REPORT- Group15

___Part1____

Creation of graph : Parsing json files, creating a dictionary and building a graph:

```
In [4]: start = time.clock()
   ...: buildGraph(process_data(pathname)[0])
   ...: print ('Execution time : ',time.clock() - start)
   ...:
Graph created with: 7771 nodes, 16489 edges
Execution time :  0.17017258666666635
```

____Part2____

Data visualization 1: Creating a subgraph based on conference id and drawing plots of centrality measures:

```
In [11]: start = time.clock()
    ...: conference_subgraph(authors_graph)
    ...: print ('Execution time : ',time.clock() - start)
    ...:
Execution time :  6.1546845866666615
```

Figure1. Plot of the subgraph

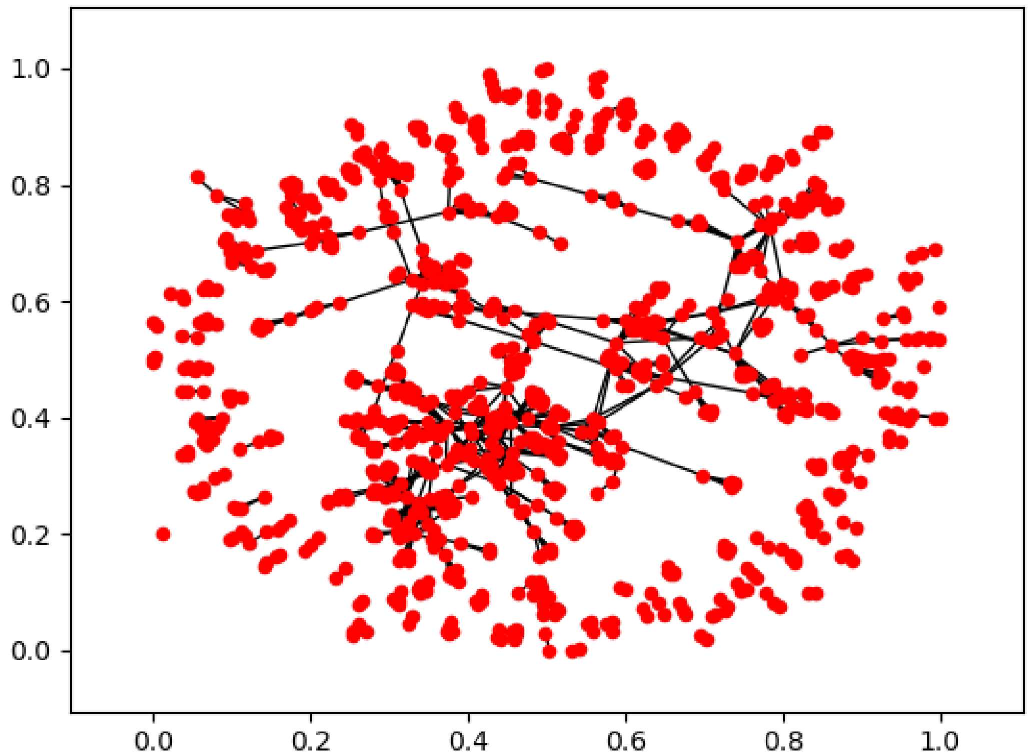


Figure2.Degree centrality

The degree centrality for a node v is the fraction of nodes it is connected to. We can see from our plot that only a small number of nodes has really high degree centrality.

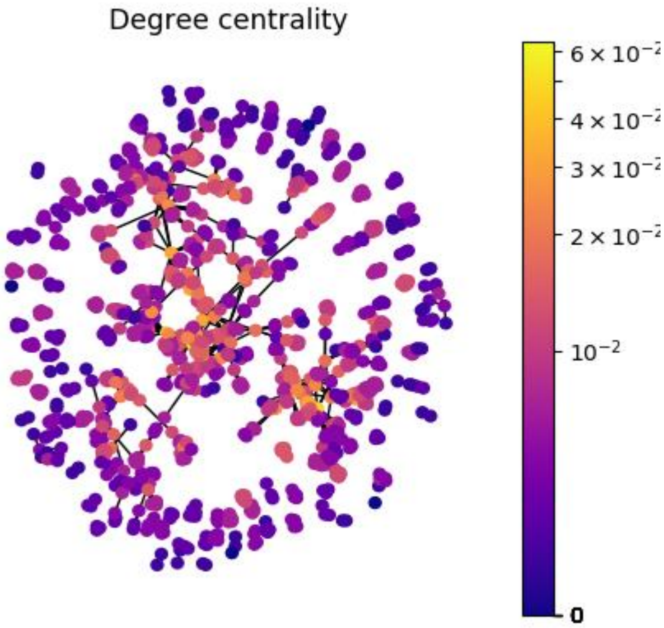


Figure3. Betweenness Centrality

Betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v.

Also here, we see that only a small number of nodes has really high betweenness centrality

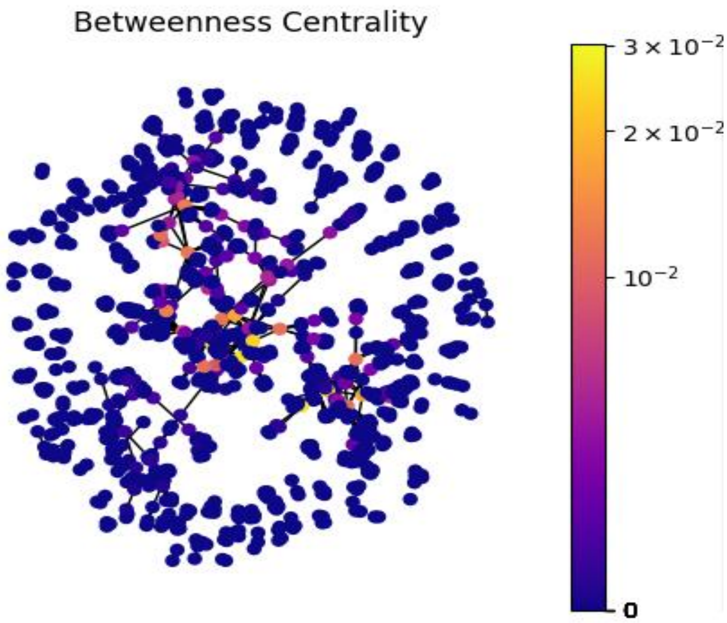
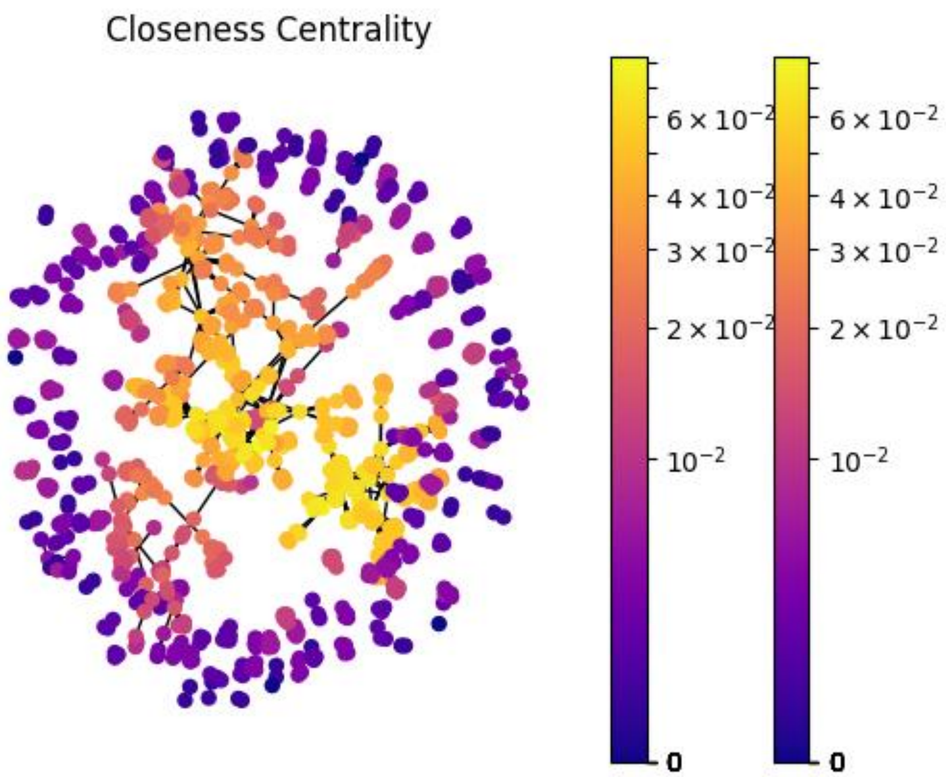


Figure4. Closeness Centrality

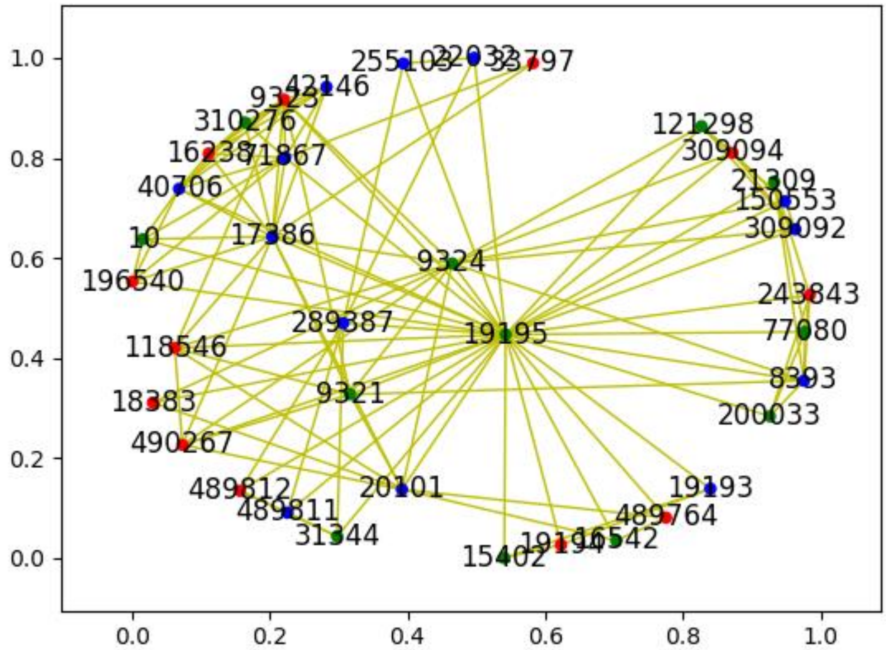
Closeness centrality of a node u is the reciprocal of the sum of the shortest path distances from u to all n-1 other nodes. Since the sum of distances depends on the number of nodes in the graph, closeness is normalized by the sum of minimum possible distances n-1. We can see that most of the nodes have low closeness centrality.



Data visualization 2: Creating a subgraph based on author id and maximum length of paths

```
In [13]: start = time.clock()
    ...: author_subgraph(authors_graph,authorid=10,d=5)
    ...: print ('Execution time : ',time.clock() - start)
    ...:

Execution time :  0.12127701333326968
```



___Part3_____

Shortest part from any node to Aris: performed by Dijkstra's algorithm