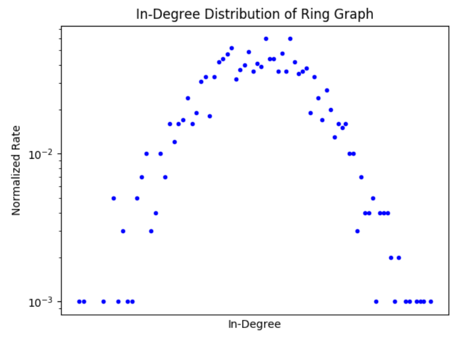
Elliot Jordan

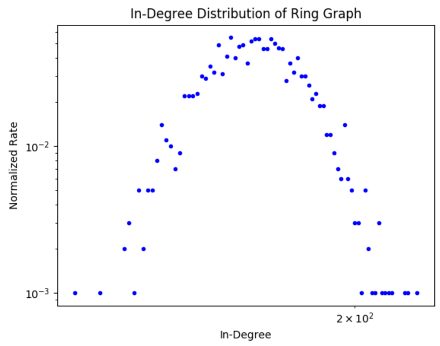
Network Analysis: Assignment

Question 1: Degree distribution of ring group graphs for p+q=0.5

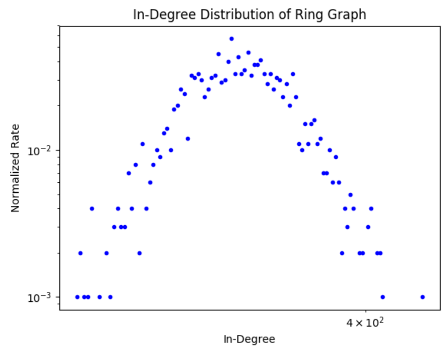
p\(m,k) (20,75) (75,20)

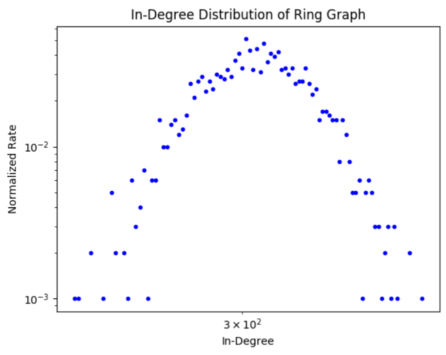
0.45
(0.05)



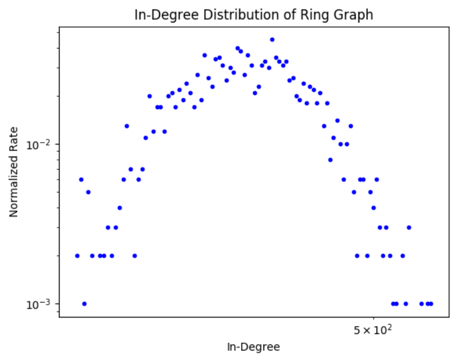


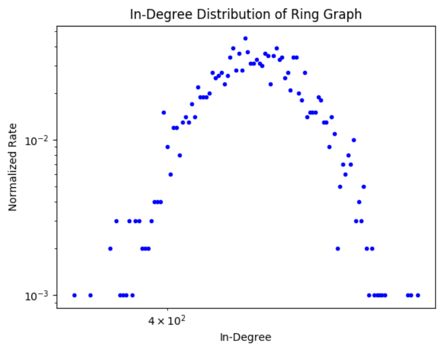
0.40
(0.10)



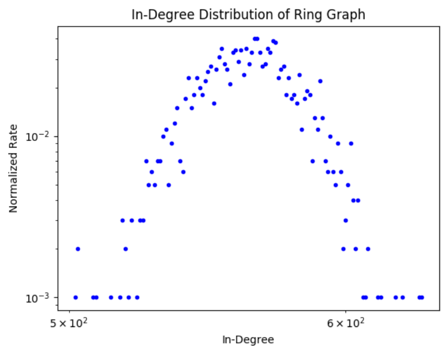


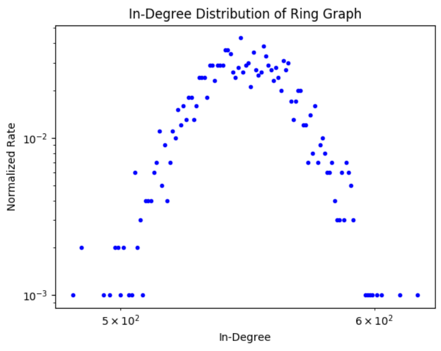
0.35
(0.15)





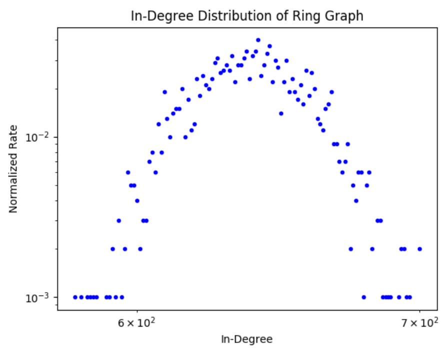
0.30
(0.20)

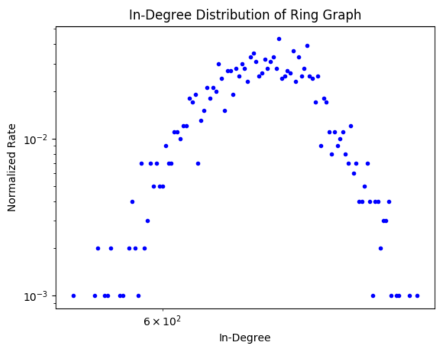




0.26
(0.24)

1

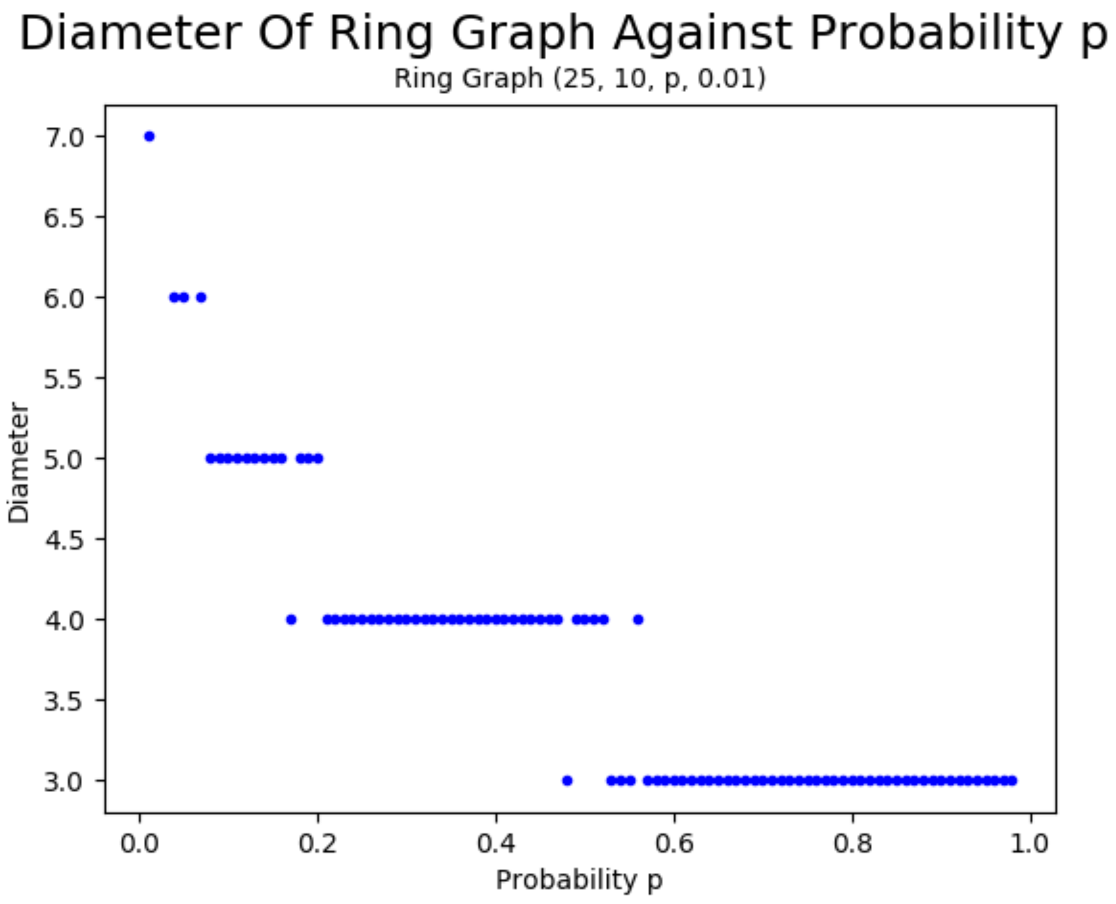




Investigating the degree distribution of ring group graphs for probabilities 0.45, 0.4, 0.35, 0.3, 0.26 found that as the probability p decreases (and q increases) the average degree of vertices in the graph increases. Simultaneously, both the shape of the graph and the normalised rate associated with each vertex degree remained consistent regardless of p and q (rate between 0.001 and 0.1). For the highest tested probability of 0.45, the average degree was approximately 175, with all degrees in the range 110-230. However, as p decreases and q increases the degree of vertices increases, to the point where for the graph p=0.26, q=0.24 the average degree is 638, with all vertex degrees falling within the range of 570-710.

These observations follow logically because as q increases, the chance that a vertex is connected to any vertex outside its group and surrounding groups is higher, and so a vertex is likely to have more connections. (e.g. If q was 0, vertex v could only be connected to 'local' vertices, a maximum of 3k).

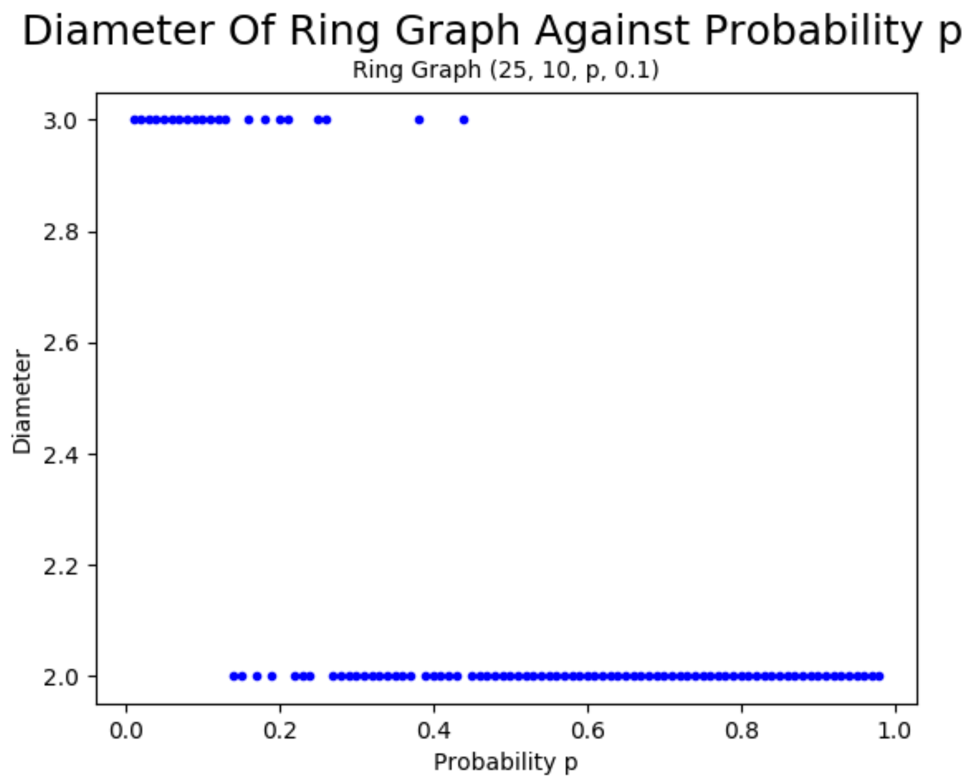
In experimenting with values of m and k many combinations were tested, however high values of m and k create more vertices and therefore more data points, which better presents the trend of the graph. The graphs above demonstrate both a large number of small groups, and a small number of larger groups, and show that regardless of the values of m and k, the trend of degree increasing as p decreases remains true.



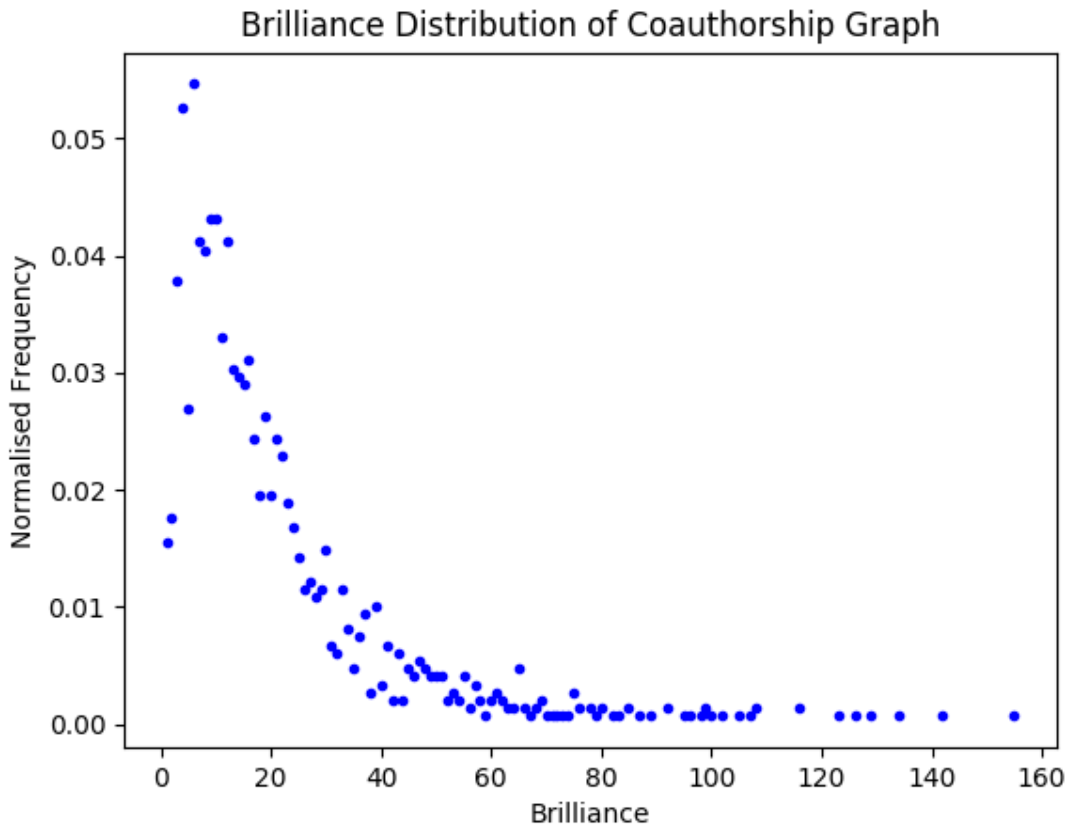
To investigate the diameter of ring group graphs, lower values of m and k were used because they created a graph which demonstrated greater variation in diameter and therefore better presented the trend of the graph. This is because for large graphs, either many pairs of vertices are not reachable, or due to large groups there is a greater chance any pair of vertices would be reachable with fewer steps. The graph above demonstrates diameter against variation in probability p, and does so with a small value of q = 0.01, because a negligible q means the ring graph will be less connected so variation in p will have a greater effect on diameter.

The graph shows that as p increases the diameter of the graph decreases, tending towards 3. In other cases (below, with a larger q), diameter tends towards 2, however both graphs demonstrate the same downwards trend. The graph above has few data points associated with lower p values, partly because pairs of vertices may not be connected when p and q are both negligible, and those that are will be connected by long paths. The graph increases exponentially as p tends to 0, which would be expected because as p reaches 0 it is very unlikely any given pair of vertices would be connected.

Both above and below (with larger or smaller q) the decrease in diameter off when p reaches 0.5, at which point two vertices in neighbouring groups are likely connected. This further increases the chance of pairs of vertices being connected. In the graph below, diameter tends to 2 as p increases because the graph is very well connected, however it is not completely connected, so there will exist pairs of vertices which are not connected (hence the path length of 2).



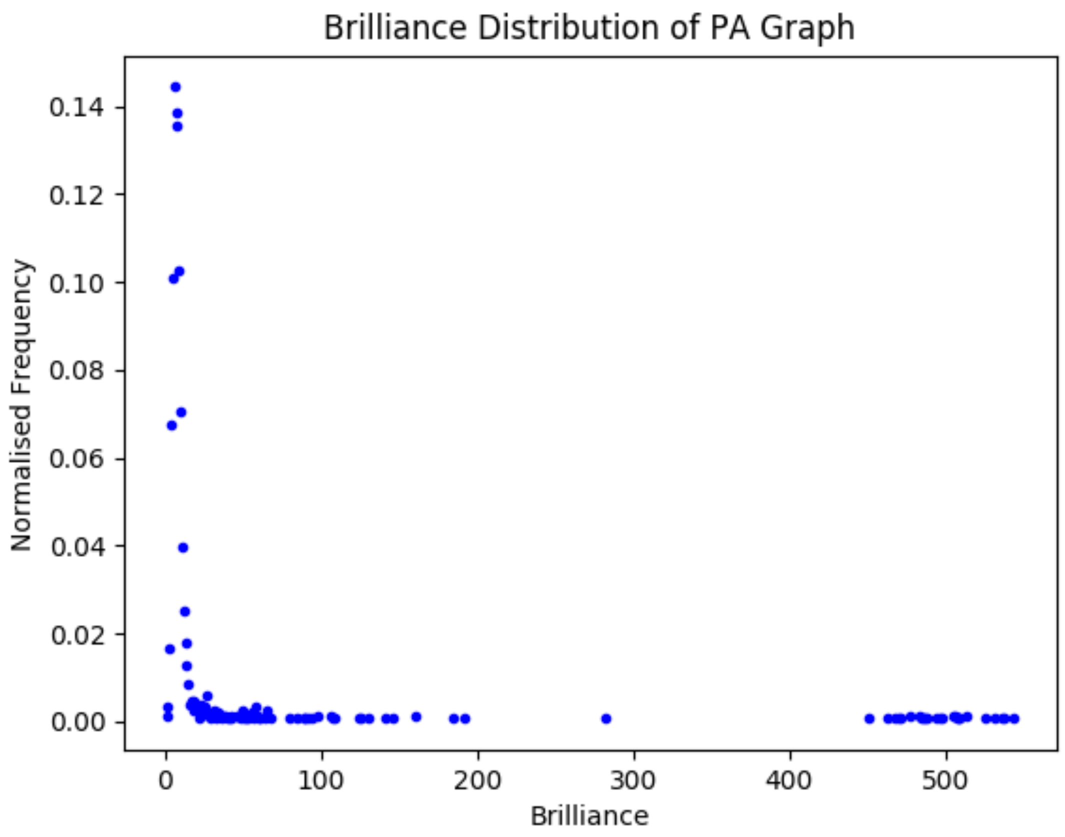
Question 2: Investigating Vertex Brilliance



The graph above shows the distribution of vertex brilliance for the graph in coauthorship.txt. Vertex brilliance is approximated using an algorithm to calculate a maximal independent set in the neighbours of the vertex in question v, which forms the k-star for v. It is maximal so no vertex can be added to improve the set, however I may not be the maximum independent set and is only an approximation.

Brilliance distribution forms an inverse Gaussian distribution – it peaks dramatically at a low value for brilliance between 5 and 10, after which it forms a curve with a long tail – where there are few or even 1 instance of high brilliance. In the context of a co-authorship graph this suggests that there is a minimum value where all authors have co-authored some papers, after which it becomes less likely an author has co-authored many papers with different co-authors who have also not co-authored with any others (hence the independent set).

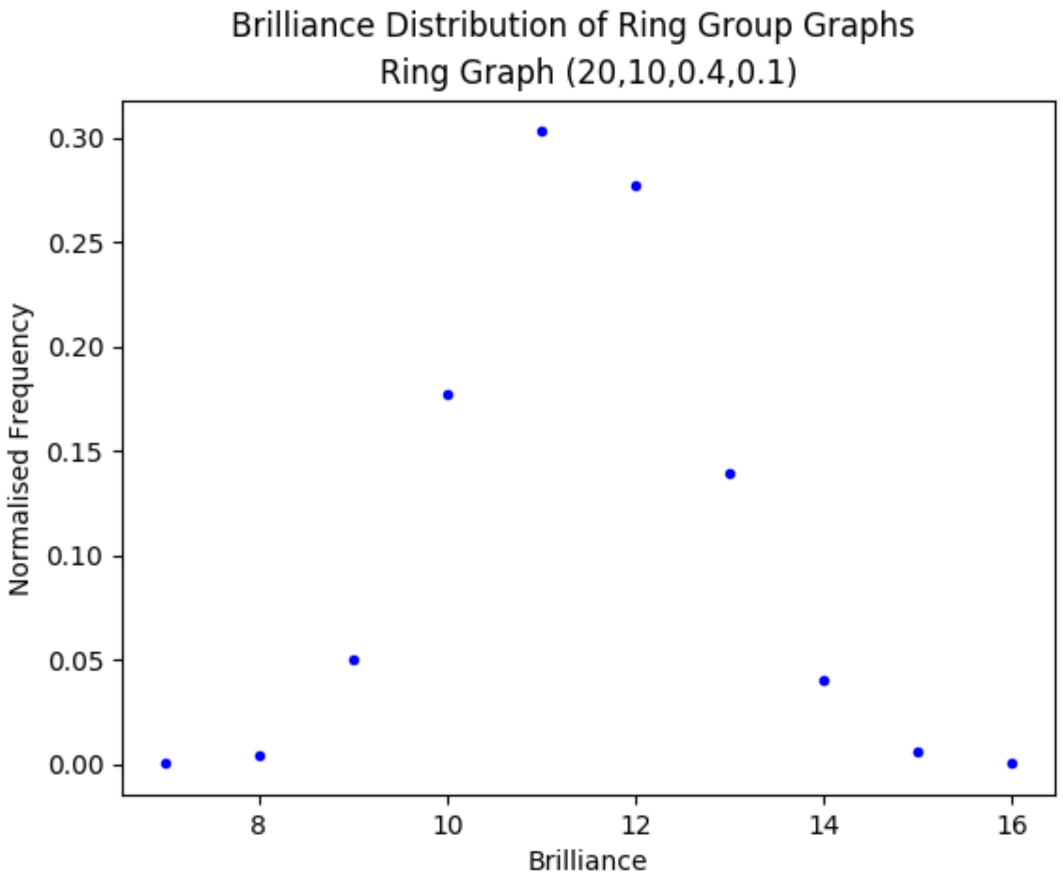
While there is a clear peak between brilliance 0 and 20, it is still a very low normalised rate of 0.055, so not many vertices share the same brilliance.



To make a PA graph with similar structure to the co-authorship graph, the number of vertices is 1559, with an out-degree of 20. From this, the above graph was created to show the distribution of vertex brilliance. The graph exhibits a similar properties to the co-authorship graph, with a peak at low brilliance and a trailing tail. The tail of this graph falls dramatically – not like the smooth curve in co-authorship – and continues to a far higher brilliance than co-authorship.

PA graphs are built by adding new vertices and connecting them to the current set according to how connected a vertex in the current set is. This model can explain the trends in the above graph. There is a cluster of instances distributed around brilliance 450 – 550 which is likely a collection of the hub vertices (likely some of the first added), which are increasingly well connected as the graph is built, and so will have a high brilliance. The normalised frequencies for the hub vertices are low, because there are only a few of them and they will have slightly different brilliance.

At the lower end of brilliance there are a large number of vertices, which is also justified by the shape of the PA graph. Vertices added later will be more poorly connected, because as vertices are added those with lots of connections will receive more while new vertices with only one connection are unlikely to be connected much further. This is why we see a large spike as brilliance approaches 0, however when brilliance is ~ 0 there are far few vertices because there are none which are unconnected, and for a graph of such a large size it is likely a given vertex is connected to a few vertices. In this graph unlike co-authorship almost the entire distribution is around the low spike in brilliance, where the normalised frequency is higher – reaching 0.14. Taking the few data points in the spike together makes up for over 0.9 (90%) of all vertices in the graph.



Investigating the distribution of brilliance in a ring group graph resulted in an output different to that of the previous graph types. For the ring graph the range of brilliance is far smaller, with many occurrences of each brilliance value and thus far higher normalised frequency. This pattern followed for larger and smaller graphs with varying m, k, p and q, all resulting in similar curves. In order to create the graph above, 100 ring graphs were used and their results combined. While this created enough data points to clearly show the curve of the graph, the data is largely the same as for a single graph.

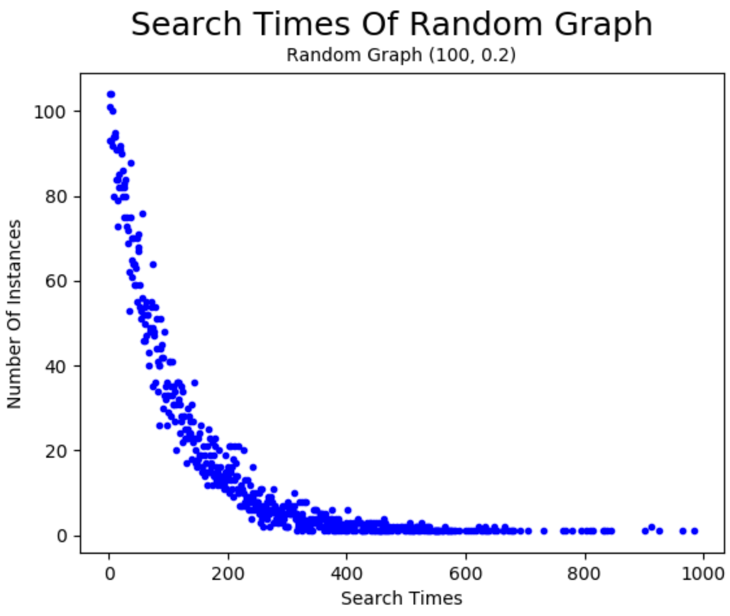
The close distribution of brilliance is likely due to the fact that vertices grouped together are more likely to be connected and therefore will not all appear in the brilliance (as they share neighbours). The number of connections of any vertex in the graph is relatively consistent – there are no hub nodes with high brilliance nor a large number of less connected vertices.

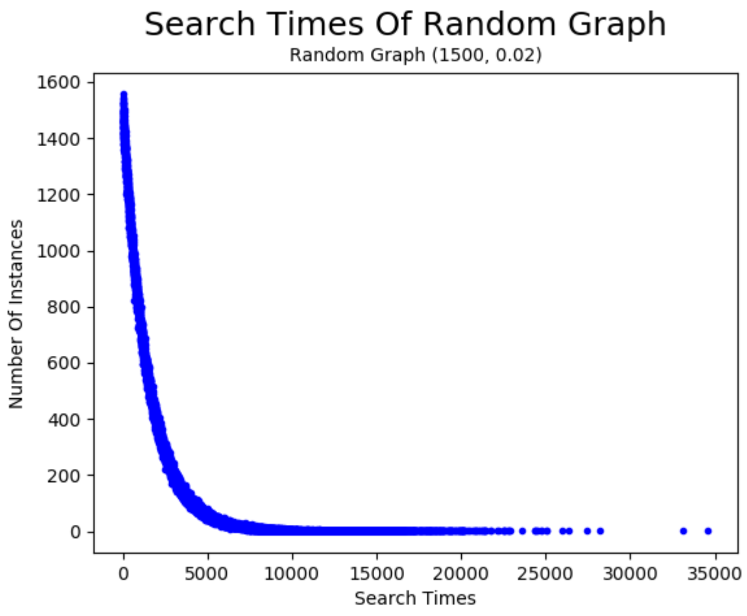
Summary

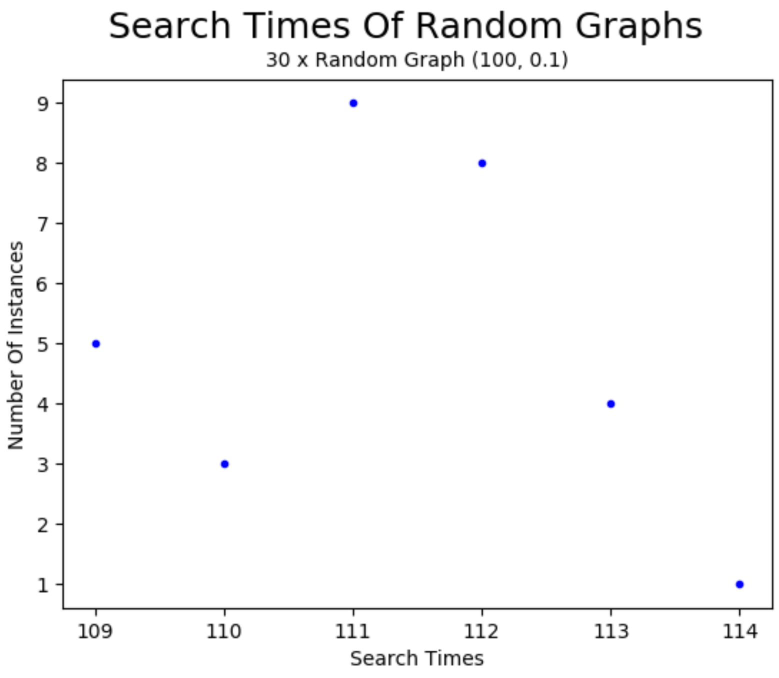
Looking at co-authorship in comparison to PA and ring graphs, it is clear that the co-authorship data shares key characteristics with the PA graph. This suggests the data follows a similar structure: A few well connected vertices where prolific authors frequently publish co-authored papers, while the majority of authors are likely to have less co-authored papers and contribute to the early peak in the graph.

The co-authorship graph does however have a less extreme spike and a more gradual tail, which suggests more of a range in the co-authorship data than in a PA graph where at the low end of brilliance a large number of vertices will be sparsely connected and therefore have a low brilliance.

Question 3: Searching In Graphs







The graphs above show the results of the random search algorithm on a random graph. The top two graphs correspond to individual graphs, where each data point is a search time between two vertices, while the graph below plots the average search times of 30 graphs (all random with 100 nodes and p = 0.1).

The algorithm used aims to minimise the number of queries made. Because querying additional vertices does not gain any additional information apart from the id of the vertex and whether it is the target, it is more efficient to only query one vertex and immediately move to that vertex, continuing until the target is found.

For example, if a vertex v is not connected to t, querying its neighbours will take search time and the result will be the same as if only one query was made – a single vertex will be selected at random from the list of queried vertices. If the vertex v is adjacent to t, then querying before making a decision on where to move would be more efficient, however for most of the search duration v and t will not be adjacent vertices, and so a random jump is far more efficient. Also, because once a move is made all information on the previous vertex is lost, there is no advantage to making more queries, because returning to that vertex will still require the queries to be made again, so a map of vertices that have been visited cannot be made (which would make an algorithm with more querying more efficient).

Due to this reasoning, the algorithm used only makes one query at each vertex and immediately moves to the new vertex until the target is found. This strategy resulted in the search times presented in the above graphs. For both a relatively small (100 vertices) and very large (1500) graphs, this method created a curve where there are a large number of search results for lower search times, and a long tail with very high search times occurring infrequently.

The algorithm tends to perform efficiently, moving quickly between vertices until the target is found, however the long tail (results of up to 1000 queries for a graph of only 100 vertices) suggests that, perhaps due to the lack of structure to the search, occasionally the search performs poorly. In a random graph the structure is consistent, so these larger search times are likely due to the chance that the search, purely by chance, doesn't choose the right edge to the target vertex when on adjacent vertices many times, meaning it visits the same vertices many times and takes longer to search.

The third graph shows the average search times across 30 random graphs generated the same way. It shows that the average search times are consistent and fall within a small range (109-114, range of 5). The graph also forms roughly a bell distribution where the central results are more likely, which suggests that the algorithm preforms consistently.

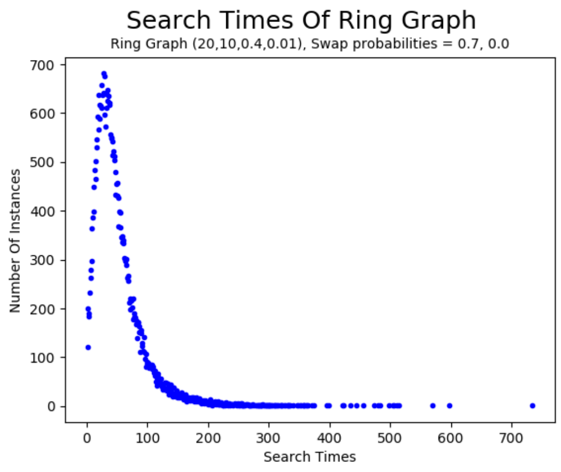
Ring Graphs

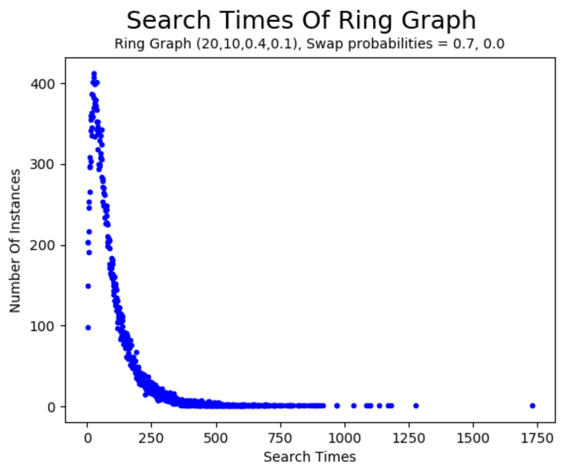
The key feature of ring group graphs which makes them more efficiently searchable is the groupings, where the target and current vertices both have groups. As adjacent groups (and within the groups themselves) are better connected (p>q) than groups further apart, an effective search strategy is to move towards the target group, and once it's reached, remain within it to search for the target.

The algorithm I designed starts at vertex s, and queries a neighbour of s randomly. It then checks whether the group of the new vertex is closer to the target group (this includes both in the forward and reverse directions as groups 0 and m-1 are considered adjacent in the ring), and moves to the new vertex if it is closer. Once the target group is reached the algorithm checks neighbours for t (because within the group it is likely they are connected) and terminates if t is found, else it moves to another member of the group and continues.

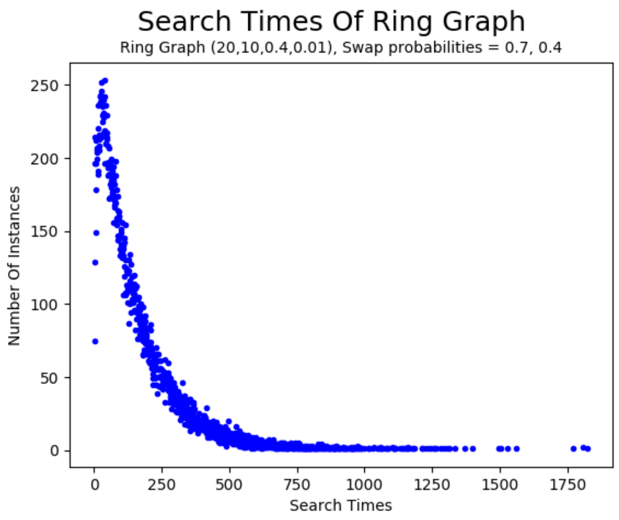
To expand on that base algorithm, additional probabilities were introduced: x, the probability that if the vertex queried is in the same group as the current vertex (not the final group) the move is made anyway, and y, the probability that if the group is further away from the target it is accepted anyway.

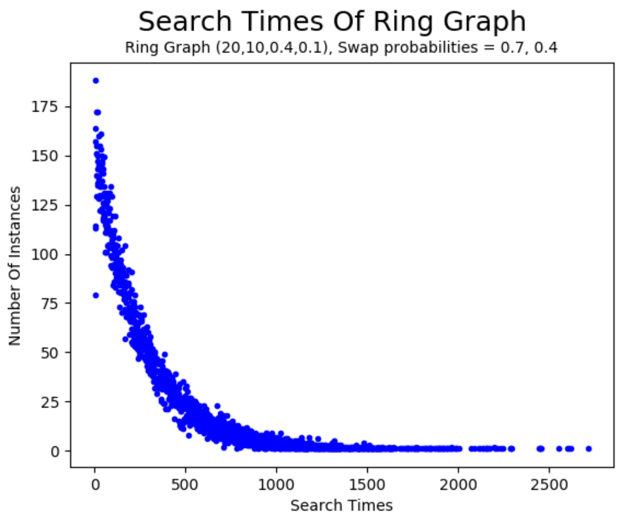
Once the current vertex is in the same group as the target, all the neighbours are checked to see if the target (in the same group so likely adjacent) is found, and if not, the search moves to another vertex in the group and checks again until t is reached.



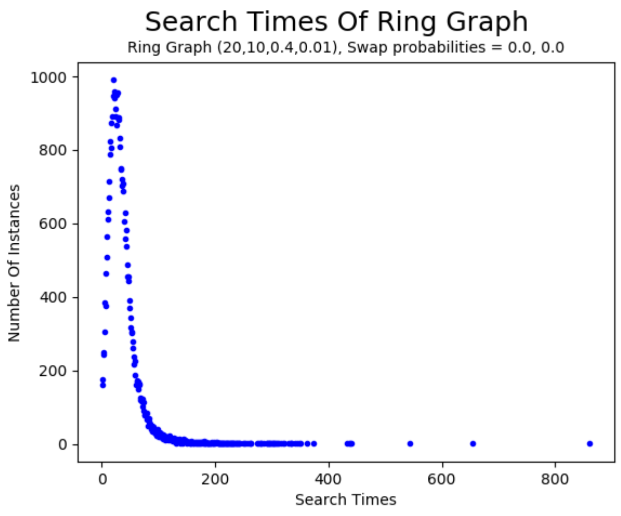


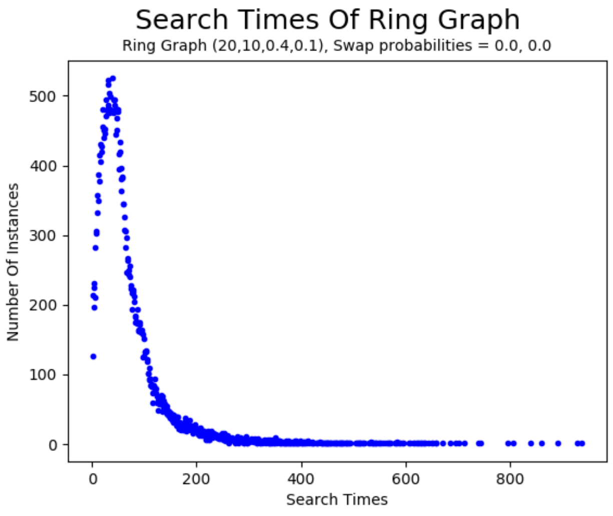
The first two graphs show distributions where the probability of moving within the same group is high: 0.7, but the probability of moving to a worse group is 0. The left graph is a ring graph with q = 0.01, so connections outside adjacent groups are unlikely, while they are more likely connected in the right graph where q=0.1. The algorithm performs better on the graph with the negligible q, because it can efficiently move towards the target by moving in adjacent groups in both cases, but when q is far smaller less queries are wasted checking vertices which are further from the target. This is shown by the higher peak in the graph (~700 vs ~400) and shorter tail, with worst case 750 against 1750 for the more connected graph (because it makes a large number of wasted queries).





In the subsequent two graphs, y (probability of moving to a worse group) has been added at 0.4, and the result is a far less efficient algorithm, with less cases performing well (the peak in the low end of search times is lower) and more long search times (longer tail). This reinforces the idea that the algorithm is efficient by moving towards the target group, and although there is theoretically a chance that skipping back to another group could find a shortcut, it is unlikely and the algorithm performs worse.





The final two graphs show the effect of reducing the probability x of moving within the same group to 0 meaning that, outside of the target group, the algorithm only makes a move to a queried vertex if it is closer to the target. This method also improved results, as shown in both the graphs. The algorithm is again far more efficient when q is 0.01, wasting less time on backwards checks, and has a very high number of instances for short search times. The tail is still evident, however the tail is very short, excluding 5 outliers, which produces a very efficient algorithm. The same can be said for the graph with q=0.1, where the tail is significantly shorter than where x > 0 (less than half the worst case when x = 0.7).

Summary

When searching ring group graphs, the most efficient algorithm proved to be one which always moved in the direction of the group of the target, and searched within the group once it was reached.

The shape of the graphs produced by the search times was consistent – an inverse Gaussian distribution, where very low search times were less likely, but frequency quickly climbed to a peak where searches for many pairs of nodes produces similarly fast results. The descent to the tail of the graph was most steep in cases where x and y were both 0 (moves only made if they improve the current group) , because nearly all results fell within a range of low search times, with far less cases of longer search times. These cases also produced the highest peaks, where most search times were similarly low.

Because the algorithm normally aims to move towards the target group and is likely be able to do so, the tails of the graphs tend to be short, excluding a few outlying cases. This is true when the algorithm doesn't move backwards, however for graphs where y = 0.4, there were more tail cases with long search times, and the worse cases were far worse, creating a longer tail.