Communities

François Théberge
theberge@ieee.org

August 2019

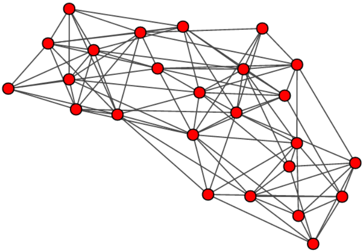
Outline

1. Definitions
2. Spectral bisection
3. Girvan-Newman clustering
4. Benchmarks: Planted partitions, LFR
5. Modularity and algorithmns

Definitions

We start from two fundamental hypothesis [REF: Barabasi, *Network Science*]:

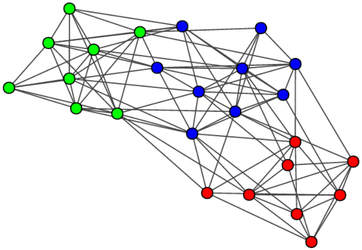
- A network's community structure is uniquely encoded in its wiring diagram.



Definitions

We start from two fundamental hypothesis [REF: Barabasi, *Network Science*]:

- A network's community structure is uniquely encoded in its wiring diagram.
- A community is a locally dense connected subgraph in a network.



Definitions

For a graph $G = (V, E)$, consider the connected subgraph C induced by a subset of nodes $V_C \subset V$ with $i \in V_C$ for some node i.

Define the *internal degree* for node $i \in V_C$ as its degree within subgraph C, denoted $d_i^{int}(C)$.

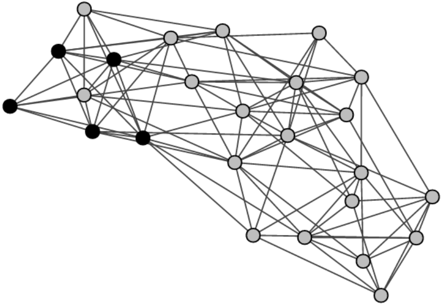
The *external degree* for node i is $d_i^{ext}(C) = d_i - d_i^{int}(C)$, where d_i is the total degree for node i in G.

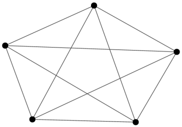
C is a **strong community** if $d_i^{int}(C) > d_i^{ext}(C)$ for each $i \in V_C$.

C is a **weak community** if $\sum_{i \in V_C} d_i^{int}(C) > \sum_{i \in V_C} d_i^{ext}(C)$ for each $i \in V_C$.

Definitions

A **clique** is a fully connected subgraph of G:





Definitions

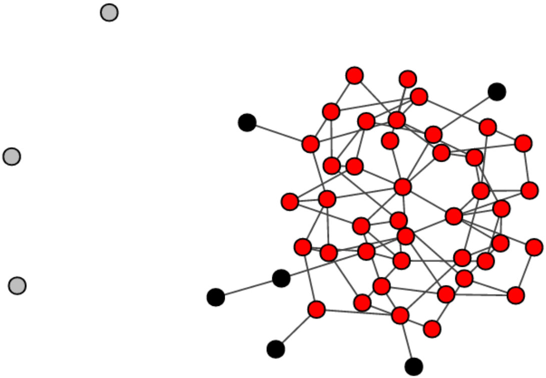
A **k-core** is a maximal connected subgraph of *G* where all nodes have degree at least k.

We can find k-cores by repeatedly deleting all nodes of degree less than k.

A node has *coreness* k if it belongs to a k-core but no (k+1)-core.

Definitions

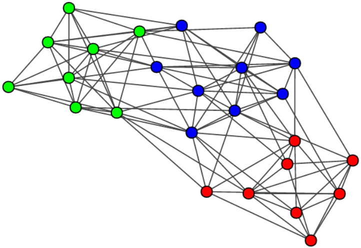
Example of a graph and vertices with coreness 0, 1 and 2:



Definitions

A **clustering** of the graph $G = (V, E)$ of size k is a partition of the nodes $V = V_1 \cup \cdots \cup V_k$ where:

- all $V_i \cap V_j = \emptyset, \; i \neq j$
- for each part (or cluster) V_i, the induced subgraph G_i is connected.



Spectral clustering

Spectral clustering is a vast topic to cover;

I will only illustrate spectral bisection;

For a good tutorial, see:

A Tutorial on Spectral Clustering

Ulrike von Luxburg
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
ulrike.luxburg@tuebingen.mpg.de

This article appears in Statistics and Computing, 17 (4), 2007.
The original publication is available at www.springer.com.

Spectral clustering

Consider unweighted, undirected graph $G = (V, E)$ with adjacency matrix A;

Let D be the matrix with node degrees on the diagonal;

$L = D - A$ is the (un-normalized) Laplacian of G.

There is a strong relation between the community structure in G, and the eigen-decomposition of L. For all $f \in \mathbb{R}^n$:

$$f^t L f = \frac{1}{2} \sum_{i,j} a_{ij}(f_i - f_j)^2$$

so minimizing the above amounts to $f_i \approx f_j$ when $a_{ij} > 0$.

Spectral clustering

We can also consider the ratio-cut bisection $V = S \cup S^c$:

$$Rcut(S, S^c) = \frac{Vol\partial S}{|S|} + \frac{Vol\partial S}{|S^c|}$$

where $Vol(\partial S) = |\{e \ : \ |E \cap S| = |E \cap S^c| = 1\}|$

This can be approximately solved as:

$$\min_{f \in \mathbb{R}^n} f^t L f \ ; \ f \perp 1, \ ||f|| = \sqrt{n}$$

where the solution is the e-vector corresponding to the first non-zero e-value of L.

Spectral clustering

L is symmetric and semi-positive definite, so all eigenvalues are real and non-negative.

L has smallest eigenvalue 0; the multiplicity of this eigenvalue corresponds to the number of connected components in G.

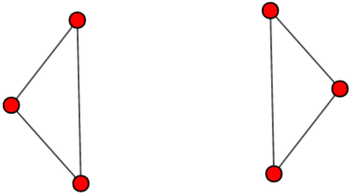
We can therefore order the eigenvalues:

$$0 = \lambda_1 \leq \lambda_2 ... \leq \lambda_n$$

and the corresponding eigenvectors $u_1, ..., u_n$.

Spectral clustering

For example, here is a simple graph with 2 connected components, and Laplacian matrix:



```
[[2, -1, -1, 0, 0, 0],
 [-1, 2, -1, 0, 0, 0],
 [-1, -1, 2, 0, 0, 0],
 [0, 0, 0, 2, -1, -1],
 [0, 0, 0, -1, 2, -1],
 [0, 0, 0, -1, -1, 2]]
```

Spectral clustering

Its eigenvalues are: $(3, 0, 3, 3, 0, 3)$, so $\lambda_1 = \lambda_2 = 0$.

The corresponding eigenvectors are:

```
[[ 0.816, -0.577,  0.31 ,  0.   ,  0.   ,  0.   ],
 [-0.408, -0.577, -0.809,  0.   ,  0.   ,  0.   ],
 [-0.408, -0.577,  0.5  ,  0.   ,  0.   ,  0.   ],
 [ 0.   ,  0.   ,  0.   ,  0.816, -0.577,  0.31 ],
 [ 0.   ,  0.   ,  0.   , -0.408, -0.577, -0.809],
 [ 0.   ,  0.   ,  0.   , -0.408, -0.577,  0.5  ]]
```

Spectral clustering

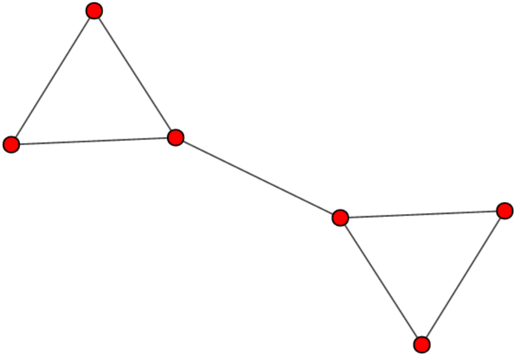
Consider a connected graph G.

In a connected graph, eigenvector u_2 corresponding to $\lambda_2 > 0$ in the *Fiedler* vector.

Spectral bisection is based on the signs of the entries in the Fiedler vector.

Spectral clustering

Consider the connected graph:



Spectral clustering

Its eigenvalues are: $(4.6, 0, 0.4, 3, 3, 3)$, so the Fiedler vector is the third column below:

```
[[-0.657,  0.408,  0.261, -0.577, -0.305,  0.101],
 [ 0.185,  0.408,  0.465,  0.289, -0.425,  0.077],
 [ 0.185,  0.408,  0.465,  0.289,  0.73 , -0.178],
 [ 0.657,  0.408, -0.261, -0.577, -0.305,  0.101],
 [-0.185,  0.408, -0.465,  0.289,  0.318, -0.735],
 [-0.185,  0.408, -0.465,  0.289, -0.014,  0.634]]
```

The two triangles are well identified.

This is called *spectral bisection*.

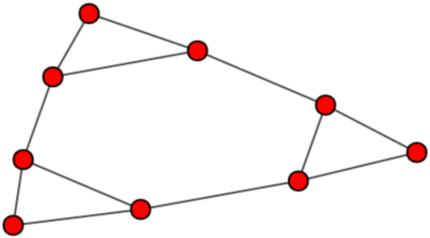
Spectral clustering

With *spectral bisection*, such a process can be applied recursively if more than 2 clusters are present;

This is an example of *divisive* hierarchical clustering.

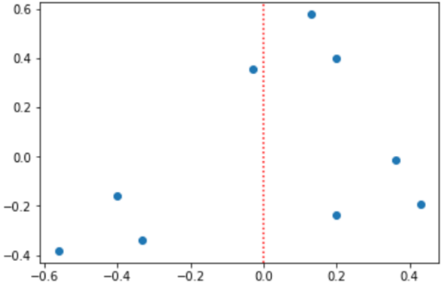
It can however behave badly, splitting some communities.

Consider the following graph:



Spectral clustering

We plot the coordinates of u_2 and u_3:



Taking only u_2, one community is splitted

To get k communities, we use $u_2 \ldots u_k$, and some clustering algorithm like k-means.

Girvan-Newman

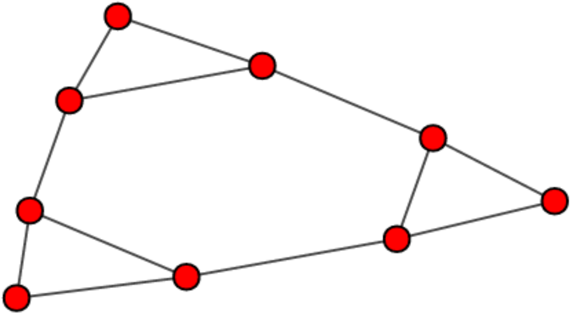
Another divisive, hierarchical clustering algorithm is the Girvan-Newman method:

- Compute the *edge betweenness* for each $e \in E$, and delete the edge with highest value.
- Split the resulting graph into connected components (the clusters) and apply the method recursively;
- This produces a hierarchy of clusterings which we can represent as a *dendrogram*.

The best partition is selected with some criterion like modularity (covered later).

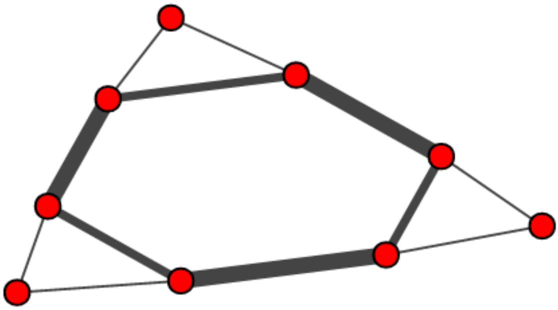
Girvan-Newman

Consider again the following graph where triangles are consecutive nodes resp. (0,1,2), (3,4,5), (6,7,8);



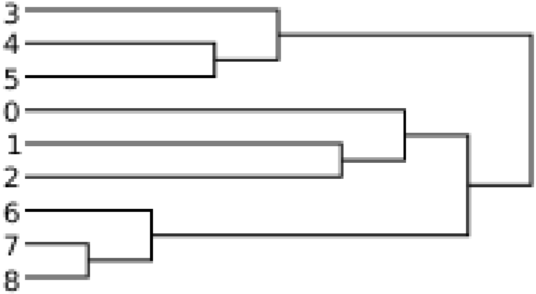
Girvan-Newman

We compute edge betweenness:



Girvan-Newman

With the GN algorithm, we get the hierarchy:



Girvan-Newman

We select the number of clusters, or find optimal modularity:

```
cl = gn.as_clustering(n=2)
cl.membership
```

```
[0, 0, 0, 1, 1, 1, 0, 0, 0]
```

```
cl = gn.as_clustering(n=3)
cl.membership
```

```
[0, 0, 0, 1, 1, 1, 2, 2, 2]
```

```
## optimal modularity
cl = gn.as_clustering()
cl.membership
```

```
[0, 0, 0, 1, 1, 1, 2, 2, 2]
```

Girvan-Newman

One issue with this algorithm is its complexity

Run time is $O(m^2 n)$

For very sparse graphs, this is still high at $O(n^3)$

We will see other algorithms can run in $O(m)$ or $O(n \log n)$ time

Benchmarks

Why benchmark models with communities?

- good way to test and compare algorithm
- control the noise level, the community sizes, etc.
- ground-truth is rarely available with real graphs
- when available, ground-truth may not coincide with the fundamental hypothesis

Planted partitions

In this model, we fix the number of nodes n and the number of communities k;

For the communities, we either:

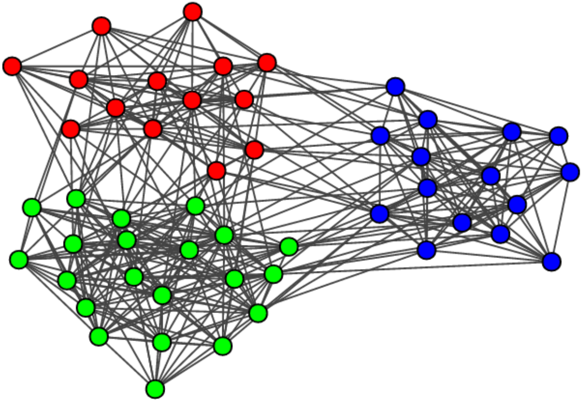
- divide the nodes equally between each community, or
- assign each node independently to community i with probability p_i with $\sum_{i=1}^{k} p_i = 1$.

For each pair of nodes resp. in communities i and j, add an edge with probability $P(i,j)$.

A special case is to assign all $P(i,i) = p_{in}$ and all $P(i,j) = p_{out}, \; i \neq j$.

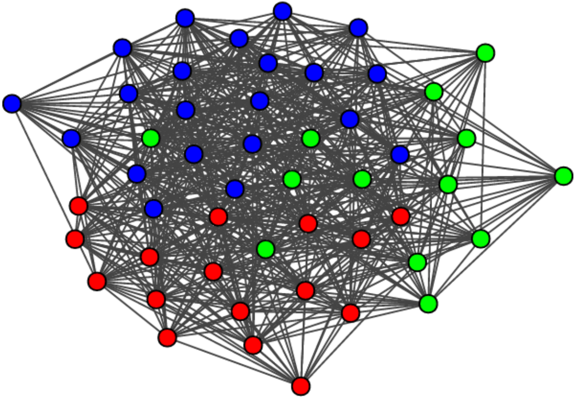
Planted partitions

PP benchmark with $p_{in} = .7$ and $p_{out} = .1$:



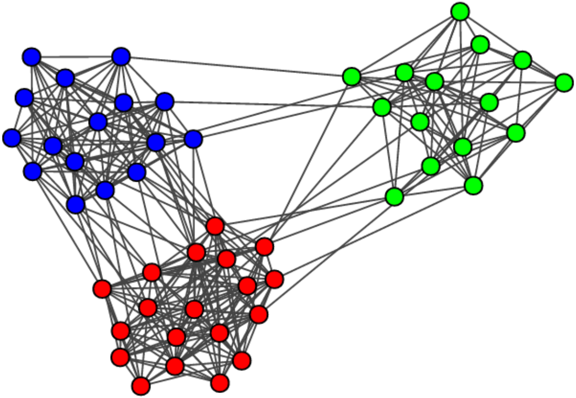
Planted partitions

PP benchmark with $p_{in} = .7$ and $p_{out} = .4$:



Planted partitions

PP benchmark with varying $P(i,j)$:



LFR

Fix the number of nodes n.
The Lancichinetti-Fortunato-Radicchi (LFR) benchmark has 3 main parameters:

- γ_1: community size follow a power law distribution with $p_n \propto n^{-\gamma_1}$; recommended values are $2 \leq \gamma_1 \leq 3$.
- γ_2: node degree follow a power law distribution with $p_k \propto k^{-\gamma_2}$; recommended values are $1 \leq \gamma_2 \leq 2$.
- $0 \leq \mu \leq 1$: for each node, this is the expected proportion of edges linking to other communities, while $(1 - \mu)$ is the proportion within its own community.

μ is called the noise level, or mixing parameter.

LFR

Each node is assigned to a single community

A variant exists which allows for overlapping communities

Extra parameters may be supplied to bound the degree distribution (average and maximum degree) and the community sizes (minimum and maximum).

The algorithms starts from the configuration model, and re-wires to approximate the target distributions

Other models such as BA can be used in the initial phase

C code available on GitHub

LFR

LFR-Benchmark_UndirWeightOvp

Extended version of the Lancichinetti-Fortunato-Radicchi Benchmark for Undirected Weighted Overlapping networks to evaluate clustering algorithms

Description

This program is an implementation of the algorithm described in the paper "Directed, weighted and overlapping benchmark graphs for community detection algorithms", written by Andrea Lancichinetti and Santo Fortunato. In particular, this program is to produce undirected weighted networks with overlapping nodes.
Each feedback is very welcome. If you have found a bug or have problems, or want to give advises, please contact us:

andrea.lancichinetti@isi.it
fortunato@isi.it

Turin, 29 October 2009

Original sources:

- Benchmark graphs for testing community detection algorithms
- Benchmarks

Reference: A. Lancichinetti, S. Fortunato, and F. Radicchi.(2008) Benchmark graphs for testing community detection algorithms. Physical Review E, 78.

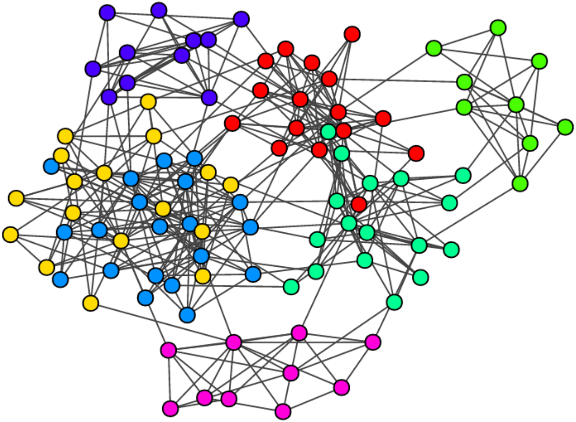
LFR

The benchmark code produces 3 files:

- a file with the list of edges with nodes labelled 1 up
- a file with the list of the nodes and their community membership, communities are also labelled 1 up
- a file with statistics such as degree distribution, community size distribution and mixing parameter.

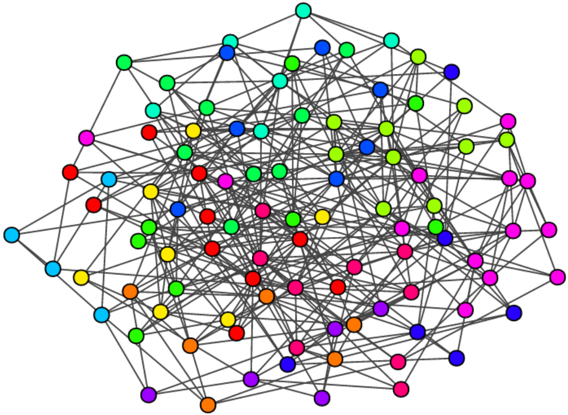
LFR

LFR graph with $n = 100$ and $\mu = .15$, with ground-truth communities identified:



LFR

LFR graph with $n = 100$ and $\mu = .55$, with ground-truth communities identified:



LFR

LFR's scalability is somewhat limited; a few scalable benchmarks are:

RMAT, which generated graphs with power law degree distribution; this is used in Graph-500.

BTER (Block Two-level ER), with power law degree distribution as well as community structure.

SBM (Stochastic Block Model), which also generates graphs with community structure. Its simplest definition is a variation of the planted partitions model.

Modularity

Barabasi's third fundamental hypothesis:

- Randomly wired networks lack an inherent community structure.

Modularity uses random wiring as a *null model* to quantify the community structure for some graph partition.

Modularity

Consider the un-directed graph $G = (V, E)$

Let $|V| = n$, $|E| = m$, d_i the degree for node i.

Let $a_{ij} = a_{ji} = 1$ if and only if $(i, j) \in E$, and 0 otherwise, and $a_{ii} = 2$ if and only if $(i, i) \in E$.

Under random wiring on the vertices,

$$p_{ij} = \frac{d_i d_j}{2m}$$

is the expected number of edge(s) between nodes i and j (in practice, the probability).

This is equivalent to the Chung-Lu model up to a factor of 2 since $\sum_{i,j} a_{ij} = 2m$.

Modularity

Let $V = C_1 \cup \cdots C_k$, a partition of the graph into k clusters.

For some cluster C_l, define:

$$q_{C_l} = \frac{1}{2m} \sum_{i,j \in C_l} (a_{ij} - p_{ij})$$

which can be written as:

$$q_{C_l} = \frac{\sum_{i,j \in C_l} a_{ij}}{2m} - \frac{\sum_{i,j \in C_l} d_i d_j}{(2m)^2}$$

Modularity

Let

$$e(C_l) = |\{e \in E\ ;\ e \subseteq C_l\}|$$

and

$$Vol(C_l) = \sum_{i \in C_l} d_i$$

We get:

$$q_{C_l} = \frac{e(C_l)}{m} - \left(\frac{Vol(C_l)}{2m}\right)^2$$

Modularity

Modularity is defined as:

$$q = \sum_{l=1}^{k} \frac{e(C_l)}{m} - \left(\frac{Vol(C_l)}{2m}\right)^2$$

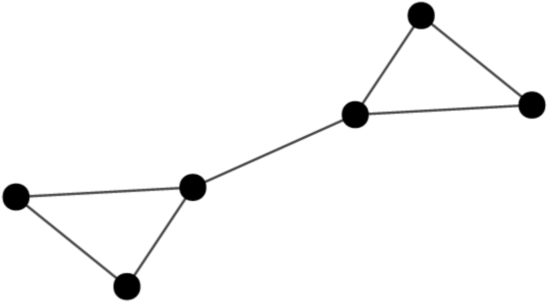
We refer to the first term above as the *edge contribution*, and the second one as the *degree tax*.

Modularity $q^*(G)$ of a graph is sometimes defined as the maximum value taken by the above over all possible partitions.

Modularity - small example

Single community:

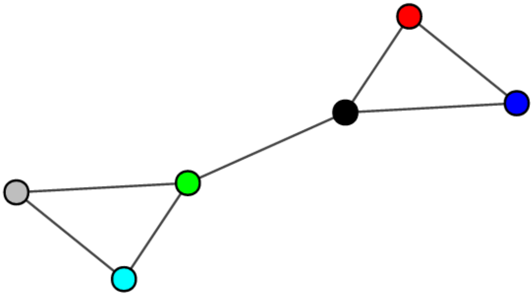
```
q = 0.0
```



Modularity - small example

Singletons:

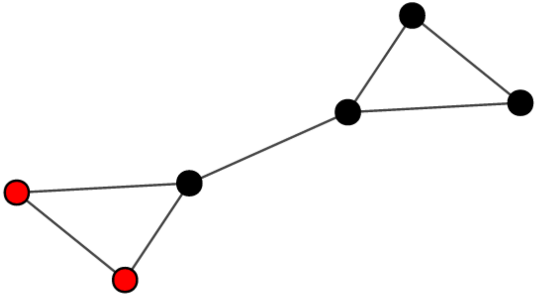
```
q = -0.173469387755102
```



Modularity - small example

Sub-optimal:

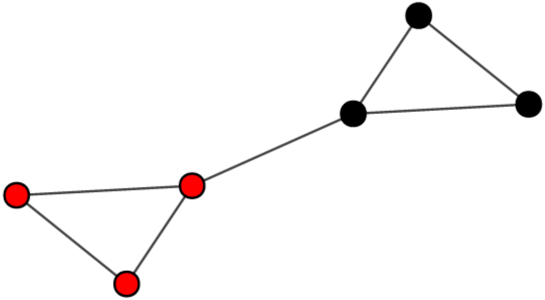
```
q = 0.12244897959183669
```



Modularity - small example

Optimal:

```
q = 0.3571428571428571
```



Modularity

Barabasi's fourth fundamental hypothesis:

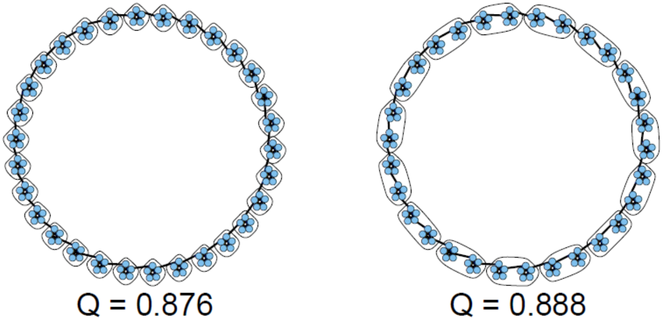
- For a given network, the partition with maximum modularity corresponds to the optimal community structure.

There are however some known issues with modularity ...

"Optimal" may not always translate to "intuitive".

Resolution Issue

Modularity-based algorithm suffer from the resolution limit issue (ring of cliques example):



Resolution Issue

Consider a ring of l cliques of size m, with $n = l \cdot m$

Grouping pairs of adjacent cliques yields a higher modularity than the natural choice of each clique forming its own cluster when $m(m-1) < l - 2$.

As we will illustrate, some modularity-based algorithms thus tend to group communities.

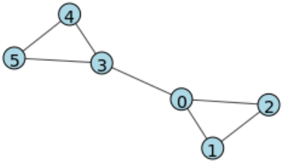
CNM

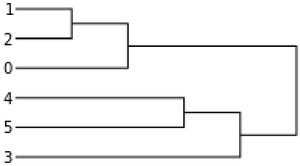
The CNM algorithm (Clauset, Newman, Moore), also known as the Fast Greedy algorithm.

- Start with each vertex in its own cluster
- Choose the pair of clusters that improves modularity the most, if any, and merge them.
- Stop when no merge can improve modularity.
- Complexity: $O(n^2)$, less for sparse graphs.

CNM

Toy example:





Louvain

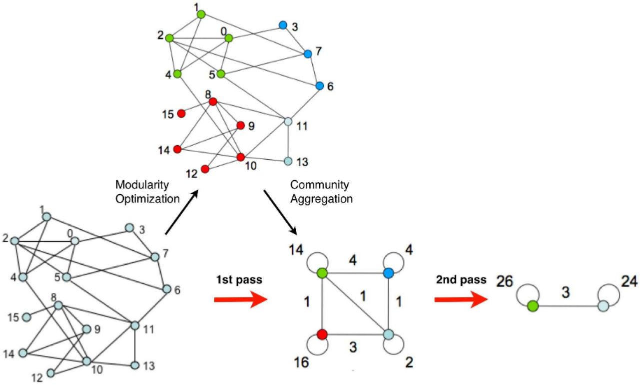
Also known as the Multilevel algorithm.

- Start with each vertex in its own cluster
- Cycle through each vertex, moving it to the community of its neighbour for which the modularity is increased the most (if any).
- Repeat the above until no move remains.
- Collapse each community into a single node and re-run the above steps; this constitutes a new level.
- Stop when the graph collapses to a single node, or when no move happens in the last level.
- Complexity: $O(n \log n)$.

Louvain

Ref: Blondel *et al.*, arXiv:0803.0476v2

Fast unfolding of communities in large networks 3



Infomap

Ref: Rosval and Bergstrom,
`www.pnas.org/cgi/doi/10.1073/pnas.0706851105`

- Infomap is based on information theory through probability flow of random walks and compression algorithm.
- Given G and a partition, encode random walks as efficiently as possible.
- Take advantage of the fact that random walks tend to stay longer in the same community.
- Optimize the *map equation*: average number of bits to describe walk between communities + average number of bits to describe walk within community.
- Its complexity is $O(n \log n)$.

Label propagation

Ref: Raghavan *et al.*, Near linear time algorithm to detect community structures in large-scale networks.

- Start with each vertex having its own cluster label
- Cycle through each vertex, with each vertex taking the most popular label of its neighbours (break ties at random)
- The algorithm stops when each vertex has the same cluster label as the most frequent label in its neighbourhood
- Complexity: $O(m)$
- Note: this algorithm is fast, but does not always converge to a solution.

Other algorithms

- **WalkTrap** is a hierarchical algorithm based on short distance random walks. Its complexity is $O(n^2 \log n)$.
- **Leading eigenvector** is based on the spectral decomposition of the modularity matrix. Its complexity is $O(n(n+m))$ for each bi-partition.

The **Louvain** and **Infomap** algorithms are currently considered state of the art.

Other algorithms

- **WalkTrap** is a hierarchical algorithm based on short distance random walks. Its complexity is $O(n^2 \log n)$.
- **Leading eigenvector** is based on the spectral decomposition of the modularity matrix. Its complexity is $O(n(n+m))$ for each bi-partition.

The **Louvain** and **Infomap** algorithms are currently considered state of the art.

More on this later...

Communities

Notebook #4