Experimental Study Term Paper - Network Science 2020

Possible topics

Below you can find five topics that we have selected. It is also possible to propose your own topic. All topics are high level: you should formulate questions that your experiments try to answer yourself.

For inspiration, consider this comparative study: Lancichinetti and Fortunato, Community detection algorithms: a comparative analysis. (Partly) reproducing and verifying the results of this later paper can be part of the experiments, however, we do expect something new (either a comparison to another algorithm, testing some hypothesis not in this paper, or an algorithm of your own). For even more inspiration on community detection algorithms, the following is a nice overview paper: Fortunato and Hric. Community detection in networks: a user guide.

6. Compare different algorithms for modularity.

There are many different algorithms that try to find community structure by optimising modularity. Compare different algorithms and/or try an algorithm of your own.

The NetworkX package provides an implementation of Clauset-Newman-Moore, and extensions of this package for the Louvain algorithm also exist. The course's books also give several examples of algorithms to optimise modularity, including the simple node moving algorithm (see lecture), the greedy algorithm from the Barabási book, and spectral decomposition. Also, you can compare to optimisation algorithms from the optimisation literature such as simulated annealing, tabu search, genetic algorithms, etc.

Note that some algorithms are for the 2-community case, while other apply to any number of communities. Comparisons should be made using either one or more variants of repeated bisection or should restrict the more general algorithms to the 2-community case.

7. Compare different measures of community structure.

Instead of comparing many algorithms for the same problem, one can fix a general-purpose community optimisation algorithm and try different measures of community structure. For example, take the simple node moving algorithm form the book and lecture (possibly modified to the multi-community case), and apply it to optimise against different objective functions.

You can do this with several base-algorithms. Logical choices would be simple node moving or and agglomerative merging algorithm (let every node start as a single community and merge), etc. One should compare algorithm performance to a ground truth that is preferably real-world data, or at the least not directly related to the measures you are comparing (do not use a statistical model to generate the data and then use maximum likelihood estimation of the communities using the same statistical model). Also, we would recommend to either fix the number of communities or select or adjust measures such that they work on any number of communities.

Measures you can optimise against include modularity, the map equation (the equation behind the InfoMap method, see the Newman book section 14.3) statistical models (see lecture, and the Newman book section 14.4), or others such as Baumes et al., Finding communities by clustering a graph into overlapping subgraphs or Lancichinetti et al., Finding Statistically Significant Communities in Networks.

8. Fast heuristics on large graphs.

One of the reasons that many community detection algorithms are heuristics, is because we want to apply them networks of ever-increasing sizes. Compare existing algorithms for community detection (you still have to program one yourself) on how they scale when considering larger and larger graphs. Compare not only the speed of the algorithms and how the running time scales, but also the quality of the computed communities relative to the input size.

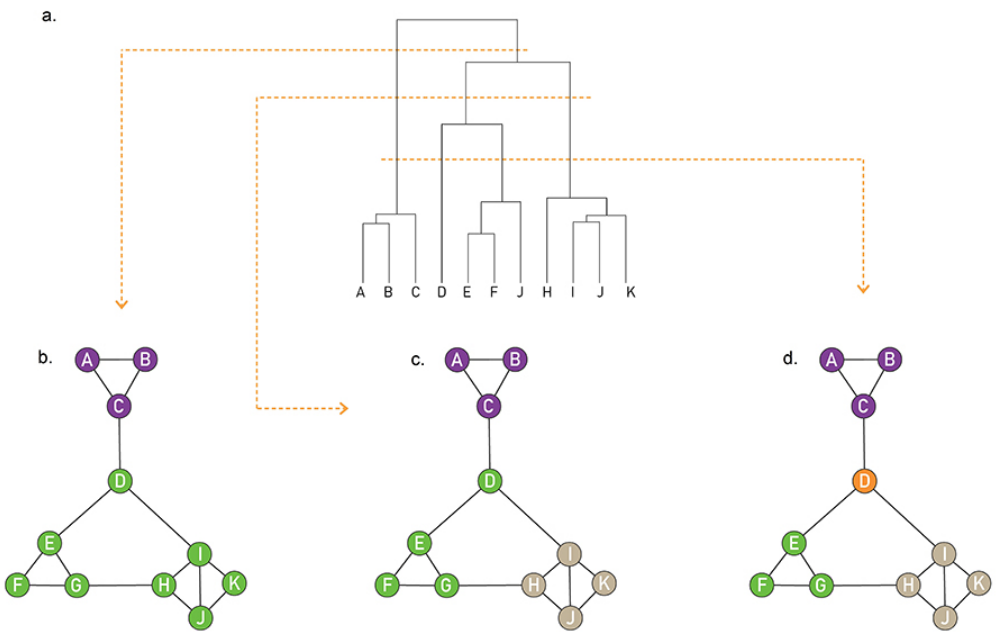
We recommend using one or more network generators (with fixed seed) to perform your experiments. Methods to compare include Louvain, InfoMap, and the algorithms in the NetworkX package such as Kernighan-Lin, Label propagation, or Fluid communities. Other methods for large graphs include Ronhovde and Nussino, Multiresolution community detection for megascale networksby information-based replica correlations, or methods from the comparative study and overview paper in the introduction above.

9. Finding overlapping communities.

Standard community detection algorithms allow a vertex to be in one community only. In some applications this is not very realistic, and therefore there are many algorithms that consider overlapping communities (e.g., see Barabási, Section 9.5, or Newman Section 14.7.1). Algorithms for overlapping communities include the clique percolation algorithm (a.k.a. CFinder), and algorithms based on clustering edges instead of vertices (hence, allowing overlapping communities), e.g., Evans and Lambiotte, Line Graphs, Link Partitions and Overlapping Communities, or the link-clustering algorithm by Ahn, Bagrow and Lehmann (Barabási book, Section 9.5.2), or a statistical method based on a random graph model with overlapping communities (e.g., see Airoldi et al.). Note that random graph models such as this last one can also be used to define a ground truth.

10. A comparison of hierarchical dendrogram-based methods.

For some categories of real-world networks, community structure exists at multiple levels, and hence these networks have a hierarchical community structure (for who took the Algorithms and Networks course, there we saw that structure in real world road networks). Dendrogram-based method are a form of hierarchical clustering/community detection algorithms that make this hierarchical structure explicit. They either work letting every vertex be its own community and merging them by some distance measure between communities, or by repeatedly splitting the vertices into two groups. As such, one does not obtain a division into communities as a result, but a tree that gives the community structure. E.g., see this dendrogram from the Barabási book:



Care should be taken to define a ground truth (e.g., through a hierarchical random graph model, see Clauset et al.) and to define a measure for the distance of a computed dendrogram to such a ground truth dendrogram. Alternatively, one could fix a method that takes the 'best-fit' division in a dendrogram to compare against non-hierarchical community structure.

Algorithms that produce dendrograms in a bottom-up way include, the algorithms of Radicchi et al. (see also Newman book Section 14.5.1). Top-down algorithms include betweenness-based methods (see the Newman book Section 14.5.1), the Ravasz algorithm (see the Barabási book, section 9.3.1), and the Girvan-Newman algorithm (Barabási book, section 9.3.2).

Note that there are many parameters one can play with here. For example, in a bottom-up approach to dendrograms one can vary the similarity measure used between vertices, how this relates to similarities between groups of vertices (e.g., see the Newman book Section 14.5.2). Or try a top-down approach many methods to split the graph into two communities can be used. Or compare methods that determine the correct level to cut a dendrogram into communities (where you do not what one large community and a few single vertex communities).

Some sources for Network data

- Girvan-Newman (GN) Benchmark. https://www.npmjs.com/package/girvan-newman-benchmark.
- Lancichinetti-Fortunato-Radicchi (LFR) Benchmark (in NetworkX).
- Stanford SNAP Stanford Large Network Dataset Collection: http://snap.stanford.edu/data/index.html.
- KONECT: The Koblenz Network Collection: http://konect.uni-koblenz.de/.
- Network Repository: http://networkrepository.com/.
- LWA (Webgraph): http://webgraph.di.unimi.it/.
- SuiteSparse (Florida): http://faculty.cse.tamu.edu/davis/matrices.html.
- 10th DIMACS implementation Challenge: https://www.cc.gatech.edu/dimacs10/downloads.shtml.