M2 – NSD (Practical Work 4)

Community detection

Maximilien Danisch, Marwan Ghanem, Lionel Tabourier

In this practical we consider algorithms for partitioning the nodes in the input graph into communities, except in exercise 5 where we consider an algorithm to compute overlapping communities.

Exercise 1 — ***Simple bechmark***

Implement an algorithm to generate the following random graph.

- The graph has 400 nodes partition into 4 clusters of size 100.
- Each pair of nodes in the same cluster is connected with a probability p
- Each pair of nodes in different clusters is connected with a probability $q \leqslant p$

Draw the obtained graphs for various values of p and q using a software of your choice. For instance: `https://networkx.github.io/documentation/stable/reference/drawing`

What is the effect of increasing or decreasing $\frac{p}{q}$ on the community structure?

Exercise 2 — ***Label propagation***

Implement the label propagation algorithm.
Run your program on the benchmark graphs generated for Exercise 1. Draw the graph and color the nodes nodes using a different color for each community.
Run your program on `http://snap.stanford.edu/data/com-Youtube.html` and draw a histogram of the sizes of the communities you obtained.
Run it 1000 times on this same graph and draw a histogram of the numbers of communities you obtained.
Comment your results.

Exercise 3 — ***New algorithm***

Three choices are available for this exercise, listed in decreasing order of difficulty:

1. Suggest your own community detection method and implement it.
2. Make your own implementation of an existing method.
3. Use an existing implementation of an existing method.

For choice 1., make an algorithm significantly different from Label Propagation and Louvain. For choices 2. and 3., pick an algorithm different from Label Propagation and Louvain.

Explain your algorithm: the intuition behind it and the implementation issues.

Exercise 4 — ***Validation***

Compare (i) the Label Propagation you have implemented in exercise 2, (ii) the Louvain algorithm (implementation available here: `https://perso.uclouvain.be/vincent.blondel/research/louvain.html`) and (iii) the algorithm in exercise 3. For this you will need to design your own experiments:

- evaluate the scalability of the algorithms/programs using graphs of different sizes and reporting the running time.
- evaluate the accuracy of the algorithms using the benchmark made in question 1, the LFR benchmark `https://sites.google.com/site/santofortunato/inthepress2` and some metrics to compare partitions.

Which algorithm(s) perform(s) the best?

Exercise 5 — *(Optional) Triangle percolation*

Implement an efficient algorithm for the k-clique percolation method (slide 31 of course 6) for $k = 3$. Make sure your algorithm is correct comparing the output of your algorithm with the output of the original algorithm (for $k = 3$): `http://www.cfinder.org/`, then compare the scalability of the two approaches.