Some genes have many partners: Detecting biological modularity

Progress report

(version 1)

Antoine Allard

antoine.allard.1@ulaval.ca

http://dynamica.phy.ulaval.ca

October 31, 2011

Abstract

This document is an attempt to sumarize and to collect information about the use of the community detection algorithm introduced by Ahn *et al.* [1] on *genes–genes* and *conditions–conditions* networks. A few basic concepts are first introduced to fix the vocabulary and comments are given about the algorithm and its performance. We then present some of the results obtained.

1 Theoretical considerations

1.1 Basic Concepts

We consider simple undirected networks composed of N nodes and M edges. Node i is connected to k_i neighbors ($i.e$ its degree) and $n(i)$ is the set of nodes that are neighbors of node i. The probability that any given node has a degree of k is given by the degree distribution p_k. The average degree $\langle k \rangle$, hence the average number of neighbors that nodes have, can be easily computed from these quantities

$$\langle k \rangle = \sum_{k=0}^{\infty} k p_k = \frac{1}{N} \sum_{i=1}^{N} k_i = \frac{2M}{N} \ .$$

The edge between node i and j is refered as e_{ij} and its weight as w_{ij}. Since edges are undirected, e_{ij} and e_{ji} correspond to the same edge and of course $w_{ij} = w_{ji}$.

1.2 The algorithm

1.2.1 Edges similarities

The algorithm used in this project [1] computes the similarity between every pair of edges that have a node in common ($e.g$ e_{ik} and e_{kj}, where node k is called the *keystone*) according to the Tanimoto coefficient:

$$T(e_{ik}, e_{kj}) = \frac{\boldsymbol{a_i} \cdot \boldsymbol{a_j}}{|\boldsymbol{a_i}|^2 + |\boldsymbol{a_j}|^2 - \boldsymbol{a_i} \cdot \boldsymbol{a_j}} \tag{1}$$

where $\boldsymbol{a_i}$ is an N-element vector that contains information about how node i is connected to its neighbors (it is mostly the i-th row of the adjacency matrix).

In the Supplementary Information document of [1], Ahn *et al.* suggest to compute $\boldsymbol{a_i}$ according to

$$[\boldsymbol{a_i}]_j = w_{ij} + \frac{\delta_{ij}}{k_i} \sum_{l \in n(i)} w_{il} \tag{2}$$

where δ_{ij} is the Kroenecker delta ($\delta_{ij} = 1$ if $i = j$ and 0 otherwise). That is the j-th element of $\boldsymbol{a_i}$ is the weight w_{ij} of e_{ij} (equals zero if no edge) except for the i-th element which is the average weight of edges that node i is attached to.

The `R` package "linkcomm" by Kalinka and Tomancak [2] computes $\boldsymbol{a_i}$ in a slightly different way:

$$[\boldsymbol{a_i}]_j = w_{ij} + \delta_{ij}$$

where instead of using the average weights of edges that node i has, the i-th element is simply 1. This affects the value of the Tanimoto coefficients twofold. First, since $|\boldsymbol{a_i}|^2 \geq 1$ with this method, Tanimoto coefficients will in general be lower than the ones computed using the first method. Second, triangles – where e_{ik}, e_{kj} and e_{ij} exist – will have a higher Tanimoto coefficient when calculated with this method than with the other one.

We do not at the moment have the knowledge to be able to state which method is better and we have arbitraryly chosen to use the one described in the Supplementary Information document of [1].

1.2.2 Communities density

To choose the threshold value, for which contiguous edges having a higher Tanimoto coefficient will be considered as part of the same community, the average density, D, of communities if calculated. The density of a community composed of n_c nodes and m_c edges is defined as

$$D_c = \frac{m_c - (n_c - 1)}{n_c(n_c - 1)/2 - (n_c - 1)} \ . \tag{3}$$

The average density of communities is then

$$D = \frac{1}{M} \sum_c m_c D_c \ . \tag{4}$$

1.3 Other considerations

Knowing the degree k_i of node i for all nodes in the network, the number of contiguous edge pairs whose similarities will be computed is given by

$$K = \sum_{i=1}^{N} \binom{k_i}{2} = N \sum_{k}^{\infty} p_k \binom{k}{2}$$

where p_k is the degree distribution (*i.e.* a fraction p_k of the nodes are of degree k) and N is the number of nodes in the network. Also, a given edge, e_{ij}, between nodes i and j will be involved in

$$(k_i - 1) + (k_j - 1)$$

contiguous pairs.

2 Algorithm's performance

Table 1 shows a few properties of the networks considered in this project as well as the computing times of the algorithms.

	gBg (cutoff 0.1)	gBg (full)	cBc (cutoff 0.1)	cBc (full)
Nodes (N)	2260	2400		1495
Edges (M)	71 907	1 565 816		1 115 538
Average degree ($\langle k \rangle$)	63.63	1304.85		1492.36
Contiguous edge pairs (K)	6 684 488	2 330 018 007		1 664 418 474
Comput. time (Tanimoto)	$\sim$20 sec.	$\sim$90 min.		$\sim$30 min.

Table 1: Properties of the gene-by-gene (gBg) and condition-by-condition (cBc) networks considered in this project as well as the computing times of the algorithms.

3 Extracting the communities

We give a detailled step by step description of how we extracted the communities from the networks and of how to use the algorithms. Note that most codes presented in this section assume that networks are undirected and should therefore be modified accordingly if used with other types of networks.

3.1 Formating the egde list

The gene-by-gene (gBg) and condition-by-condition (cBc) networks were received under the format of adjacency matrices where each line contained a single entry of the matrix. Because these networks are *undirected* and *not fully connected*, the original files contained a lot of useless information. We used the code `adjmat2wpairs.cpp` to remove this information and to generate a weighted edge list in the `.wpair` format[1]. This code assigns contiguous integer IDs to nodes and uses them in the `.wpairs` file. A `.numid2name` file containing the correspondence between the original node names and the numerical IDs is also generated.

3.1.1 Usage

Compilation:

```
    g++ -O3 -o adjmat2wpairs adjmat2wpairs.cpp
```

Execution:

```
    ./adjmat2wpairs ROOT_OF_FILENAME OPTION THRESHOLD
```

where ***`ROOT_OF_FILENAME`*** is the common name, without extention, of the files used and produced by the software (*e.g.* *`ROOT_OF_FILENAME`*`.wpairs` and *`ROOT_OF_FILENAME`*`.numid2name`.). If the networks are provided in the format of an edge list where each undirected edge appears only once, the extra parameter ***`OPTION`*** can be specified and set to 1 (setting it to 0 corresponds to the case where the parameter is omitted). The optional parameter ***`THRESHOLD`*** is the minimum weight that edges need to have to be kept in the output `.wpairs` file. If omitted, the default threshold value is 10^{-12}.

3.2 Computing the Tanimoto coefficients

We used the code `compute_tanimoto.cpp` to compute the Tanimoto coefficient for every contiguous edges (*i.e.* edges that share a same node) from the `.wpairs` file. The Tanimoto coefficients are computed following the guidelines given in the Supplementary Information document of Ahn *et al.* [Eq. (1)–(2)] if

[1]The `.wpairs` format is simply the weighted version of the `.pairs` format where each line correspond to an edge in the network (contains the numerical IDs of the two nodes and the weight of the edge).

option 0 is specified, or according to the definition used by Kalinka *et al.* [2] if option 1 is specified. A `.tanimoto` file containing the K Tanimoto coefficients is generated.

3.2.1 Usage

Compilation:

```
    g++ -O3 -o compute_tanimoto compute_tanimoto.cpp
```

Execution:

```
    ./compute_tanimoto ROOT_OF_FILENAME OPTION START_NODE END_NODE
```

with *START_NODE* and *END_NODE* indicating the range of *keystone nodes* that will be covered. CAUTION: the algorithm will cover the range [*START_NODE*,*END_NODE*) and therefore will not use *END_NODE* as a keystone. Also, the numerical ID of the first node must be zero. Finally, if *END_NODE*$>N$, the algorithm will set *END_NODE*$=N$ by itself. If *OPTION*=0, the algorithm uses the definition provided by Ahn *et al.* when computing the Tanimoto Coefficients, or uses the definition used by Kalinka *et al.* if *OPTION*=1.

Once all keystone nodes have been covered, we compiled all Tanimoto coefficient using the UNIX command

```
    cat ROOT_OF_FILENAME.tanimoto_* > ROOT_OF_FILENAME.tanimoto
```

3.2.2 Relation to the original codes

The file `compute_tanomoto.cpp` is the weighted version of the file `calcAndWrite_Jaccards.cpp` written by Jim Bagrow that computes the Jaccard coefficients between contiguous edges from a given edge list.

3.3 Hierarical Clustering

To build the communities, we used the code `cluster_communities.cpp` that merges two contiguous edges in a single community if their Tanimoto coefficient is higher than a given threshold. It produces three files:

1. `.clusters`: each line corresponds to a cluster and contains all edges that are part of if in the format *NODE1,NODE2*;
2. `.cluster_stats`: each line corresponds to a cluster and contains the number of edges and the number of nodes that are part of it [useful to compute the density of communities using (3)];
3. `.density`: this files contains only one line, itself containing the threshold value, the number of edges, the mean density of communities computed with definition #1, the number of edges that are part of communities of size greater than two, the mean density of communities when communities of size two are disregarded (definition #2), the number of communities detected by the algorithm, and the mean densities of the communities computed using definitions #5, #3 and #4, respectively. See Progress Report II of August 19$^{\text{th}}$ for details on the density definitions.

Note that these three files have their name appended with the threshold value for which they have been generated.

3.3.1 Usage

Compilation:

```
    g++ -O3 -o cluster_communities cluster_communities.cpp
```

Execution:
```
    ./cluster_communities ROOT_OF_FILENAME THRESHOLD
```

3.3.2 Relation to the original codes

The file `cluster_communities.cpp` is the original code `clusterJaccsFile.cpp` written by Jim Bagrow, exept for a few minor modifications (inputs/outputs, density definitions added).

3.4 Extracting useful information from the results

Numerical IDs have been given to nodes in order to be easier to deal with in the algorithm. To give back the nodes their original names and to generate the detected communities in a useful format, we have used the code `getting_communities.cpp`. This code uses the "`.clusters`" and "`.cluster_stats`" generated by the community detection algorithm of Ahn *et al.* as well as the "`.numid2name`" generated by `adjmat2wpairs` to generate a file where each line corresponds to an edge in the network and contains the following information:

- the name of the two end nodes of the edge;
- the numerical ID of the community the edge belongs to;
- the density of the community.
- the weighted density od the community (same as the previous one but multiplied by the average weight of edges in it)

3.4.1 Usage

Compilation:
```
    g++ -O3 -o getting_communities getting_communities.cpp
```

Execution:
```
    ./getting_communities ROOT_OF_FILENAME THRESHOLD
```

Caution must be used when writing the threshold value as it must be exaclty identical to the way it is written in the filename. For example, if the filename ends like `clusters_0.35000`, the threshold value given to the program must be `0.35000`.

4 Results

4.1 Conditions network

We have extracted the communities from the conditions network following the steps presented in section 3. Fig. 1 shows the number of communities detected by the algorithm on the conditions network and their average density for different values of threshold. Probably the most striking feature is the fact that the average density curve is not bell shaped.

The reason is simple. Looking at Tab. 1, we see that the full networks are densely connected[2]. Hence for the lower threshold values where the algorithm detects only one community (the whole network), the average density is the density of the network which is in this case almost one.

[2] In the case of the conditions network, the network is almost fully connected!

Choosing the threshold value that maximizes the average density is therefore no longer a proper choice. This is the reason the number of communities detected has also been ploted in Fig. 3 in order to guide the choice of the threshold value.

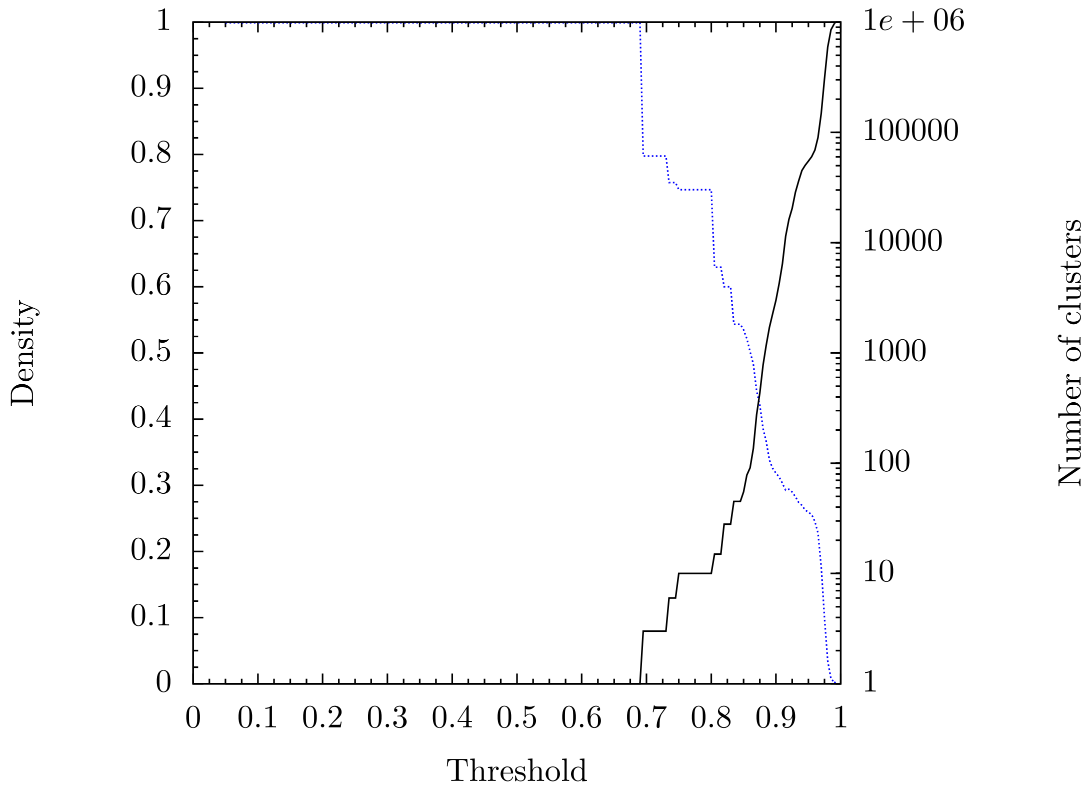

Figure 1: Number of communities detected (solide black line) by the algorithm on the cBc network and their average density (doted blue line) for different values of threshold.

4.2 Genes network

We have extracted the communities from the genes network following the steps presented in section 3. Fig. 2 shows the number of communities detected by the algorithm on the conditions network and their average density for different values of threshold. Again, the average density curve is not bell shaped. See the previous section for the probable cause.

References

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, *Link communities reveal multiscale complexity in networks*, Nature, 466 (2010), pp. 761–764.

[2] A. T. Kalinka and P. Tomancak, *linkcomm: an r package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type*, Bioinformatics, 27 (2011), pp. 2011–2012.

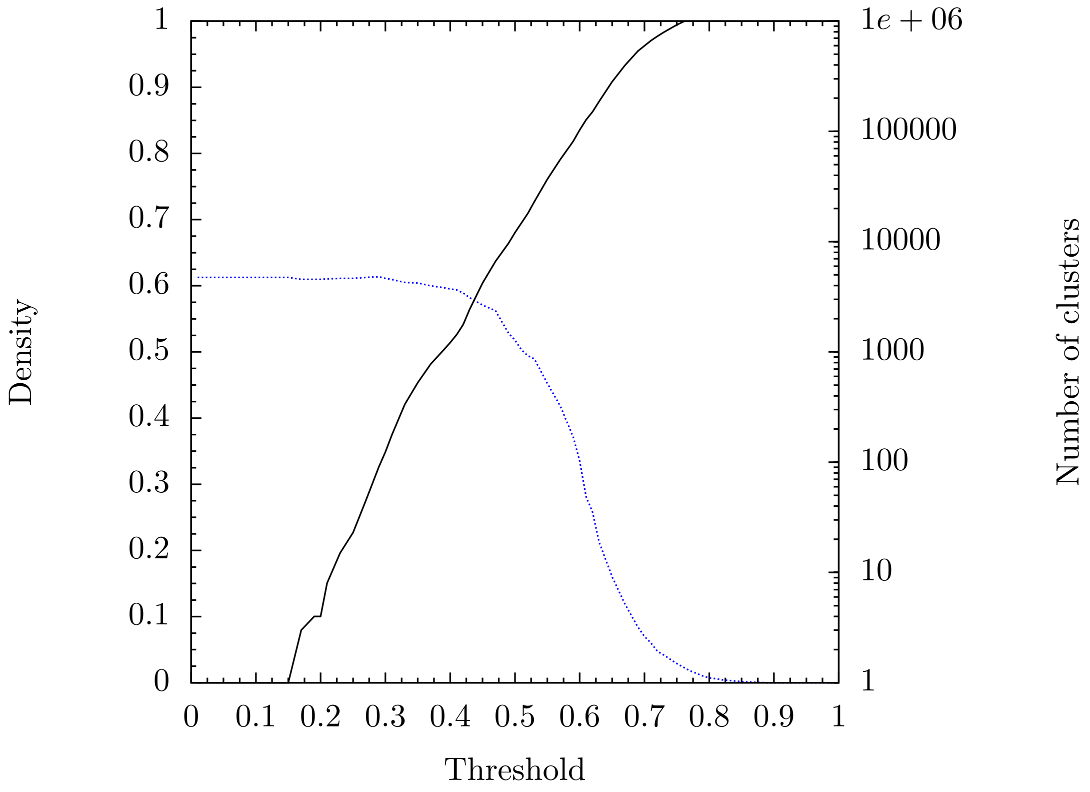

Figure 2: Number of communities detected (solide black line) by the algorithm on the gBg network and their average density (doted blue line) for different values of threshold.