REPORT(good)

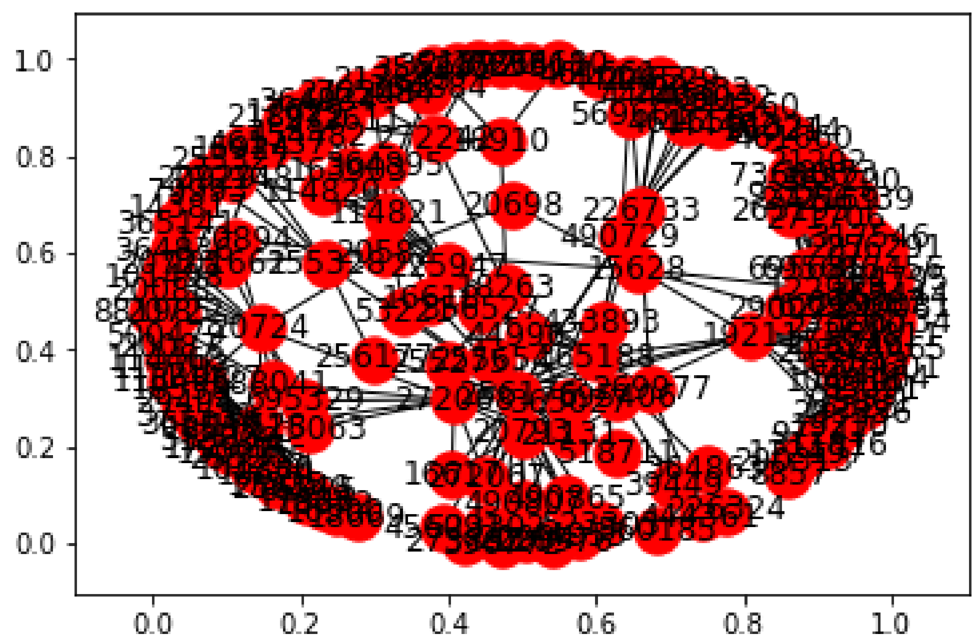
December 21, 2017

```
In [68]: import json
         import networkx as nx
         import matplotlib.pyplot as plt
         import itertools
         import heapq
         import Modules

In [61]: with open('/Users/Dario/Desktop/Aris_Homeworks/AMD_Homework_4/full_dblp.json') as data_
             data = json.load(data_file)

In [69]: import importlib
         importlib.reload(Modules)
```

```
Enter the number of part you want (2a, 2b, 3a, 3b) or type 'save me', if you want to quit: 2b
Please, give me an ID of the Author: 256176
Please, tell me, what distance of neighbors do you want?: 3
```



```
Enter the number of part you want (2a, 2b, 3a, 3b) or type 'save me', if you want to quit: save
It was pleasure to meat you
```

```
Out[69]: <module 'Modules' from '/Users/Dario/Desktop/Modules.py'>
```

```
In [62]: dictAuthor = Modules_2.Author(data)
         dictPubl = Modules_2.Publ(data)
         dictConf = Modules_2.Conf(data)
```

```
In [63]: Gall = nx.Graph()
         for k,v in dictAuthor.items():
             Gall.add_node(k, id = v[0])
```

```
In [64]: for k,v in dictPubl.items():
             for i in itertools.combinations(v,2):
                 Gall.add_edge(i[0],i[1], weight = Modules_2.JaccardDistance(dictAuthor[i[0]],di
```

```
In [65]: print(nx.info(Gall))
```

```
Name:
Type: Graph
Number of nodes: 904664
Number of edges: 3679473
Average degree:   8.1345
```

```
In [24]: n = int(input())
         subgraph = Gall.subgraph(dictConf[n])
```

```
5262
```

1 Homework 4: Group 19

2 Algorithmic Methods of Data Mining

2.0.1 Through this PDF, we're going to explain all the results we obtained in the project.

2.0.2 First of all we created the Graph composed by:

- Number of nodes: 904.664
- Number of edges: 3.679.473

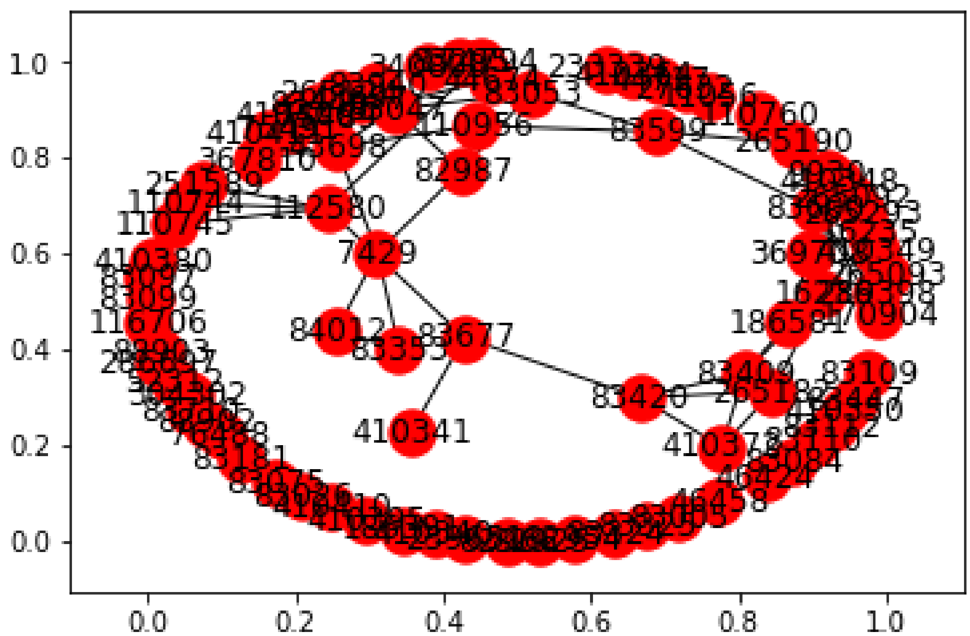
2.0.3 At point 2.a, given a conference in input, we had to return the subgraph induced by the set of authors who published at the input conference at least once.

2.0.4 The number of the conference has to be given by input, but for showing the subgraph we've used as example the conference number 5262 with:

- Number of nodes: 85
- Number of edges: 138

2.0.5 obtaining the following subgraph:

In [25]:



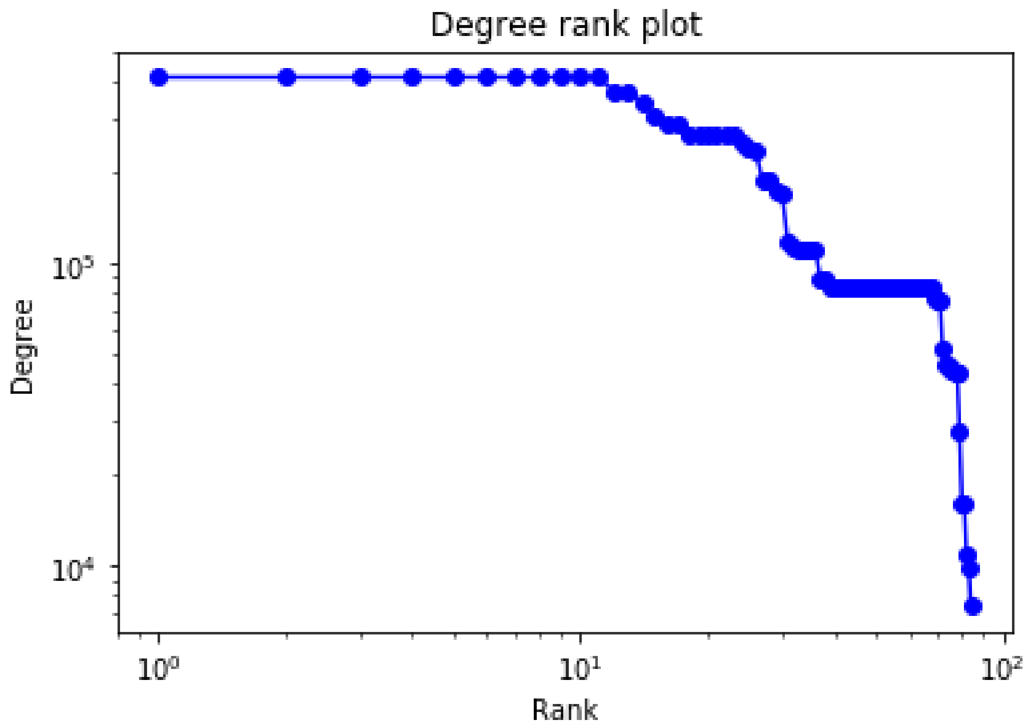
2.0.6 On this subgraph we computed some centralities measures (degree centrality, closeness centrality, betweeness centrality) and plotted them.

2.0.7 Before we show the results, we'll explain what these measures does and how they are defined.

3 Degree Centrality:

3.0.1 This measure is defined as the number of ties that a node has. If the graph were directed, we define two separate measures of degree centrality, namely *indegree* and *outdegree*. * Indegree* is a count of the number of ties directed to the node and *outdegree* is the number of ties that the node directs to others. In this case, our graph is not directed.

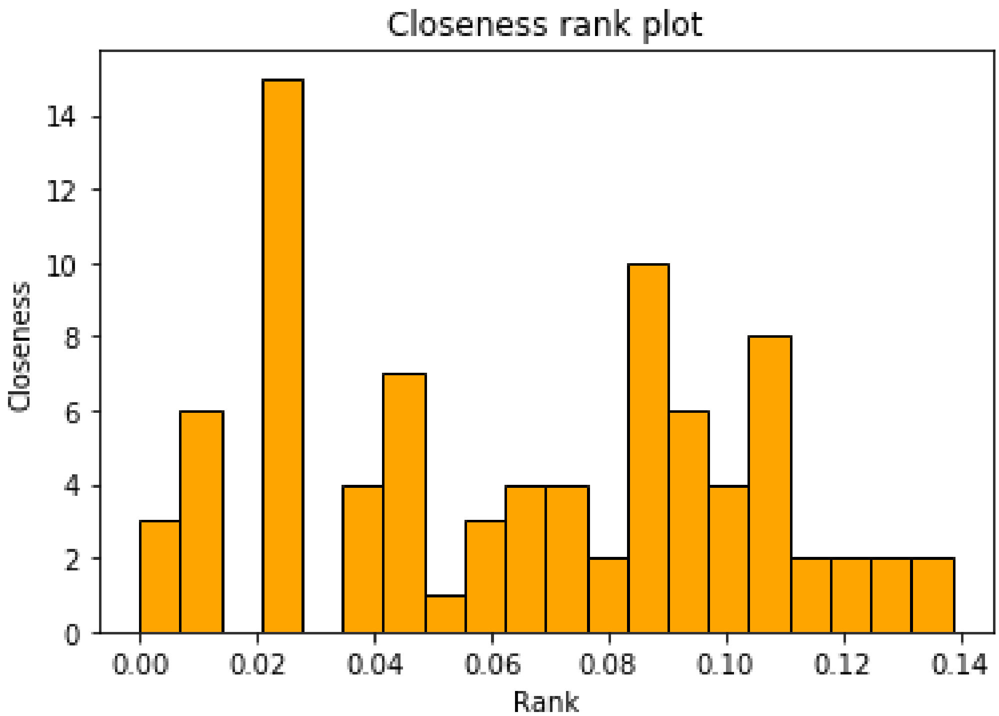
In [32]:
```
Modules.degree(subgraph)
```



4 Closeness Centrality:

4.0.1 In a connected graph, the normalized closeness centrality of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus the more central a node is, the closer it is to all other nodes.

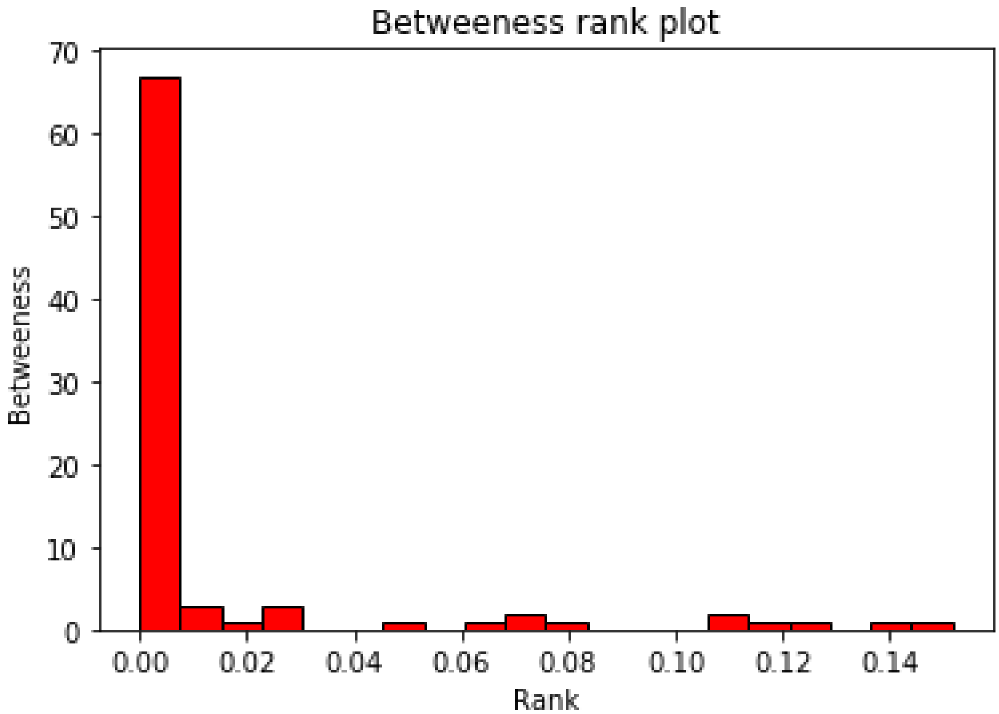
```
In [33]: Modules.closeness(subgraph)
```



5 Betweeness Centrality:

5.0.1 Betweeness Centrality represents the degree of which nodes stand between each other. It means that a node with higher betweenness would have more control over the network, because more information will pass through that node.

```
In [37]: Modules.betweeness(subgraph)
```

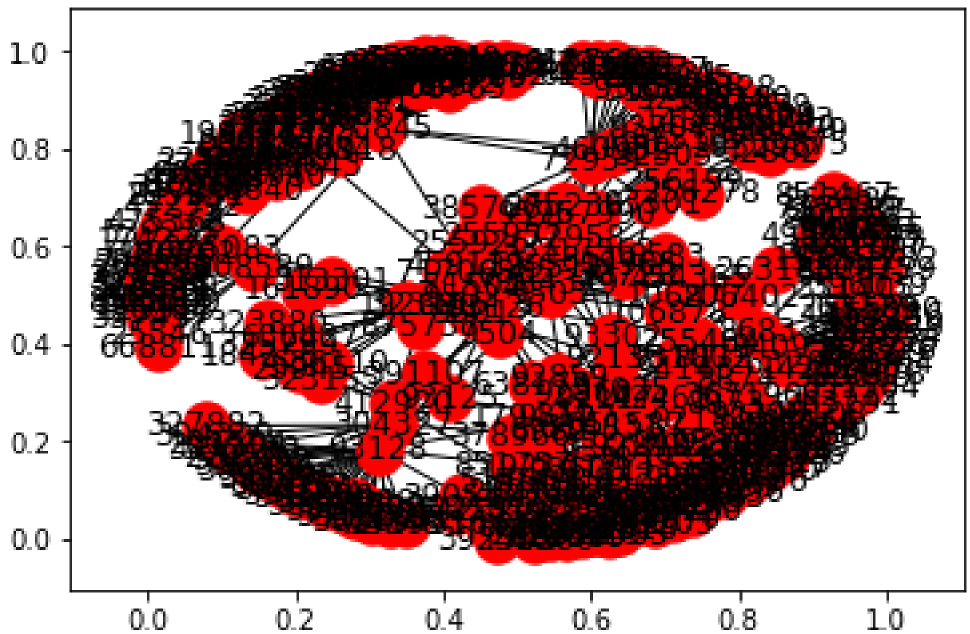


5.0.2 After statistics visualization, we created the subgraph induced by the nodes that have *hop distance* at most equal to an integer *d* with an input author.

5.0.3 The result is shown below.

```
In [77]: Modules.hop_distance(Gall)
```

```
Please, give me an ID of the Author: 93126
Please, tell me, what distance of neighbors do you want?: 2
```



5.0.4 In the exercise 3.a we had to implement the Dijkstra algorithm which calculate a generalized version of the Erdos number. In this case Erdos is Aris and our output will be the distance between an input author and Aris.

```
In [72]: Modules.dijkstra(Gall,256176,256177)
```

```
Out[72]: 0.9565217391304348
```