STATISTICAL ANALYSIS AND DOCUMENT MINING

CM11: Community detection in networks – Part 2

Pedro L. C. Rodrigues

☞ ○ Community detection
  - Spectral community detection
  - Louvain algorithm
- ○ Other topics in graph analysis
- ○ About the exam 😬
- ○ Final remarks

We are going to study four different methods for community detection in graphs.

Each one of them is based on a different **network measure** as defined previously.

- Hierarchical clustering
  - Agglomerative **[similarity / dissimilarity]**
  - Divisive **[edge betweenness]**
- Spectral community detection **[modularity]**
- Louvain algorithm **[modularity]**

Spectral community detection

Some of the main issues with the previous hierarchical methods are that:

- They follow a **greedy** approach, possibly missing the global optimum
- The target quantity being minimized is not naturally related to clustering

A very popular and successful approach is to use the **modularity** as objective function

Recall that modularity measures a “statistically surprising arrangement of edges”

In other words, modularity is (up to a multiplicative constant)

$$Q = \boxed{\text{ACTUAL NUMBER OF EDGES FALLING WITHIN GROUPS}} - \boxed{\text{EXPECTED NUMBER OF EDGES FALLING WITHIN GROUPS FOR A RANDOM NETWORK}}$$

depends of the structure of the network and to which group how we assign assign each node

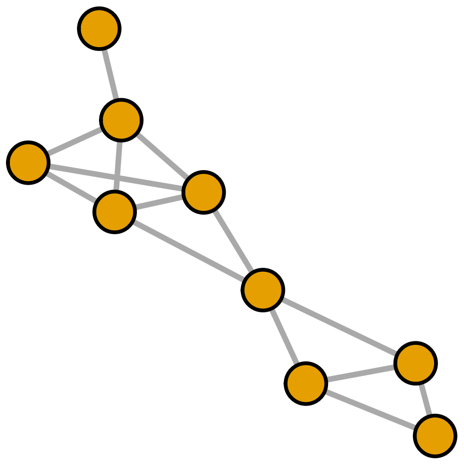
depends only of the structure of the network

Spectral community detection

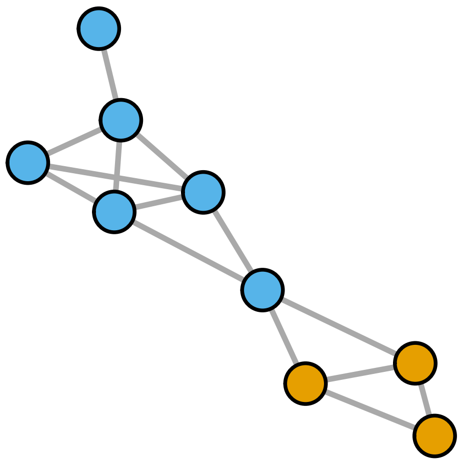
$$Q = \frac{1}{2m}\sum_{ij}\underbrace{\left(A_{ij} - \frac{k_i k_j}{2m}\right)}_{\text{network structure}} \delta_{g_i g_j}$$

node assignment

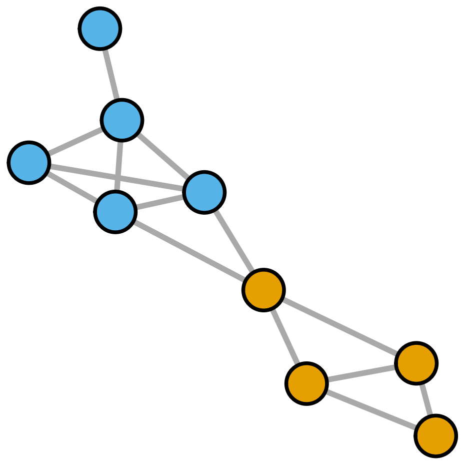
Q = 0



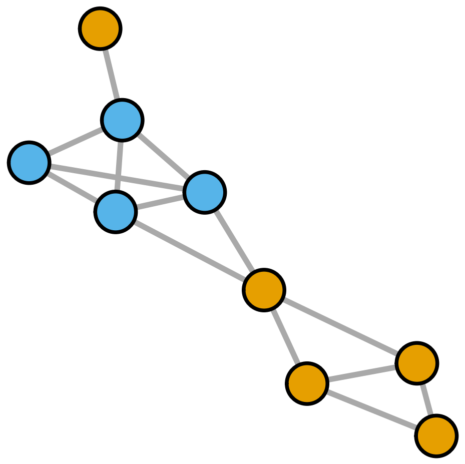
Q = 0.265



Q = 0.347



Q = 0.283



Spectral community detection

Some of the main issues with the previous hierarchical methods are that:

- They follow a **greedy** approach, possibly missing the global optimum
- The target quantity being minimized is not naturally related to clustering

A very popular and successful approach is to use the **modularity** as objective function

Supposing we have two clusters, we can assign cluster $s_i \in \{-1, +1\}$ to each node *i*

$$\delta_{g_i g_j} = \frac{1}{2}(s_i s_j + 1) \longrightarrow Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right) \times \frac{1}{2}(s_i s_j + 1)$$

and

$$Q = \frac{1}{4m}\mathbf{s}^\top \mathbf{B}\mathbf{s} \quad \text{with} \quad B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

Spectral community detection

Our goal is to maximize the modularity Q, so we can see our problem as that of finding the values of the vector ***s*** for a given matrix ***B*** encoding the structure of the graph.

Since **B** is a symmetric matrix that (we assume) has *n* different eigenvectors, we know that

$$\mathbf{s} = \sum_i (\mathbf{u}_i^\top \mathbf{s})\mathbf{u}_i$$

where the $\mathbf{u}_i$ are eigenvectors of the modularity matrix with eigenvalues λ_i and then

$$Q = \frac{1}{4m}\left(\sum_i (\mathbf{u}_i^\top \mathbf{s})\mathbf{u}_i\right)^\top \mathbf{B}\left(\sum_j (\mathbf{u}_j^\top \mathbf{s})\mathbf{u}_j\right) = \frac{1}{4m}\sum_i (\mathbf{u}_i^\top \mathbf{s})^2 \lambda_i$$

Assuming $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ we see that we want to choose **s** to **concentrate as much as weight as possible** in the terms of the sum involving the largest (most positive) eigenvalues.

We could simply choose **s** to be proportional to $\mathbf{u}_1$ but the vector is in $\{-1, +1\}^n$ 🧐

Spectral community detection

Our goal is to maximize the modularity Q, so we can see our problem as that of finding the values of the vector ***s*** for a given matrix ***B*** encoding the structure of the graph.

- One reason why this maximization problem is a **difficult** one is that the s_i can take only a discrete set of values—each s_i is restricted to the values +1 and -1. If, on the other hand, the s_i were continuous-valued variables, allowed to take any real values, the problem would be much easier—we could just **differentiate** to the find the maximum.

We follow a **relaxation** approach: we consider that the s_i can have values in the real line and assume that their values will give as us good clues as to how to split the graph into
-1 and +1 classes.

- If ***s*** is to be seen as a n-dimensional vector, the requirement of its values being always either +1 or -1 means that such vector always points to one of the corners of an n-dimensional hypercube. When we relax the discrete-value constraint, we should impose that the sum of the squares of the components of ***s*** is n, i.e.

$$\sum_i s_i^2 = n$$

Spectral community detection

So now community detection problem becomes an optimization problem under constraints

$$\underset{\mathbf{s}\in\{-1,+1\}^n}{\text{maximize}} \ \frac{1}{4m}\mathbf{s}^\top \mathbf{B}\mathbf{s} \quad \Rightarrow \quad \underset{\|\mathbf{s}\|^2=n}{\text{maximize}} \ \mathbf{s}^\top \mathbf{B}\mathbf{s}$$

with an augmented Lagrangian given by

$$\mathcal{L}(\mathbf{s},\lambda) = \frac{1}{4m}\mathbf{s}^\top \mathbf{B}\mathbf{s} + \lambda(n - \mathbf{s}^\top \mathbf{s})$$

Taking the derivatives we see that the vector that maximizes the loss function is...

$$\mathbf{B}\mathbf{s} = \lambda\mathbf{s}$$

In other words, the optimal *s* is the **leading eigenvector** of the modularity matrix **B**.

Spectral community detection

Question How to relate this vector of real values to the actual **discrete-valued** vector?

Take the signs of the elements of ***s*** and assign the communities accordingly !

➢ **Attention**: this will not necessarily give the global solution to the modularity maximization problem, but it is a good heuristic. It is interesting to note that the **magnitudes** of the elements of the vector have an interpretation: these values indicate how firmly we assign each vertex to one community or the other. Those with large values are strong members of the communities, whereas smaller values are more ambivalent members.

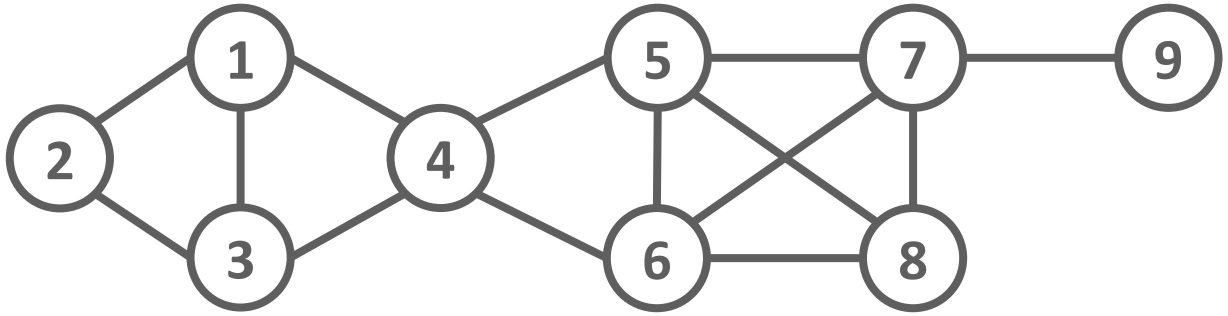
The algorithm (Spectral community detection)

1) Build the modularity matrix **B**
2) Calculate the eigenvector of this matrix corresponding to the largest eigenvalue
3) Assign vertices to communities according to the signs of their elements, i.e. positive signs in one group and negative signs in the other group

Newman (2013) – "Spectral methods for network community detection and graph partitioning"

Spectral community detection

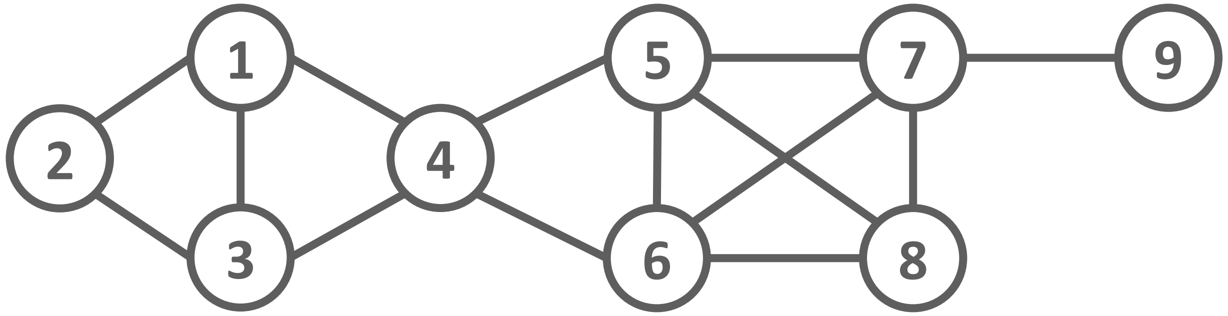
Example: in the graph below, the leading eigenvector is



+0.4382574
+0.3792501
+0.4382574
+0.1738957
-0.2853926
-0.2853926
-0.3754298
-0.3444000
-0.1390457

Spectral community detection

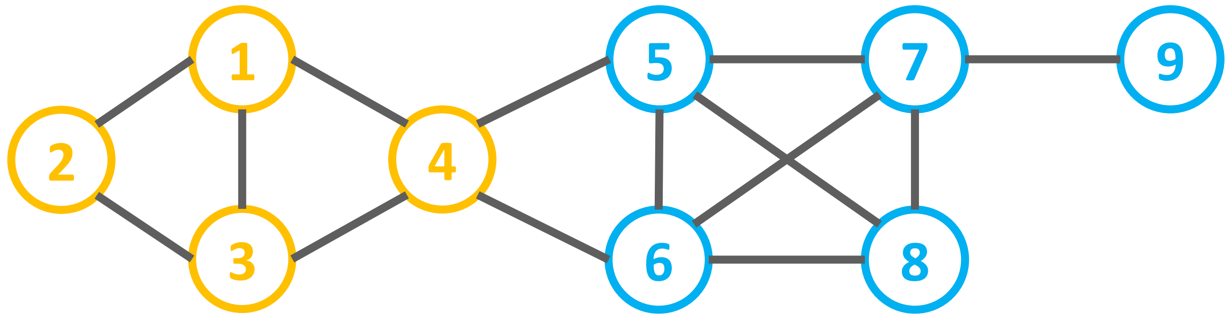
Example: in the graph below, the leading eigenvector is



+0.4382574	+1
+0.3792501	+1
+0.4382574	+1
+0.1738957	+1
-0.2853926	-1
-0.2853926	-1
-0.3754298	-1
-0.3444000	-1
-0.1390457	-1

Spectral community detection

Example: in the graph below, the leading eigenvector is



	+0.4382574	+1
	+0.3792501	+1
	+0.4382574	+1
Ambivalent vertex $\longrightarrow$	+0.1738957	+1
	-0.2853926	-1
	-0.2853926	-1
	-0.3754298	-1
	-0.3444000	-1
	-0.1390457	-1

Spectral community detection

It is interesting to note that the modularity matrix **B** defined as follows

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

has two important properties:

$$\sum_i B_{ij} = \sum_i A_{ij} - \frac{k_j}{2m} \sum_i k_i = k_j - \frac{k_j}{2m} 2m = 0$$

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i}{2m} \sum_j k_j = k_i - \frac{k_i}{2m} 2m = 0$$

Which means that such matrix **always** have a zero eigenvalue associated to vector **s = 1**$_N$

Spectral community detection

It is interesting to note that the modularity matrix **B** defined as follows

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

Remarks:

- Matrix **B** always have a zero eigenvalue associated to vector **s = 1**$_N$
- Matrix **B** is symmetric but not necessarily positive definite. Therefore, it always has real eigenvalues (which can be positive or negative) and real eigenvectors.

Conclusion:

If the largest eigenvalue of **B** is zero, it means that there is no way of splitting the graph into communities with a higher modularity than a random graph (**stop criterium**)

Question Is this the same thing as *spectral clustering*? NO!

A very well common procedure for clustering a set of data points is to

1. Embed the data points into an Euclidean space based on the Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & \mathbf{x}^\top \mathbf{L} \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x}^\top \mathbf{x} = 1 \end{aligned}$$

2. Run k-means to find the clusters

This is called spectral clustering and is different from the method presented in previous slides.

↳ Different matrix, ad-hoc normalization, two-step procedure, no stopping criteria, Euclidean approach

Spectral community detection

The spectral approach to community detection is a **relaxation heuristic** to the solution of the modularity maximization problem. Therefore, the results are not necessarily the global optimum (although it works really well in practice!). It is possible to **fine-tune** the output of the algorithm using a second simple algorithm for community detection:

The algorithm (Fine-tuning of the spectral algorithm)

1) Split the graph using the leading eigenvector of **B**
2) For each node in the graph, calculate how much the modularity would change if it was moved from one group to another group.
3) Choose among these changes the node for which the modularity increases the most. Change the group of that node.
4) Repeat **Steps 2** and **3** without moving the nodes that have already been moved.
5) When all nodes have been moved exactly once, finish the current round.
6) Go through all of the changes in positions in the round that has just ended and choose the configuration with the largest modularity. Go back to **Step 1** using this split of the graph as starting point for another round (i.e. **Steps 2-6**)
7) Stop the algorithm when the modularity no longer improves. Use the division with the largest modularity of the last round.

Spectral community detection

Remarks:

- For a graph with **n** vertices and **m** edges, the node-moving strategy has time complexity of **O(n(m+n))** or simply **O(n^2)** for sparse matrices (i.e. with few edges)

- If done naively, the time complexity of the spectral approach would be the same as that for the node-moving approach. However, the matrix operations of this procedure can be simplified by taking into account that a vertex usually has only a few neighbors. The actual complexity of the algorithm is actually **O(n log(n))** for sparse graphs.

- So far we have only considered graphs being split into two communities. Extending the spectral approach to more communities can be done via **bissection**: we first find two main communities and then re-run the algorithm on each of these communities to find two other communities, etc.

- The node-moving approach extends more naturally to more groups, but is rather slow.

We are going to study four different methods for community detection in graphs.

Each one of them is based on a different **network measure** as defined previously.

- Hierarchical clustering
  - Agglomerative **[similarity / dissimilarity]**
  - Divisive **[edge betweenness]**
- Spectral community detection **[modularity]**
- Louvain algorithm **[modularity]**

Louvain algorithm

The **Louvain** algorithm is inspired by the node-moving approach as presented before, but manages to have a time complexity equivalent to that of the spectral approach.

In other words, it has the advantage of extending naturally to multiple groups (as in the node-moving procedure) while having a time complexity equivalent to that of the spectral approach, i.e. **O(n log(n))**

Because of its numerical advantages, the Louvain method is one of the most important and used algorithms for community detection nowadays. In the paper where they proposed the method, the authors of the algorithm claimed that “Contrary to all the other community detection algorithms, the network size limits that we are facing with our algorithm are due to **limited storage capacity rather than limited computation time**: identifying communities in a 118 million nodes network took only 152 minutes.”

Blondel et al. (2008) – “Fast unfolding of communities in large networks”

Louvain algorithm

Assume we have a graph with ***n*** nodes.

The **Louvain** algorithm consists of **two phases** that are repeated iteratively:

Phase 1: First, we assign a different community to each node of the network. In this initial partition there are as many communities as there are nodes.

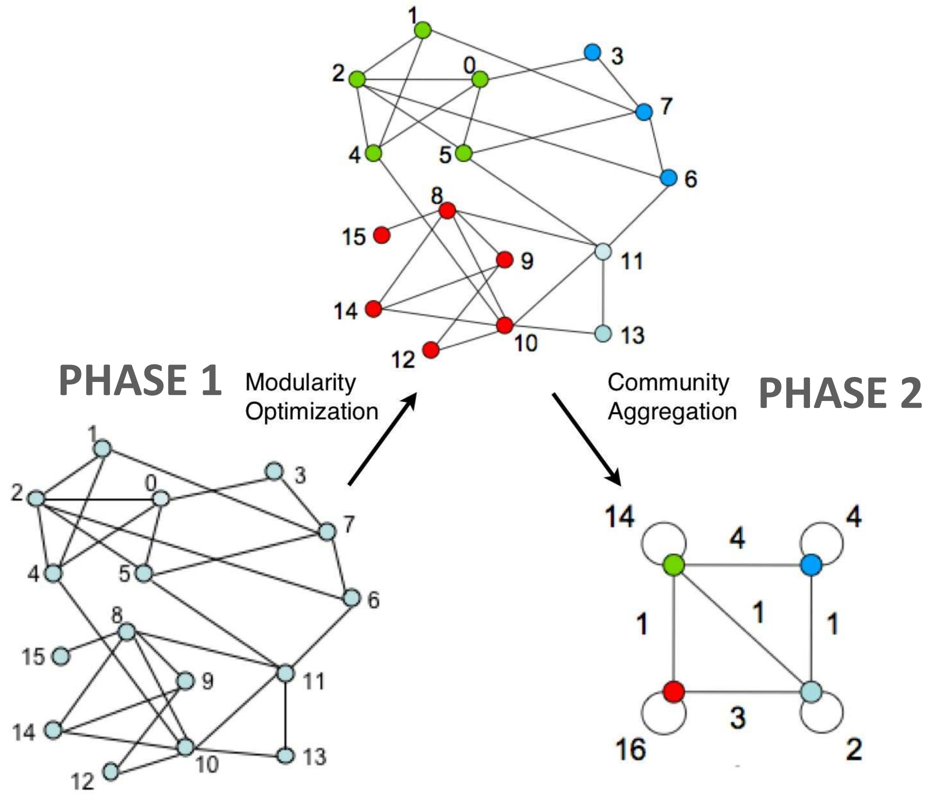
Then, for each node ***i*** we consider the neighbours ***j*** of ***i*** and we evaluate the gain of modularity that would take place by removing ***i*** from its community and by placing it in the community of ***j***. The node ***i*** is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. If no positive gain is possible, ***i*** stays in its original community.

This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved. The first phase stops when a local maxima of the modularity is attained, i.e., when no individual move can improve the modularity.

AGGLOMERATIVE STYLE

Louvain algorithm

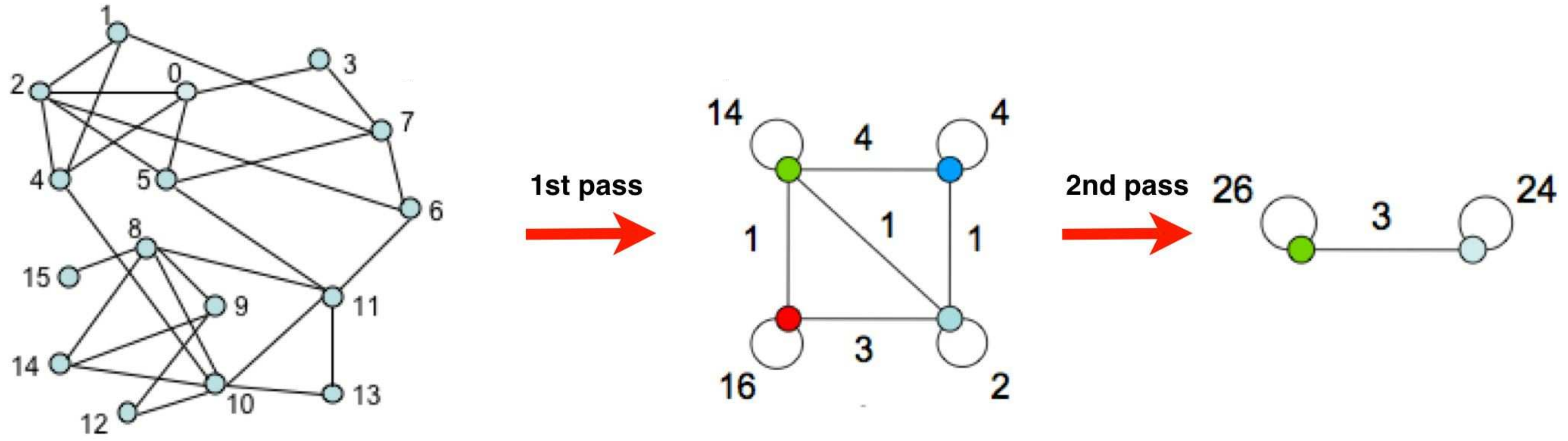
Phase 2: The second phase of the algorithm consists in building a new network whose nodes are now the communities found during the first phase. To do so, the weights of the links between the new nodes are given by the sum of the weight of the links between nodes in the corresponding two communities. Links between nodes of the same community lead to self-loops for this community in the new network.



Louvain algorithm

Once the second phase is **completed**, it is possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate.

Calling a “pass” the combination of these two phases, we see that by construction the **number of meta-communities decreases at each pass**, and as a consequence most of the computing time is used in the first pass. The passes are iterated until there are no more changes and a maximum of modularity is attained.



Louvain algorithm

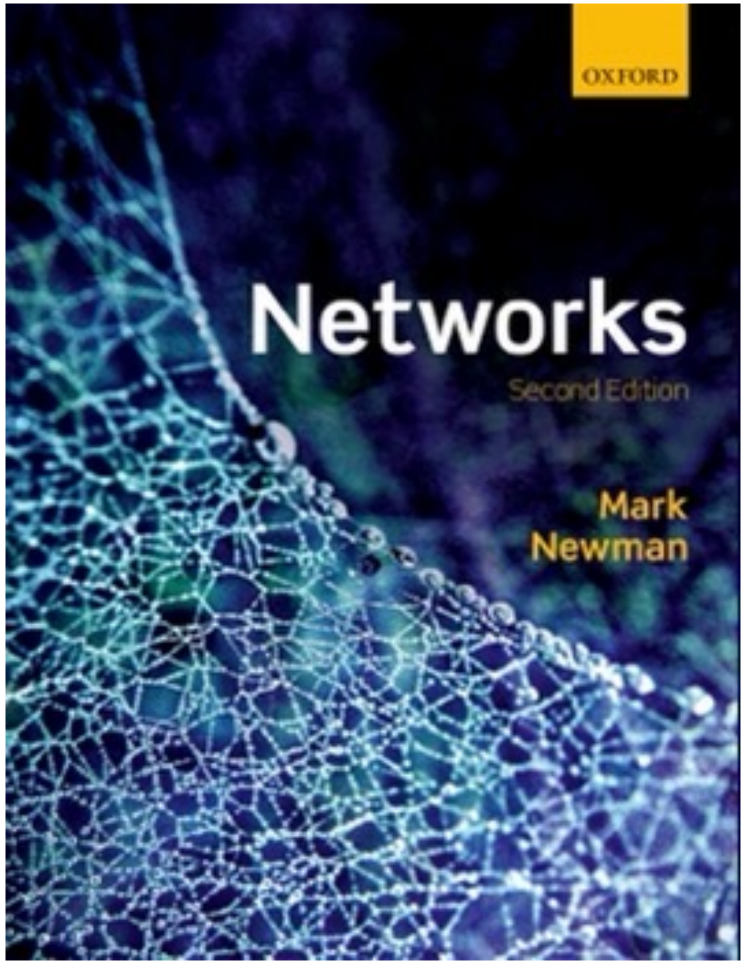
The Louvain algorithm has several **advantages**:

1. Its steps are intuitive and easy to implement, and the outcome is unsupervised.
2. The algorithm is extremely fast, i.e., computer simulations on large ad-hoc modular networks suggest that its complexity is linear on typical and sparse data.
3. The algorithm provides a decomposition of the network into communities for different levels of organization. This means that the intermediate solutions found by the algorithm may also be meaningful and that the uncovered hierarchical structure may allow the end-user to zoom in the network to observe its structure with the desired resolution.

- Community detection
  - Spectral community detection
  - Louvain algorithm
- ☞ Other topics in graph analysis
- About the exam 😬
- Final remarks

Other topics in graph and network theory

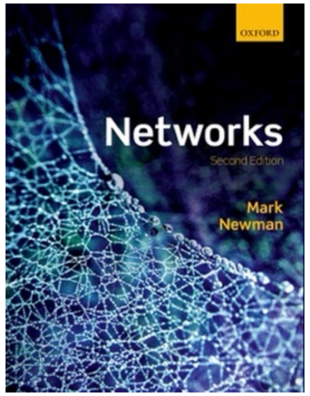
The main reference for what we have seen in the past three CMs is



We covered most of Chapters 6, 7, and 14

Other topics in graph and network theory

But there are many other interesting topics to learn, such as



- Software and algorithms for network analysis and visualization
- Network statistics and measurement error in communications
- Network models and their statistical inference
- Percolation and network resilience
- Epidemics on networks
- Etc.

There has also been a lot of interest in the study of **graph methods** for machine learning

↳ **Graph signal processing:** Fourier basis of a graph, filters over graphs, sub-sampling
Ortega et al. "Graph Signal Processing: Overview, Challenges and Applications" – **arxiv.org/pdf/1712.00468.pdf**

↳ **Geometric deep learning:** Neural networks over graphs
Bronstein et al. "Geometric deep learning: going beyond Euclidean data" – **arxiv.org/pdf/1611.08097.pdf**

Don't hesitate to come talk with me (or write me) if you want to know more on this…

- Community detection
  - Spectral community detection
  - Louvain algorithm
- Other topics in graph analysis
- About the exam 😬
- Final remarks

ENSIMAG

Exam is **practical** and you have **three hours**.

The exam has four obligatory parts.

M1AM

Exam is **written** and you have **two hours**.

The exam has one obligatory part, you choose two out of the other three.

The first part of the exam consists of multiple choice questions.
↳ You should check the SADM course website for several examples

ENSIMAG: The three other parts will use R but will also have some conceptual questions.
↳ No, you won't have to demonstrate anything

M1AM: The three other parts are written and conceptual questions.

You are expected to demonstrate stuff related to multiple linear regression and PCA

Part four of the exam will ask questions related to the article by Mark Newman *"Modularity and community structure in networks"* (2006) available in SADM website

You should check the list of relevant keywords in the SADM website

- Community detection
  - Spectral community detection
  - Louvain algorithm
- Other topics in graph analysis
- About the exam 😬
- ☞ Final remarks

Final remarks

If you’ve enjoyed this course, I would suggest you to take a look on these ones too

↳ **L'apprentissage face à la malédiction de la grande dimension:** Stéphane Mallat’s course at the Collège de France on some topics that we’ve seen and many others. Link [here]

↳ **CS229 at Stanford:** Andrew Ng’s course on Machine Learning at Stanford. Link [here]

↳ **CS4780 at Cornell:** Kilian Weinberger’s course on Machine Learning at Cornell University. Link [here]

In the MSIAM master you also have some relevant courses.

Don’t hesitate to come talk with me (or write me) if you want to know more on this…

Thanks for you attention and presence! 🫶