Artificial Intelligence: Assignment 2

Part 2

Aoibh Tully: 20376871

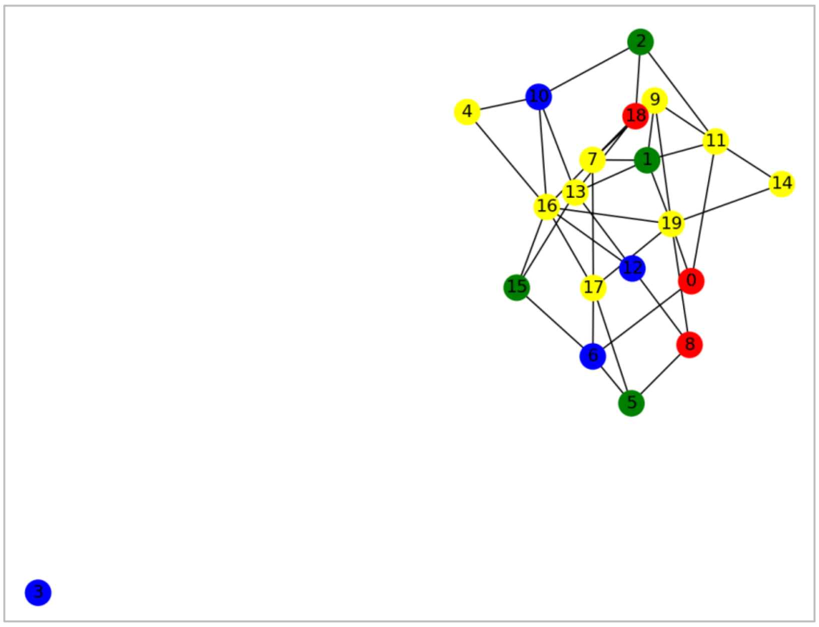
Code Link:
https://github.com/aoibhhtullyy/ArtificialIntelligenceAS02/tree/main/PartBCode

Research Question: Does changing the topology of a generated network change the colouring effects?

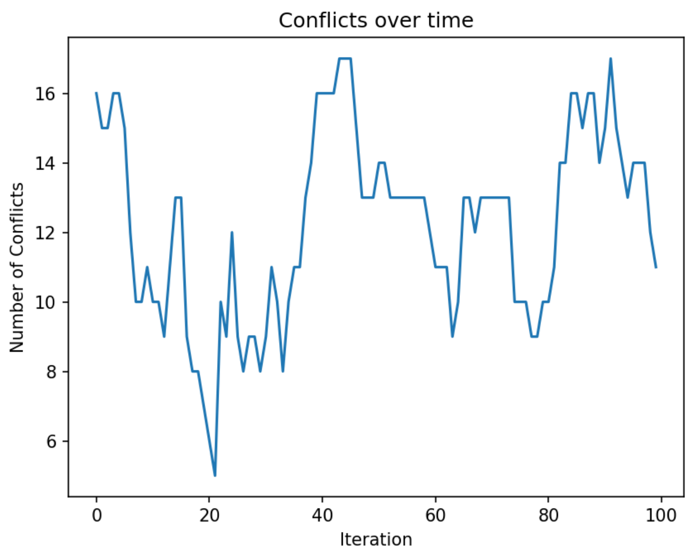
The avenue I decided to explore in expansion from Part A, is to see the difference in using other network topologies. In my first part of this assignment I focused on using random geometric graphs as provided by the networkx libraries. In this part, I will apply the same experiments to the graphs, but using different topologies. The topologies I will be using are, an Erdos_Renyi , and a small world graph (in networkx as barabasi_albert_graph).

Erdos Renyi Graph

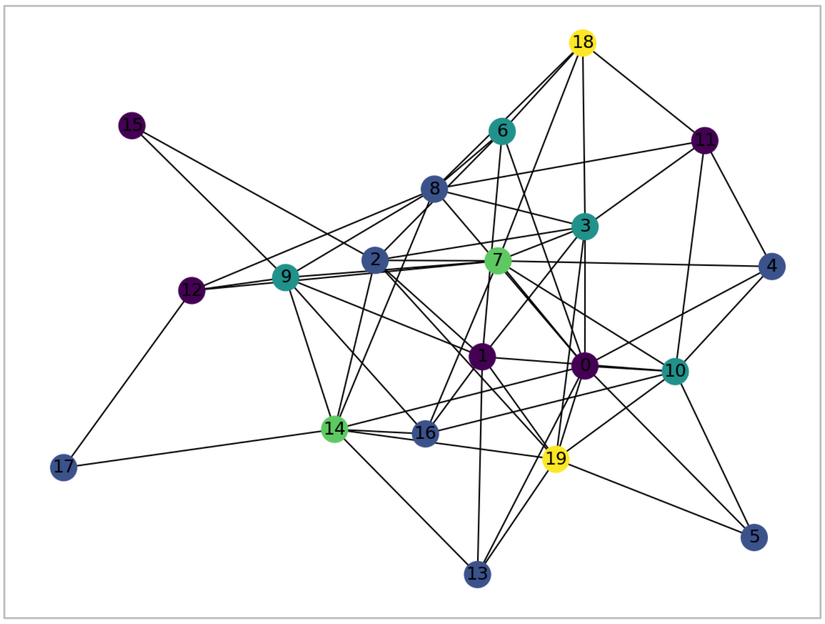
To begin, this is how a generated erdos-renyi graph looks, when nodes are assigned random colours:



Similar to part A the first experiment I carried out was to iterate through the nodes and assign them random colours and mark their conflict values. The values of conflict were then plotted over time:



I then applied the greedy colouring algorithm to this type of network topology and the following graph was produced:

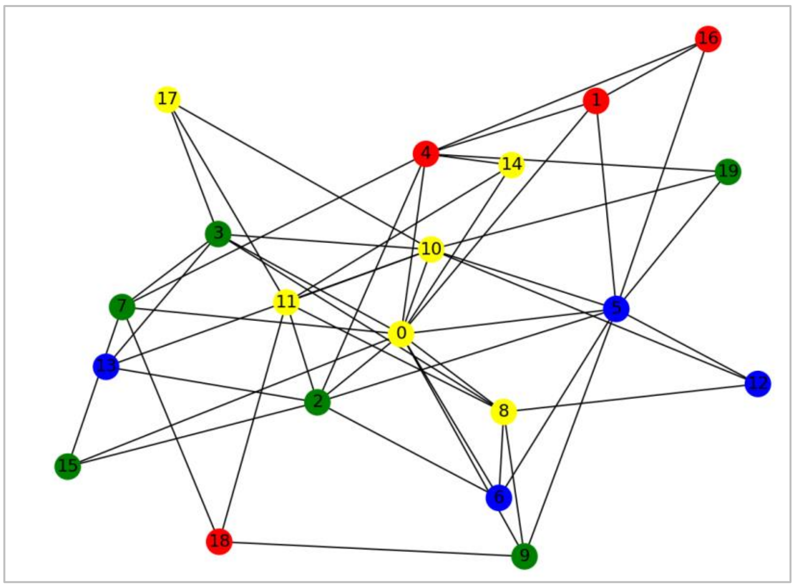


Just like the results from Part A, the greedy colouring algorithm provided the most optimal results.

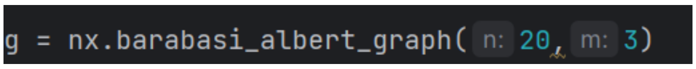
As erdos-renyi graphs are similar to the networks I used in part A, in that they are randomly generated graphs, the results don't differ much between them.

Small World Networks (Preferential Attachment)

Within the networkx library there is functionality to create a preferential attachment style network graph. When using this, and applying random colouring to nodes, the following is produced:



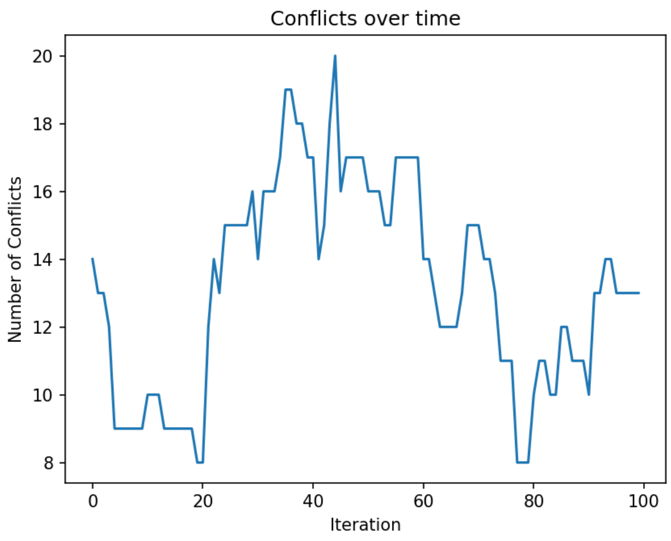
The first notable difference when using this type of network graph, is when defining it you must define the number of edges to be connected to each node, like this:



```
g = nx.barabasi_albert_graph( n: 20, m: 3)
```

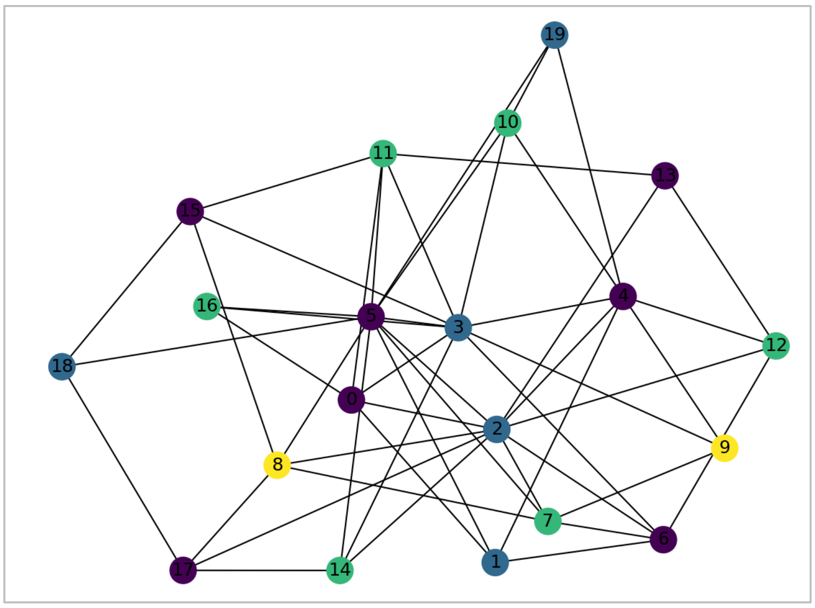
This creates a network graph with 20 nodes and each node has 3 edges connected.

To continue with the experimentation, the approach to assign a different colour value to each node over 100 iterations was run next. This was the graph of the conflict values collected over the iterations:



As this code is run with random colours, it provides random results similar to the results above. Within the first few iterations a low value of 8 conflicts is reached. However the possibility of there being a conflict value of 20 exists, as seen on the graph.

Following this the greedy colouring algorithm was applied to the generated graph, and this was the result:



Comments on Results

Overall, I don't see many differences between the network graphs in terms of colouring when using different topologies. What I take from this is that the greedy colouring algorithm is very efficient in its ways of correctly colouring graphs, with relatively low chromatic values.

In the future, I think a more efficient approach to understanding and dissecting the colouring approach to network graphs, would be to work with the concept of assigning random colours to the nodes. It may include noting during the iterations, the graphs which provide the lowest conflicts, and displaying them as the final result.

The graphs of conflict show that, even with a system that only uses 4 colours, there are many options in how to colour a graph, each leading closer to the correct colouring (with no conflicts, as seen when observing the greedy colouring graphs).