

Community detection

Maximilien Danisch

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 benchmark*

Implement an algorithm to generate the following random graph.

- The graph has 400 nodes partition into 4 clusters of 100 nodes.
- 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 $q \leq p$

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 using a different color for each community.

Exercise 3 — *New algorithm*

Suggest your own community detection method and implement it.

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) your own algorithm (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 report the running time and memory consumption.
- 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 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.cfindexer.org/>, then compare the scalability of the two approaches.