Community detection in graphs

Fabien Tarissan

In this practical we consider algorithms for partitioning the nodes of the input graph.

1 What is expected

You are asked to study community detections algorithms for graphs with an experimental approach. This requires you to:

1. Propose your own community detection method and implement it.
2. Compare it to existing algorithms (either by implementing them or using existing implementations).

In the report, it is expected that you present:

- a description of the community detection problem
- a description of the algorithms used (most important part) and an evaluation of their complexity,
- experimental results : the datasets used (and their features), optimization score (e.g. modularity), features of the communities obtained (e.g. size distribution), computation times, criteria used to compare the algorithms, ...
- you can conclude on a discussion of your results : performances, variations or upgrades or any thought that you consider relevant.

2 Some leading exercices

Exercise 1 — ***Simple bechmark***

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 $q \leqslant p$

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 — ***Test a community detection approach***

Choose and adapt/implement an algorithm proposed in Section 3.
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.

Exercise 3 — ***Experimental evaluation***

Compare several methods by designing your own experiments:

- evaluate the scalability of the 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://github.com/eXascaleInfolab/LFR-Benchmark_UndirWeightOvp` and some metrics to compare partitions: Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), ...
- evaluate the accuracy of the algorithms on network with ground-truch community structures: `http://snap.stanford.edu/data/index.html#communities`

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

Exercise 4 — ***New algorithm***

Suggest your own community detection method and implement it.

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

Add it in the experimental evaluation (exercise 2).

3 Possible community detection algorithms

- Louvain algorithm:

  Fast unfolding of communities in large networks, Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, 2008.

  Implementation: `https://perso.uclouvain.be/vincent.blondel/research/louvain.html`

- Divisive edge-betweenness algorithm :

  Community structure in social and biological networks, Girvan and Newman, 2002.

  Implementation in python : `github.com/kjahan/community`

- Random walk based algorithm :

  Computing Communities in Large Networks Using Random Walks, Pons and Latapy, 2005.

  Implementation in C++ : `www-complexnetworks.lip6.fr/~latapy/PP/walktrap.html`

- Leading eigenvector algorithm :

  Finding community structure in networks using the eigenvectors of matrices, Newman, 2006.

  Implementation in R : `igraph.org/r/doc/cluster_leading_eigen.html`

- Simulated annealing algorithm :

  Functional cartography of complex metabolic networks, Guimera and Amaral, 2005.

  Implementation in C : `seeslab.info/downloads/network-c-libraries-rgraph`

- Label propagation algorithm :

  Near linear time algorithm to detect community structures in large-scale networks, Ragahavan et al., 2007.

  Implementation in R : `igraph.org/r/doc/cluster_label_prop.html`

- K-cliques based algorithm (overlapping communities) :

  Uncovering the overlapping community structure of complex networks in nature and society, Palla et al., 2005.

  Implementation in R : `igraph.wikidot.com/community-detection-in-r`

- Physics inspired algorithm (overlapping communities) :

  Detecting fuzzy community structures in complex networks with a Potts model, Reichardt and Bornholdt, 2004.

  Implementation in R : `igraph.org/r/doc/cluster_spinglass.html`

- Map equation algorithm :

  Maps of random walks on complex networks reveal community structure, Rosvall and Bergstrom, 2007.

  Implementation in R : `igraph.org/r/doc/cluster_infomap.html`