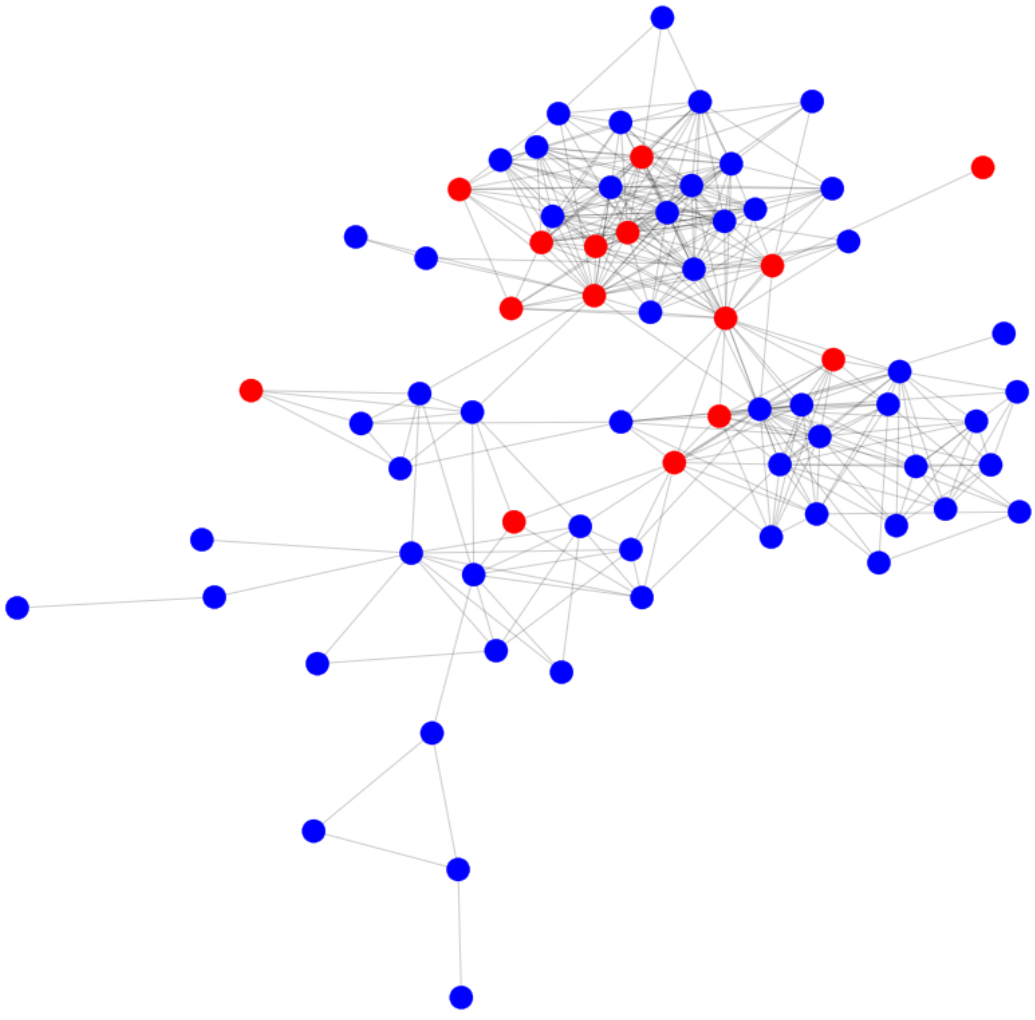
Exporting graph using Vk api

Out[113]:

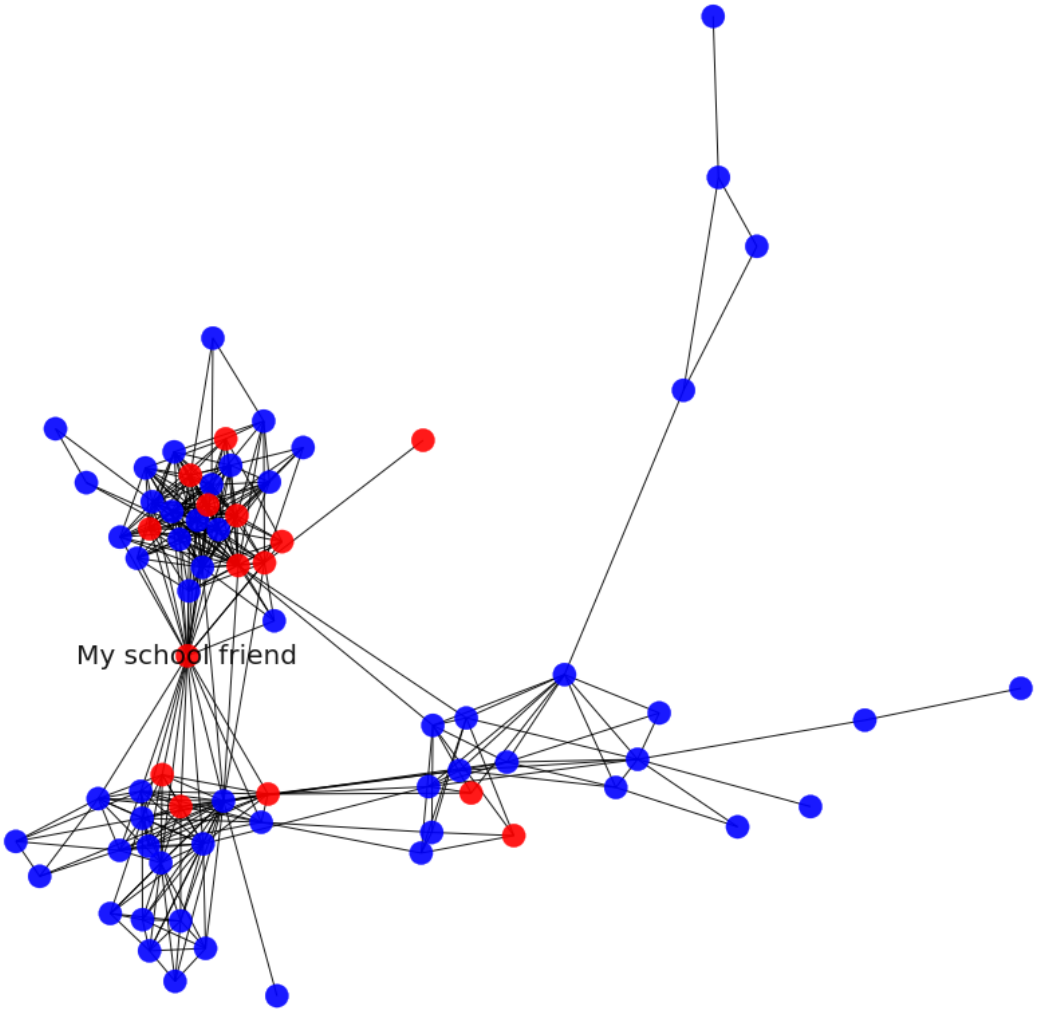
The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

As a graph, I used my own account in the social network vkontakte. To export the data, I created the VKontakte application, as well as several tokens, which allow making calls to Api. Also, to increase productivity, I took advantage of the advanced features of the VKontakte and created on their platform a special script that allows using 25 calls to api with 1 request. Code for exporting SN graph can be found in jupyter-notebook, but it needs tokens to work, so I pickled already downloaded one.

Below you can see the resulting graph, where red verticles are women and blue are men. You can see quite clearly that the graph has a clear structure and has 3 clusters, which I can interpret as - my school friends, my friends from the university and my friends from the sports class.



Using another layout function, we can get a clearer presentation of clustering. This graph clearly shows that I have a friend who knows quite a lot of people both from my university and from school - this is the red vertex that connects 2 large clusters.

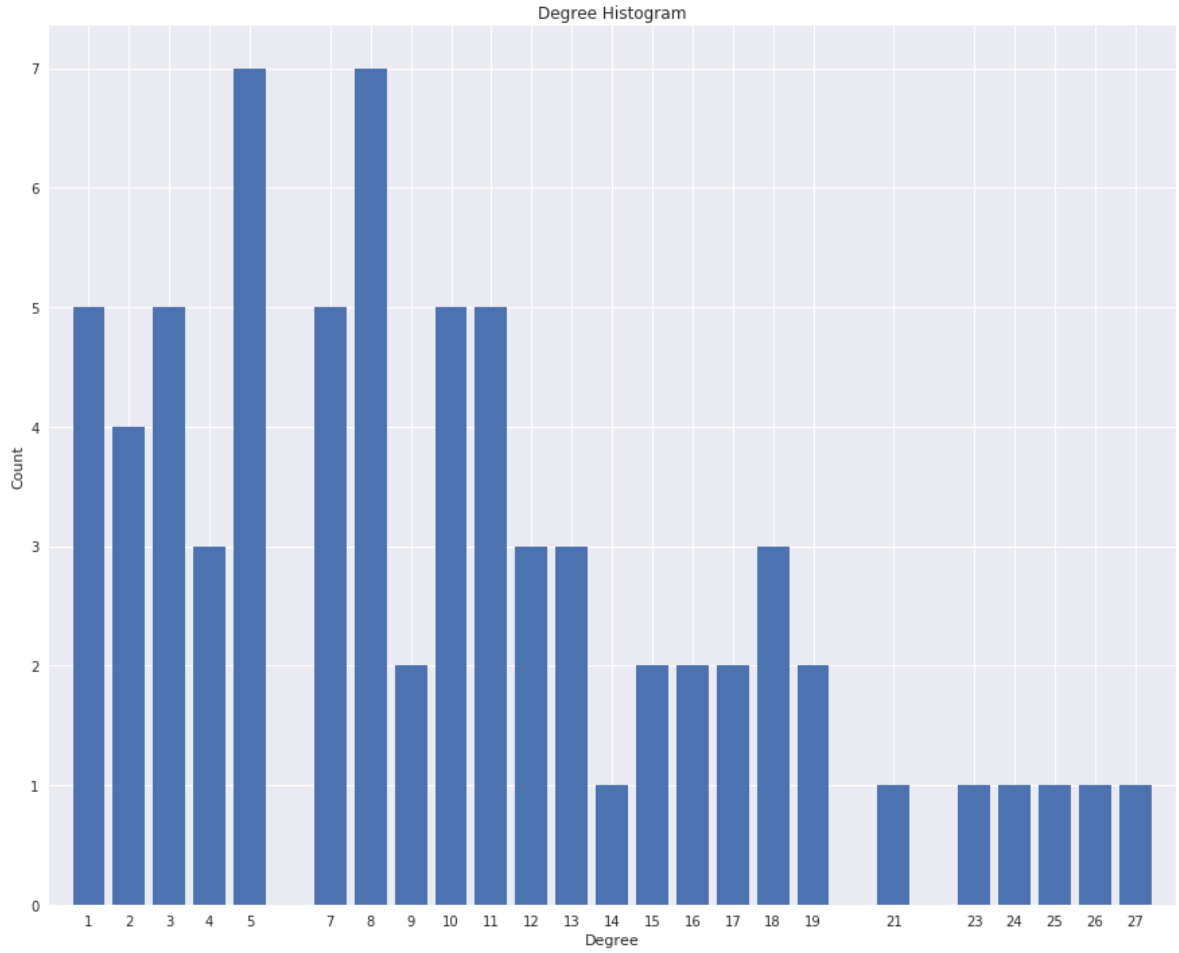


Graph has 72 verticles and 356 edges.

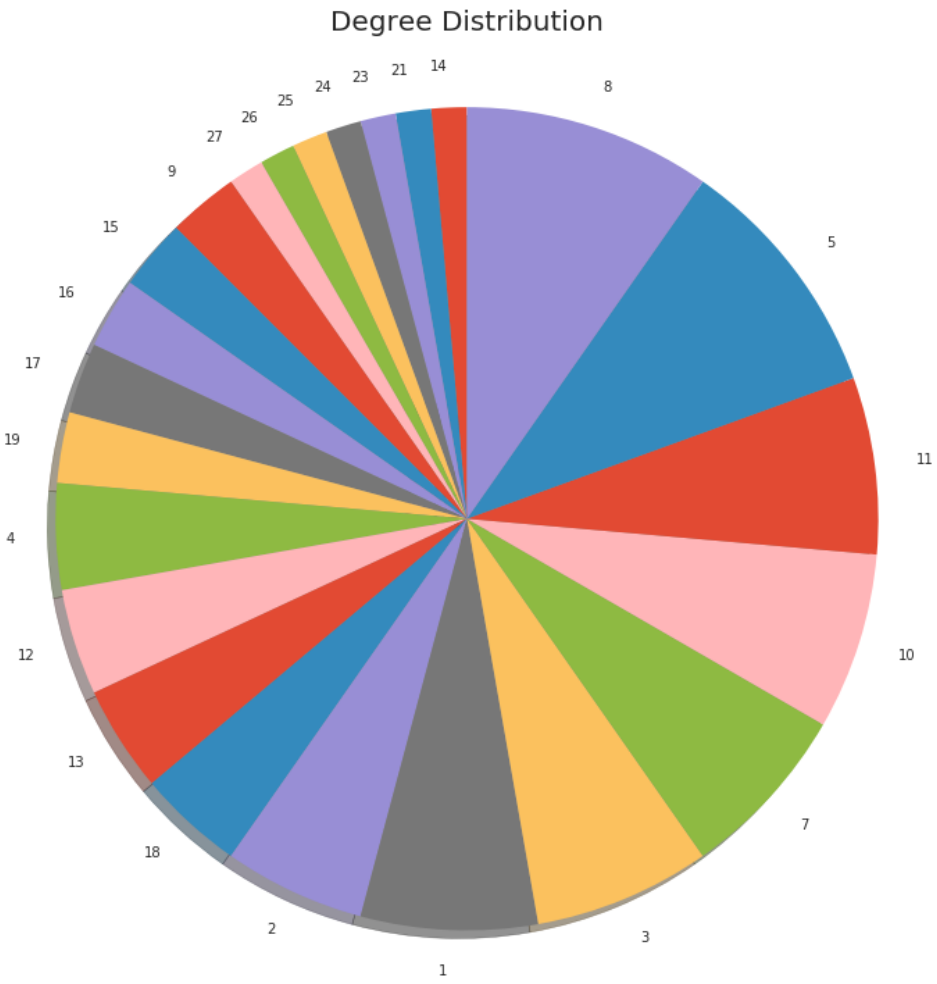
Analisys

Degree distribution

As can be seen from the graph, the vertices with the number of edges equal to 5 and 8 are the most.

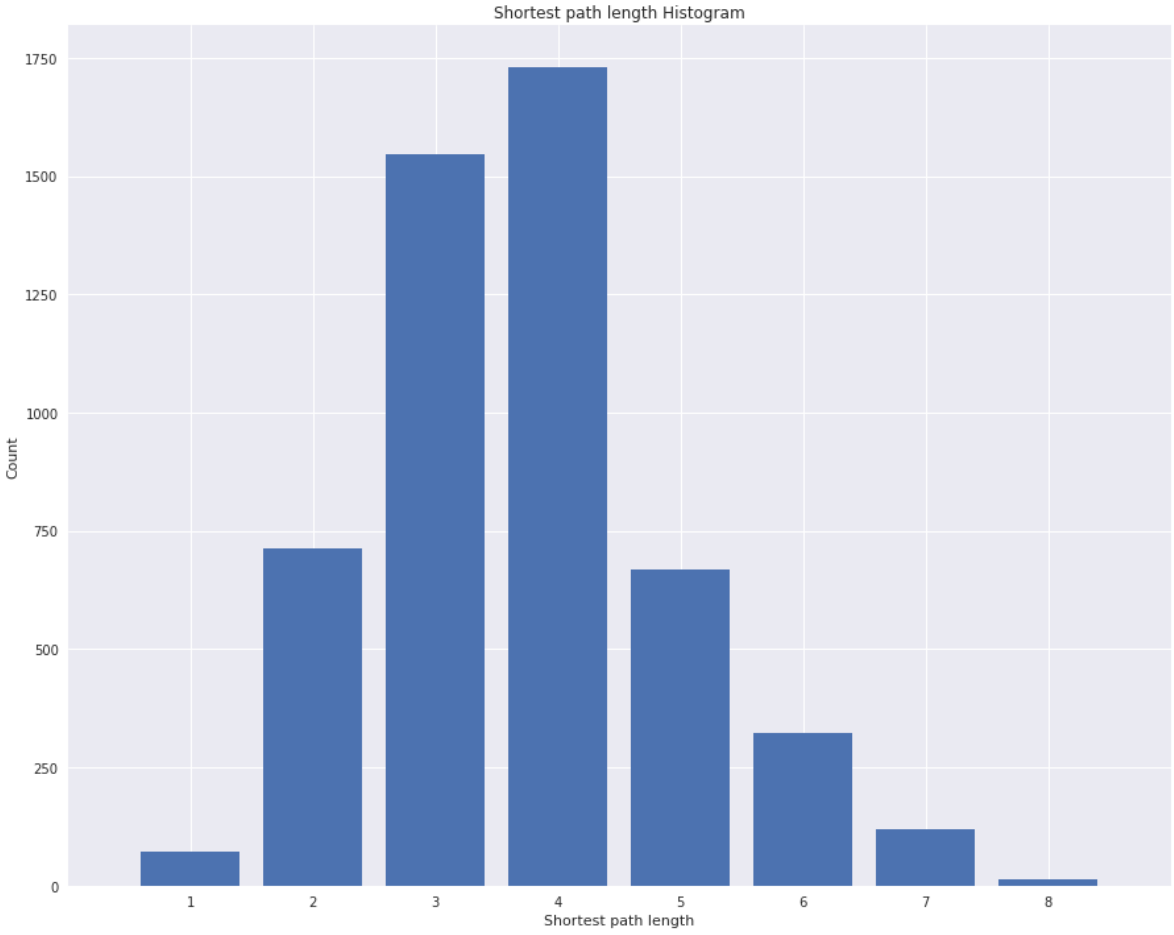


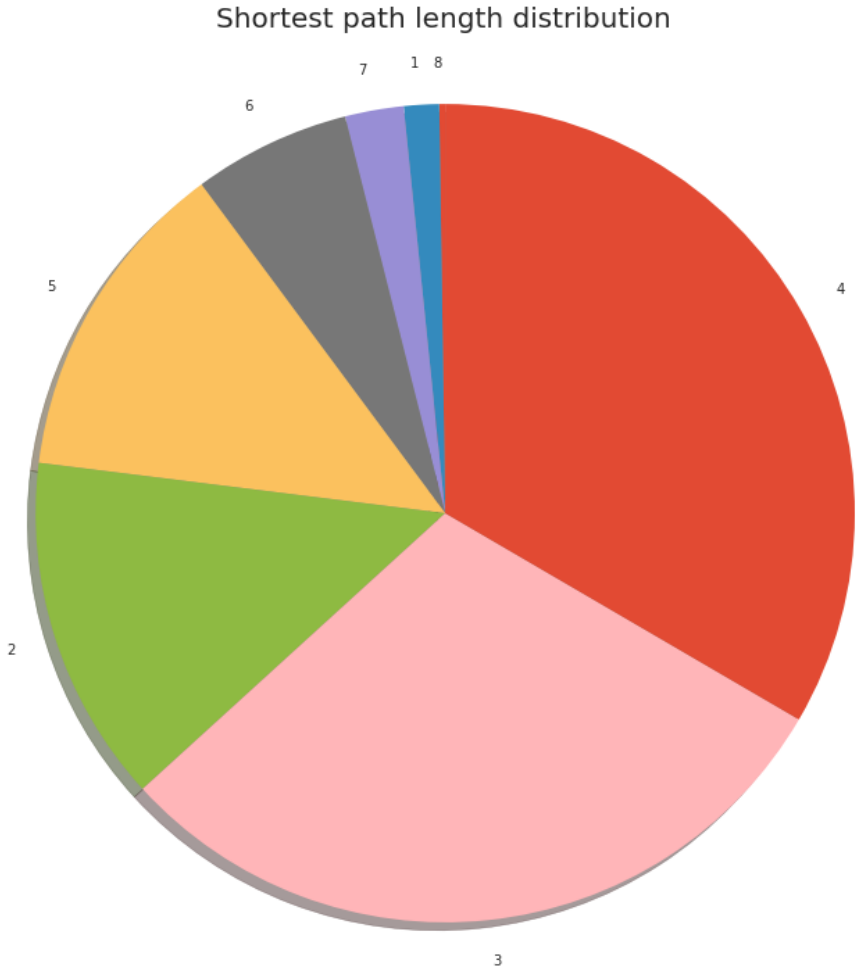
And pie-chart:



Shortest Path distribution

The most common path length in the graph is 4, which means that most of my friends know each other through a chain of 2 people.

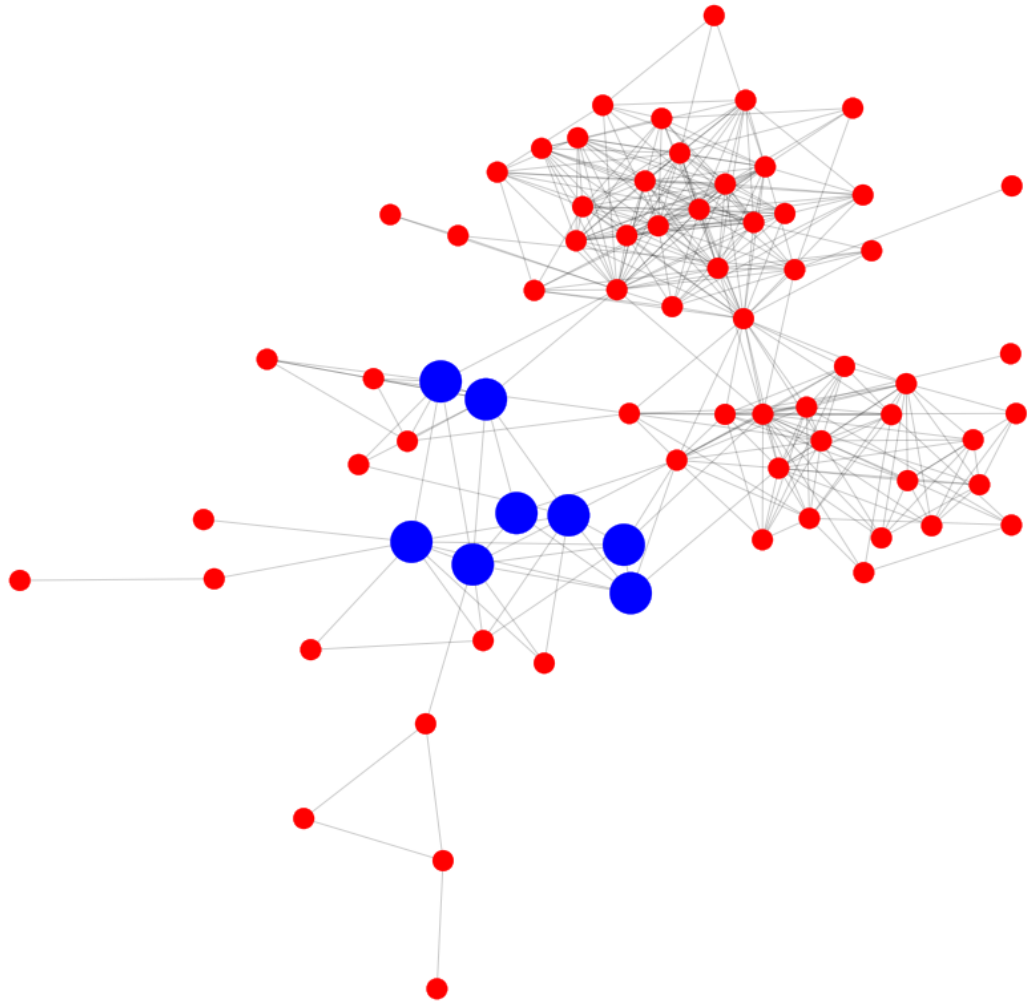

Shortest path length Histogram
Count
Shortest path length
0
250
500
750
1000
1250
1500
1750
1
2
3
4
5
6
7
8


Shortest path length distribution
1
2
3
4
5
6
7
8

Diameter, Radius, Center

```
Diameter: 7
Center: [196301700, 341452870, 374450143, 16035041, 228163684, 2782678
78, 67849056, 273877653]
Radius: 4
```

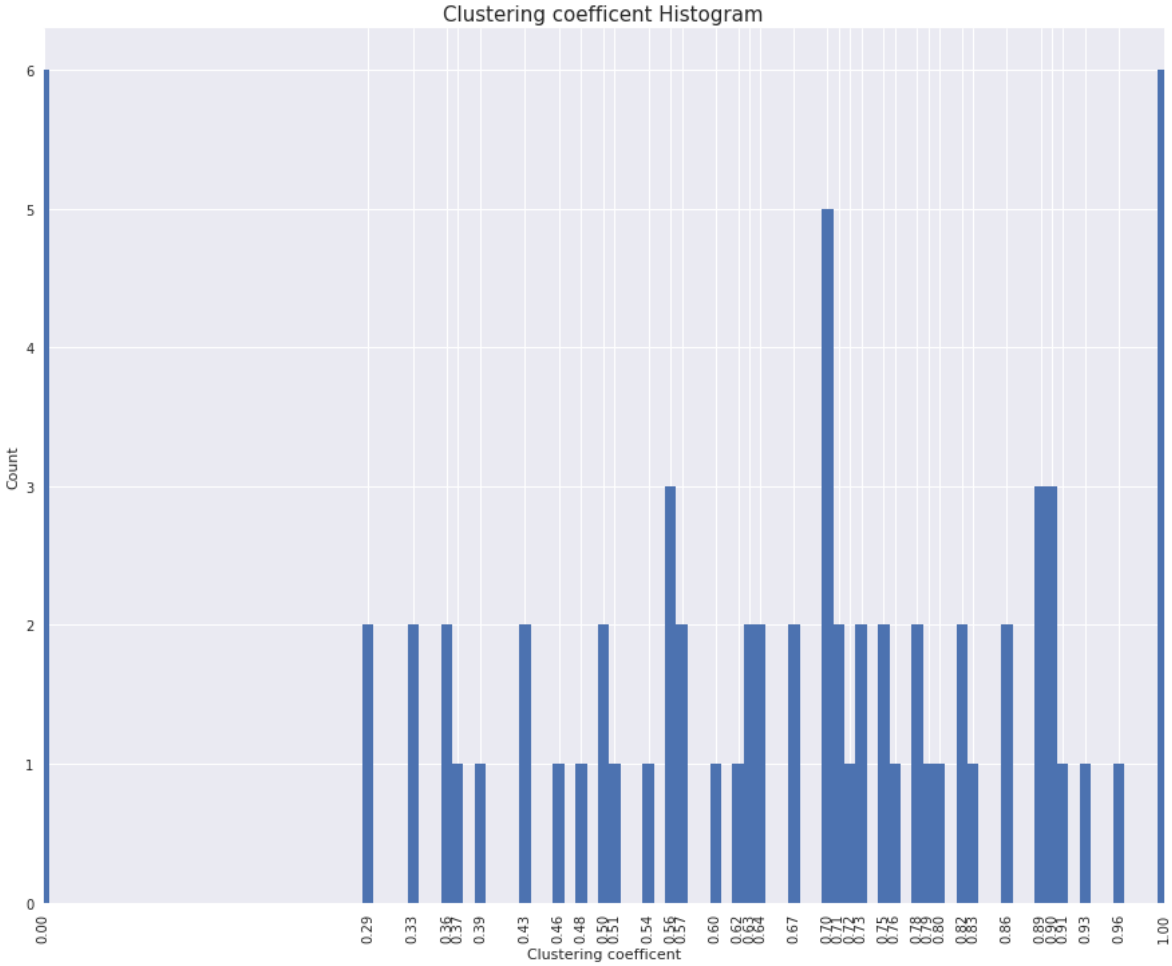
Big blue verticles are central.



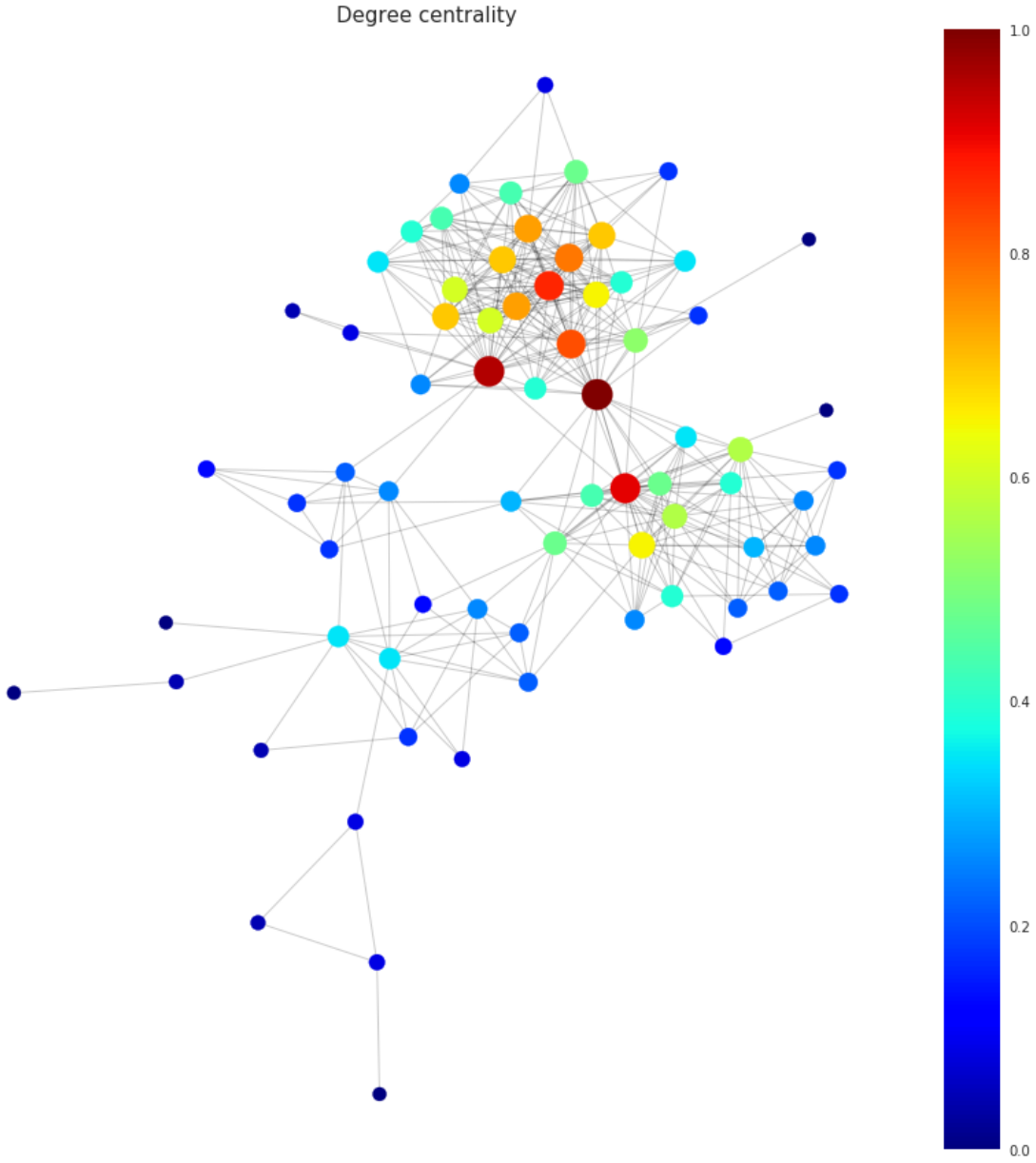
Clustering Coefficients

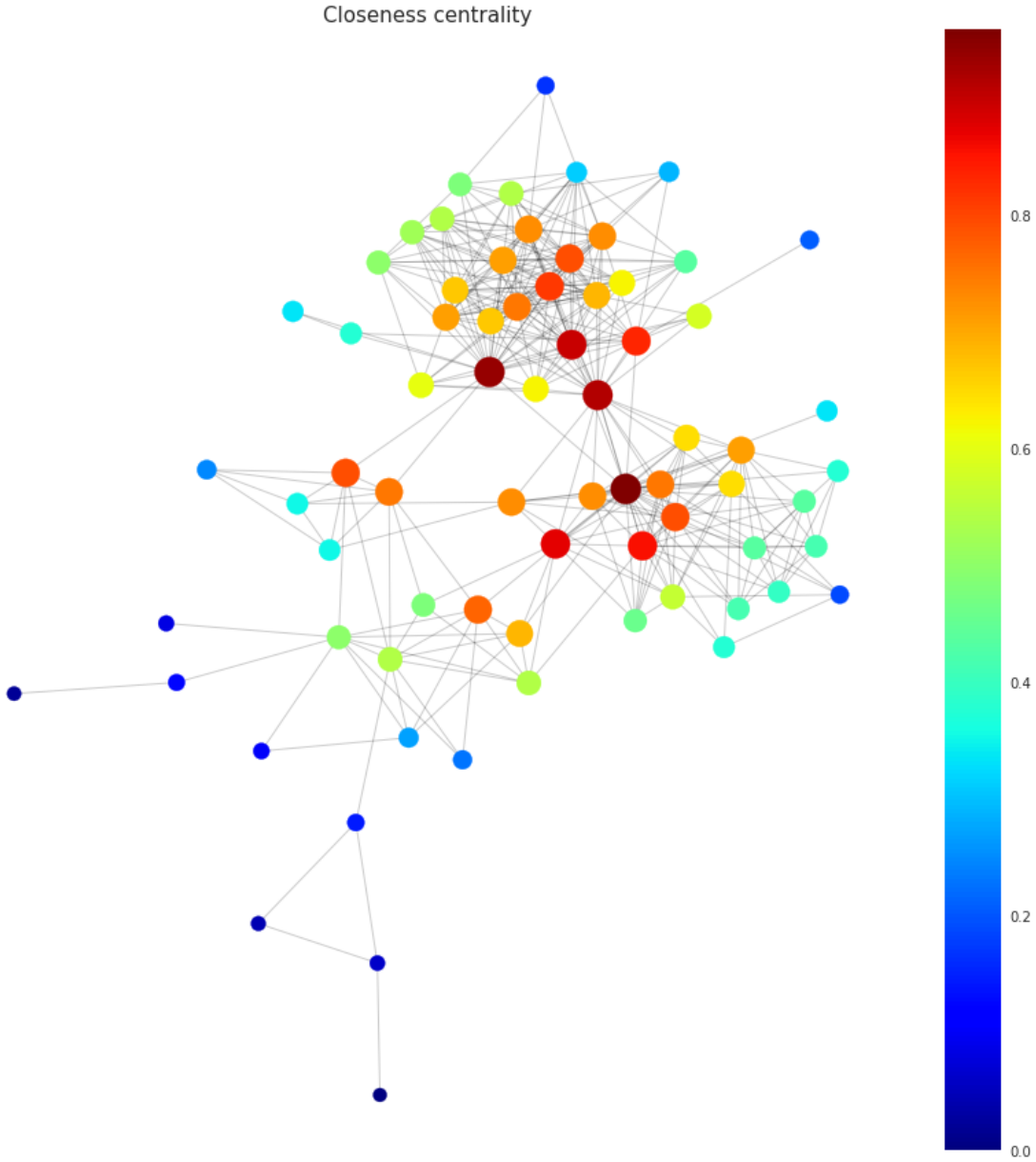
As it is observed, graph has pretty big clustering coefficent, which means almost everyone knows each other.

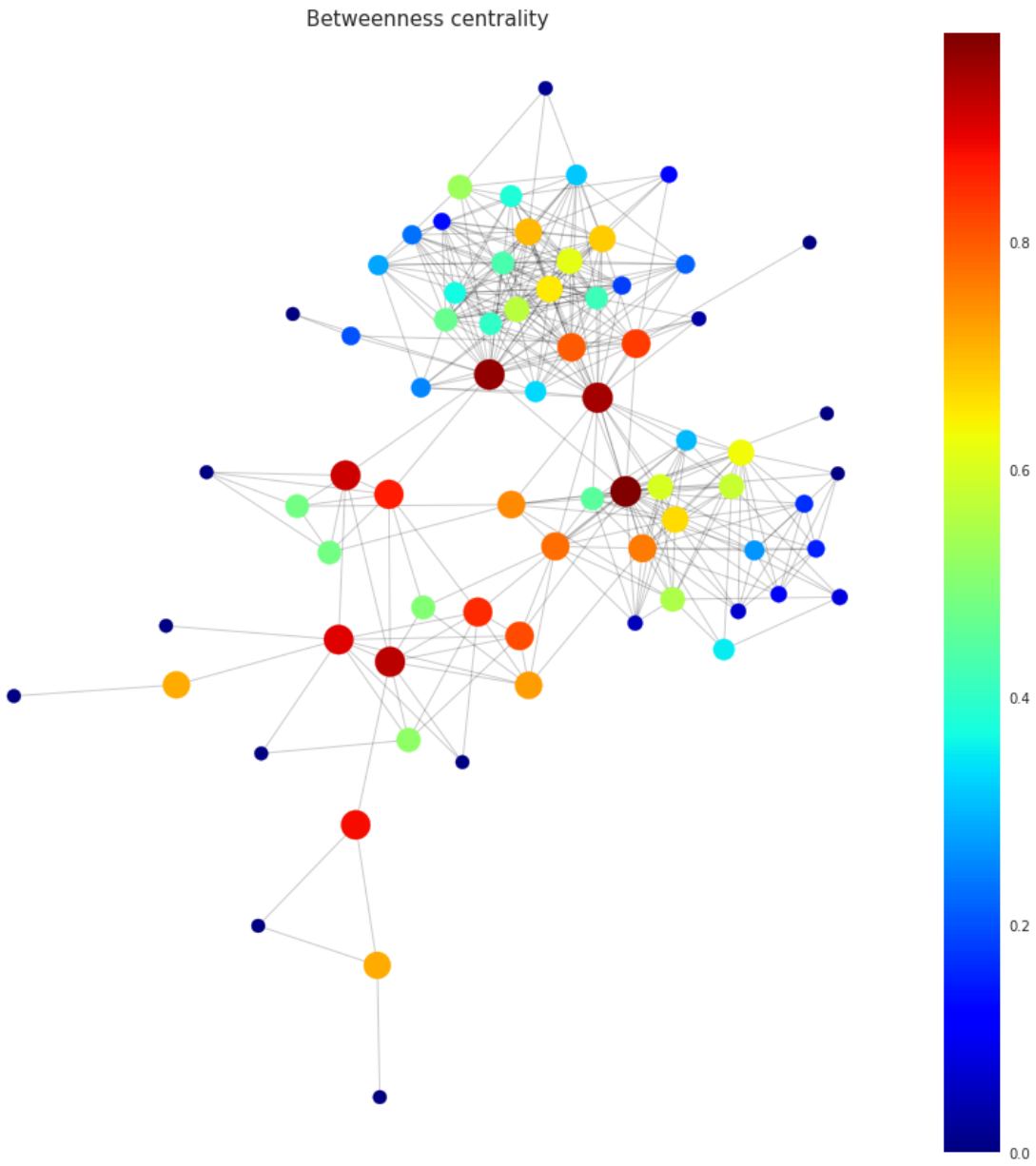
```
Clustering coefficent: 0.6303389756818139
```



Degree/Closeness/Betweenness centralities


Degree centrality
1.0
0.8
0.6
0.4
0.2
0.0


Closeness centrality
0.8
0.6
0.4
0.2
0.0



Top nodes interpretation

```
Top by closeness:
Плотников Георгий
Павлов Сергей
Чижова Даша

Top by betweenness:
Плотников Георгий
Павлов Сергей
Чижова Даша

Top by degree:
Чижова Даша
Павлов Сергей
Плотников Георгий
```

For all 3x metrics, 3 vertices with the highest value are clearly visible: 1 the vertex belongs to the cluster 'school', 1 to the cluster 'university' and 1 between them (I wrote about it earlier). This, of course, is my friend from school, my friend from University and a school friend who came with me to the university.

Page-Rank. Comparison with centralities

Again, there are the same persons.

```
Page Rank:
Плотников Георгий 0.03426548281742275
Павлов Сергей 0.030152489726071815
Чижова Даша 0.029453479498658883
```

Assortative Mixing according to node attributes

The coefficient is rather low, which means that people are connected regardless of sex.

```
Out[142]:
```

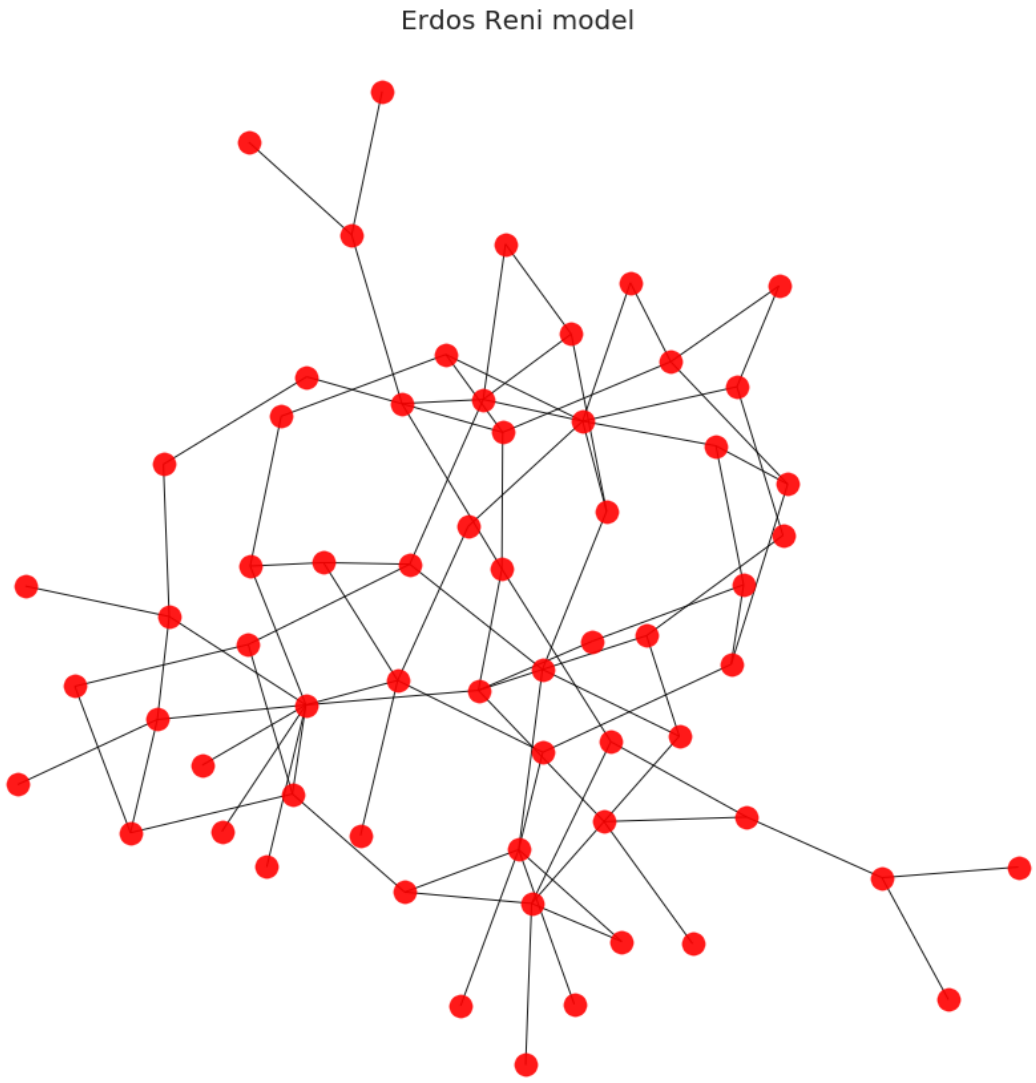
```
0.06723635208868595
```

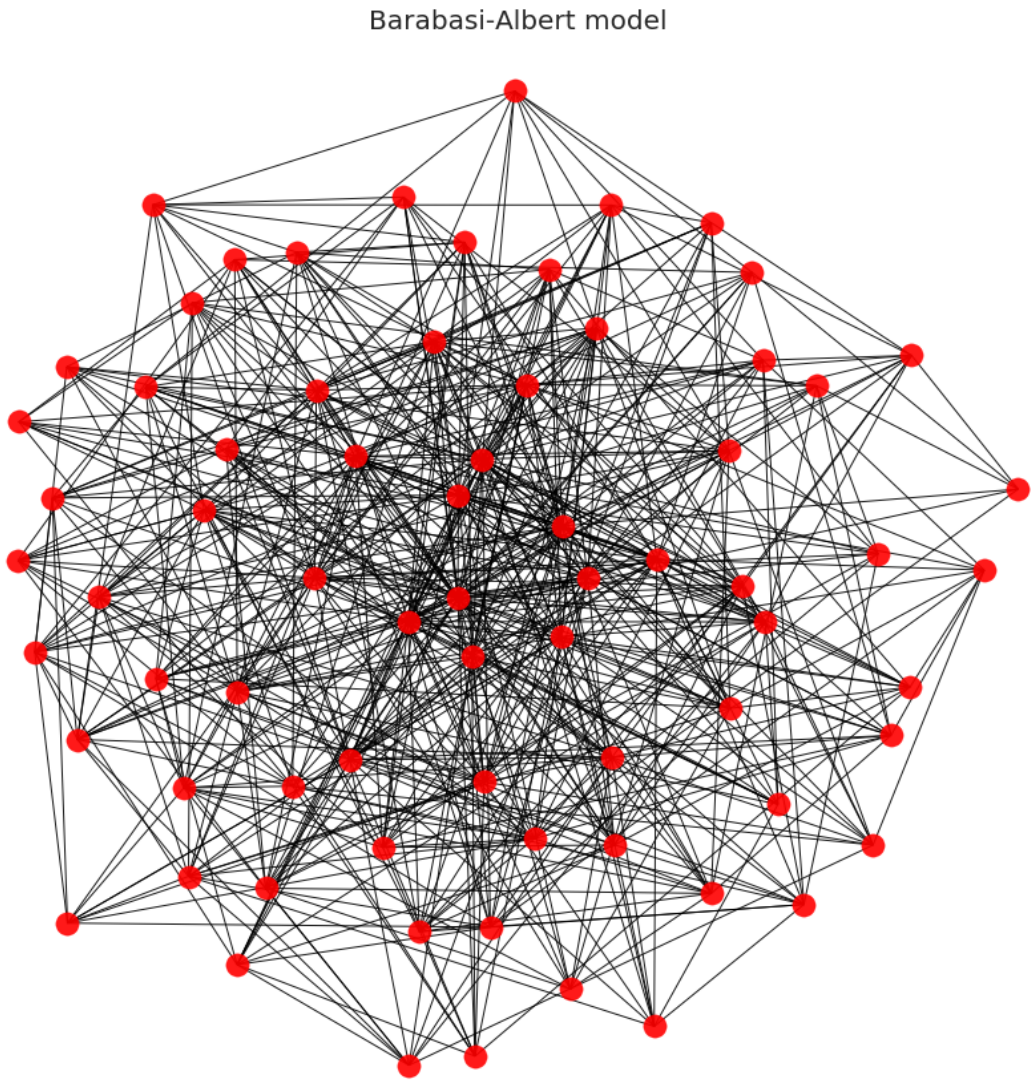
Node structural equivalence/similarity

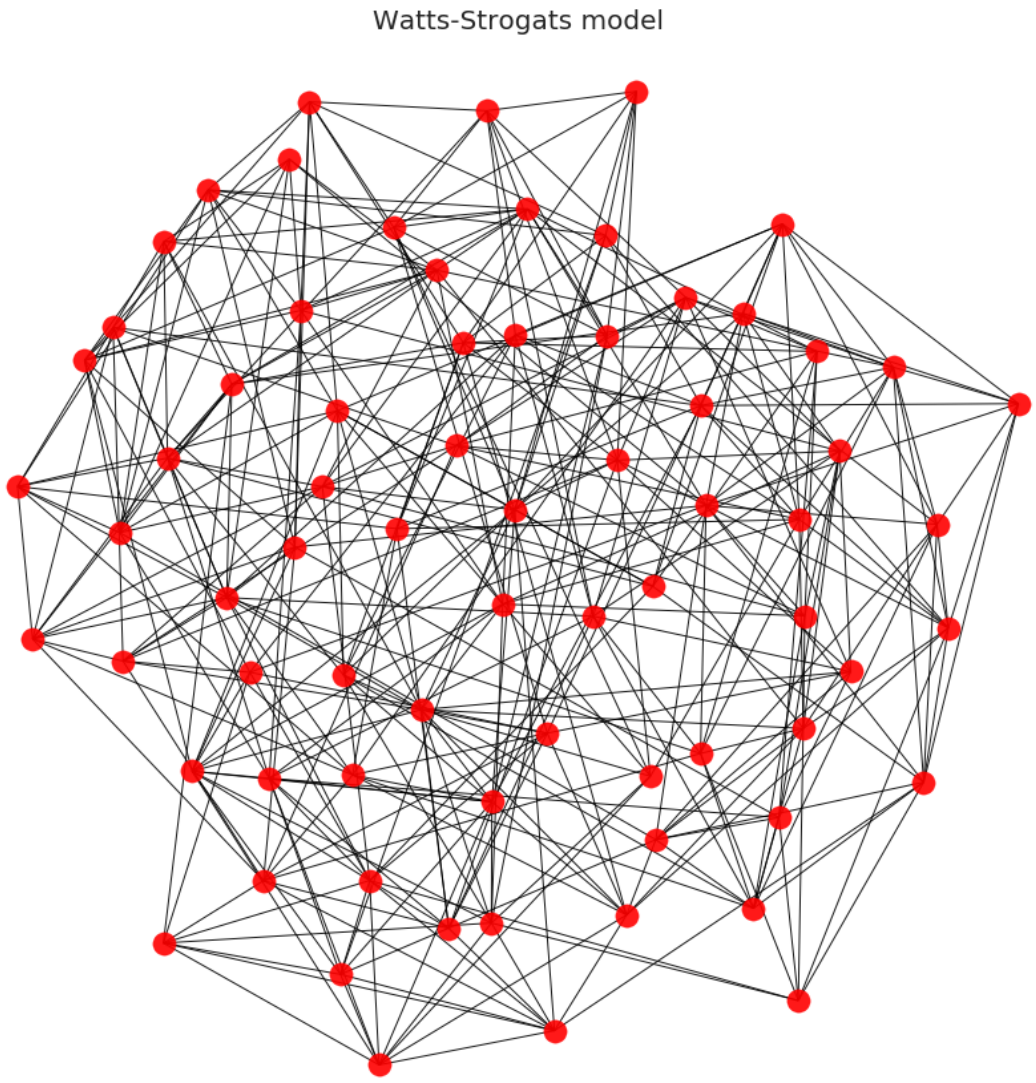
In my graph there are no verticles that belongs to same equivalent class. This is logical, because if people have a common circle of communication, most likely they are also friends or at least know each other.

The closest random graph model similar to your SN

I created several graphs using Erdos-Reni, Barabasi-Albert and Watts-Strogats random graph models and metrics from my own graph.


Erdos Reni model


Barabasi-Albert model



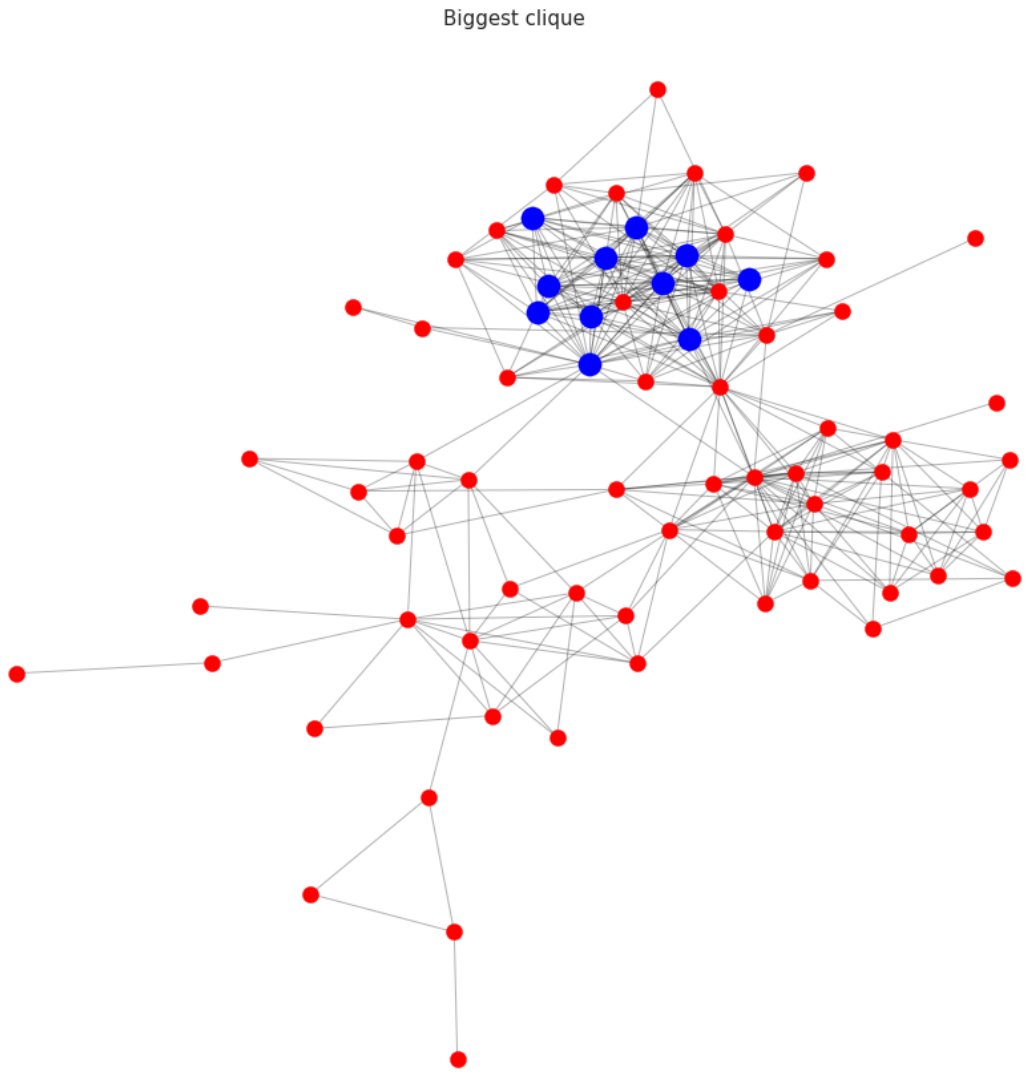
Out[283]:

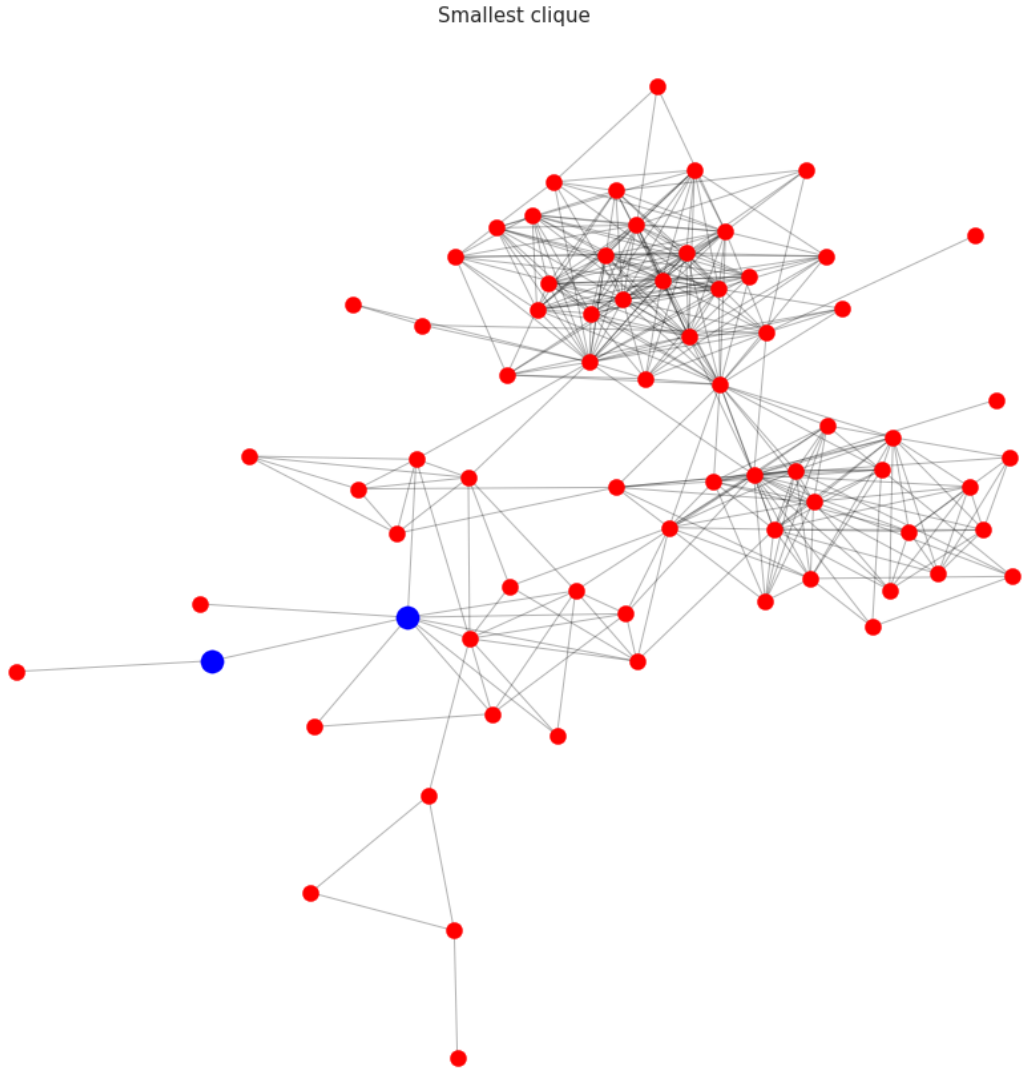
	Clustering coefficent	Average path length	Diameter	Average degree	Edges num
Original	0.630339	2.755477	7	9.888889	356
Barabasi-Albert	0.342684	1.770736	3	17.222222	620
Erdos-Reni	0.028943	3.938657	8	2.472222	89
Watts-Strogats	0.244006	2.129499	3	10.000000	360

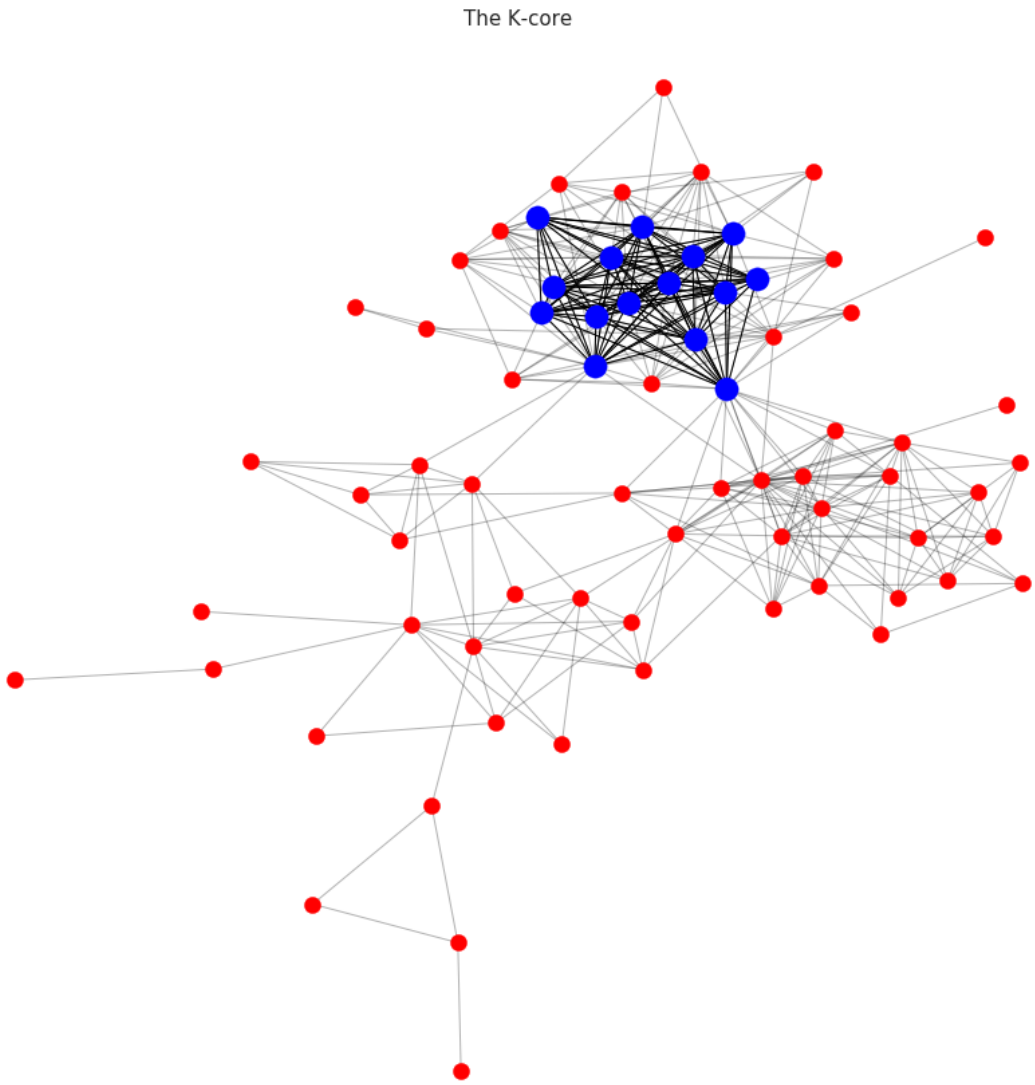
In this table, you can see quite clearly that the metrics of my graph are most similar to that of a graph created using a Watts-Strogats model.

Community Detection

Cliques, Hierarchical k-cores, k-plexes

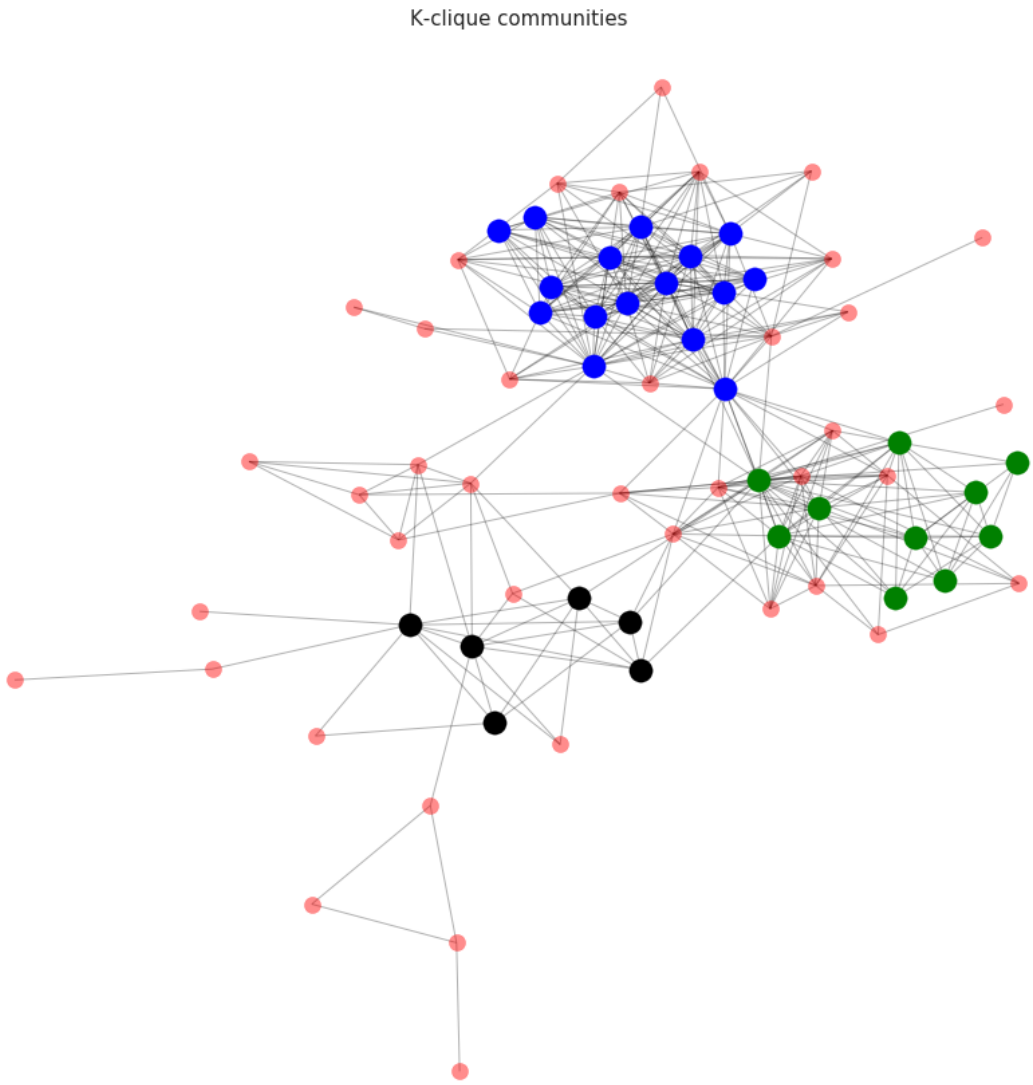

Biggest clique


Smallest clique

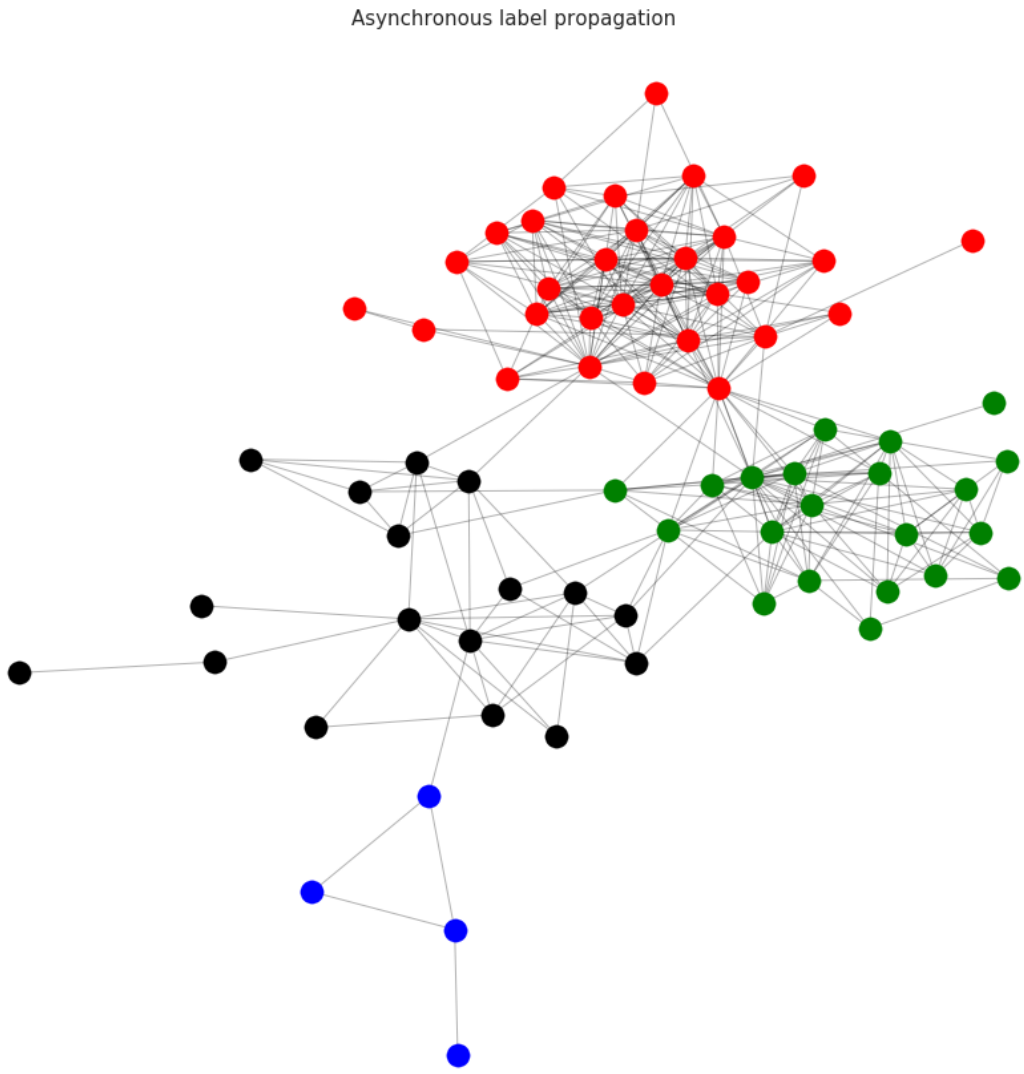


Community detection algorithms

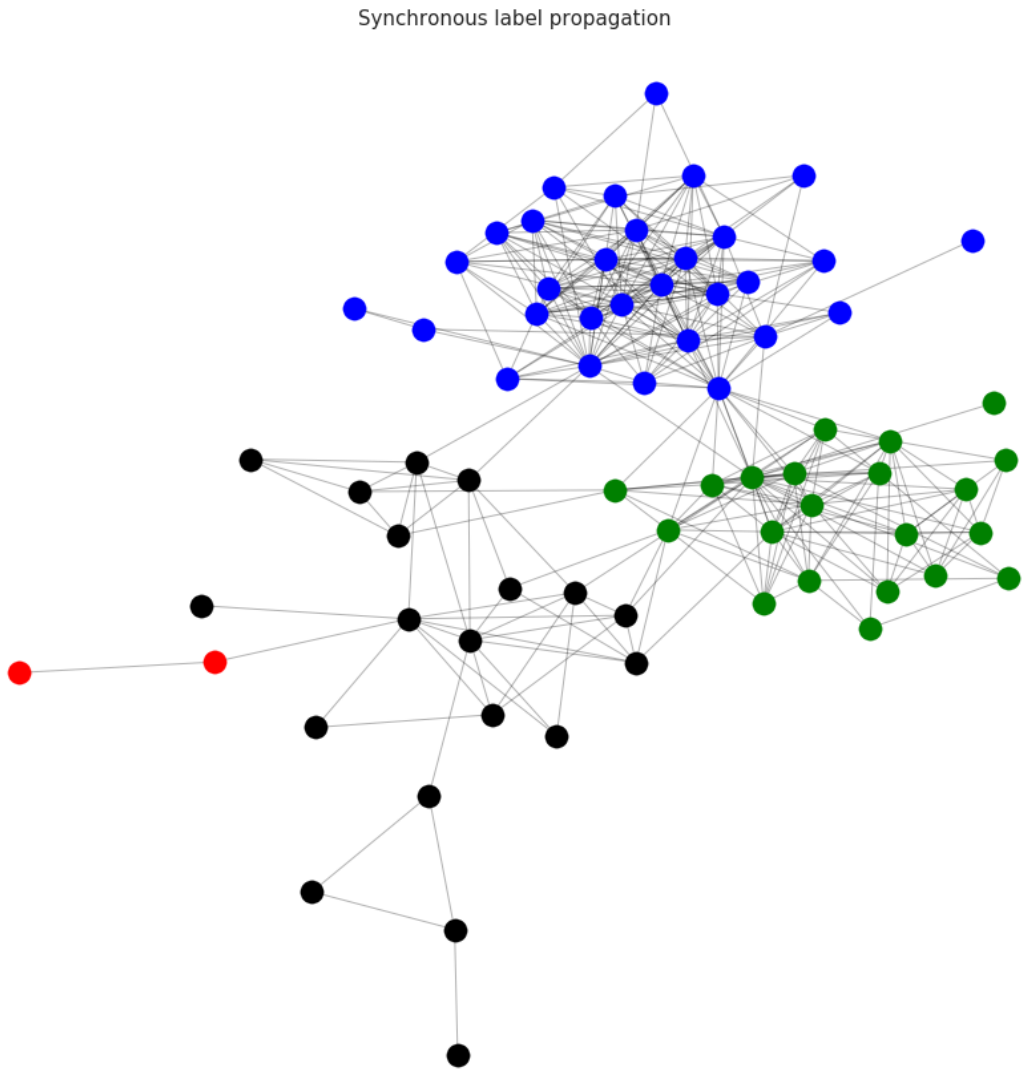
The algorithm showed a fast work speed, but the results are not very impressive, since the communities found in most cases were only small parts of the real ones.



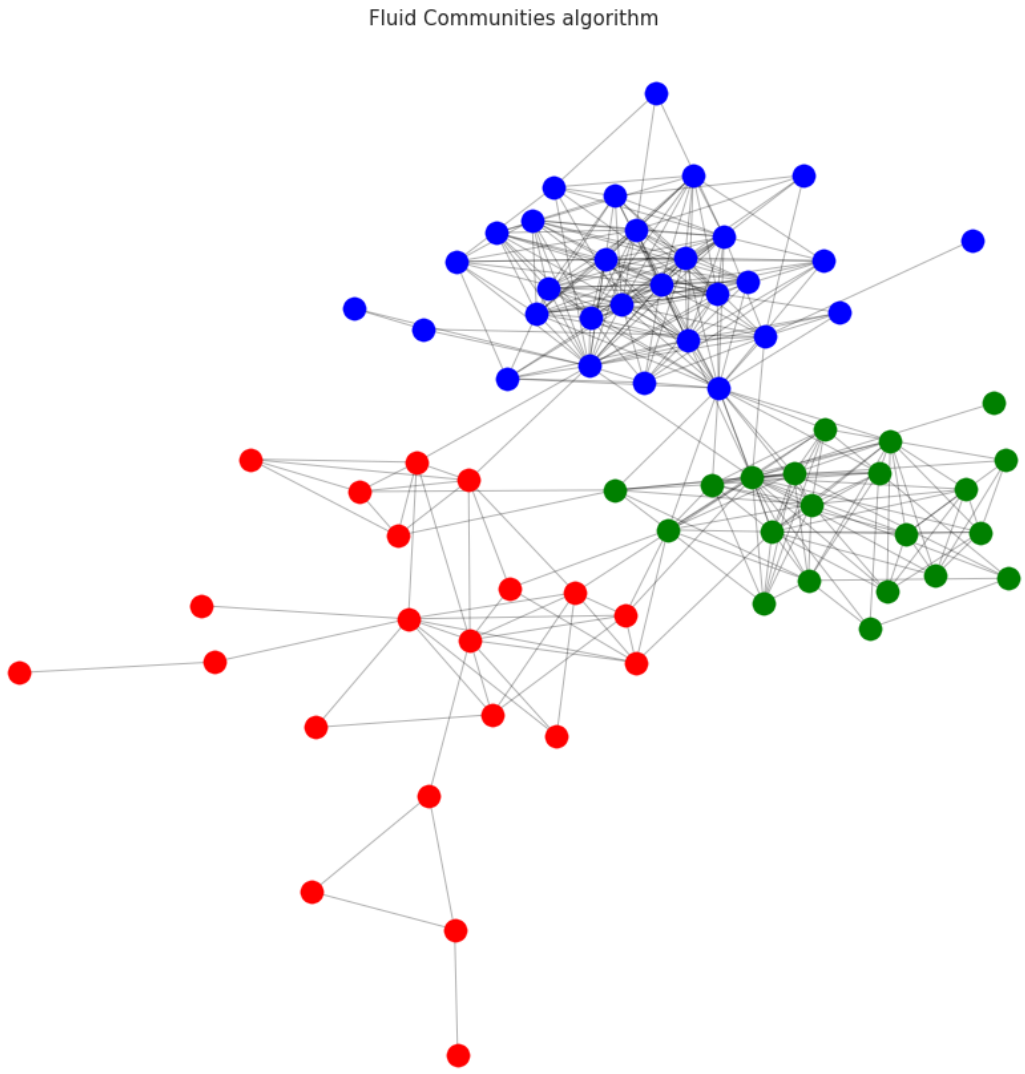
The algorithm is one of the slowest, but it can be called one of the best. The probability of giving an incorrect result is practically equal to zero and even if there is an error, it is insignificant.



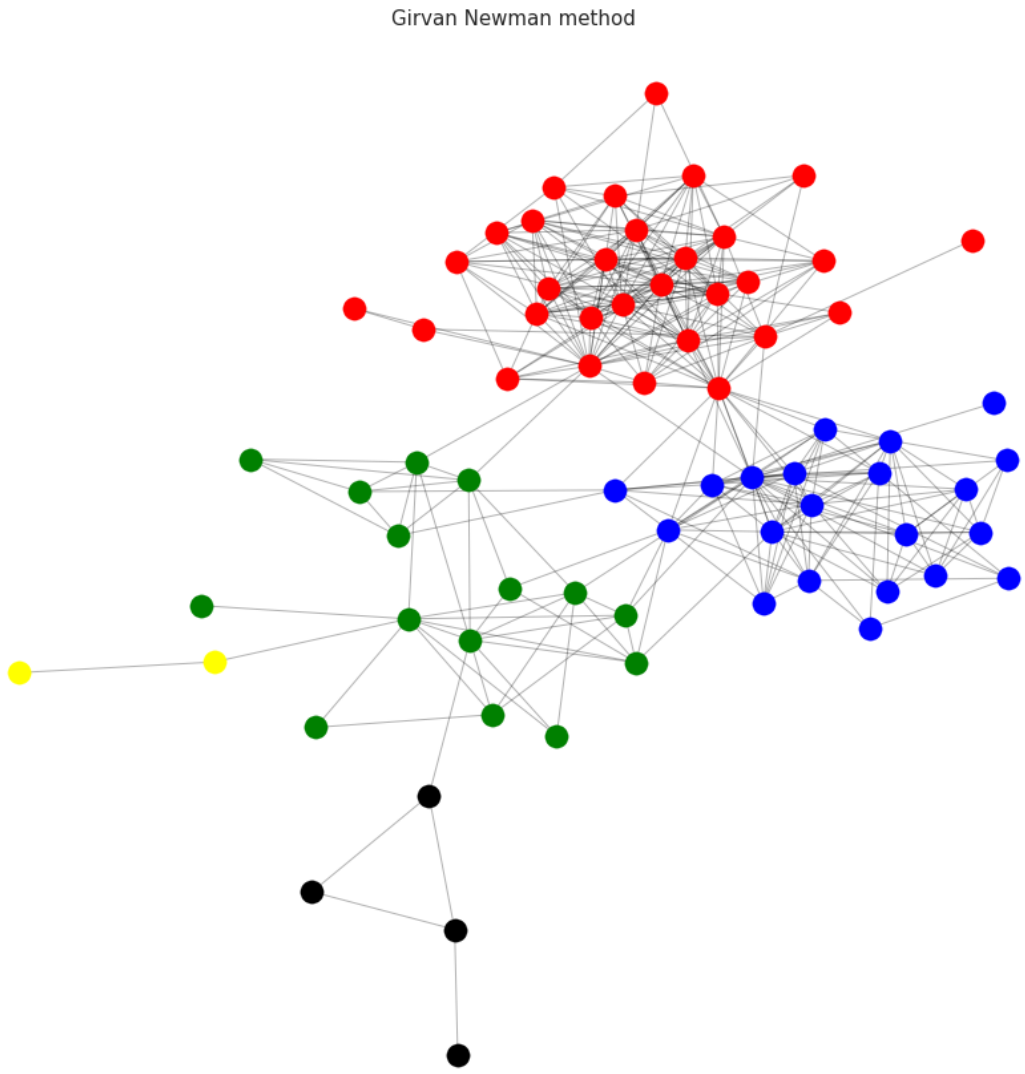
Differs from the previous work speed, exceeding the previous analog at times.



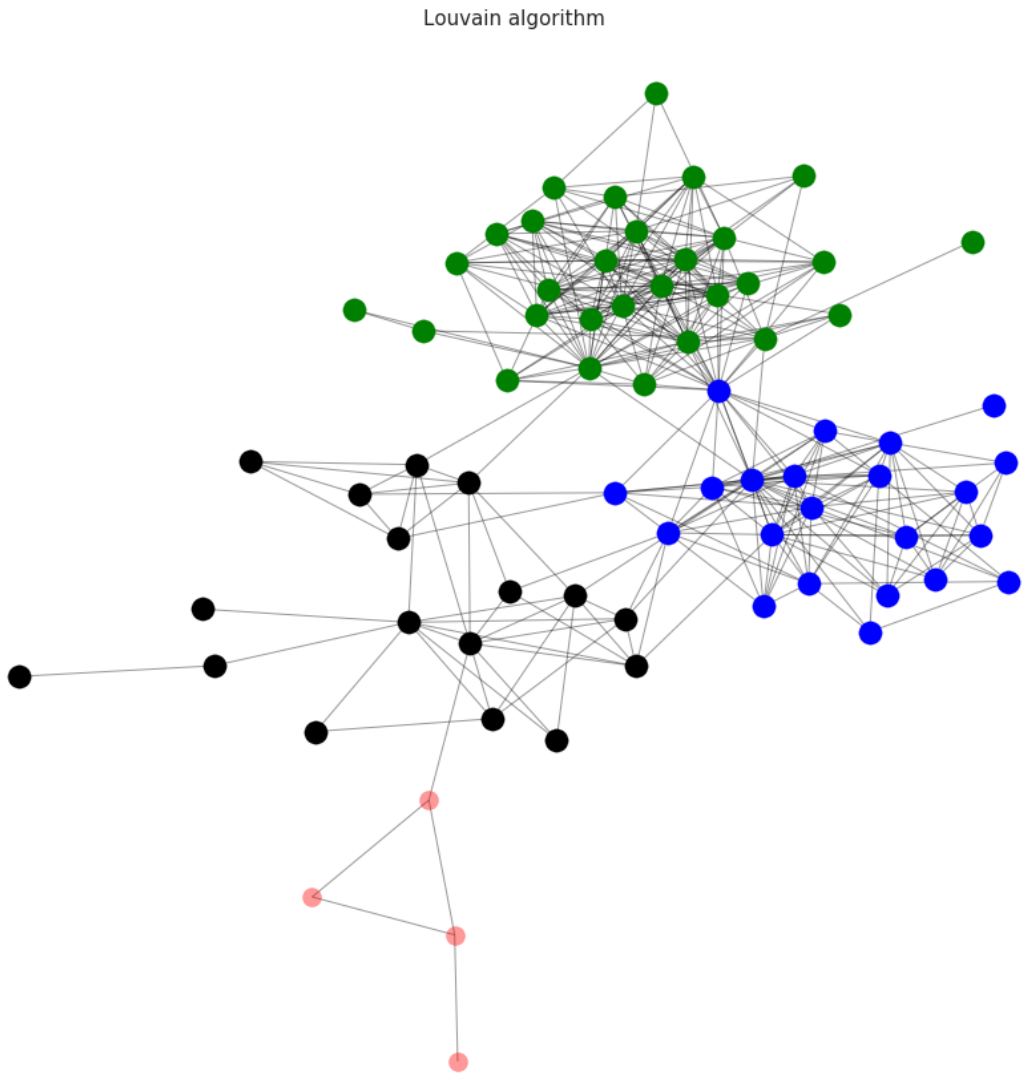
Fast, however, an extremely unreliable algorithm that gives an error in half the cases - can even divide a clear cluster into 2 parts.



It produces good results, but it works very long.



One of the best algorithms, combining high speed and quality.


Louvain algorithm