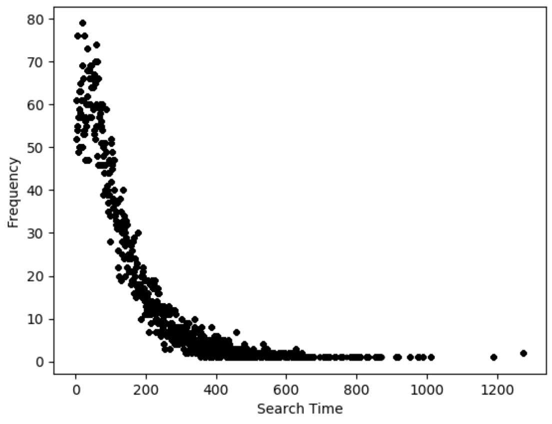


(c) Random Walk

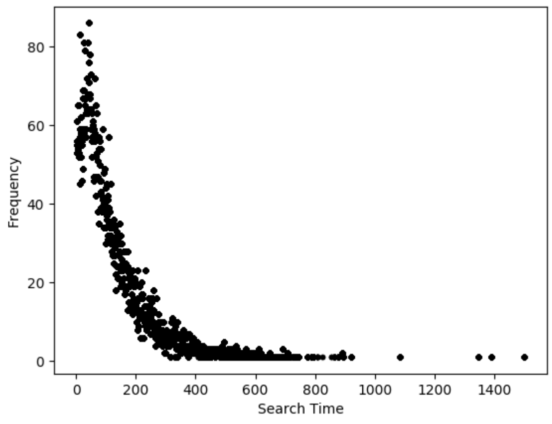
Figure 12: Random graph, p = 0.5

	Search Time		
	Mean	**Variance**	**Range**
Query All Neighbours	78.437	5648.181	822
Partial Querying	83.892	7238.326	954
Random Walk	111.768	13036.946	989

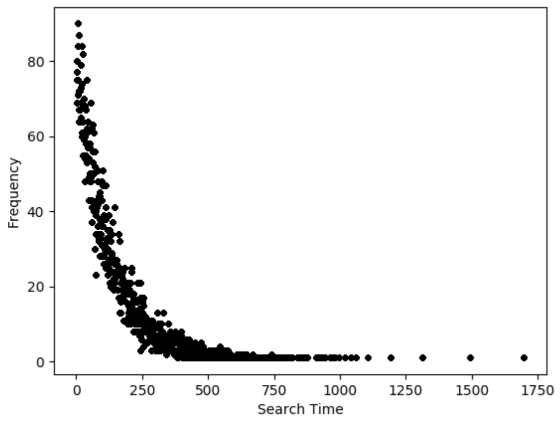
Table 2: Random graph, p = 0.5



(a) Query All



(b) Partial Query



(c) Random Walk

Figure 13: Random graph, p = 0.25

	Search Time		
	Mean	**Variance**	**Range**
Query All Neighbours	68.513	4462.571	832
Partial Querying	72.729	5341.012	903
Random Walk	83.752	7124.257	864

Table 3: Random graph, p = 0.25

3.2 Ring Group Graphs

Due to the structure of a ring group graph relative to that of a random graph, searching is more complex but heuristics can be used to reduce the search time. p > q, meaning nodes in the same or adjacent group to the target vertex are more likely to be connected. For the purposes of this report, 3 algorithms were created and compared between each other.

1. Query All: As above query every vertex and if the target is not found move to a random vertex.
2. Group Hierarchy: Create a list of all possible groups in the graph, ordered by preference. The preference will attempt to maximise the chance that the next vertex moved to is adjacent to the target vertex by either moving to the relevant adjacent/specific group or moving in an either clockwise or counter-clockwise direction round the ring so as to move closer to the target group and increase the probability of finding the target vertex.
3. Partial Querying: This method is intelligent and similar to the previous. However, it utilises a random walk, hoping to find a group adjacent to or containing the target vertex by chance. If adjacent to or in the same group, it queries all neighbours looking for the target, however if not, it randomly moves to another neighbour and avoids wasting search time by querying all nodes to look for a closer option.

Algorithm 2 Group Hierarchy Searching Algorithm

```
Require: $Graph, m, k, startVertex, targetVertex$
1: $searchTime \leftarrow 0$
2: $startGroup \leftarrow$ get startVertex Group    ▷ Function gets the group of a vertex
3: $targetGroup \leftarrow$ get targetVertex Group
4: $currentVertex \leftarrow startVertex$
5:
6: $orderedGroups \leftarrow groupPriority(m, k, startGroup, targetGroup)$
7:     ▷ List specifying preference for groups. Takes into account direction around the ring
8:
9: while $TargetVertexNotFound$ do
10:     $neighbours \leftarrow graph[currentVertex]$    ▷ The order of vertices in neighbours is random
11:     $oldGroup \leftarrow startVertex$
12:     $bestGroupFound \leftarrow$ lowest ranked group
13:     for neighbour in neighbours do    ▷ Loop through all neighbours of the current vertex
14:         $searchTime \leftarrow searchTime + 1$
15:         if $neighbour == targetVertex$ then return searchTime
16:         else    ▷ Return search time if target found
17:             $neighbourGroup \leftarrow groupNumber(targetVertex, k)$
18:             if oldGroup != neighbourGroup and ranked(bestGroupFound) < ranked(group) then
19:                 $bestGroupFound \leftarrow neighbourGroup$
20:                 $currentVertex \leftarrow neighbour$
21:                 if bestGroupFound equal to or adjacent to the targetGroup then
22:                     Break
```

The group hierarchy algorithm was the first that came to mind. The idea behind the implementation was that it is desirable to move as close to the target group and its adjacent groups as possible, as these vertices have a probability p of being adjacent to the target. Therefore, the algorithm ranks each of the groups in terms of its 'desirability', that in the immediate future, entering that group

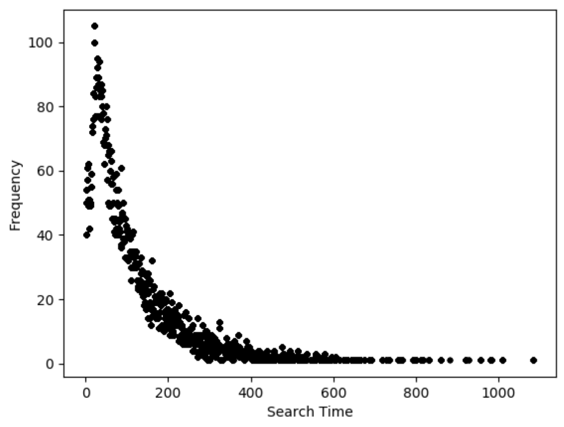
Algorithm 3 Partial Querying Searching Algorithm

```
Require: $Graph, m, k, p, q, startVertex, targetVertex$
 1: $searchTime \leftarrow 0$
 2: $targetGroup \leftarrow gettargetVertexGroup$
 3: $highGroups \leftarrow$ Target group and 2 adjacent groups
 4: $vertexChanged \leftarrow False$
 5:
 6: while $TargetVertexNotFound$ do
 7:     $currentGroup \leftarrow$ get currentVertex Group
 8:     $neighbours \leftarrow graph[currentVertex]$        ▷ The order of vertices in neighbours is random
 9:     if group(currentVertex) in highGroups) then
10:         for neighbour in neighbours do
11:             $neighbourGroup \leftarrow getneighbourGroup$
12:             $searchTime \leftarrow searchTime + 1$
13:             if $neighbour == targetVertex$ then return searchTime
14:             if neighbourGroup in highGroups then
15:                 $currentVertex \leftarrow neighbour$
16:                 $vertexChanged \leftarrow True$
17:         if vertexChanged then        ▷ If a vertex from the high group was not found
18:             $currentVertex \leftarrow$ random neighbour
19:     else
20:         $searchTime \leftarrow searchTime + 1$
21:         $currentVertex \leftarrow$ random neighbour
```

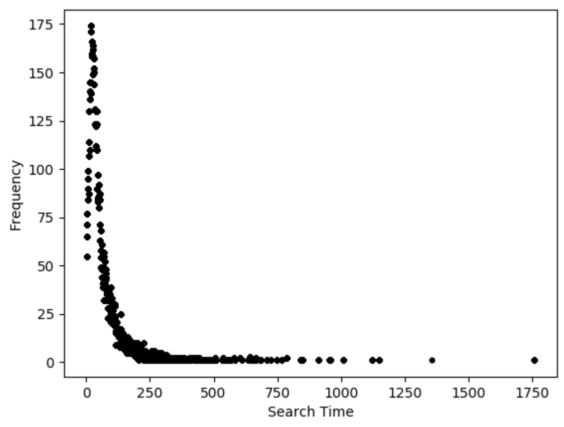
will yield a higher probability of finding the target vertex. If the current vertex is not adjacent to another vertex in the target group, then it should move to the groups adjacent to the target. If neither of these are possible, moving to another group is an option, ideally as close to the desirable groups as possible. Because of adjacency between groups, moving around the ring is possible for when the value of q is really low. In this case, one wants to move either clockwise or anti-clockwise round the ring depending on which route is shortest. This is what the group hierarchy decides. Once in the group or those adjacent, the algorithm stays inside the 3 groups because it is here that the highest chance of finding the target exists. The algorithm is therefore very good if presented with a relatively sparse ring graph with lots of small groups as it can effectively navigate between them. The issue with this implementation is the almost unnecessary checking of every neighbour, which is very heavy on the search time. When there is a low probability of finding the target, there is little reason to check every neighbour but the algorithm still has to in order to find the best possible group it can advance to next.

This is what inspired the Partial Querying algorithm. The algorithm follows much the same logic as group hierarchy except it only pays attention to the target group and those adjacent to it. At a vertex, if it is not in the same or adjacent group to the target, the algorithm will simply randomly choose a neighbour to move to recursively. If the group is correct, every neighbour will be queried looking for the target. If it is not found, another vertex adjacent to the current one will be moved to as long as it is in the same or adjacent group.

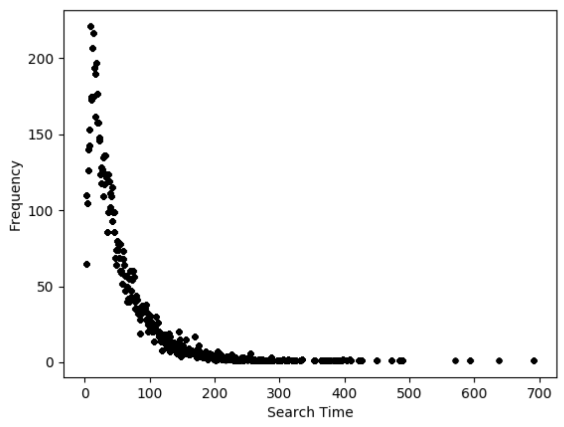
The two algorithms balance between them querying more often for more intelligent decision making, with spontaneous choice. Experiments were conducted on the a Ring Group Graph with parameters: m=k=10, p=0.4 and q=0.1. The sample size for calculation was 100,000 with the final results representing an average over each.



(a) Query All



(b) Group Hierarchy Algorithm



(c) Partial Query Algorithm

Figure 14: Ring Group Graph: m=k=10, p=0.4, q=0.1

	Search Time		
	Mean	**Variance**	**Range**
Query All	111.018	13211.223	1770
Group Hierarchy	57.927	6744.383	1758
Partial Query	41.616	2980.924	1639

Table 4: Ring Group Graph: m=k=10, p=0.4, q=0.1

Interestingly, the partial query algorithm was much more consistent at producing better results according to table 4. It yielded the lowest variance quite significantly and as can be seen in figure 14(c), and the lowest max search time. In comparison, group hierarchy has the highest maximum search time even when compared to a random query all strategy. The mean search time is lowest for partial query consistently but is closely followed by group hierarchy. Further tests would be required, experimenting with values of m,k,p and q to vary the graph structure and see which copes best. However, under the tests done here partial query has the lowest max search time, mean and variance between searches so is the logical choice for the search algorithm.

An additional feature was also added in latter stages of development, nicknamed the 'break chance'. In the incredibly unlikely possibility that either algorithm gets caught in a loop between the vertices in a group that are not directly connected to the target vertex (in the event of much lower p values), the algorithm will move to a random neighbour under a certain probability, currently set to:

$$BreakChance = c \cdot \frac{q}{pmk} \ where\ c\ is\ some\ constant$$

The rationale behind this is that for high values of q, the chances increase and if for higher values of any other parameter they decrease. High values of p mean the probability, if in the same group, target are higher. Likewise, for larger values of k, there are more vertices that have a chance of being adjacent to the target. High values of m mean more groups so if leaving the target room is will be harder to return. The break chance technique was experimented with and included for completeness but is only necessary for more extreme parameters. Break chance was implemented to prevent looping in experimenting sample sizes over 1 million where it could occur.