Sarantopoulos Ilias

# Graph Algorithms on Spark GraphX and Apache Giraph

Status report 11/5

I have set up Spark and GraphX on a single node cluster for developing purposes. I have studied the Spark and GraphX papers to familiarize myself with spark's RDD operations and graph processing.

In order to do that, I took a brief introduction into functional programming concepts using Scala programming language and after that I studied the Spark and GraphX programming guides.

Next, I started studying Centrality measures from this Wikipedia article, various online sources as well as the MOOC available from Coursera “Graph Analytics for Big Data” which copes with relative issues.

I ran PageRank on an extracted dataset from my twitter network using the algorithm implemented in the GraphX library and as shown in the programming guide.

So far I have implemented the algorithms for Degree and Closeness Centrality.

First I started with **Degree Centrality** calculating each vertex degree using a normalization factor. So for each vertex the degree centrality measure equals the vertex degree  divided by the total number of possible edges it could have if it was/is connected to all other vertices in the graph (V-1 possible edges).

$$d(u)=\frac{degree(u)}{N-1}$$

Running this algorithm on the graphx test data(followers.txt , users.txt) gives us the following output:

```
User with name: Martin Odersky has a degree centrality of 0.8
User with name: Barack Obama has a degree centrality of 0.6
User with name: Matei Zaharia has a degree centrality of 0.6
User with name: John Resig has a degree centrality of 0.6
User with name: Goddess of Love has a degree centrality of 0.4
User with name: Justin Bieber has a degree centrality of 0.2
```

For **Closeness Centrality** I used the following formula :

$$c(u)=\frac{1}{\sum_{v}^{N} dist(v,u)}$$

where $\sum_{v}^{N} dist(v,u)$ is the sum of all the shortest paths from vertex u to all other vertices. In order to calculate the shortest paths I used the *ShortestPaths* algorithm available from the GraphX lib which returns a new Graph on which the VertexRDD has a HashMap as an attribute for each vertex which contains the shortest distance from each other vertex to the vertex u.

Both Closeness and Betweeness centrality measures require calculating all-pairs shortest paths, which is computationally expensive, so I do not have results for a large dataset yet.