Quantitative exercise

2018.09.17

Mohamed Gabr (mohgab@kth.se)
Óttar Guðmundsson (ottarg@kth.se)

Disclaimer: Figures and tables are unnamed. They will be titled appropriately in the final report if used.

Edit from group review: Maybe have a small intro about the project or possibly explain what the data represents.

The world is producing data more than ever before. Analyzing this massive amount of information interests the academic and the corporate field. Such material can provide valuable insights for companies and useful results for researchers. Therefore, developing efficient tools to perform these tasks is paramount in the field of Big Data. Information comes in different forms and structures; graphs are one of them. It is a data structure composed of nodes that are connected between each other by edges. Such a collection can describe connections in a social network for instance.

Edit from group review: What do these libraries do? How do they perform? How is that measured?

Several frameworks have been released to perform graph processing like GraphX in Spark and Gelly in Flink. Nevertheless, it is challenging to choose the appropriate one for a particular task; the community lacks comparisons between these tools. Our project aims to benchmark GraphX and Gelly regarding runtime, CPU and memory usage on diverse tasks with varying complexities. It intends to help future users picking the tool that serves their problem the best and hence saving their time.

Datasets adopted in this project consist of connected, undirected graphs from a public repository. They describe different types of information; one depicts connections between a group of jazz musicians, another pictures mail interchanges among members of a University.

Edit from group review: Shouldn't the data analysis include execution time between the two libraries.
This paper presents the checks and analysis done on the data that the frameworks will process their algorithms on.

For this exercise, we chose to use python, networkx[1], and Graph-tool[2] for analyzing the data. The preprocessing was done by opening each file and trimming empty spaces of every line. Space between values was trimmed to a single space to easily open and import into a graph file structure. Thus, data was changed from the format

```
 1   3    1
3   2    1
1   2   1
2     3 1
```

To

```
1 3 1
3 2 1
1 2 1
2 3 1
```

Note that all nodes have an extra 1 at the end. This the weight of the edge if the network was weighted. One can argue that this extra number can be removed from the research, but we chose to keep this to extend upon further research in the future so the weights can easily be imported using the same data pipeline.

This data was loaded inside a jupyter notebook and constructed into a new graph, where the imported tools were then used on the graphs to further analyze them.

Expected property: There does not exist a vertex that is unconnected.
Description
All of these graphs of this research are undirected and fully connected, making all nodes connected by at least one edge from any direction. That hints that the expected property of the datasets concludes that there is no edge unconnected, so the minimum degree of the graph should not be lower than 1.
Condition met. Our data has the expected property, this can be seen by the Min degree variable for every dataset (pages 3,4 and 5).

Expected property: The adjacent connectivity of the matrix should be symmetric.
Description
The algorithms that are being researched are all performed on undirected, unweighted graphs. If a graph is undirected, each edge between node has a connection that goes from A to B and B to A. This means that in the dataset that if there exists a line “2 3 1” (A connection from node 2 to 3 with weight 1) there should also exist “3 2 1” (A connection from 3 to 2 with weight 1).

[1] https://networkx.github.io/documentation/networkx-1.10/index.html#
[2] https://graph-tool.skewed.de/static/doc/index.html

Condition not met. For each dataset downloaded and defined as undirected, there does not exist a counter connection between nodes. This is intentional to save data downloaded by half of its size. Files that have directed graphs often have a readme file that describes in which direction the edges of the file are, or an extra column that exists that defines the direction. In our datasets, all edges are imported to Spark/Flink with the property *EdgeDirection.EITHER,* simply creating edges both ways in the graph structure.

Our data does not have any dependent or independent variables since our research is not associated with vertex variables or any other values that a node might have. To further analyze the networks used in our research, a few important properties were investigated by either writing our own code or using inbuilt algorithms[3].

Finally, each vertex is plotted against it degree to see the distribution of vertices. These plots provide valuable insights into the data. Firstly, one can notice that the distribution degree is quite evenly spread in all datasets. Secondly, there exist at least one heavily connected node in each set, indicating that there might either be a center vertex or a really influential one.

Results of analysis:

As mentioned above, this paper presents the checks and analysis done on the data that the frameworks will process their algorithms on while the research is about analyzing the execution time of two libraries (GraphX and Gelly) and their performance on both batches and streams of data. Analyzing the data that is going to be processed is important for the results of our research to have a deeper understanding of what factors might affect the results.

These datasets were picked to see how the performance will change as size increases from around 100, to 1,000 and finally to 10,000 nodes. From the information about the data, it is clear that the size of the networks does increase since there are more nodes and more connections between them.

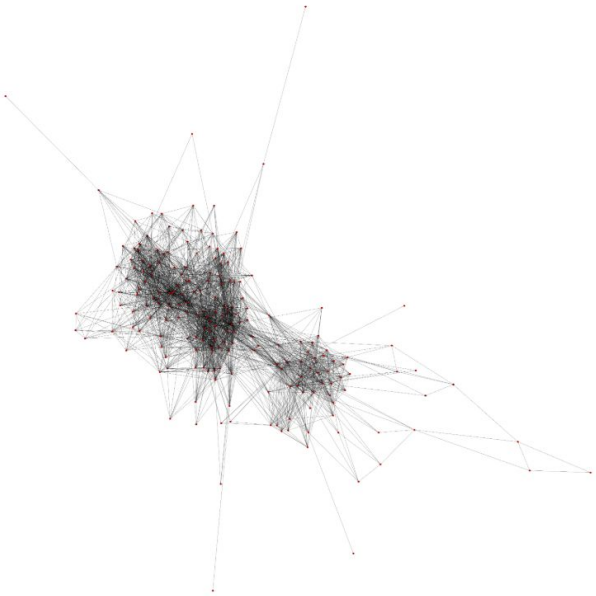
However, we do notice that the first and the second dataset have double the density of the largest dataset. The triangle count drops in the second data set but increases heavily in the third one. Finally, the average degree of vertices seems to drop as the number of vertices in the networks increases.

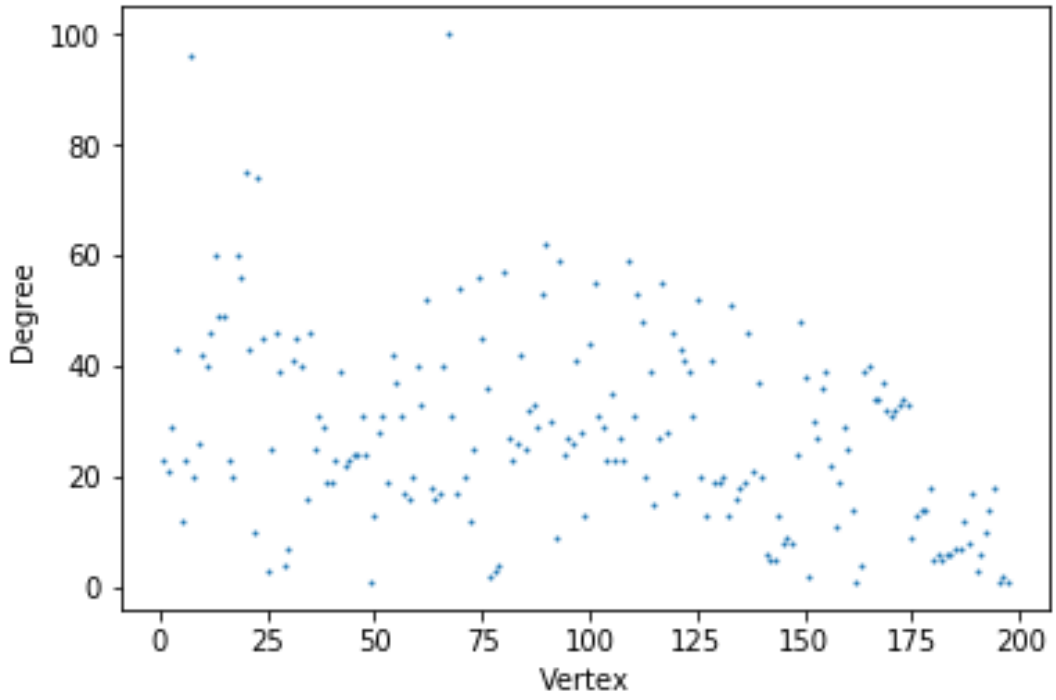
This might be an indicator of further research on algorithm performance in graph processing, to see if these properties of the network do affect the libraries in any way.

[3] https://networkx.github.io/documentation/stable/reference/algorithms/index.html

Jazz Musicians

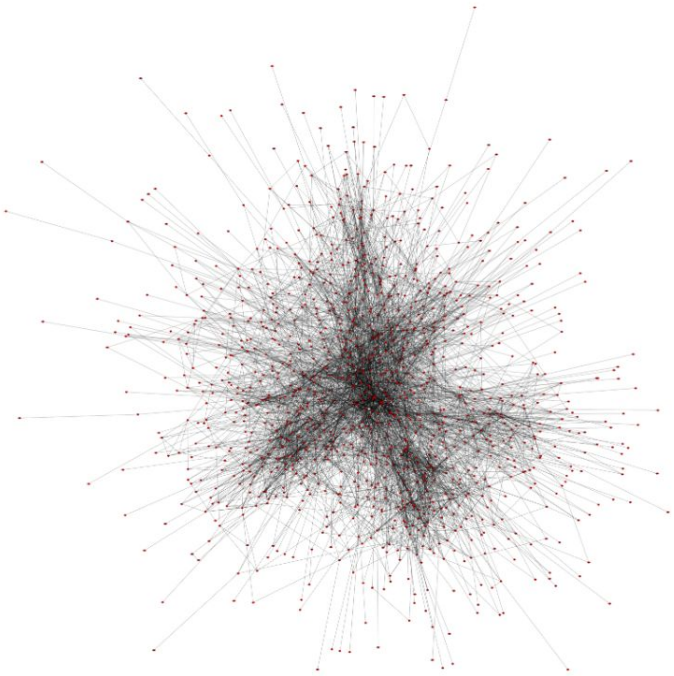
Number of nodes	198
Number of edges	2,742
Max degree	100
Min degree	1
Average degree	27,697
Density	0.14059375
Triangle count	17,899

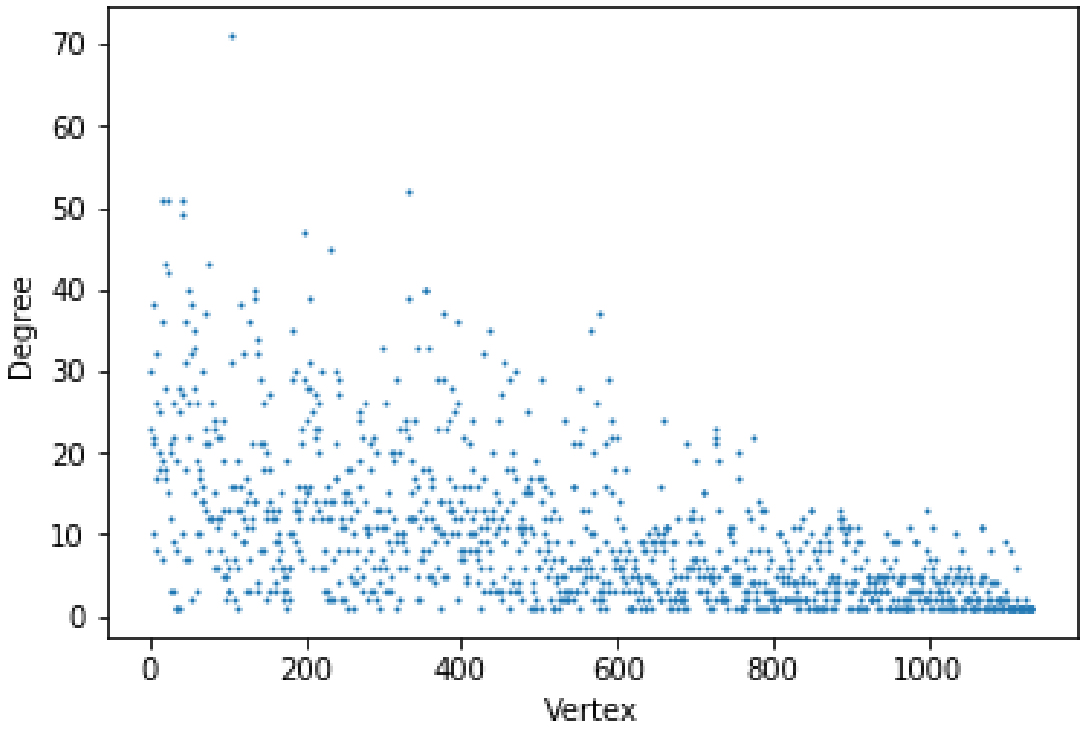




U. Rovira i Virgili

Number of nodes	1,133
Number of edges	5,451
Max degree	71
Min degree	1
Average degree	9.6222
Density	0.00850021
Triangle count	5,343





Pretty Good Privacy

Number of nodes	10,680
Number of edges	24,316
Max degree	205
Min degree	1
Average degree	4.5536
Density	0.0004264
Triangle count	54,788