Universitat Rovira i Virgili

Complex Networks

Practical Report

Report of the Exercise 2: Community detection in complex networks

Authors:
Julià Camps Sereix

Supervisor:
Sergio Gmez Jimnez

April 22, 2016

Contents

Motivation

This exercise is motivated by the importance of understanding the different algorithms and methods studied for communities detection in complex networks. Being able to distinguish among the strong and weak points of them, due to their characteristics. In order to be able to distinguish in a proper way which method should be used on each oncoming facing problem on the future.

The algorithms presented on this report have been all tested with already well known network analysis applications, which are briefly described the following part of this section.

The algorithms presented in this report are:

1. **Fast algorithm (Newman, 2004):** Is a "greedy" algorithm which retrieves as an output an *dendrogram*, on whose every level has been built trying to maintain the modularity measure to be maximum. Then at the end it chooses the reviewed partition to achieve the maximum modularity measure during the process performed. This algorithm is implemented in *Radatools*, and we will use from this implementation.

2. **Louvian method (Blondel et al., 2008):** This algorithm, is an iterative algorithm, that joins nodes to form communities, trying to maximise the modularity function, and at the end of each iteration considers each community generated as a single node with n self loops, where n is the number of edges in the underlying community. This "consideration" is called the *coarsening phase*. The implementation used for this algorithm, is the one found in *Pajek* (Batagelj and Mrvar, 2003) software (this implementation corresponds to Rotta and Noack (2011), however, in the paper it can be seen that when using the single level case, this algorithm corresponds to Blondel et al. (2008)). The configuration that will be choosed for performing the experiments in an homogeneous notion, will be the *default*[1] one offered by *Pajek*.

3. **Spectral algorithm (Newman, 2006):** Firs of all, notice that this approach is a top-down[2] approach, while the other two algorithms are botom-up[3]. It computes the leading eigenvector of the modularity matrix and divide the vertices into two groups according to the signs of the elements in this vector. It has a very interesting property, since the algorithm has the ability not only to divide networks effectively, but also to refuse to divide them when no good division exists. The networks in this latter case will be called indivisible. That is, a network is indivisible if the modularity matrix has no positive eigenvalues. That is an important feature of the algorithm. This algorithm is implemented in *Radatools*, and we will use from this implementation.

The programs used for this practical work are:

- **Radatools:** Is an aggregation of different programs devoted to network analysis, with other tools related to work with different format files and different functionalities. As a whole, it provides for every needed functionality when intending to perform community analysis on complex networks (it only lacks of a visualisation tool).

- **Pajek (Batagelj and Mrvar, 2003):** Is a very complete program devoted to complex networks analysis, from it is very easy to load and save different versions of a same network. While operating on it. In this case, it disposes of very complete visualisation tools.

[1] For applying the Louvian method in Pajek, one of the possible configurations is choosing the *Coarsening + Single Refinement* option with the *default* configuration of parameters, which implies: *Resolution parameter:1*, *Number of Random Restarts:1*, *Maximum Number of Levels in each Iteration:20*, *Maximum Number of Repetitions in each Level:50*.

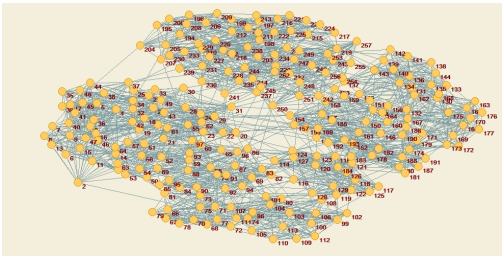
[2] It starts with only one cluster and splits it trying to maintain *modularity* measure to be maximised.

[3] It starts with individual nodes and joins them into communities trying to maximise *modularity* of the partition

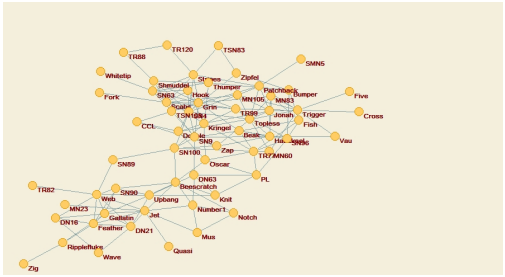
For the testing part of this report it was decided to perform all experiments to three sets of different networks:

- **Model:** This are big networks generated with some of the models reviewed in the previous *practical exercise*.
- **Real:** This networks have been extracted form real cases (e.g. social interaction among people enrolled in the Complex Networks subject).
- **Toy:** This networks are very tiny networks of extremely different shapes. The experiments among them will be very interesting for identifying what are the algorithms exactly doing.

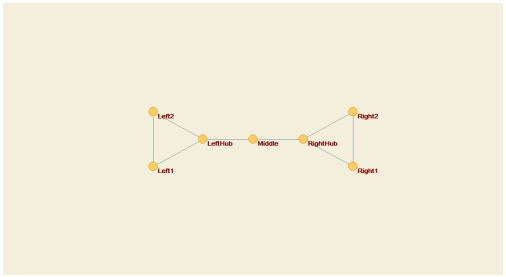
Here we show a plot of this three different types of networks:



(a) Model network.



(b) Real network.



(c) Toy network.

Figure 1: Sample of the different types of networks.

In figure 1 we can observe a sample of the different types of networks used for this exercise.

Community analysis of the networks

In this section the different types of networks will be analysed using the algorithms mentioned in the previous section. The results of all the experiments performed are commented at the end of the section from a global perspective, in order to be able to comment on the algorithms, rather than the concrete cases of each typology.

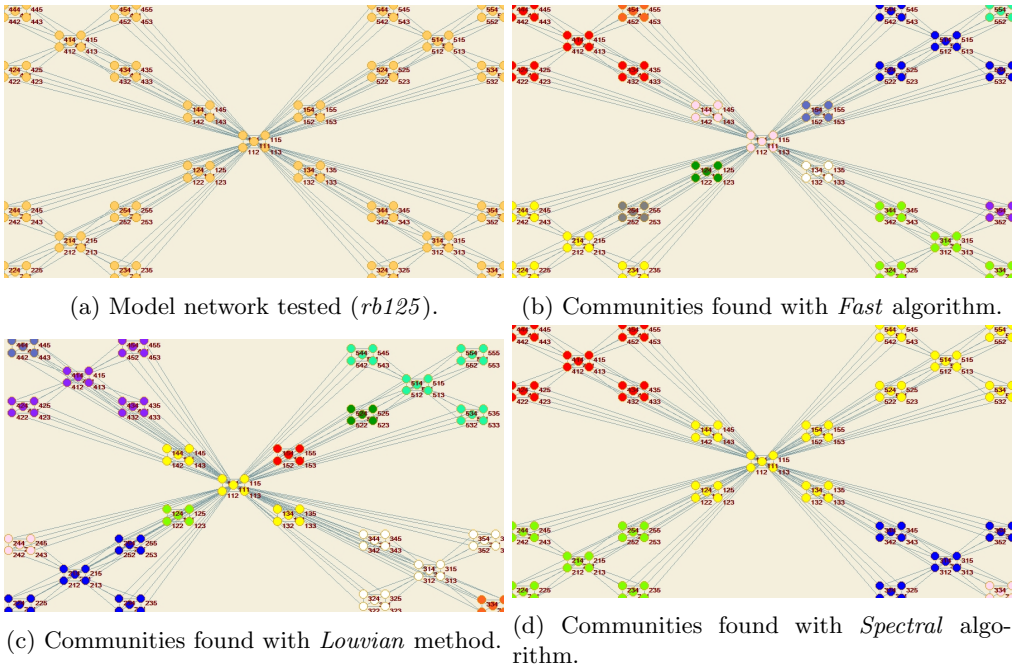
Analysis of Model networks

In this section we will analyse the set of model networks provided for this exercise, as it has been requested.

Network	Software	Algorithm	Parameters	Communities	Modularity
rb125	Radatools	Fast	UN	12	0.608733
rb125	Pajek	Louvian	*default*	11	0.6085
rb125	Radatools	Spectral	UN	5	0.554099
256_4_4_2_15_18_p	Radatools	Fast	UN	9	0.76566
256_4_4_2_15_18_p	Pajek	Louvian	*default*	16	0.781804
256_4_4_2_15_18_p	Radatools	Spectral	UN	16	0.70222

Table 1: Results from using different methods and configurations in for the "model" networks.

Here we show some plots to provide a better understanding on the tests performed in this section.



(a) Model network tested (*rb125*).

(b) Communities found with *Fast* algorithm.

(c) Communities found with *Louvian* method.

(d) Communities found with *Spectral* algorithm.

Figure 2: Sample of tests performed in this section, using "model" networks.

Analysis of Real networks

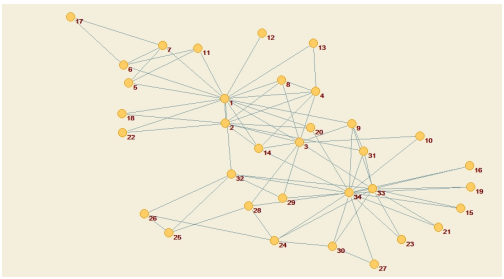
In this section we will analyse the set of real networks provided for this exercise, as it has been requested.

In this section we will find some references to the "true" partitions of a network, this partitions are provided as the known partitions before applying the algorithms (the real ones observed in the environment).

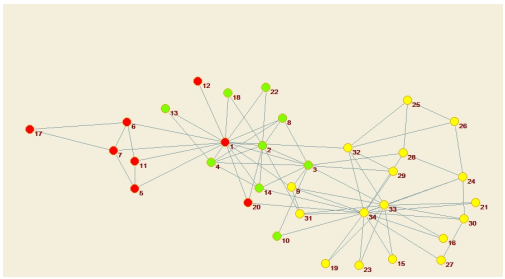
Network	Software	Algorithm	Parameters	Communities	Modularity
cat_cortex_sim	Radatools	Fast	UN	3	0.260436
cat_cortex_sim	Pajek	Louvian	*default*	3	0.256044
cat_cortex_sim	Radatools	Spectral	UN	4	0.251026
cat_cortex_sim	-	*true*	-	4	0.246
dolphins	Radatools	Fast	UN	4	0.495491
dolphins	Pajek	Louvian	*default*	5	0.523338
dolphins	Radatools	Spectral	UN	2	0.389858
dolphins	-	*true*	-	2	0.373482
football	Radatools	Fast	UN	6	0.549741
football	Pajek	Louvian	*default*	10	0.60457
football	Radatools	Spectral	UN	2	0.37572
football	-	*true*	-	12	0.553973
zachary_unwh	Radatools	Fast	UN	3	0.380671
zachary_unwh	Pajek	Louvian	*default*	4	0.41979
zachary_unwh	Radatools	Spectral	UN	2	0.371466
zachary_unwh	-	*true*	-	2	0.371466

Table 2: Results from using different methods and configurations in for the "real" networks.

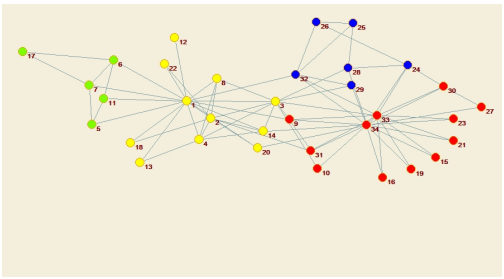
Here we show some plots to provide a better understanding on the tests performed in this section.



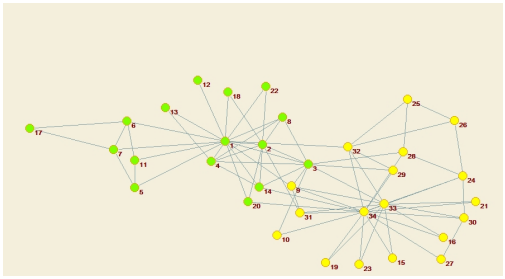
(a) Real network tested (*zachary_unwh*).



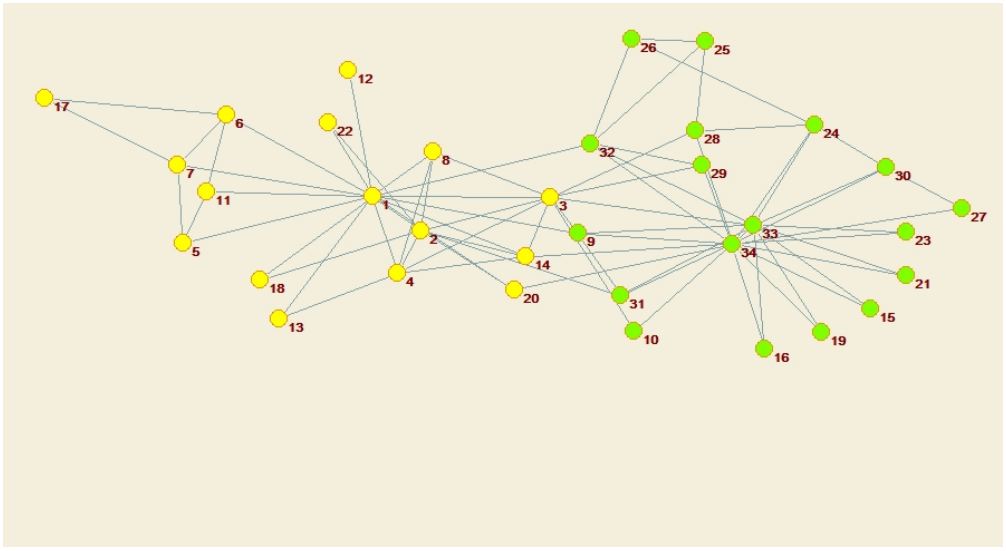
(b) Communities found with *Fast* algorithm.



(c) Communities found with *Louvian* method.



(d) Communities found with *Spectral* algorithm.



(e) True communities of the network.

Figure 3: Sample of tests performed in this section, using "real" networks.

Since in this section we are testing real well studied networks, we know the real communities that exist on this networks. Though, we can compare the real communities underlying in the network with the ones computed using the tested algorithms. In figure 3 it can be visualised the kind of experiments that are being performed in this section, together to the real communities in the network tested.

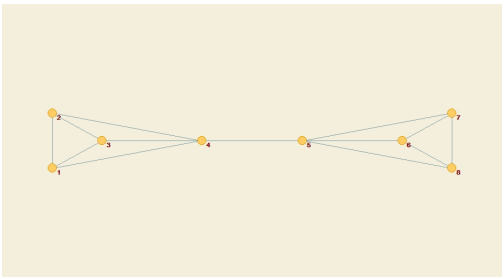
Analysis of Toy networks

In this section we will analyse the set of *toy*[4] networks provided for this exercise, as it has been requested.

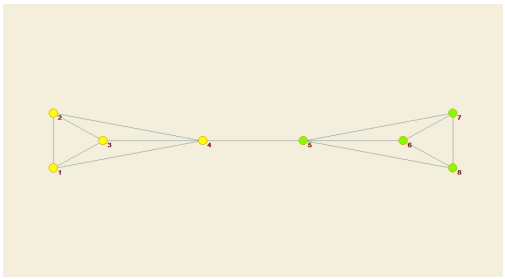
Network	Software	Algorithm	Parameters	Communities	Modularity
graph4+4	Radatools	Fast	UN	2	0.423077
graph4+4	Pajek	Louvian	*default*	3	0.204142
graph4+4	Radatools	Spectral	UN	2	0.423077
20x2+5x2	Radatools	Fast	UN	3	0.542579
20x2+5x2	Pajek	Louvian	*default*	3	0.54257855
20x2+5x2	Radatools	Spectral	UN	3	0.542579
graph3+1+3	Radatools	Fast	UN	2	0.367188
graph3+1+3	Pajek	Louvian	*default*	2	0.3671875
graph3+1+3	Radatools	Spectral	UN	2	0.367188
star	Radatools	Fast	UN	1	0.0
star	Pajek	Louvian	*default*	1	0.0
star	Radatools	Spectral	UN	1	0.0

Table 3: Results from using different methods and configurations in for the "toy" networks.

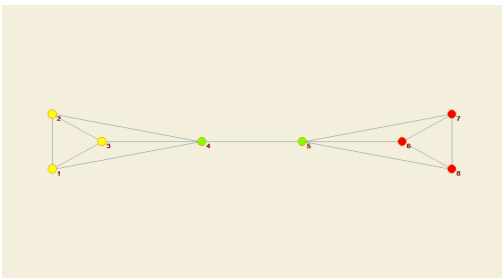
Here we show some plots to provide a better understanding on the tests performed in this section.



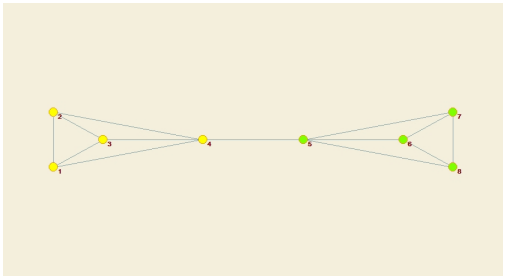
(a) Toy network tested (*zachary_unwh*).



(b) Communities found with *Fast* algorithm.



(c) Communities found with *Louvian* method.



(d) Communities found with *Spectral* algorithm.

Figure 4: Sample of tests performed in this section, using "toy" networks.

[4] Interesting shapes, generated on porpoise (by hand).

Discussion on the Results

During this section it have been shown all the obtained results during the tests with the reviewed algorithms and software, on the three types of tests.

It can be observed that in most cases, the algorithms retrieving partitions of communities with higher modularity values, are: *Fast(Newman, 2004)* and *Louvian(Blondel et al., 2008)*. However, *Spectral(Newman, 2006)*, as can be seen in 2, is the algorithm that retrieves more times a number of communities that matches with the *true* number of communities on the network. Moreover, it can be seen on the plots reported, that as higher are the modularity retrieved for a given partition, more intuitive seams the solution to be correct, despite of the fact that many times did not correspond to it.

I was surprised from the robustness of this algorithms, since they have shown to be able to deal with several different types of networks, and from the fact that every method is retrieving different partitions, in most cases, and then, it can easily be deduced, that deciding among this different results is not a trivial problem.

From this variety on the results retrieved, it seams a good scenario for ensemble methods, such as voting aggregation mechanisms for taking into account the results of different methods, in order to decide the best community partition given some network.

Strengths and Weaknesses

In this section I will comment some issues that were not planned and altered the planification of performing the exercise.

Strengths

- The literature provided by the teacher was very complete, although difficult to understand at some point the methods on the literature, we were able to choose among many different alternatives. Though, finding an easier algorithm, if the one being reviewed showed to be unfeasible (in complexity terms, due to the weak background on the area).
- So the only kind of research needed was for clarification and support to the already available, to double check concepts and algorithms meanings.
- The amount of time provided was widely more than needed, which permitted to get a deeper understanding on the target topics.

Weaknesses

- It showed up that the computer that I have available for working, did not support the installation of the *igraph* nor *Scipy* python library. This implied loosing a lot of time trying to get it to work and having to invest extra time in implementing the desired functionalities from scratch.
- Due to the OS installed, it also came up that *graph_tool* Python library, was also impossible to install in the computer.
- From all this complications, I ended up doing the exercise using the software reviewed and some checking using *networkx* Python library.
- The technologies available for this practical exercise, were widely unknown for us, due to loosing some lab sessions when they were deeply commented. From this, we had to struggle with the documentation of the implemented methods and the GUI provided by the programs used.

Conclusions and Future Work

- From this exercise we learnt different methods community in networks analysis, with several different strategies (e.g. using betweeness, random-walkers, modularity optimisation, VOS metrics, top-down,bottom-up, etc.), even if we have not used all the strategies, we reviewed them all in order to decide which techniques to use during the testing. So from an existing network we could easily distinguish if we are facing a network with or without communities, and with a bit of intuition (deep enough understanding on the algorithms reviewd), be able to decide which approach could give better results for showing this communities.
- As future work, it wold be interesting to obtain a deeper insight view on the algorithms discarded due to their complexity (e.g. (Arenas et al., 2008)).

References

A Arenas, A Fernndez, and S Gmez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008. URL `http://stacks.iop.org/1367-2630/10/i=5/a=053039`.

Vladimir Batagelj and Andrej Mrvar. Pajek – analysis and visualization of large networks. In *GRAPH DRAWING SOFTWARE*, pages 77–103. Springer, 2003.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008, 2008. URL `http://stacks.iop.org/1742-5468/2008/i=10/a=P10008`.

M E Newman. Modularity and community structure in networks. *Proc Natl Acad Sci U S A*, 103(23):8577–8582, June 2006. doi: 10.1073/pnas.0601602103. URL `http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=retrieve&db=pubmed&list_uids=16723398&dopt=AbstractPlus`.

M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004. doi: 10.1103/PhysRevE.69.066133. URL `http://link.aps.org/doi/10.1103/PhysRevE.69.066133`.

Randolf Rotta and Andreas Noack. Multilevel local search algorithms for modularity clustering. *J. Exp. Algorithmics*, 16:2.3:2.1–2.3:2.27, July 2011. ISSN 1084-6654. doi: 10.1145/1963190.1970376. URL `http://doi.acm.org/10.1145/1963190.1970376`.