# Report

December 22, 2017

## 1 Homework 4: Group 19

### 1.1 Algorithmic Methods of Data Mining

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

#### 1.1.2 First of all we created the Graph:

```
In [2]: data = Modules.loader('/Users/Dario/Desktop/Aris_Homeworks/AMD_Homework_4/full_dblp.json
```

```
In [3]: dicAuthor = Modules.Author(data)
        dicPubl = Modules.Publ(data)
        dicConf = Modules.Conf(data)
```

```
In [4]: Gall = Modules.createGraph(dicAuthor)
        Gall = Modules.addEdges(dicPubl)
```

#### 1.1.3 composed by:

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

#### 1.1.4 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.

```
In [6]: n = int(input())
        subgraph = Gall.subgraph(dicConf[n])
```

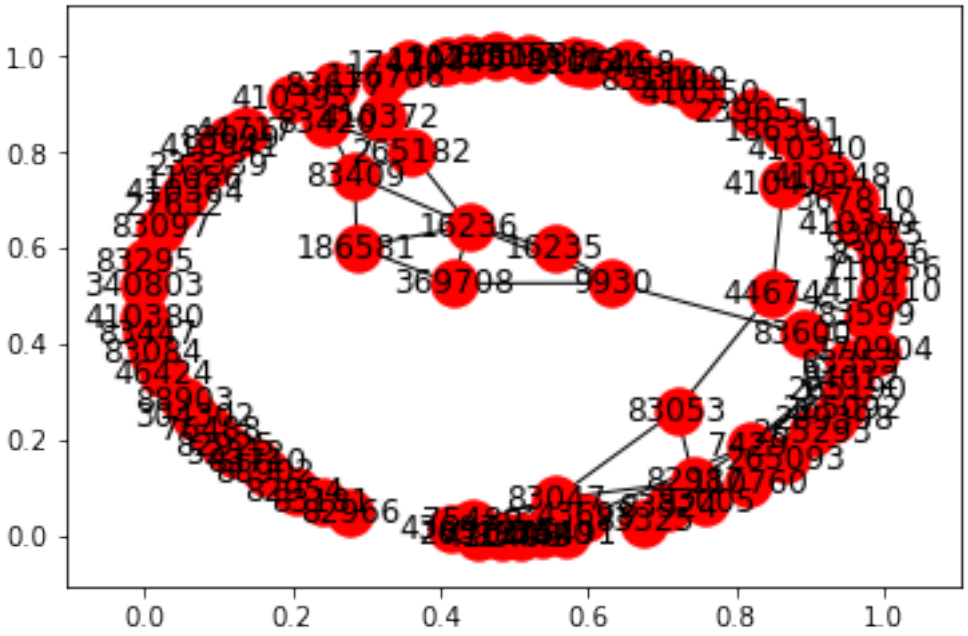
```
5262
```

#### 1.1.5 Like we told above 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

#### 1.1.6 obtaining the following subgraph:

```
In [7]: Modules.draw_graph(subgraph)
```



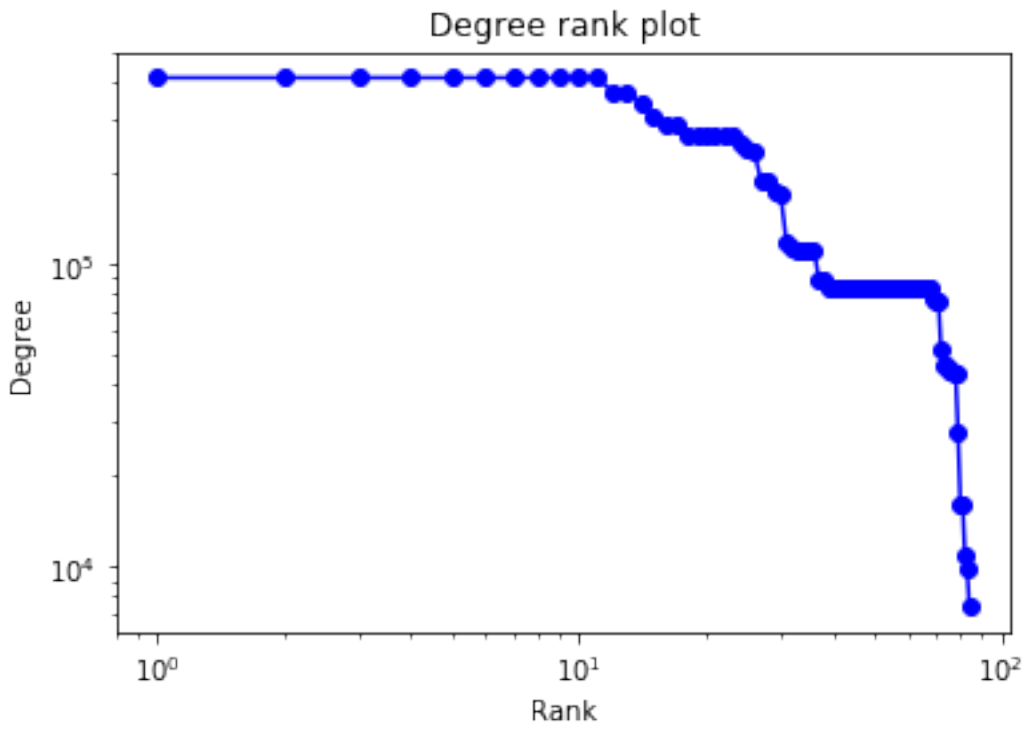

#### 1.1.7 On this subgraph we computed some centralities measures (degree centrality, closeness centrality, betweeness centrality) and plotted them.

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

## 2 Degree Centrality:

#### 2.0.1 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 [8]: Modules.degree(subgraph)
```

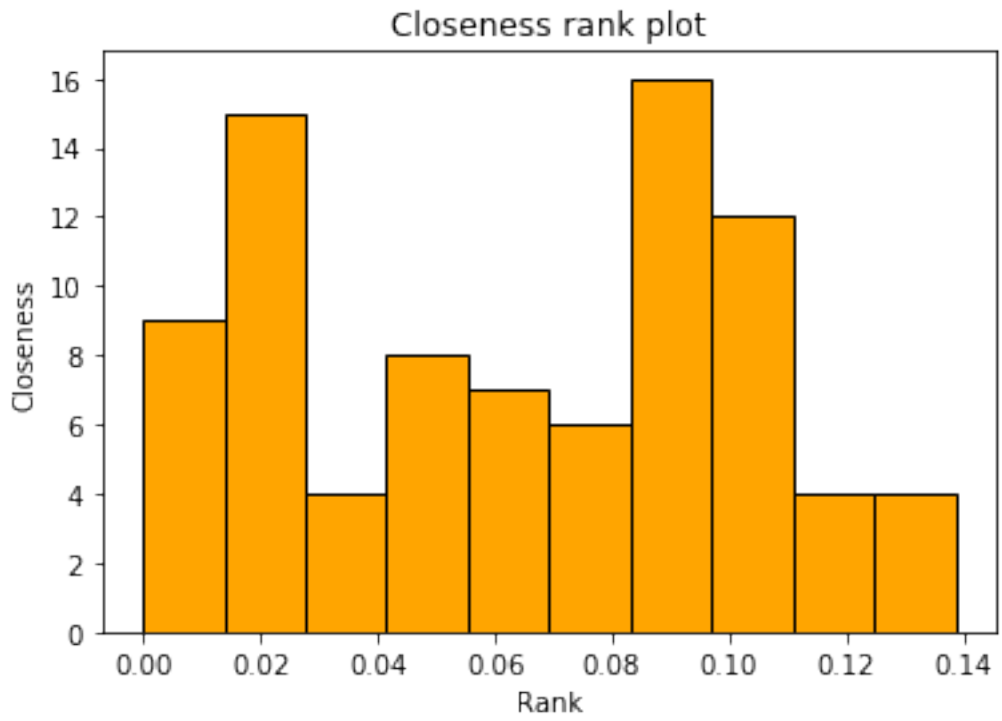



**2.0.2 As we can see from the log-log plot, the behaviour which our nodes have belongs to a specific type of distribution called *power law*.**

## 3 Closeness Centrality:

**3.0.1 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.**

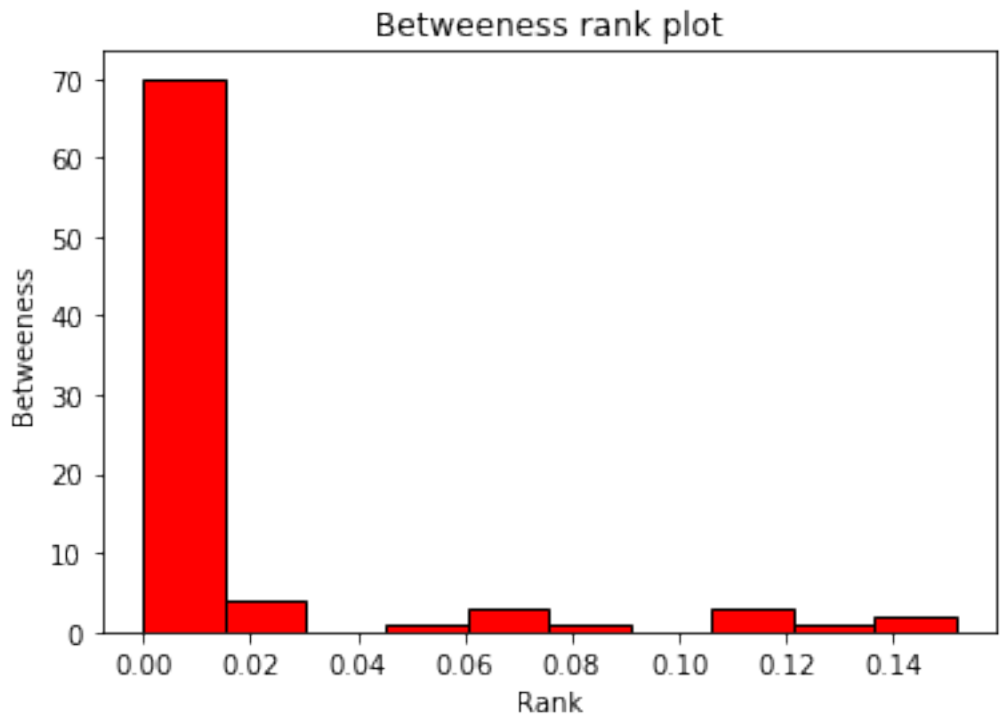
```
In [9]: Modules.closeness(subgraph)
```




# 4 Betweeness Centrality:

#### 4.0.1 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 [10]: Modules.betweeness(subgraph)
```

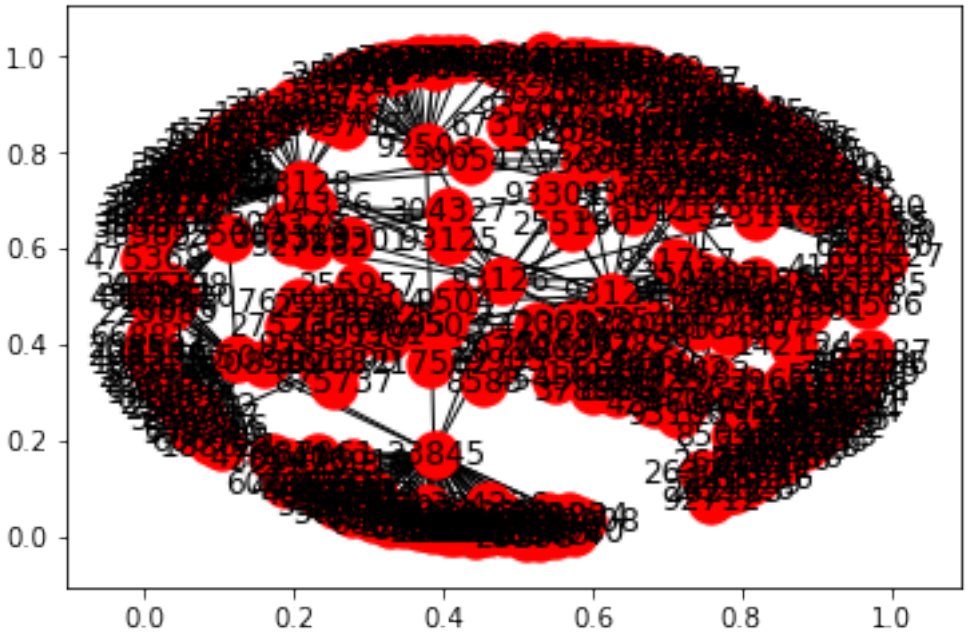



#### 4.0.2 After statistics visualization, we created the subgraph induced by the nodes that have *hop distance* at most equal to an integer *d* with an input author.

#### 4.0.3 The result is shown below:

```
In [11]: Modules.hop_distance(Gall)

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




#### 4.0.4 In the exercise 3.a we had to implement the Dijkstra algorithm which calculate a generalized version of the Erdos number.

#### 4.0.5 In this case Erdos is Aris and our output will be the distance between an input author and Aris.

```
In [12]: Modules.dijkstra(Gall,256176,256177)
```

```
Out[12]: 0.9565217391304348
```