

Community detection in networks

R. Lambiotte
Department of Mathematics
University of Namur, Belgium

Community detection in networks

1. Introduction
2. Why modularity?
3. Graph partitioning
4. Newman-Girvan modularity
5. Modularity and dynamics
6. Recent trends

“I think the next century will be the century of complexity.”

Stephen Hawking

com.plex

[adj., v. kuh m-pleks, kom-pleks; n. kom-pleks]

- 1) composed of many interconnected parts; compound; composite: a complex highway system
- 2) characterized by a very complicated or involved arrangement of parts, units, etc.: complex machinery
- 3) so complicated or intricate as to be hard to understand or deal with: a complex problem

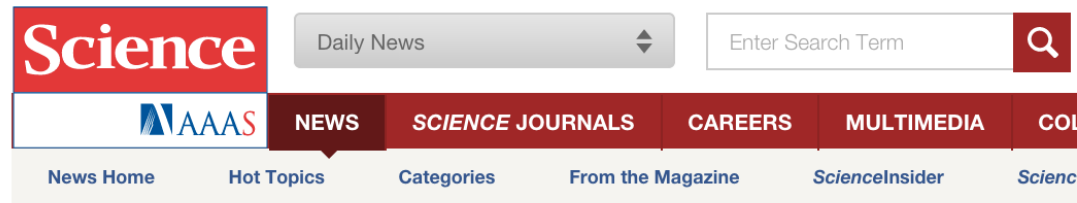
Source: Dictionary.com

Box 1.1

“Complex systems consist of a large number of interacting components. The interactions give rise to emergent hierarchical structures. The components of the system and properties at systems level typically change with time.”

H.J. Jensen, in Encyclopedia of Complexity and Systems Science

Importance of metaphors, analogies and common languages



News > Math > How bird flocks are like liquid helium

LATEST NEWS



COBBS LAB, ISC-CNR

All together now. Flocks of starlings fly over Rome's city center.

How bird flocks are like liquid helium

Tool box

Agent-based and numerical simulations

Information theory

(Non-linear) Dynamical systems

Data mining and optimisation

Networks

Tool box

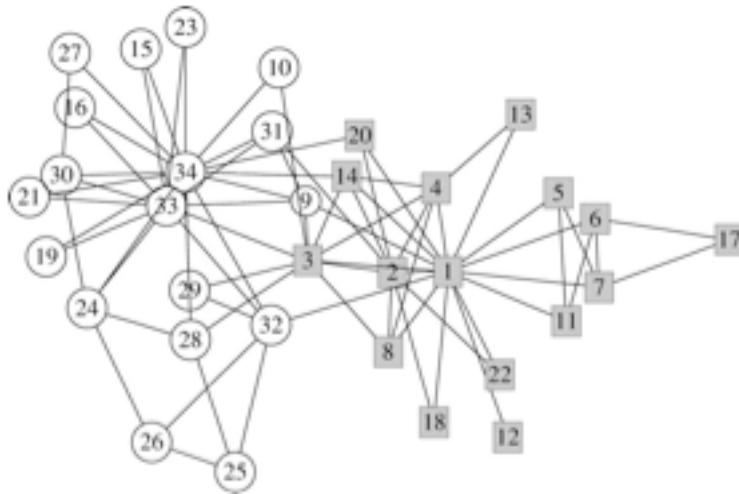
Agent-based and numerical simulations

Information theory

(Non-linear) Dynamical systems

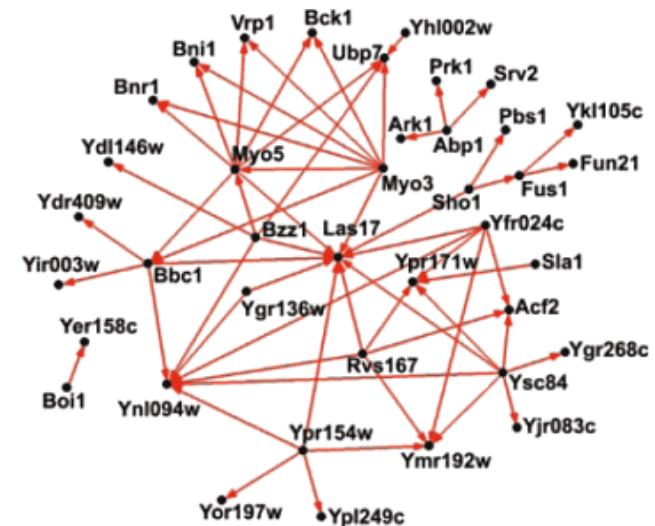
Data mining and optimisation

Networks



Social networks

- Collaboration networks
- Communication networks
- Online social networks



Biological networks

- Protein-protein interaction networks
- Neural networks
- Food webs

Network representation

$$A_{ij} = \begin{cases} 1 & \text{if node } v_i \text{ is adjacent to node } v_j, \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

If the network is weighted, A_{ij} can take positive values different from one, representing the weight of the link. In general, undirected and directed networks will yield symmetric and asymmetric adjacency matrices, respectively. The adjacency matrix of the network illustrated in Fig. 3.1 is given by

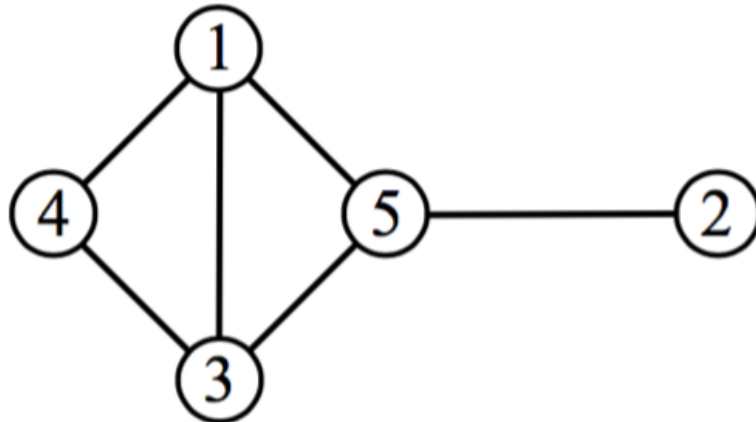
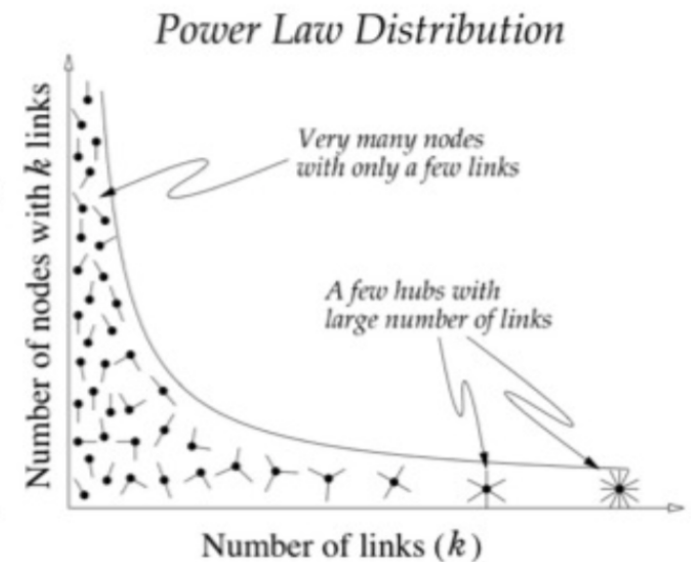
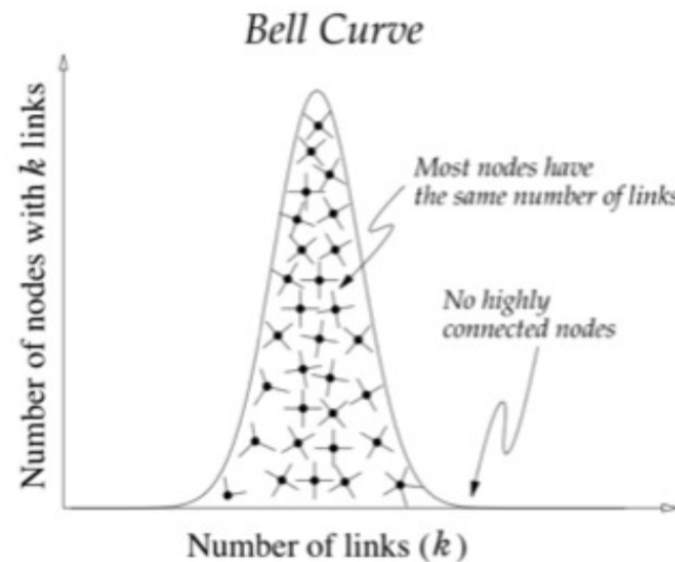

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}. \quad (3.3)$$

Fig. 3.1 An undirected and unweighted network.

Properties: Scale-free (and heterogeneous) distributions

$$k_i = \sum_{j=1}^N A_{ij} \left(= \sum_{j=1}^N A_{ji} \right)$$



Properties: Scale-free (and heterogeneous) distributions

$$p(x) = Cx^{-\alpha} \quad (x \geq x_{\min})$$

$$\int_{x_{\min}}^{\infty} x^{\beta} p(x) dx = \frac{\alpha - 1}{\alpha - 1 - \beta} x_{\min}^{\beta} \quad (\beta < \alpha - 1)$$

Moments diverge when: $\beta \geq \alpha - 1$

In particular, the variance diverges when: $2 < \alpha \leq 3$

When a moment diverges, its empirical measurement diverges as the number of sample increases

Are power-laws everywhere or do we see them everywhere?

quantity	n	$\langle x \rangle$	σ	x_{\max}	\hat{x}_{\min}	$\hat{\alpha}$	n_{tail}	p
count of word use	18 855	11.14	148.33	14 086	7 ± 2	1.95(2)	2958 ± 987	0.49
protein interaction degree	1846	2.34	3.05	56	5 ± 2	3.1(3)	204 ± 263	0.31
metabolic degree	1641	5.68	17.81	468	4 ± 1	2.8(1)	748 ± 136	0.00
Internet degree	22 688	5.63	37.83	2583	21 ± 9	2.12(9)	770 ± 1124	0.29
telephone calls received	51 360 423	3.88	179.09	375 746	120 ± 49	2.09(1)	$102 592 \pm 210 147$	0.63
intensity of wars	115	15.70	49.97	382	2.1 ± 3.5	1.7(2)	70 ± 14	0.20
terrorist attack severity	9101	4.35	31.58	2749	12 ± 4	2.4(2)	547 ± 1663	0.68
HTTP size (kilobytes)	226 386	7.36	57.94	10 971	36.25 ± 22.74	2.48(5)	6794 ± 2232	0.00
species per genus	509	5.59	6.94	56	4 ± 2	2.4(2)	233 ± 138	0.10
bird species sightings	591	3384.36	10 952.34	138 705	6679 ± 2463	2.1(2)	66 ± 41	0.55
blackouts ($\times 10^3$)	211	253.87	610.31	7500	230 ± 90	2.3(3)	59 ± 35	0.62
sales of books ($\times 10^3$)	633	1986.67	1396.60	19 077	2400 ± 430	3.7(3)	139 ± 115	0.66
population of cities ($\times 10^3$)	19 447	9.00	77.83	8 009	52.46 ± 11.88	2.37(8)	580 ± 177	0.76
email address books size	4581	12.45	21.49	333	57 ± 21	3.5(6)	196 ± 449	0.16
forest fire size (acres)	203 785	0.90	20.99	4121	6324 ± 3487	2.2(3)	521 ± 6801	0.05
solar flare intensity	12 773	689.41	6520.59	231 300	323 ± 89	1.79(2)	1711 ± 384	1.00
quake intensity ($\times 10^3$)	19 302	24.54	563.83	63 096	0.794 ± 80.198	1.64(4)	$11 697 \pm 2159$	0.00
religious followers ($\times 10^6$)	103	27.36	136.64	1050	3.85 ± 1.60	1.8(1)	39 ± 26	0.42
freq. of surnames ($\times 10^3$)	2753	50.59	113.99	2502	111.92 ± 40.67	2.5(2)	239 ± 215	0.20
net worth (mil. USD)	400	2388.69	4 167.35	46 000	900 ± 364	2.3(1)	302 ± 77	0.00
citations to papers	415 229	16.17	44.02	8904	160 ± 35	3.16(6)	3455 ± 1859	0.20
papers authored	401 445	7.21	16.52	1416	133 ± 13	4.3(1)	988 ± 377	0.90
hits to web sites	119 724	9.83	392.52	129 641	2 ± 13	1.81(8)	$50 981 \pm 16 898$	0.00
links to web sites	241 428 853	9.15	106 871.65	1 199 466	3684 ± 151	2.336(9)	$28 986 \pm 1560$	0.00

TABLE 6.1

Basic parameters of the data sets described in this section, along with their power-law fits and the corresponding p -value (statistically significant values are denoted in **bold**).

Critical Truths About Power Laws: Michael P. H. Stumpf and Mason A. Porter

<https://people.maths.ox.ac.uk/porterm/papers/critical.pdf>

Power laws, Pareto distributions and Zipf's law, M. E. J. Newman

<http://arxiv.org/pdf/cond-mat/0412004v3.pdf>

Power-law distributions in empirical data, Aaron Clauset et al.

<http://arxiv.org/pdf/0706.1062v2.pdf>

Models of networks

Mechanistic models, whose goal is to understand the mechanisms leading to certain structures observed in empirical networks. In general, such models are defined by simple rules on how nodes and links are created or destroyed in the course of time. Examples include the **growing network model** (e.g. BA). Comparison between networks generated by the model and empirical networks allows us to identify potential forces having driven the evolution of the empirical networks.

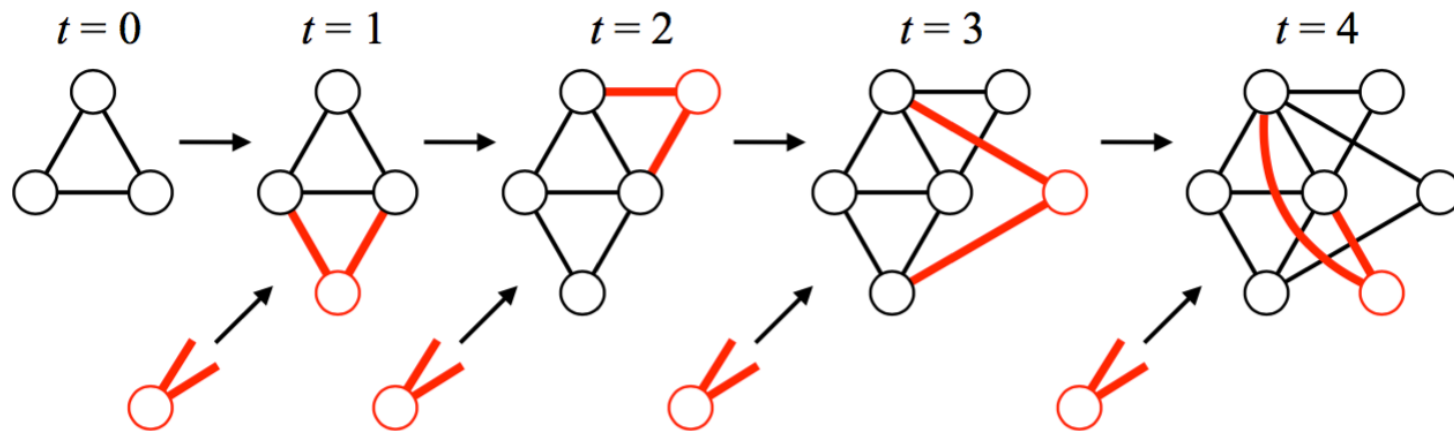


Fig. 3.3 First several stages of the BA model. The bold lines represent new links. We set $m_0 = 3$ and $m = 2$.

Models of networks

Growing networks with **preferential attachment** model was proposed by Barabasi and Albert in 1999 (BA or PA model), while the model had been known for longer time [de Solla Price (1976); Szyman´ski (1987); Mahmoud et al. (1993)]. The model is an instance of a family of multiplicative stochastic models, starting around a century ago with the Polya urn model and the Yule process. Historically, the mechanism of preferential attachment was also identified qualitatively by the sociologist Robert Merton, who called it the Matthew effect, also called **rich-gets-richer**, after a passage in Biblical Gospel of Matthew. The Yule process was studied by the economist Herbert Simon, interested in the distribution of wealth, who showed that it produces power-law distributions. This work inspired the Price’s network model [de Solla Price (1976)].

Models of networks

Random graph models in which links are random variables with certain constraints. The most fundamental model of random graph is the **Erdos-Renyi model**, and other examples include the **configuration model**, the exponential random graph models and random networks with communities. These models provide neither an explanation for the values taken by the parameters nor the reason for certain constraints to emerge in an empirical network. Instead, they have nice mathematical and statistical properties. For this reason, random graphs provide a useful baseline, or **null model**, for deciding whether patterns observed in empirical data are significant. In practice, if a value of a measurement observed in empirical data is significantly different from the expected value for the random graph model, the model does not represent the process behind the empirical data.

Erdos-Renyi networks

One of the simplest random graph models is the Erdős-Rényi random graph, introduced by Hungarian mathematicians Erdős and Rényi in 1959 (see [Bollobás (2001)] for detailed exposure). The model, also called the Poisson or binomial random graph, is denoted by $\mathcal{G}(N, q)$ and has two parameters, the number of nodes, N , and the probability q that a link exists between a pair of nodes. The self-loops are excluded. For each pair of nodes, consider a Bernoulli process that determines whether or not they are connected by a link. In fact, $\mathcal{G}(N, q)$ does not represent a single network, but a **random ensemble** of them in the probabilistic sense. Any network without multiple edges or self-loops is generated by the random graph $\mathcal{G}(N, q)$ as long as $0 < q < 1$.

The degree distribution is a binomial distribution

$$p(k) = \binom{N-1}{k} q^k (1-q)^{N-1-k}$$

well approximated by a Poisson distribution when N is sufficiently large

$$p(k) = \frac{\langle k \rangle^k}{k!} e^{-\langle k \rangle}$$

Erdos-Renyi networks

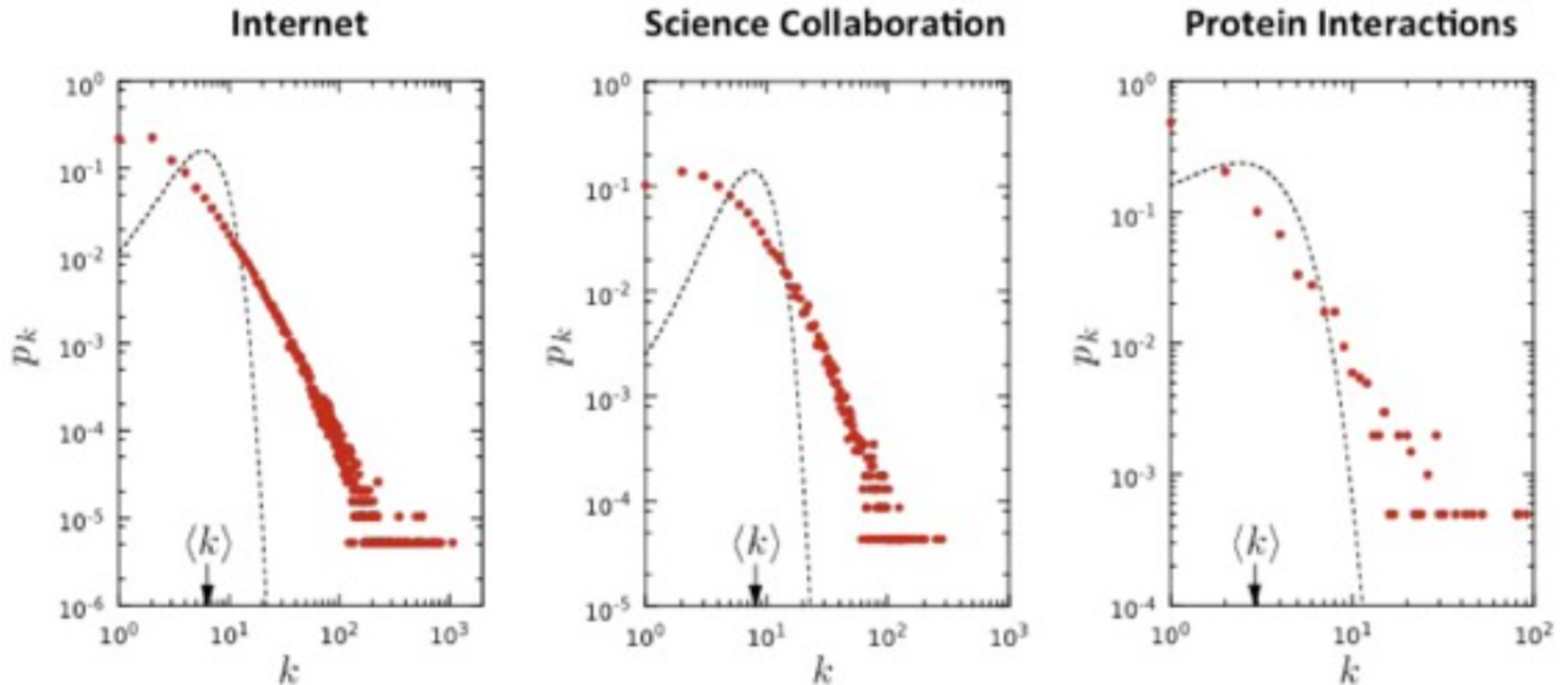


Image 3.5

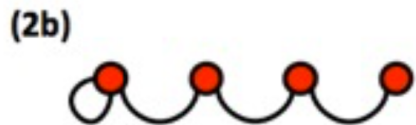
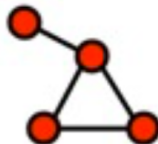
Degree distribution of real networks.

The degree distribution of the Internet, science collaboration network, and the protein interaction network of yeast (Table 2.1). The dashed line corresponds to the Poisson prediction, obtained by measuring $\langle k \rangle$ for the real network and then plotting Eq. (8). The significant deviation between the data and the Poisson fit indicates that the random network model underestimates the size and the frequency of highly connected nodes, or hubs.

Configuration model

The configuration model is a generalisation of the Erdős-Rényi random graph to the case of an arbitrary but given degree of each node. It is used to inspect the effect of heterogeneous degree distributions because it does not have more specific features such as high clustering. The model is defined as a random graph in which all possible configurations appear with the same probability under the constraint that node v_i has degree k_i ($1 \leq i \leq N$). The degree sequence $\{k_i\}$ is often generated by a given degree distribution $p(k)$ under the constraint that the sum of the degrees is an even number to satisfy the handshaking lemma.

(1) $k_1 = 3$ $k_2 = 2$ $k_3 = 2$ $k_4 = 1$



The density of self-loops and multiple edges goes to zero when N is sufficiently large.

Configuration model

The configuration model is a generalisation of the Erdős-Rényi random graph to the case of an arbitrary but given degree of each node. It is used to inspect the effect of heterogeneous degree distributions because it does not have more specific features such as high clustering. The model is defined as a random graph in which all possible configurations appear with the same probability under the constraint that node v_i has degree k_i ($1 \leq i \leq N$). The degree sequence $\{k_i\}$ is often generated by a given degree distribution $p(k)$ under the constraint that the sum of the degrees is an even number to satisfy the handshaking lemma.

Because v_i owns k_i stubs, the expected number of links between v_i and v_j is given by

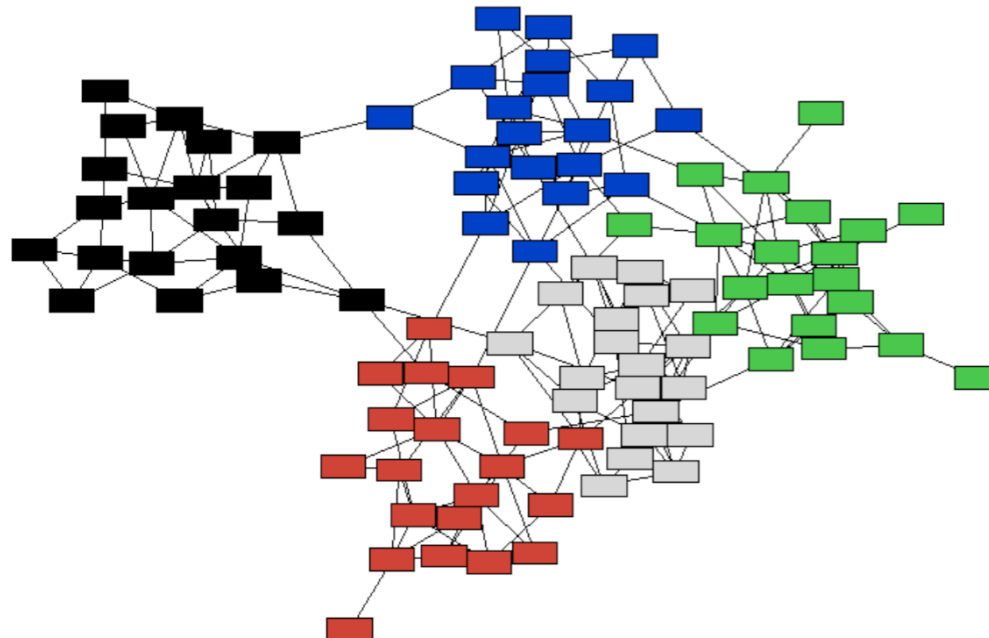
$$A_{ij}^* = \frac{k_i k_j}{2M}, \quad (3.64)$$

where A_{ij}^* represents the statistical average of the adjacency matrix.

Beyond the degree distribution

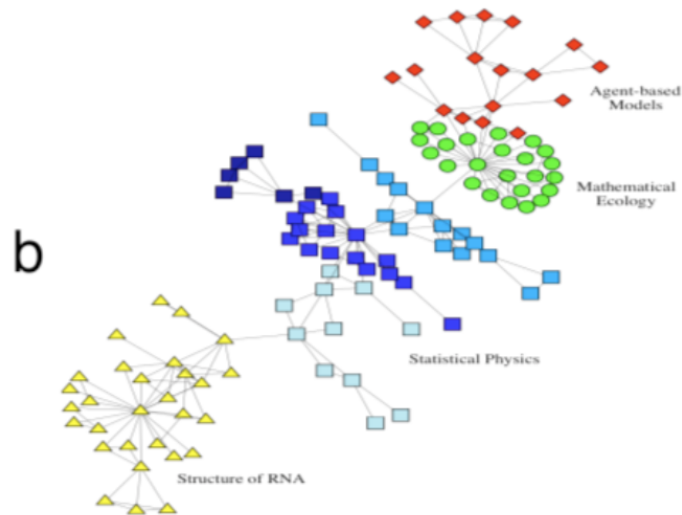
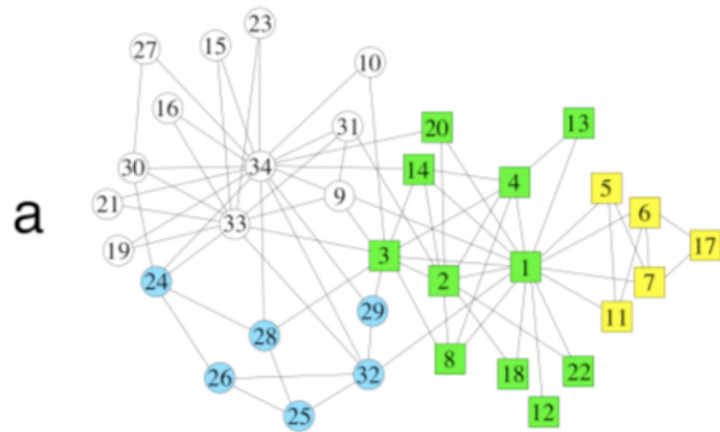
clustering coefficient
density of cliques
motifs
k-core
degree-degree correlations

Modularity:
Many networks are inhomogeneous and are made of modules: many links within modules and a few links between different modules



Properties: Modularity

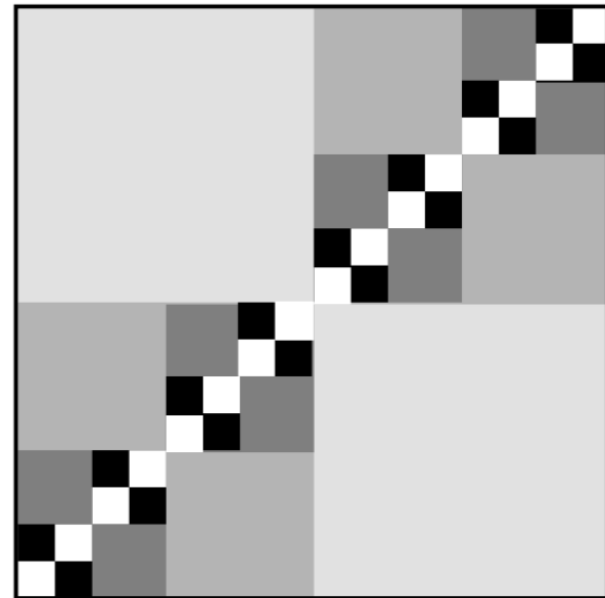
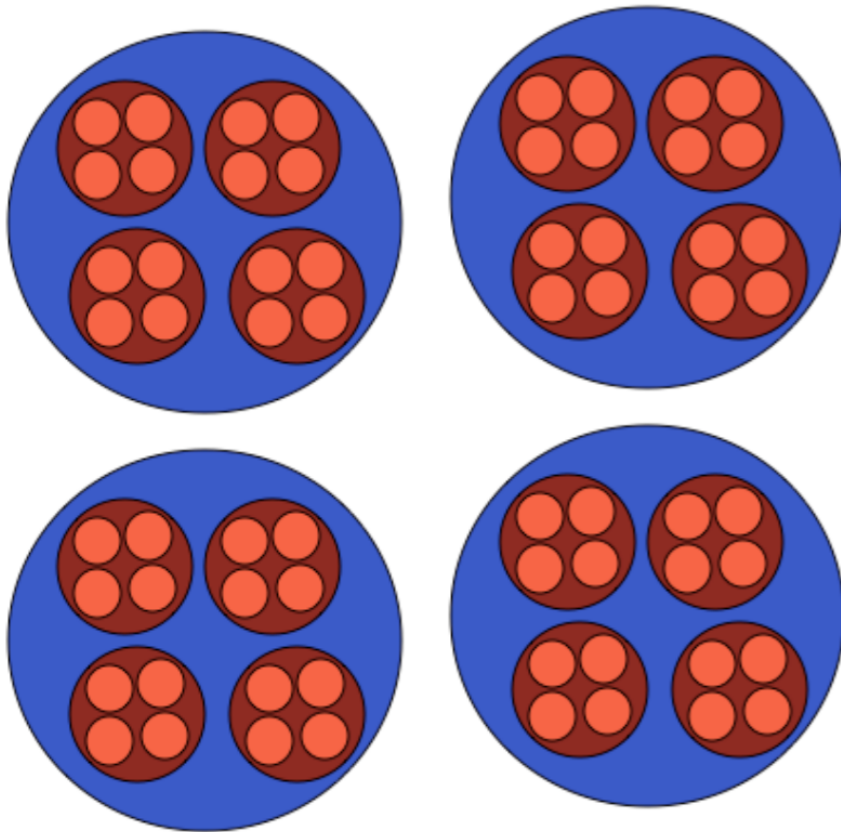
Observed in social, biological and information networks



Properties: Multi-level modularity (hierarchy)

Networks have a hierarchical/multi-scale structure: modules within modules

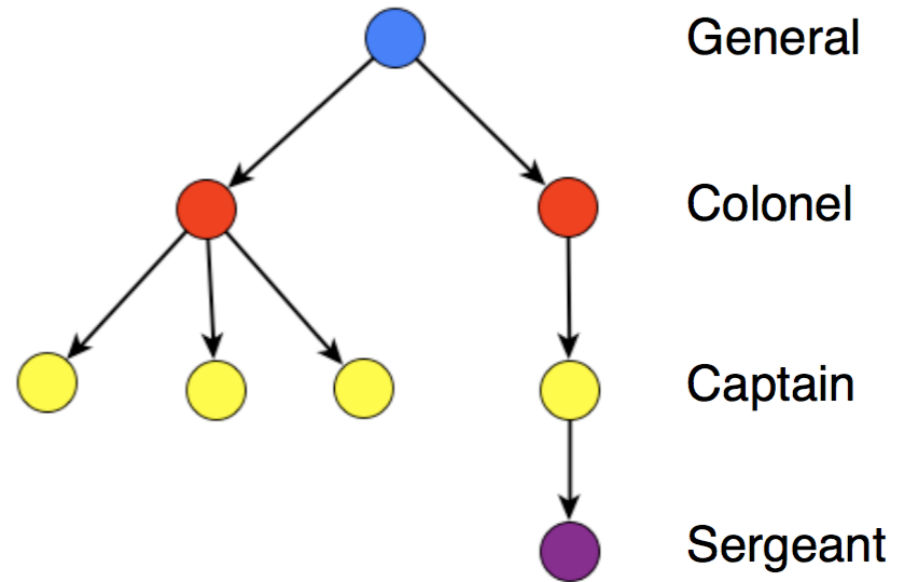
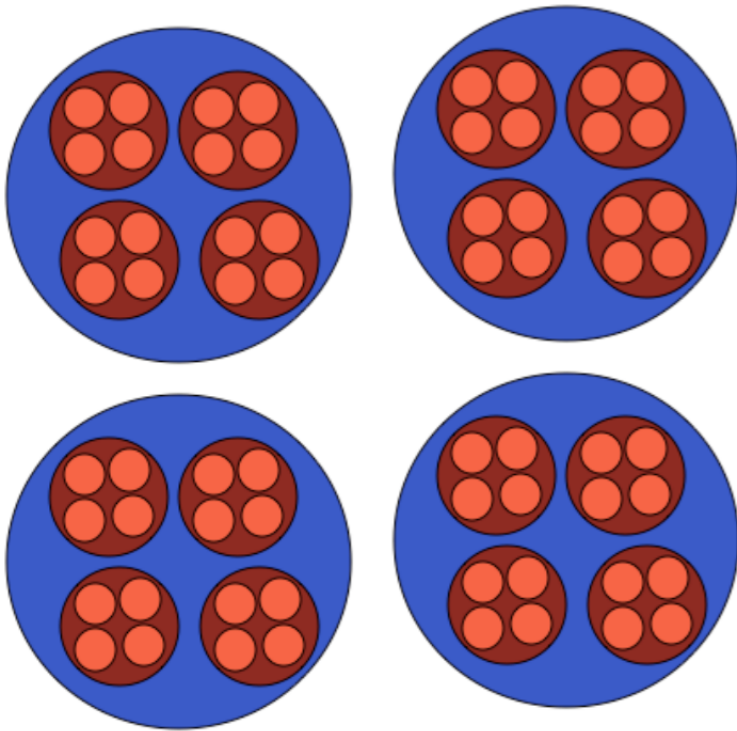
Nested organization



nb. Different notions of hierarchy

Hierarchy = multi-scale structure: modules within modules

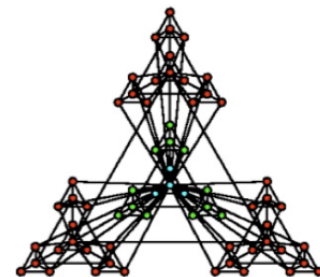
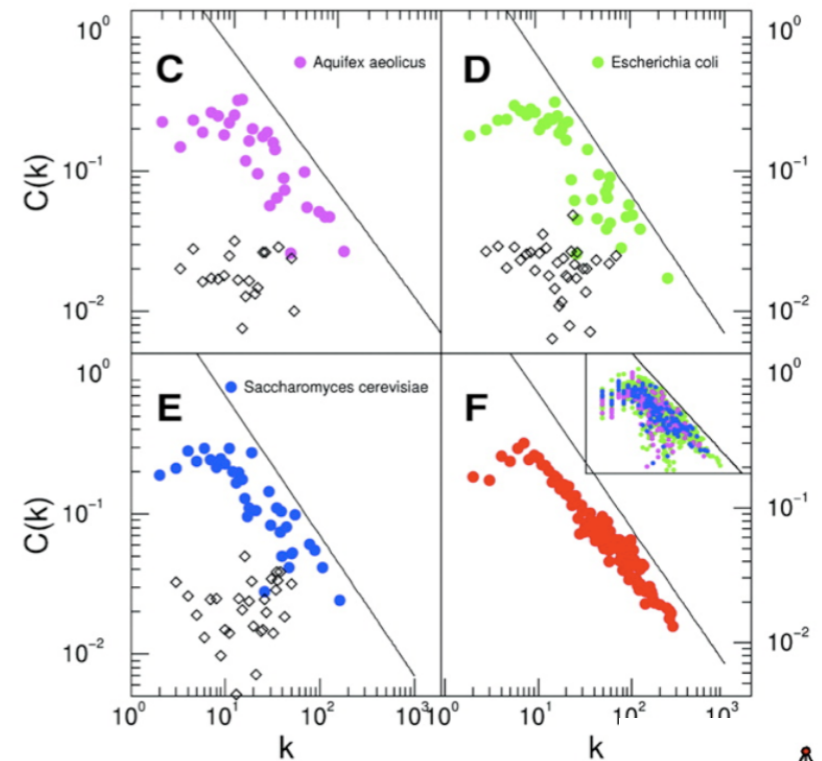
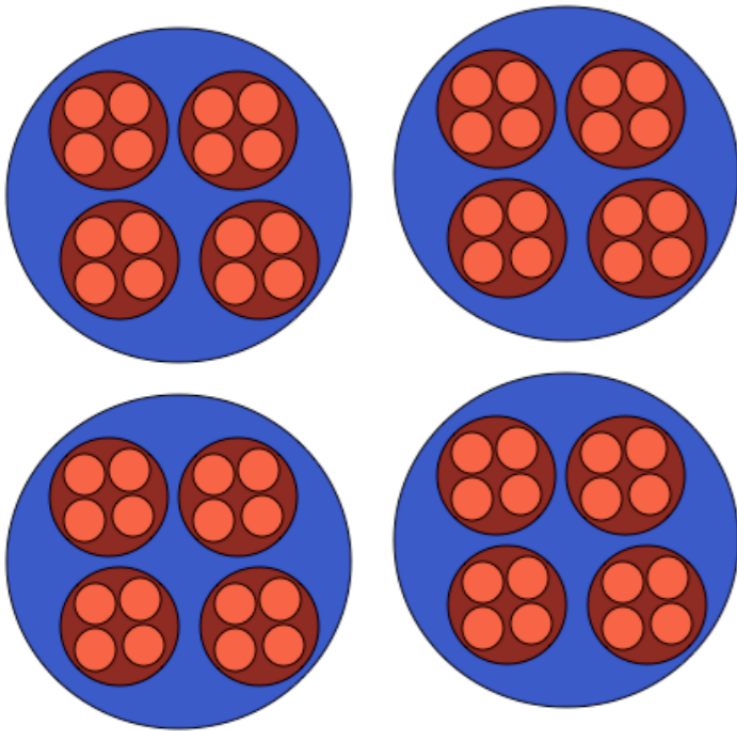
Hierarchy = subordination



nb. Different notions of hierarchy

Hierarchy = multi-scale structure: modules within modules

Hierarchy of nodes with different degrees of “modularity” (clustering)

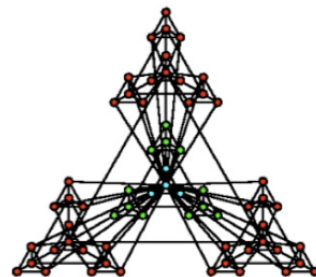


Hierarchical Organization of Modularity in Metabolic Networks, E. Ravasz et al.

<http://barabasi.com/f/108.pdf>

Modularity in questions

- Why do we observe modularity in complex systems?
- What is the impact of modularity on the function and dynamics of systems?
- Is it possible to uncover communities in networks in an algorithmic fashion?



Why modularity?

Generic mechanisms driving the emergence of modularity,
“à la rich-gets-richer”?

Why modularity?

Generic mechanisms driving the emergence of modularity, “à la rich-gets-richer”?

Simple systems evolve more rapidly if there are stable intermediate forms (modules) than if there are not present.

Among possible complex organizations, modular hierarchies are observed because they are the ones that have had the time to evolve.

Simon, H. (1962). The architecture of complexity.

<http://www.cs.brandeis.edu/~cs146a/handouts/papers/simon-complexity.pdf>

Why modularity?

Watchmaker parable:

“There once were two watchmakers, named Hora and Tempus, who made very fine watches. The phones in their workshops rang frequently; new customers were constantly calling them. However, Hora prospered while Tempus became poorer and poorer. In the end, Tempus lost his shop. What was the reason behind this?

The watches consisted of about 1000 parts each. The watches that Tempus made were designed such that, when he had to put down a partly assembled watch (for instance, to answer the phone), it immediately fell into pieces and had to be reassembled from the basic elements.

Hora had designed his watches so that he could put together subassemblies of about ten components each. Ten of these subassemblies could be put together to make a larger sub-assembly. Finally, ten of the larger subassemblies constituted the whole watch. Each subassembly could be put down without falling apart.”

Simon, H. (1962). The architecture of complexity.

<http://www.cs.brandeis.edu/~cs146a/handouts/papers/simon-complexity.pdf>

Why modularity?

Probability that an interruption occurs while a piece is added, say $p=0.01$

Tempus

- must complete 1 assembly of 1000 elements
- loses on average 100 pieces ($1/0.01$)
- finishes an assembly with probability $(1-0.01)^{1000} \sim 0.000004$
- Time to finish a watch: $100 / (1-0.01)^{1000}$

Hora

- must complete 111 assemblies of 10 elements
- loses on average 5 pieces (random between 0 and 9)
- finishes an assembly with probability $(1-0.01)^{10} \sim 0.9$
- Time to finish a watch: $111 * 5 / (1-0.01)^{10}$

It will take Tempus **4000 times** as long to assemble a watch

Importance of disturbance due to the environment

Robust intermediate steps during evolution: if the system breaks down (whatever the reason), evolution does not restart from scratch, but from intermediate, stable solutions (back-up!).

Random networks with communities

“These networks consist of 128 vertices divided into 4 communities of 32 nodes each. Each vertex pair is connected by an edge with one of two different probabilities, one for pairs in the same group and one for pairs in different groups, with values chosen so that the expected degree of each vertex remains fixed at 16. As the average number b of between-group connections per vertex is increased from zero, the community structure in the network, stark at first, becomes gradually obscured until, at the point where between- and within-group edges are equally likely, the network becomes a standard Poisson random graph with no community structure at all.”

Connection to stochastic block-models (see later).

Random networks with communities

Generalization to arbitrary number of communities, distribution of community size and degree distribution

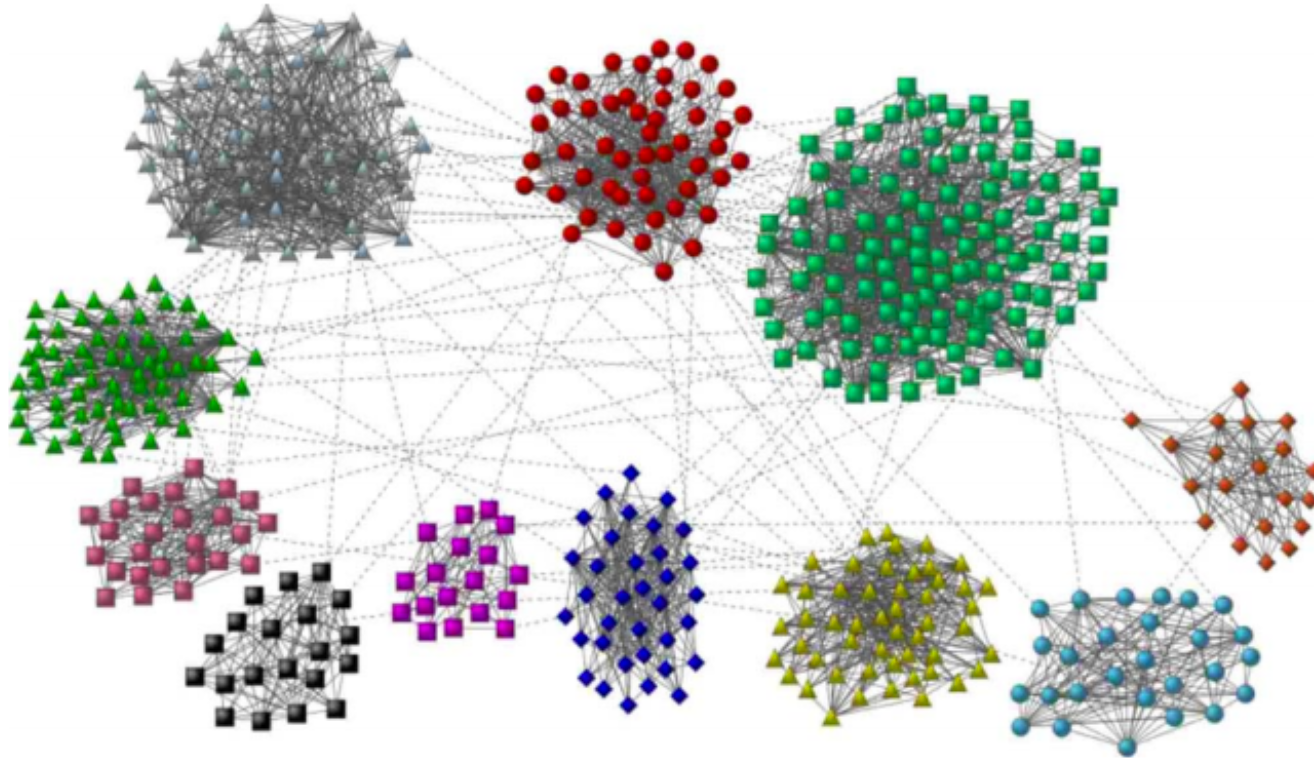


FIG. 1. (Color online) A realization of the new benchmark, with 500 nodes.

Benchmark graphs for testing community detection algorithms, Andrea Lancichinetti et al.

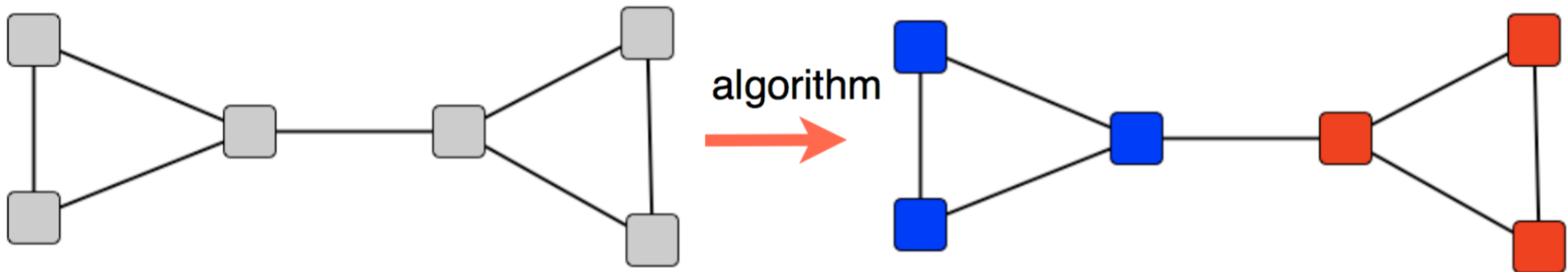
<https://6c131308-a-62cb3a1a-s-sites.googlegroups.com/site/andrealancichinetti/benchmark.pdf>

What is Community Detection?

Is it possible to uncover the (multi-scale) modular organisation of networks in an automated fashion? And please avoid false positives.

Given a graph, we look for an algorithm able to uncover its modules without specifying their number nor their size

The method should be scalable to accommodate very large networks, as often observed in the real-world.



Why community detection?

Graphs help us to comprehend in a visual way the global organisation of the system. This works extremely well when the graph is small but, as soon as the system is made of hundreds or thousands of nodes, a brute force representation typically leads to a meaningless cloud of nodes.

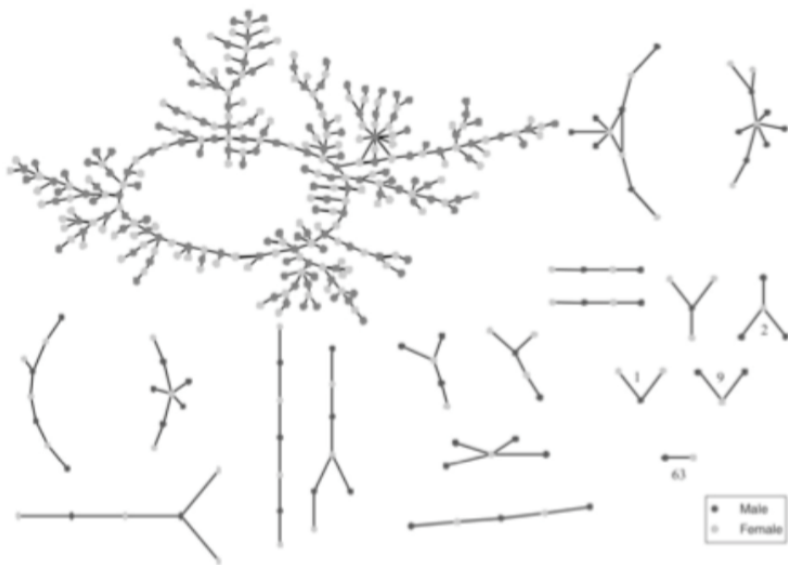
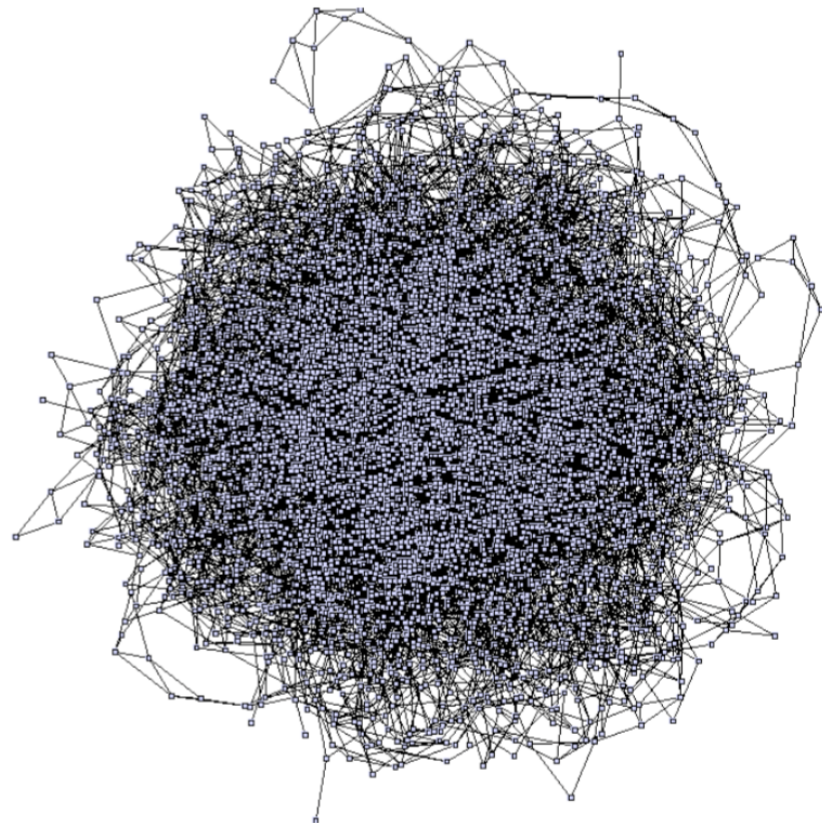
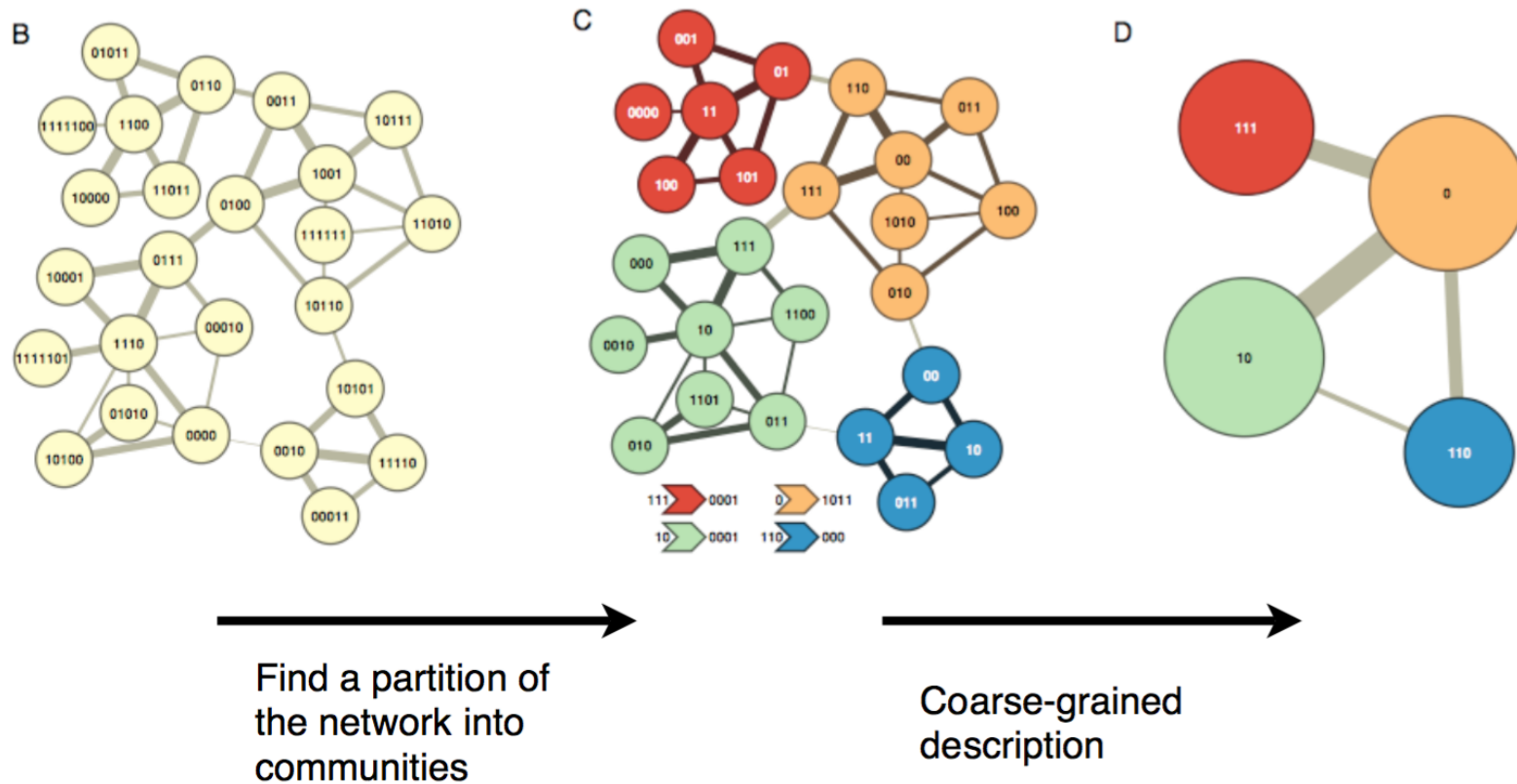


FIG. 2.—The direct relationship structure at Jefferson High



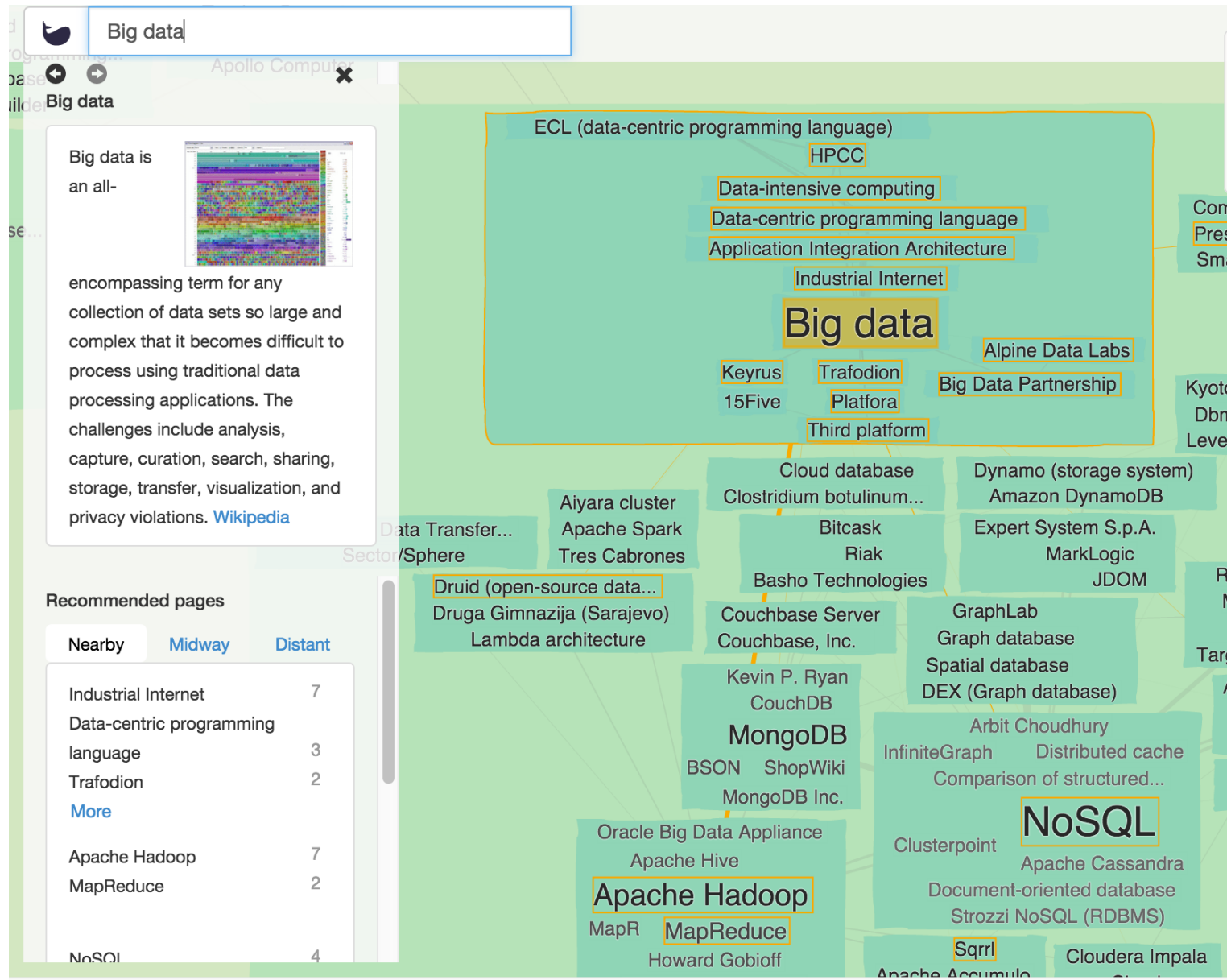
Why community detection?

Uncovering communities/modules helps to change the resolution of the representation and to draw a readable map of the network



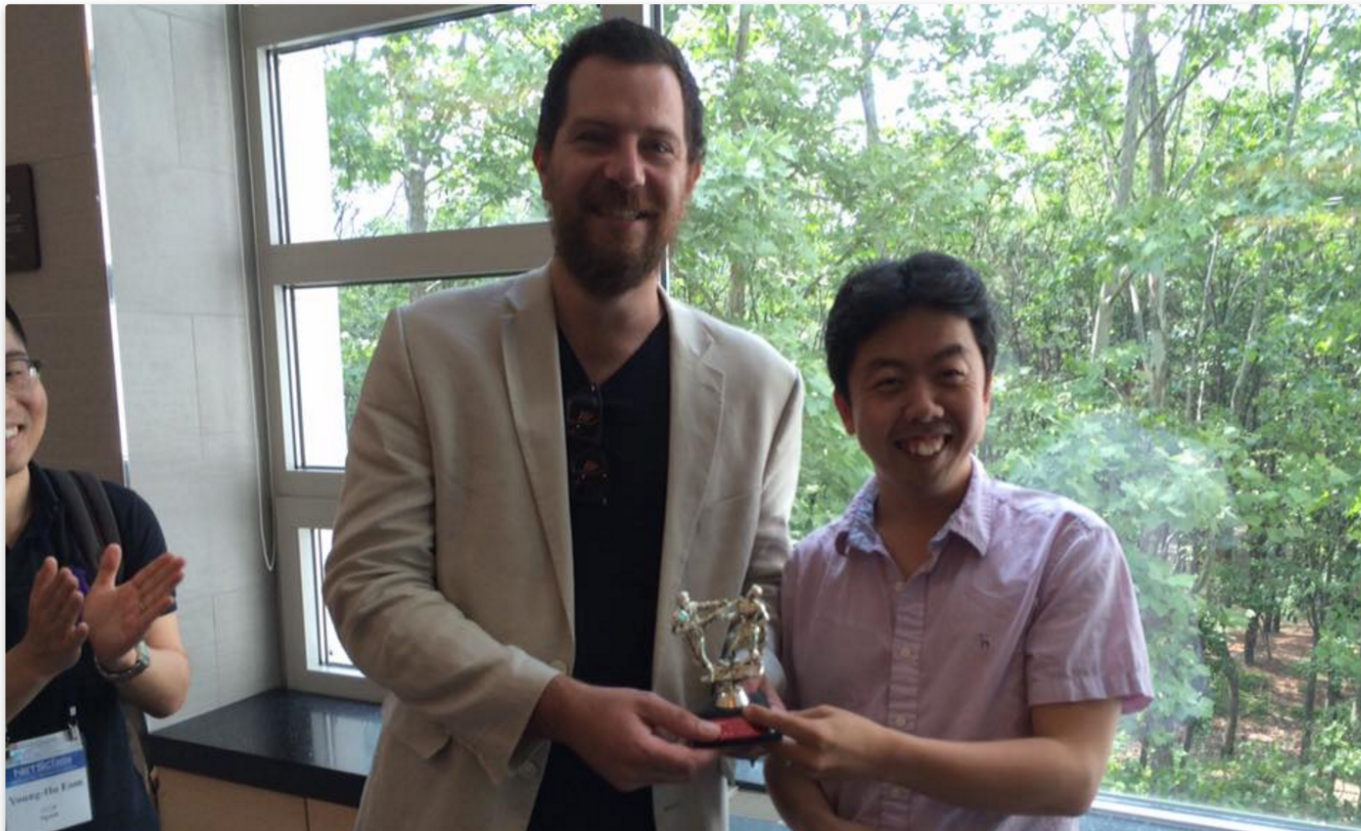
Why community detection?

Uncovering communities/modules helps to change the resolution of the representation and to draw a readable map of the network



Why community detection?

Network Scientists with Karate Trophies



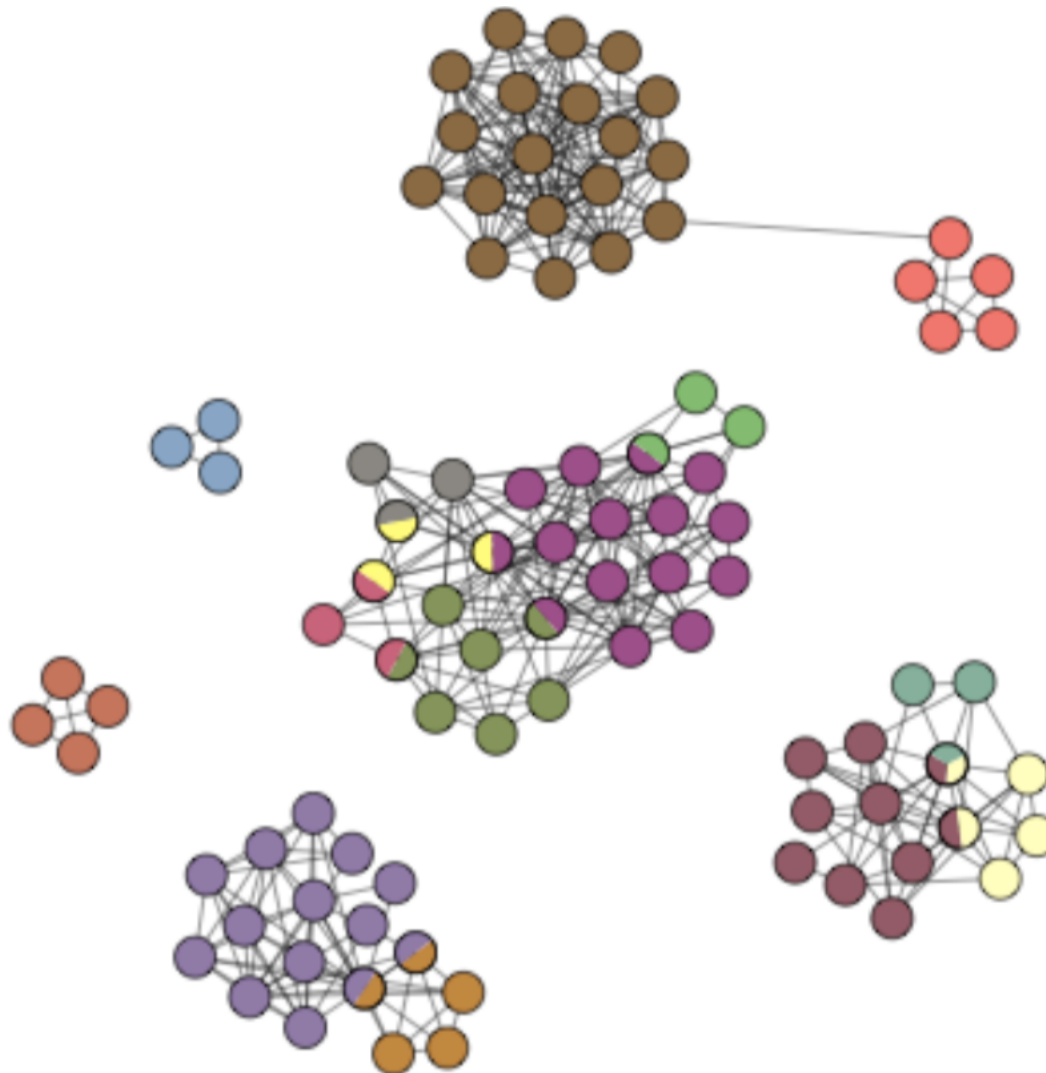
The first scientist at any conference on networks who uses Zachary's karate club as an example is inducted into the Zachary Karate Club Club, and awarded a prize. This tumblr records those moments.

 [RSS](#)

 [ARCHIVE](#)

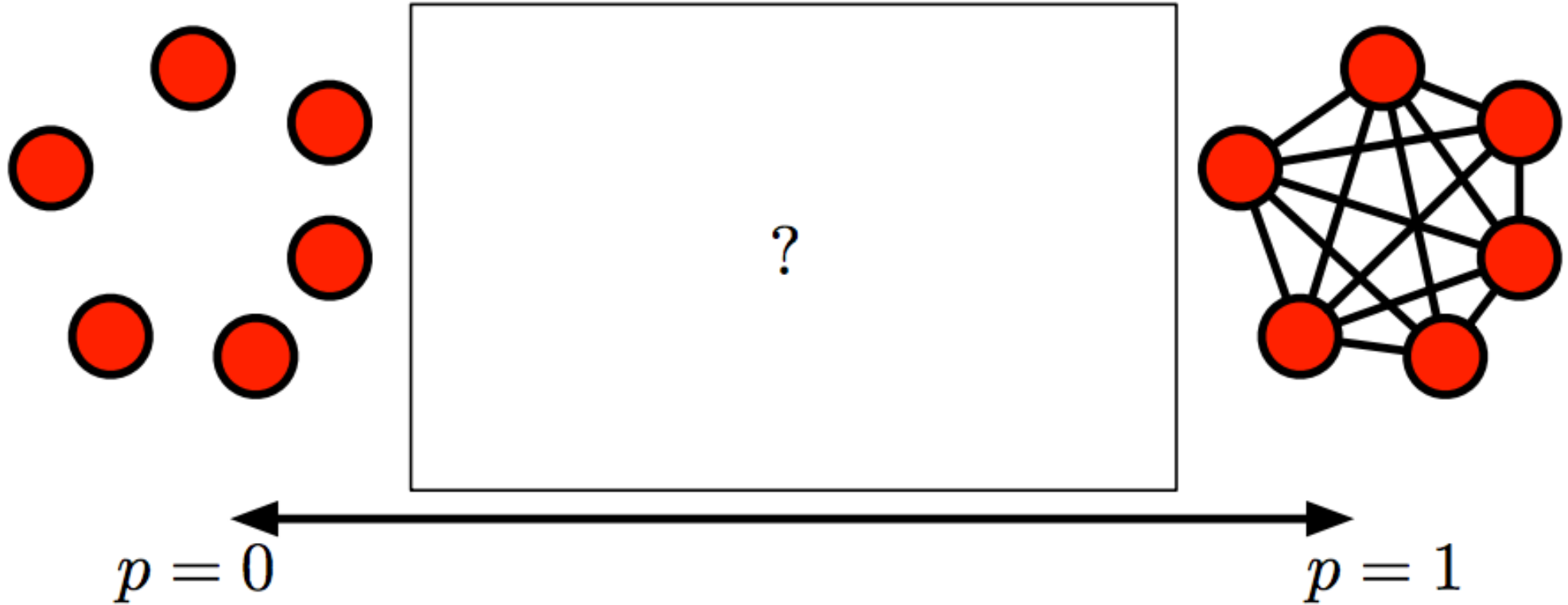
What is a “good” community?

A connected component is certainly a good community, in case of several components

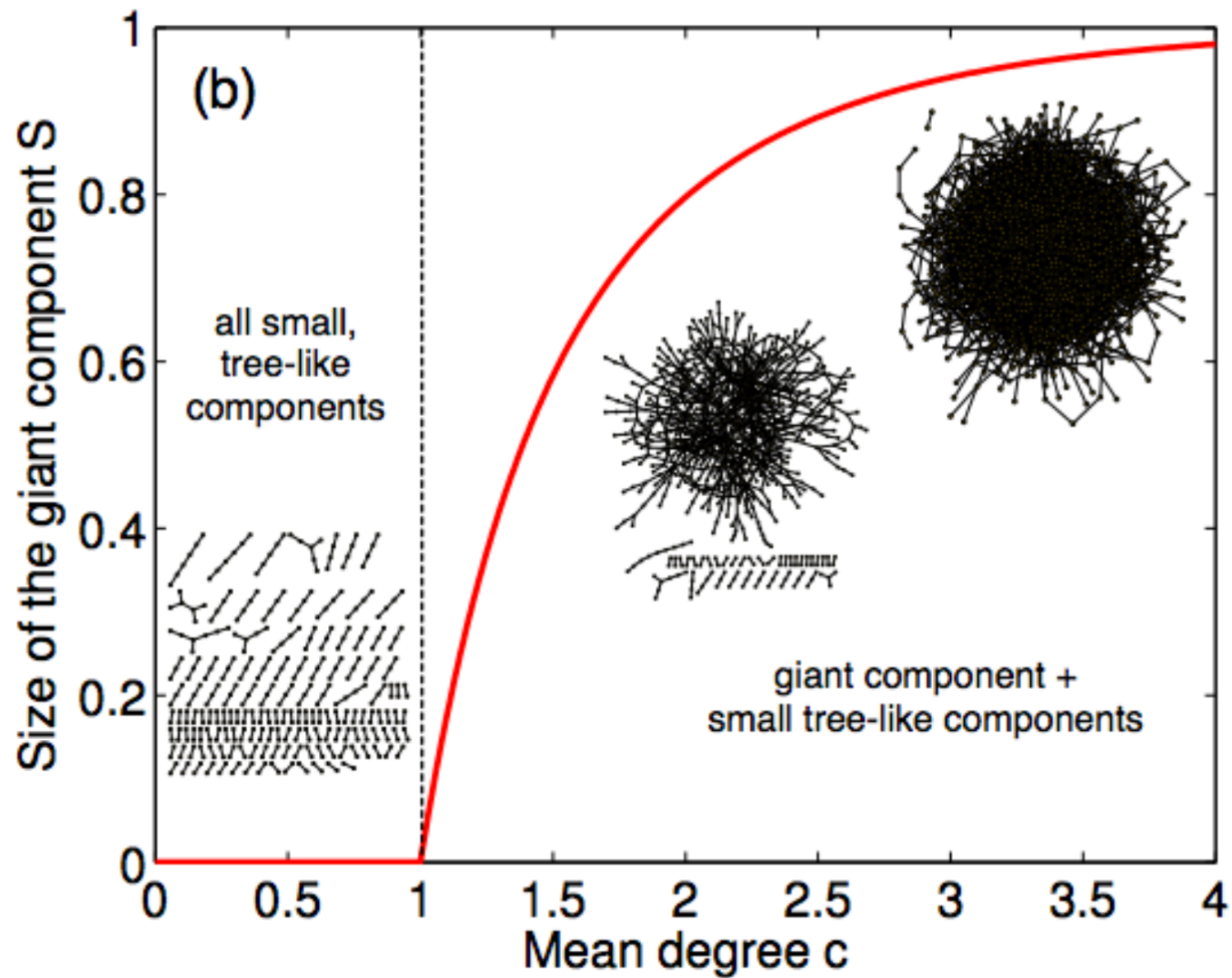


Percolation as a phase transition

Take the Erdos-Renyi network: how many disconnected components shall we expect depending on p ?



Percolation as a phase transition



Community detection versus network partitioning

Both terms refer to the division of a network into dense groups

Graph partitioning: the number and size of the groups is **fixed** by the user. For instance, in a typical bisection problem: what is the best division of a network into 2 groups of equal size such that the number of links between the groups is minimised.

Application parallel computing: minimise inter-processor communication. Divides the system into the required number of groups, whatever the organisation of the system.

Community detection: the number and size of the groups are unspecified, but determined by the organisation of the network. Ideally, the method should be able to uncover a mixture of groups of different size in the same system. It should divide a network only when a good subdivision exists and leave it undivided otherwise.

Community detection versus network partitioning

Both terms refer to the division of a network into dense groups

Graph partitioning: the number and size of the groups is **fixed** by the user. For instance, in a typical bisection problem: what is the best division of a network into 2 groups of equal size such that the number of links between the groups is minimised.

Application parallel computing: minimise inter-processor communication. Divides the system into the required number of groups, whatever the organisation of the system.

Community detection: the number and size of the groups are unspecified, but determined by the organisation of the network. Ideally, the method should be able to uncover a mixture of groups of different size in the same system. It should divide a network only when a good subdivision exists and leave it undivided otherwise.

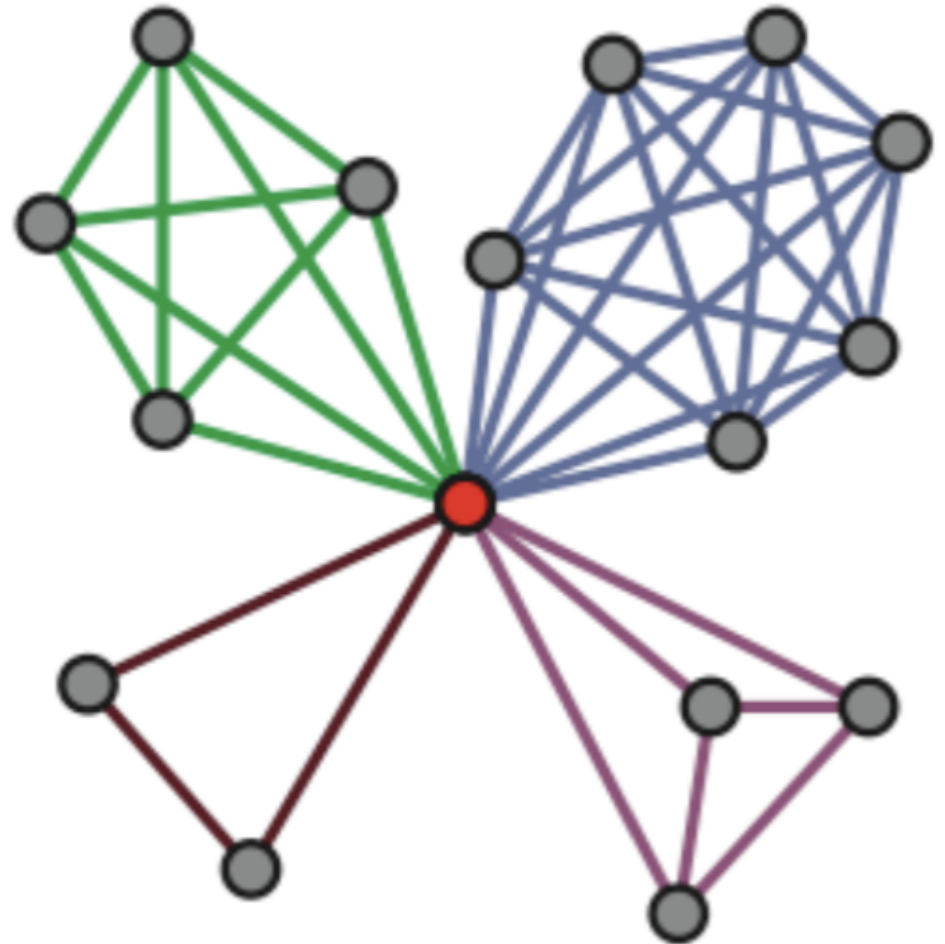
Different but similar methods

In both cases:

- 1) How to formalise the problem? **Definition of a good community**
- 2) How to solve it in practice? **Optimisation techniques to find it**

Community detection versus network partitioning

Other difference: communities can be overlapping (in some systems, at certain resolutions, it is the norm rather than the exception)



Graph bipartition

Definition of the problem:

Find the best division of a network into 2 groups of size n_1 and n_2 such that the cut size is minimal, where the cut size is the total number of links between different groups.

Solving the problem:

Looking through all bi-partitions and choose the one with the smallest cut size?

Impossible in practice, as the exhaustive search is extremely costly in terms of computer time

E.g. The number of ways to divide a network of $2n$ nodes into two groups of n and n nodes is:

$$\frac{(2n)!}{n!n!} \xrightarrow{\text{Stirling}} \frac{2^{n+1}}{\sqrt{n}}$$

No method solving exactly the problem in polynomial time for all networks...
But several existing heuristics allow to find approximate solutions in non-prohibitive times.

Spectral methods

$$R = \frac{1}{2} \sum_{ij \text{ in different groups}} A_{ij}$$

Let us denote by $s_i = \pm 1$ the assignment of node i

$$R = \frac{1}{2} \sum_{ij} A_{ij}(1 - s_i s_j) = \frac{1}{4} \sum_{ij} L_{ij} s_i s_j$$

By performing a spectral decomposition of the Laplacian matrix, one finds:

$$L_{ij} = \sum_{\alpha=2}^N \lambda_{\alpha} v_{\alpha,i} v_{\alpha,j}$$

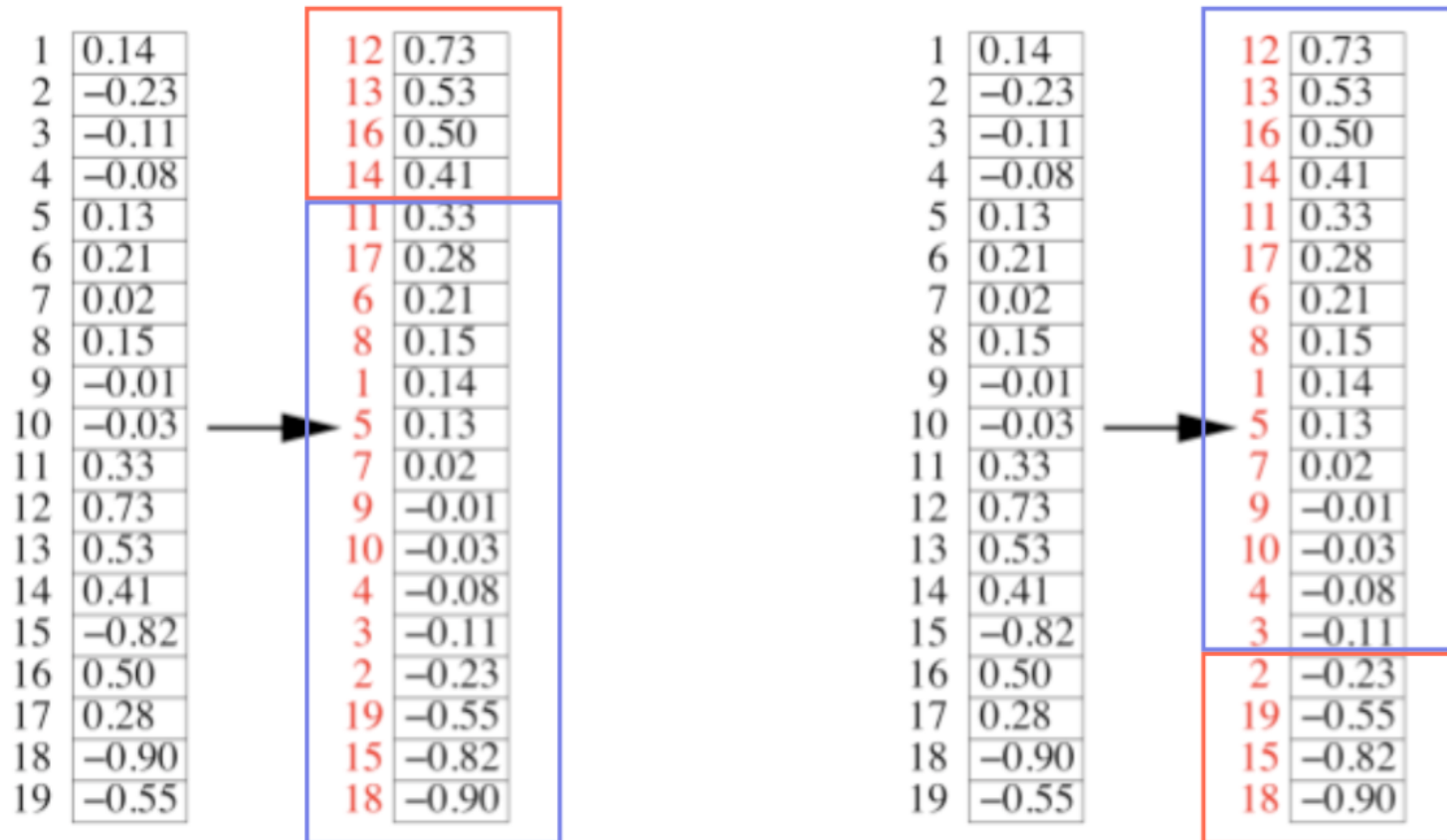
If there is no condition on s_i , the optimal solution would be $s_i = v_{2,i}$

$$R = \frac{1}{4} \lambda_2$$

But this solution is not a partition (except in extremely trivial situations) and it probably does not satisfy the required size of the groups.

Spectral methods

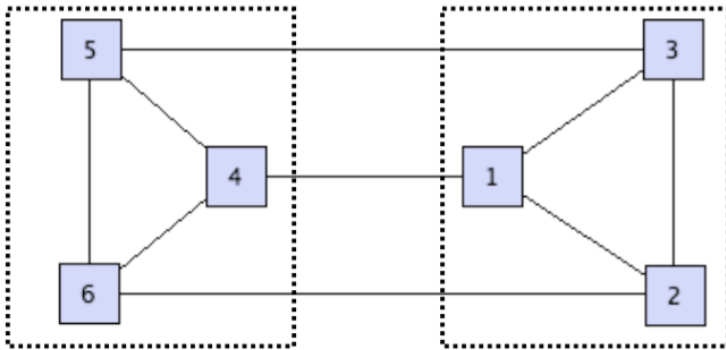
Approximation: If one wants a split into n_1 and $n_2 = n - n_1$ vertices, one orders the components of the Fiedler vector from the largest positive to the smallest negative and picks the n_1 largest (smallest) components of the Fiedler vector



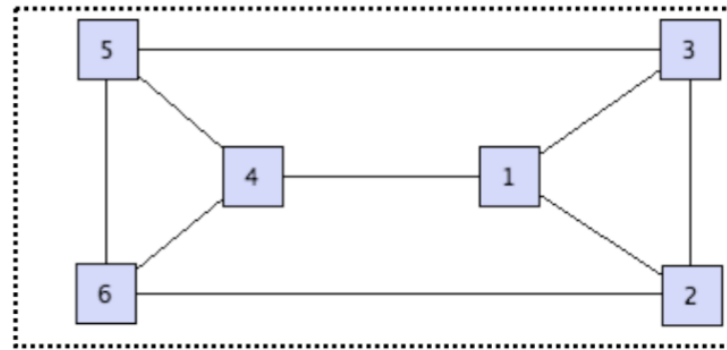
Complexity: $O(N^2)$ on sparse networks

Community detection

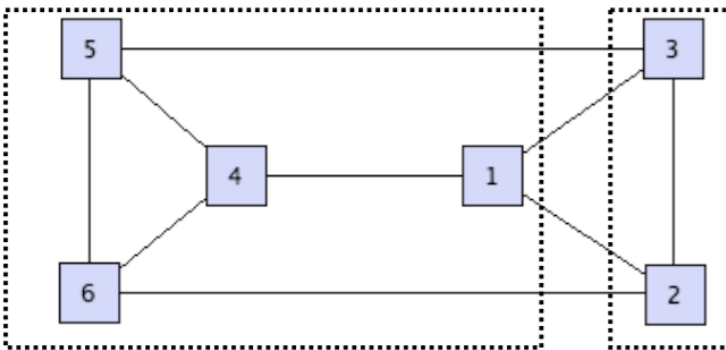
What is the best partition of a network into modules?
How do we rank the quality of partitions of different sizes?



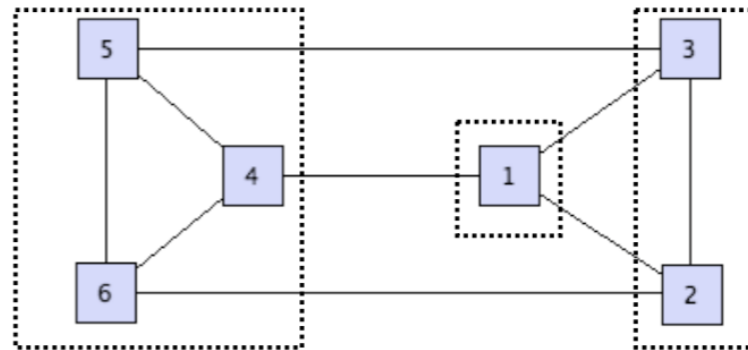
Q1



Q2



Q3



Q4

.....

Newman-Girvan Modularity

Q = fraction of edges within communities - expected fraction of such edges

Let us attribute each node i to a community c_i

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - P_{ij} \right] \delta(c_i, c_j)$$

$P_{ij} = \frac{k_i k_j}{2m}$ expected number of links between i and j

$$Q_C = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - k_i k_j / 2m \right] \delta(c_i, c_j) \quad Q_C \in [-1/2, 1]$$

Allows to compare partitions made of different numbers of modules

Note on the null model

Random network with constrained degrees $P_{ij} = \frac{k_i k_j}{2m}$

What if one has extra information about the nodes?

Directed networks -> $P_{ij} = k_i^{\text{in}} k_j^{\text{out}} / m$

Spatially-embedded networks -> $P_{ij}^{\text{Spa}} = N_i N_j f(d_{ij})$

Or if the information on the degrees is expected to be irrelevant:

$$P_{ij} = \langle k \rangle^2 / 2m = \langle k \rangle / N$$

$$\sum_{ij} P_{ij} = \sum_{ij} A_{ij} = 2m$$

Modularity

Property 1 *A partition where all the vertices are grouped into the same community has a modularity equal to zero. This proves to be simply shown from the definition of the null model $\frac{k_i k_j}{2m}$ for which $\sum_{i,j} \frac{k_i k_j}{2m} = 2m$ and from the expression of modularity in this particular case*

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] = 0 . \quad (3)$$

This property implies that any partition with a positive modularity is better than this trivial one, but also that it is always possible to find a partition such that $Q \geq 0$.

Modularity

Property 2 *If a partition contains a disconnected community, it is always preferable (in terms of modularity) to split this community into connected communities. Let us consider, for the sake of simplicity, the case of a disconnected community C_1 formed by two connected subgraphs C_{11}, C_{12} . In this case, modularity is given by*

$$\begin{aligned} Q &= \frac{1}{2m} \left[\sum_{C \neq C_1} \sum_{i,j \in C} \left(A_{ij} - \frac{k_i k_j}{2m} \right) + \sum_{i,j \in C_1} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \right] \\ &= \frac{1}{2m} \left[\sum_{C \neq C_1} \sum_{i,j \in C} \left(A_{ij} - \frac{k_i k_j}{2m} \right) + \sum_{i,j \in C_{11}} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \right. \\ &\quad \left. + \sum_{i,j \in C_{12}} \left(A_{ij} - \frac{k_i k_j}{2m} \right) + 2 \sum_{i \in C_{11}, j \in C_{12}} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \right]. \end{aligned} \quad (4)$$

Given that $A_{ij} = 0$ if $i \in C_{11}, j \in C_{12}$, the sum $\sum_{i \in C_{11}, j \in C_{12}}$ is composed uniquely of negative terms and it is thus preferable to split the community into two subcommunities.

This property implies that any partition made of disconnected communities is sub-optimal and that the optimal partition of a graph is only made of connected communities.

Modularity Optimization

Optimization of modularity is an NP-complete problem

Need for efficient heuristics

Optimization: Spectral methods

Similar method to one for minimizing the cut, based on the spectral properties of the modularity matrix Q

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

Let us first focus on the best division of the network into 2 communities.

Let us denote by $s_i = \pm 1$ the assignment of node i $\delta(c_i, c_j) = \frac{1}{2}(s_i s_j + 1)$

$$Q = \frac{1}{2m} \sum_{ij} Q_{ij} \delta(c_i, c_j) = \frac{1}{4m} \sum_{ij} Q_{ij} s_i s_j$$

By performing a spectral decomposition of the modularity matrix, one finds:

$$Q_{ij} = \sum_{\alpha=1}^N \lambda_{\alpha} v_{\alpha,i} v_{\alpha,j}$$

s_i is chosen to be as similar to the dominant eigenvector

$$\begin{aligned} s_i &= 1 \text{ if } v_{N,i} > 0 \\ s_i &= -1 \text{ if } v_{N,i} < 0 \end{aligned}$$

Optimization: Greedy optimization

Louvain: multi-scale, agglomerative and greedy

The algorithm is based on two steps that are repeated iteratively. **First phase:**

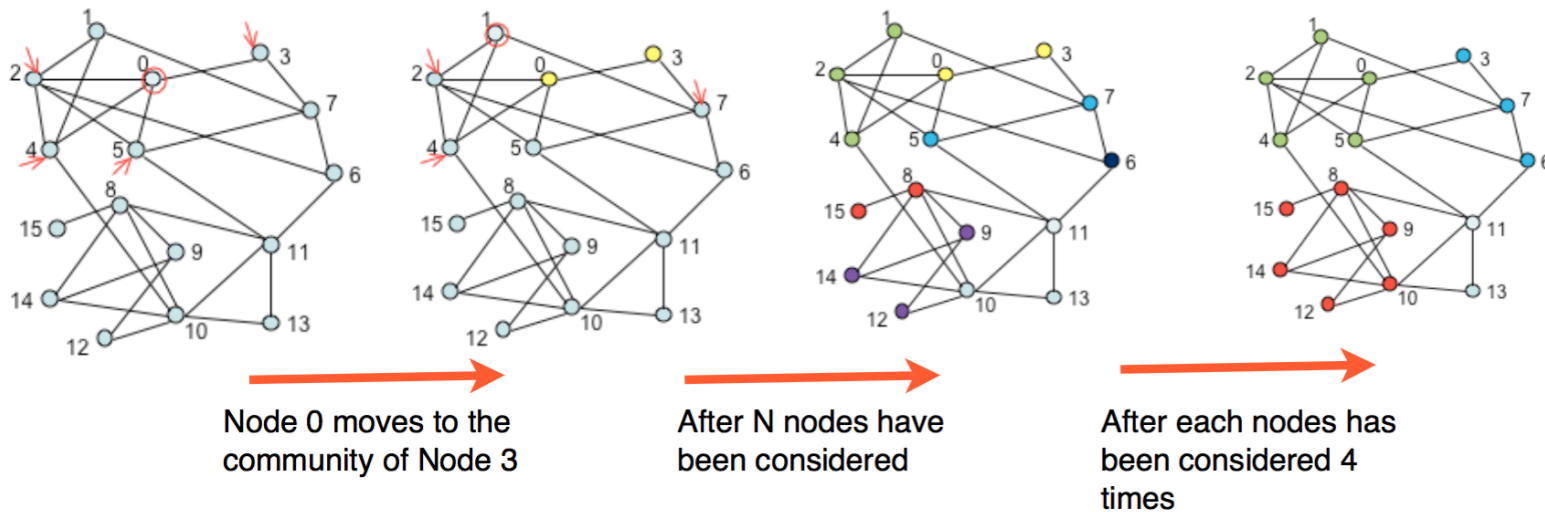
Find a local maximum

1) Give an order to the nodes (0,1,2,3,....., N-1)

2) Initially, each node belongs to its own community (N nodes and N communities)

3) One looks through all the nodes (from 0 to N-1) in an ordered way. The selected node looks among its neighbours and adopt the community of the neighbour for which the increase of modularity is maximum (and positive).

4) This step is performed iteratively until a local maximum of modularity is reached (each node may be considered several times).

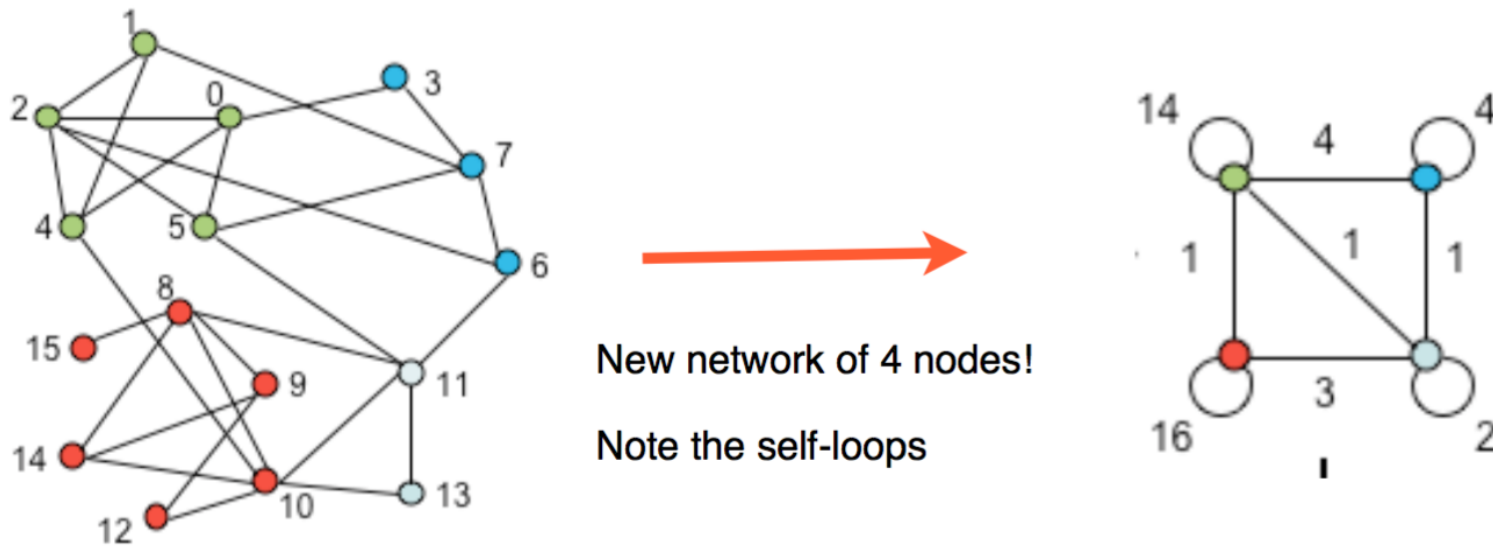


Optimization: Greedy optimization

Louvain: multi-scale, agglomerative and greedy

Once a local maximum has been attained, **second phase**:

We build a new network whose nodes are the communities. The weight of the links between communities is the total weight of the links between the nodes of these communities.



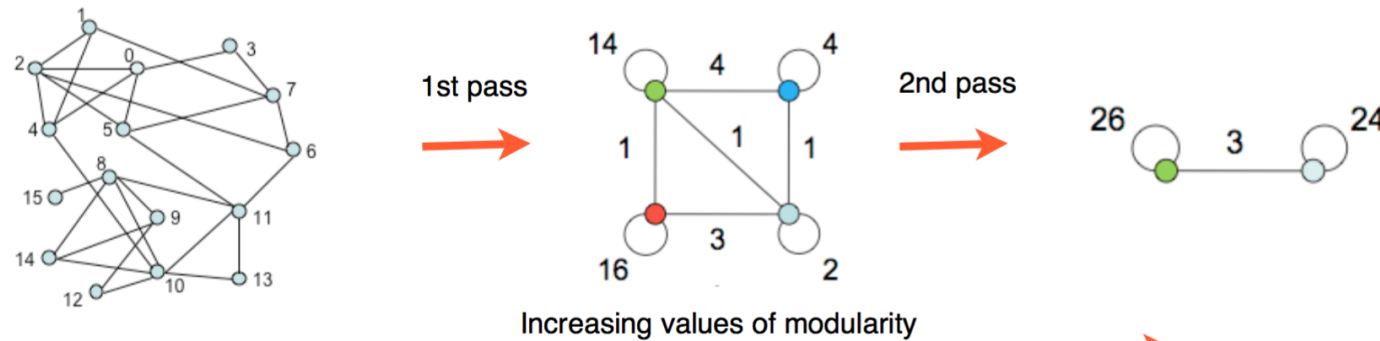
In typical realizations, the number of nodes diminishes drastically at this step.

Optimization: Greedy optimization

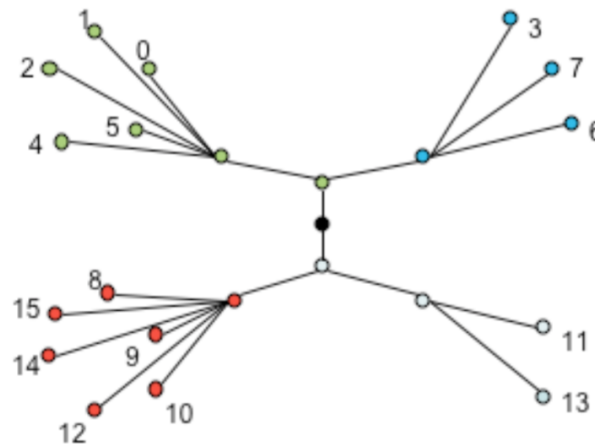
Louvain: multi-scale, agglomerative and greedy

The two steps are repeated iteratively, thereby leading to a hierarchical decomposition of the network.

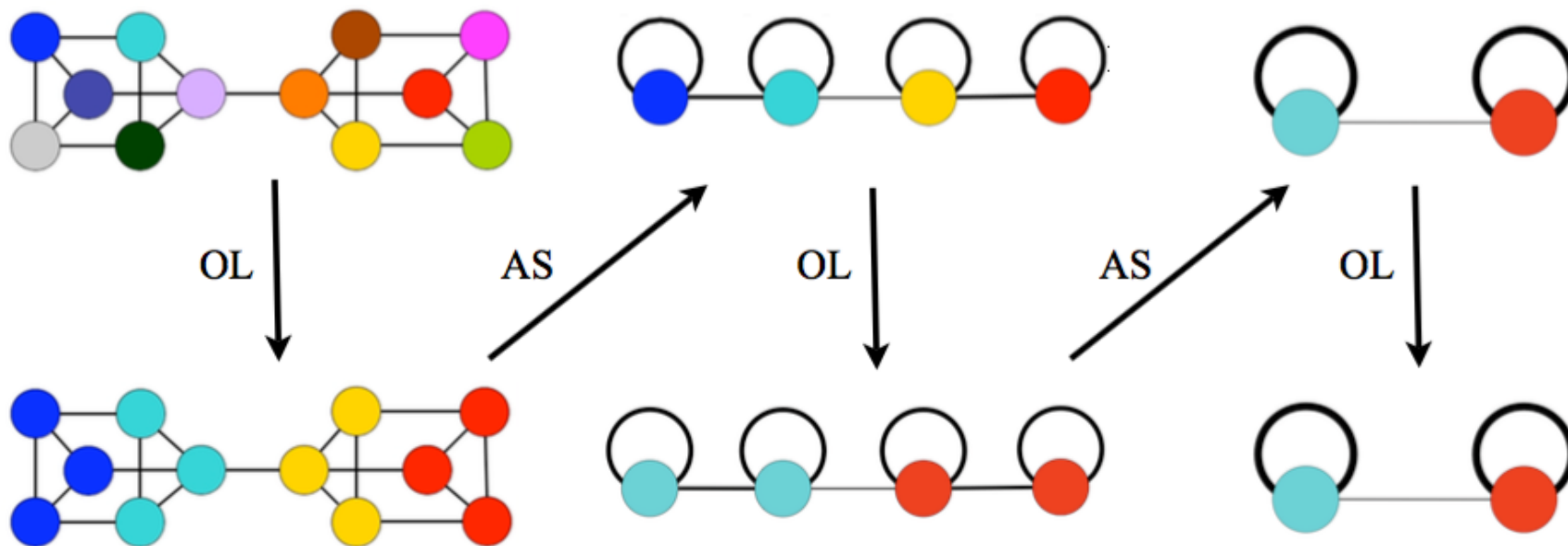
Multi-scale optimisation: local search first among neighbours, then among neighbouring communities, etc.



Hierarchical representation



Optimization: Greedy optimization



Partition initiale
(12 communautés, $Q=-0.08$)



Après la première passe
(4 communautés, $Q=0.38$)



Après la seconde passe
(2 communautés, $Q=0.45$)



Louvain

Algorithm 1 Pseudo-code of the community detection algorithm.

```
1: Community detection  $G$  initial graph
2: repeat
3:   Place each vertex of  $G$  into a single community
4:   Save the modularity of this decomposition
5:   while there are moved vertices do
6:     for all vertex  $n$  of  $G$  do
7:        $c \leftarrow$  neighboring community maximizing the modularity increase
8:       if  $c$  results in a strictly positive increase then
9:         move  $n$  from its community to  $c$ 
10:      end if
11:    end for
12:  end while
13:  if the modularity reached is higher than the initial modularity, then
14:     $end \leftarrow false$ 
15:    Display the partition found
16:    Transform  $G$  into the graph between communities
17:  else
18:     $end \leftarrow true$ 
19:  end if
20: until  $end$ 
```

Louvain

The efficiency of the algorithm partly resides in the fact that the variation of modularity Δ_{ij} obtained by moving a vertex i from its community to the community of one of its neighbors j can be calculated with only **local** information. In practice, the variation of modularity is calculated by removing i from its community $\Delta_{remove;i}$ (this is only done once) then inserting it into the community of j $\Delta_{insert;ij}$ for each neighbor j of i . The variation is therefore:

$$\Delta_{ij} = \Delta_{remove;i} + \Delta_{insert;ij}.$$

Let us calculate the variation of modularity when a vertex x is removed from its community. Assume that x is not alone in its community (the opposite case is trivial). By removing x from its community, the size of the community of x is decreased $C_x \rightarrow C_x \setminus \{x\}$ and a new community only containing x is created C'_x . The original modularity is:

$$\begin{aligned}
 Q &= \sum_C \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \\
 &= \sum_{C \neq C_x} \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] + \frac{1}{2m} \sum_{i,j \in C_x} \left[A_{ij} - \frac{k_i k_j}{2m} \right], \tag{7}
 \end{aligned}$$

and after removing the vertex x from C_x , the modularity becomes:

$$\begin{aligned}
 Q' &= \sum_{C \neq C_x} \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] + \frac{1}{2m} \sum_{i,j \in C_x \setminus \{x\}} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \\
 &\quad + \frac{1}{2m} \left[A_{xx} - \frac{k_x^2}{2m} \right] \\
 &= Q - \frac{1}{m} \sum_{i \in C_x \setminus \{x\}} \left[A_{ix} - \frac{k_i k_x}{2m} \right], \tag{8}
 \end{aligned}$$

where we used the fact that A_{ij} is symmetric. The modularity variation is given by:

$$\Delta_{remove} = Q' - Q = -\frac{1}{m} \sum_{i \in C_x \setminus \{x\}} \left[A_{ix} - \frac{k_i k_x}{2m} \right]. \tag{9}$$

Let us consider the situation where a vertex x is alone in a community and where it is moved into another community C_1 . The original modularity is:

$$\begin{aligned}
Q &= \sum_C \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \\
&= \sum_{C \neq (C_x, C_1)} \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] + \frac{1}{2m} \sum_{i,j \in C_1} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \\
&\quad + \frac{1}{2m} \left[A_{xx} - \frac{k_x^2}{2m} \right],
\end{aligned} \tag{10}$$

and after movement of x to C_1 , which becomes C'_1 , the modularity becomes:

$$\begin{aligned}
Q' &= \sum_{C \neq C'_1} \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] + \frac{1}{2m} \sum_{i,j \in C'_1} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \\
&= \sum_{C \neq C'_1} \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] + \frac{1}{2m} \sum_{i,j \in C_1} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \\
&\quad + \frac{1}{m} \sum_{i \in C_1} \left[A_{ix} - \frac{k_i k_x}{2m} \right] + \frac{1}{2m} \left[A_{xx} - \frac{k_x^2}{2m} \right].
\end{aligned} \tag{11}$$

The modularity variation is given by:

$$\Delta_{insert} = Q' - Q = \frac{1}{m} \sum_{i \in C_1} \left[A_{ix} - \frac{k_i k_x}{2m} \right]. \tag{12}$$

In both cases, whether it concerns removal or insertion, the calculations of variations are performed using only local information on x and its neighbors.

Optimization: Greedy optimization

Louvain: multi-scale, agglomerative and greedy

Very fast: $O(N)$ in practice. The only limitation being the storage of the network in main memory

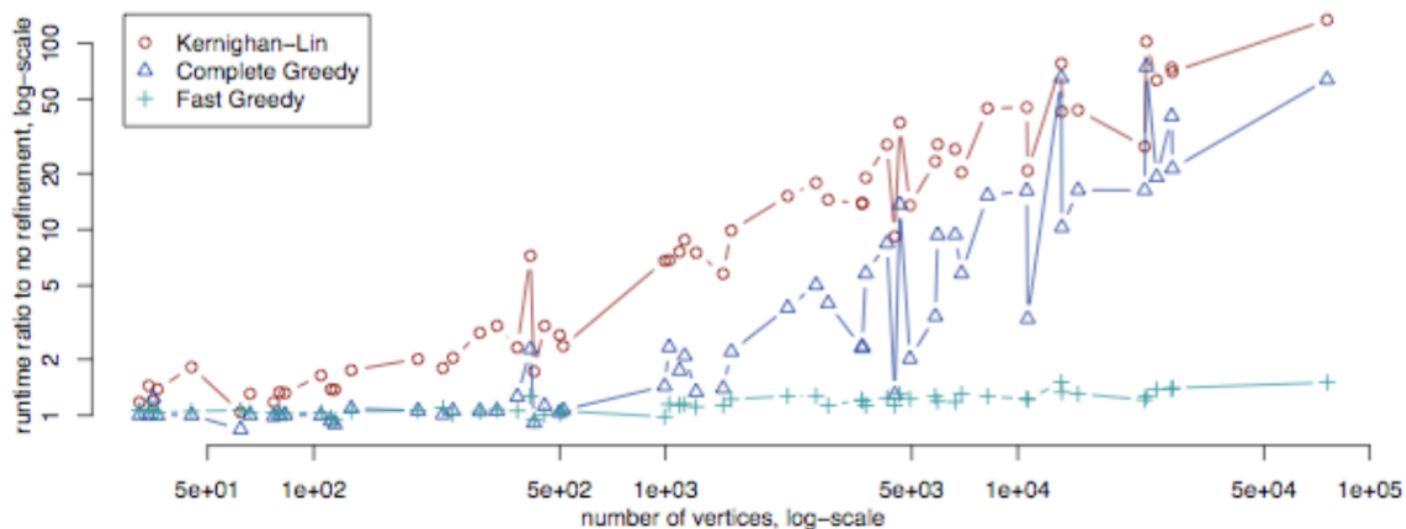
Good accuracy (among greedy methods)

	Karate	Arxiv	Internet	Web nd.edu	Phone	Web uk-2005	Web WebBase 2001
Nodes/links	34/77	9k/24k	70k/351k	325k/1M	2.04M/5.4M	39M/783M	118M/1B
CNM	.38/0s	.772/3.6s	.692/799s	.927/5034s	-/-	-/-	-/-
PL	.42/0s	.757/3.3s	.729/575s	.895/6666s	-/-	-/-	-/-
WT	.42/0s	.761/0.7s	.667/62s	.898/248s	.553/367s	-/-	-/-
Our algorithm	.42/0s	.813/0s	.781/1s	.935/3s	.76/44s	.979/738s	.984/152mn

V.D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech., P10008, 2008.

How to test the methods?

Test the heuristics: what is the value of Q obtained for different algorithms? Time complexity?



graph	size	subdivision	coarsening	local search	math prog	SS+ML
karate [42]	34	[29] .419	[41] .4198	[12] .4188	[1] .4197	.4197
dolphins [22]	62	[29] .4893	[31] .5171	[33] .5285	[40] .5285	.5276
polBooks [21]	105	[29] .3992	[37] .5269	[4] .5204	[1] .5272	.5269
afootball [14]	115	[39] .602	[41] .605	[4] .6045	[1] .6046	.6002
jazz [15]	198	[29] .442	[9] .4409	[12] .4452	[1] .445	.4446
celeg_metab [12]	453	[29] .435	[36] .450	[12] .4342	[1] .450	.4452
email [17]	1133	[29] .572	[9] .5569	[12] .5738	[1] .579	.5774
Erdos02 [16]	6927	[29] .5969	[32] .6817	[33] .7094		.7162
PGP_main [5]	11k	[29] .855	[9] .7462	[12] .8459		.8841
cmat03_main [25]	28k	[29] .723	[41] .761	[12] .6790		.8146
ND_edu [2]	325k		[7] .927	[4] .935		.9509

How to test the methods?

Comparison with real-world data: do modules reveal nodes having similar meta-data?

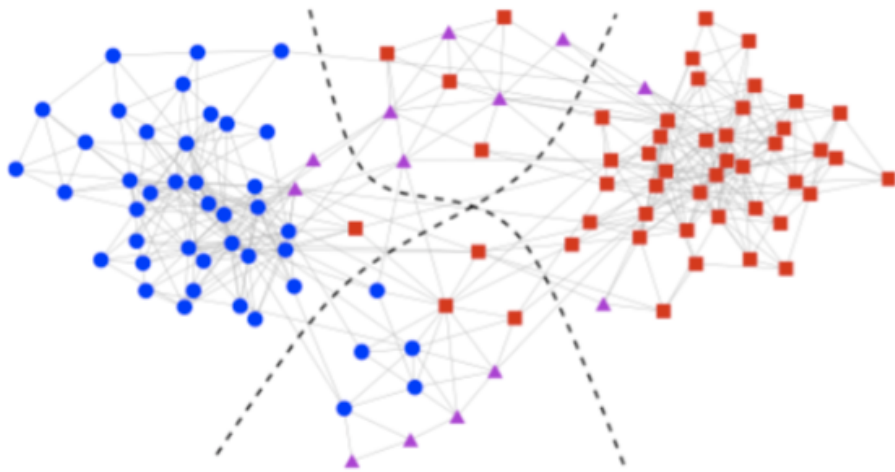
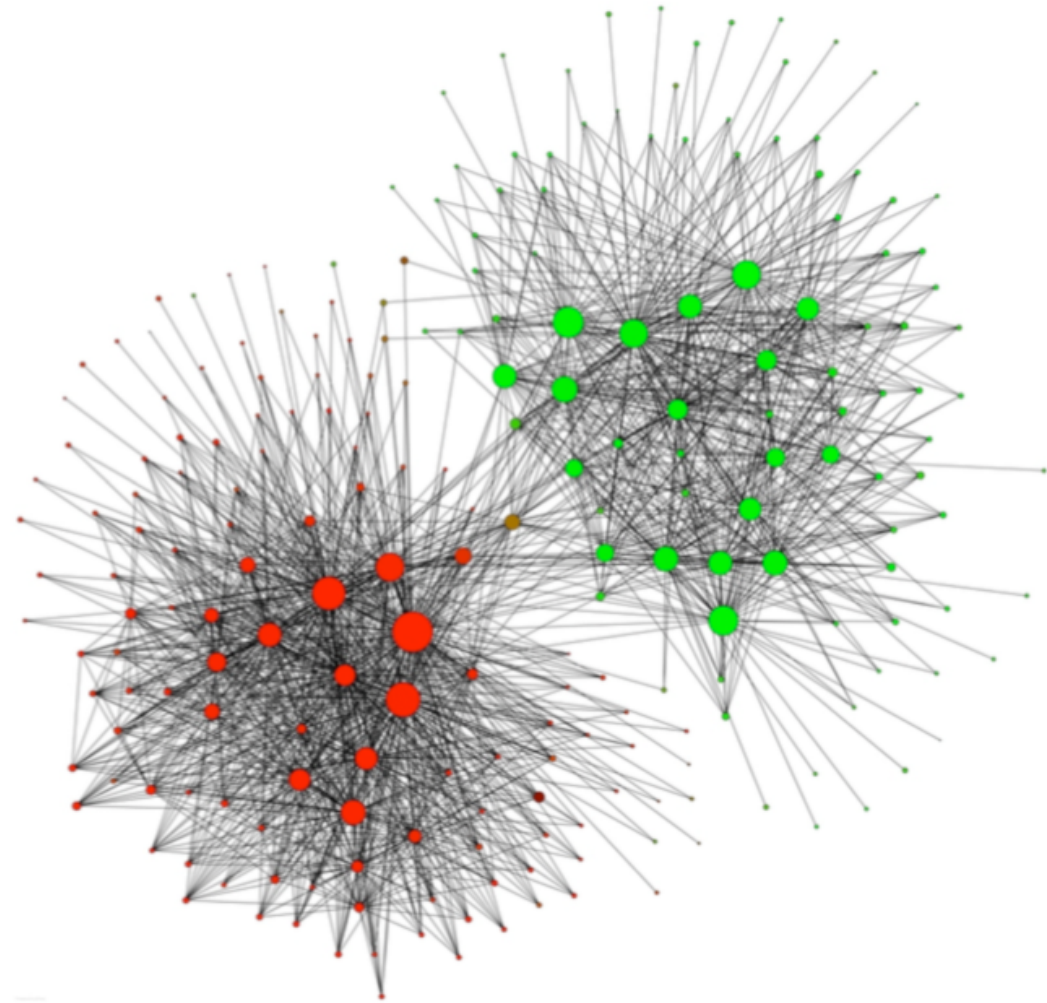


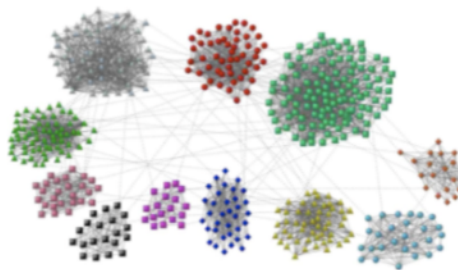
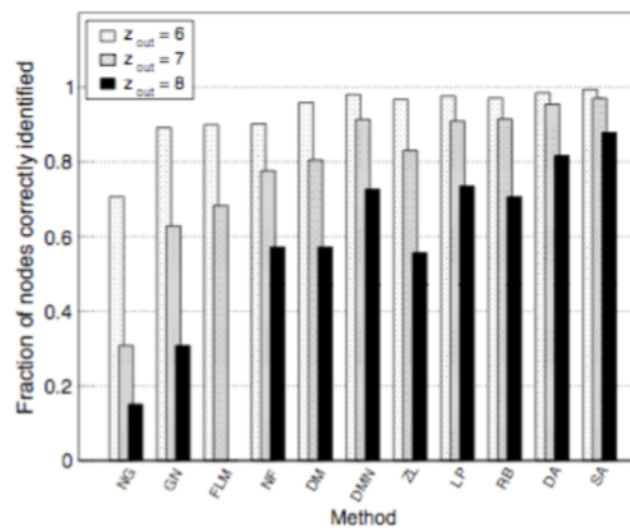
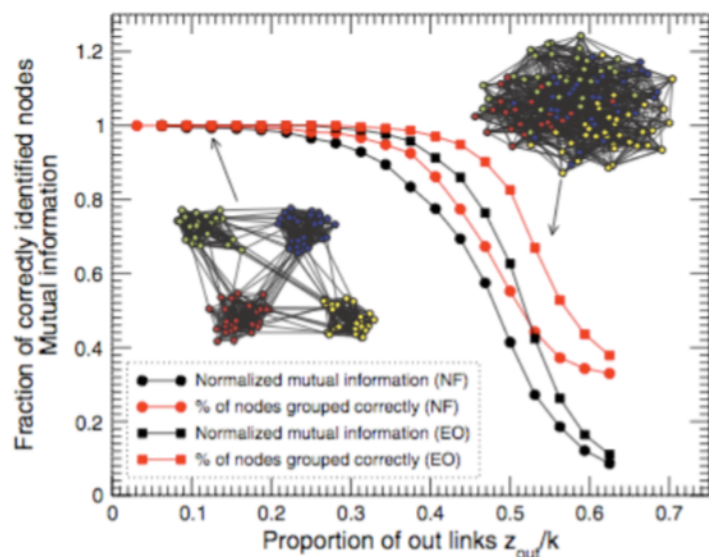
FIG. 3: Krebs' network of books on American politics. Vertices represent books and edges join books frequently purchased by the same readers. Dotted lines divide the four communities found by our algorithm and shapes represent the political alignment of the books: circles (blue) are liberal, squares (red) are conservative, triangles (purple) are centrist or unaligned.



But: meta-data are often unknown. No insurance that modular organization coincides with semantic/cultural organisation

How to test the methods?

Benchmarks: artificial networks with known community structure.



But: random networks (their structure is quite different from real-world networks). In the way the benchmark is built, there is a (hidden) choice for what good partitions should be

Comparing partitions

A popular way to compare two partitions \mathcal{P}_1 and \mathcal{P}_2 is the normalized variation of information⁴³

$$\hat{V}(\mathcal{P}_1, \mathcal{P}_2) \equiv \frac{H(\mathcal{P}_1|\mathcal{P}_2) + H(\mathcal{P}_2|\mathcal{P}_1)}{\log N}, \quad (15)$$

where $H(\mathcal{P}_1|\mathcal{P}_2)$ is the conditional entropy of the partition \mathcal{P}_1 given \mathcal{P}_2 , namely the additional information needed to describe \mathcal{P}_1 once \mathcal{P}_2 is known. The conditional entropy is defined in a standard way for the joint distribution $P(C_1, C_2)$ that a node belongs to a community C_1 of \mathcal{P}_1 and to a community C_2 of \mathcal{P}_2 . The normalized variation of information, which has been shown to be a true metric on the space of partitions, belongs to the interval $[0, 1]$ and vanishes only when the two partitions are identical.

How to test the methods?

Ajk the people!

about

[EN](#) [ES](#) [FR](#) [PT](#)

On Facebook, you only have *friends*. In real life however, these friends are part of different groups: family, close friends, co-workers, childhood friends, etc. The way you communicate with them likely depends on the group they belong to. And yet, on Facebook, you reveal **everything** to **everybody**.

There are ways to chose those with whom you want to share some information (be it a picture, a status update, a link, etc.), but we think that those are too complex. They require you to add your friends one by one to friend lists, which might take a tremendous amount of time if you have hundreds of contacts.

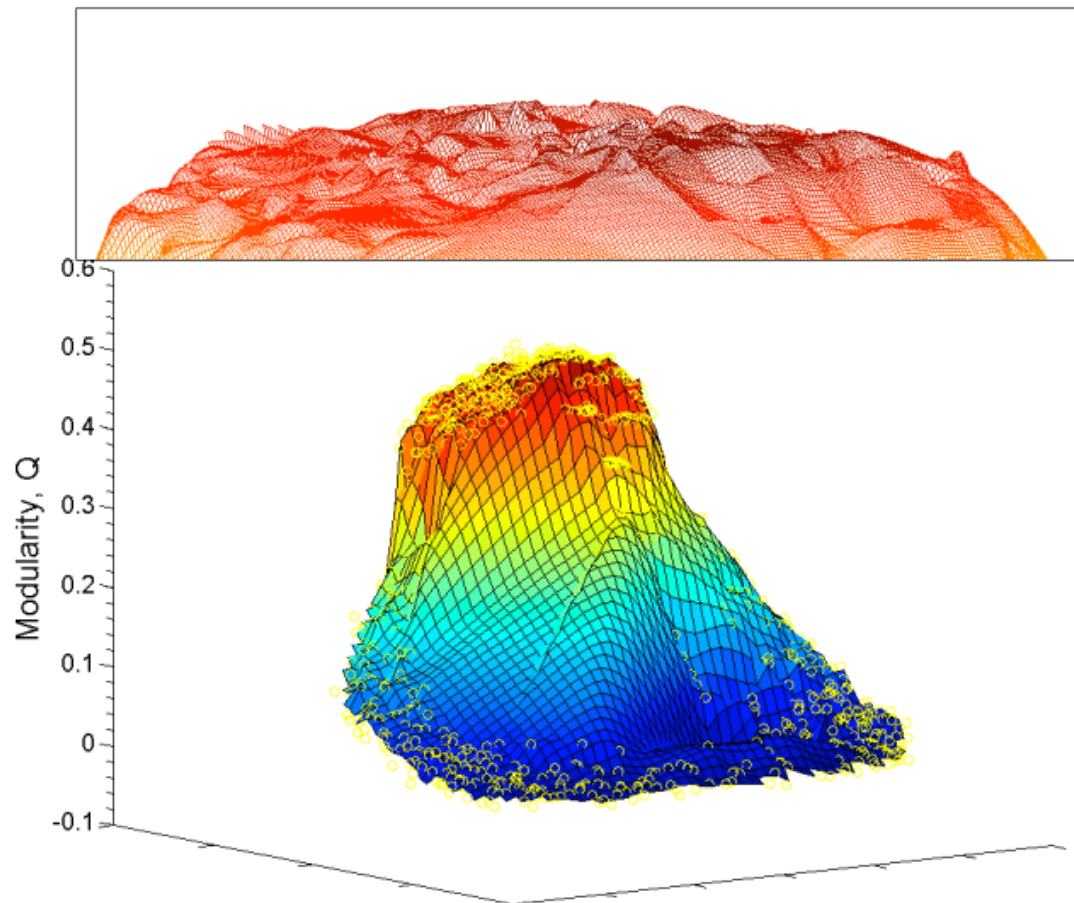
We are working on a way to automatically generate those groups of friends, using only the information on "who knows who". By

fellows

start

Limitations of modularity (1)

The modularity landscape tends to be very rugged, with many partitions, possibly very different, having similar value of modularity.

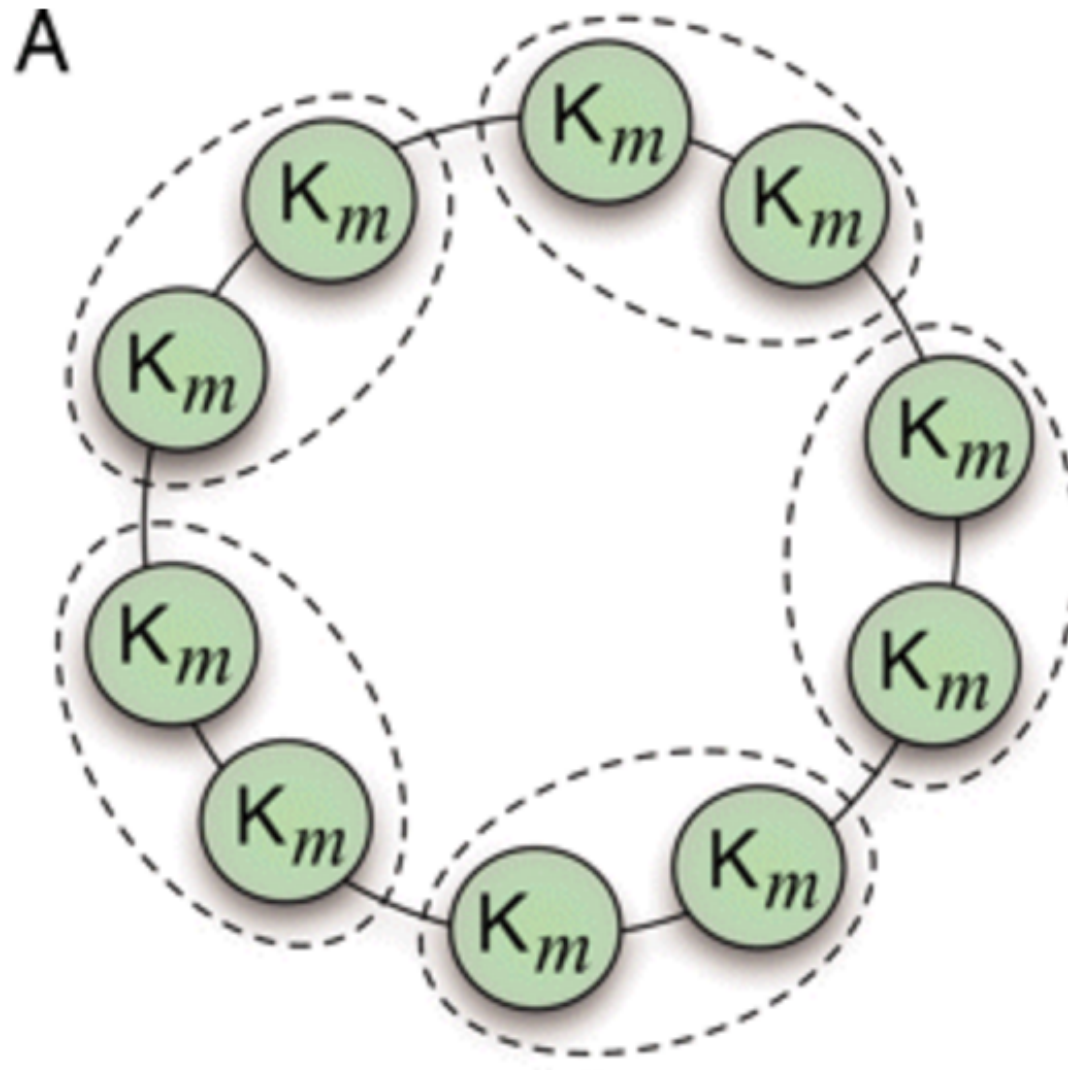


Limitations of modularity (2)

Second, Q exhibits a resolution limit, because using Q it is impossible to detect dense clusters of nodes that are smaller than a certain scale [Fortunato and Barthélemy (2007)]. The resolution limit originates from the dependency of the null model on $2M$. The dependency decreases when the number of links, M , is increased. Then, modularity maximisation tends to favour larger communities. In the limit $M \rightarrow \infty$, the null model is neglected and modularity optimisation simply uncovers the connected components. Modularity-based methods implicitly favour communities having a certain size, depending on the size of the entire network, not only on its internal structure.

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

Limitations of modularity (2)



Limitations of modularity (3)

Finally, although modularity allows us to compare partitions of the same network, it is by no means intended to compare modularity values of different networks. Therefore, Q should not be used as a measure of the modularity of a network. For instance, the modularity of the best partition of a random network tends to $Q = 1$ when the network is sufficiently large, whereas this network is by no means modular [Guimerà *et al.* (2004)].

Connections with the field of statistical inference.

The model most commonly used in this context is the so-called stochastic block model, which is a random graph model of a network with community structure [12, 23, 24]. One takes some number n of nodes, initially without any edges, and divides them into q groups in some way, with g_i being the group to which node i is assigned, as previously. Then one places edges between nodes independently at random, with the probability, which we denote ω_{rs} , of an edge between a particular pair of nodes depending on the groups r and s to which the nodes belong. Thus there is a symmetric $q \times q$ matrix of parameters ω_{rs} which determine the probabilities of edges within and between every pair of groups. If the diagonal elements ω_{rr} of this matrix are larger than the off-diagonal elements, then networks generated by the model have a higher probability of edges within groups than between them and hence have traditional community structure.

Connections with the field of statistical inference.

Maximisation of the log-likelihood of observing the network given the model parameter.

In the case of the degree-corrected stochastic block model.

$$\log P(\mathbf{A}|\mathbf{\Omega}, \mathbf{g}) = \frac{1}{2} \sum_{i,j} \left(A_{ij} \log \omega_{g_i g_j} - \frac{k_i k_j}{2m} \omega_{g_i g_j} \right)$$

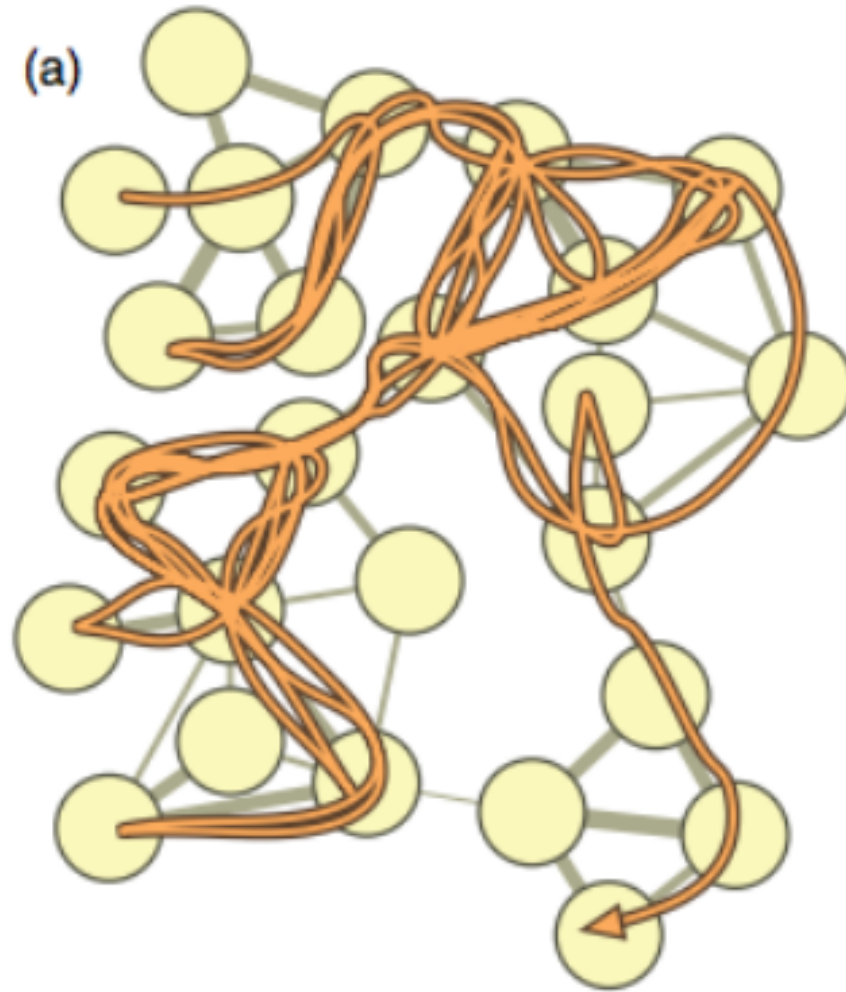
The planted partition model [30, 31] is a special case of the stochastic block model in which the parameters ω_{rs} describing the community structure take only two different values:

$$\omega_{rs} = \begin{cases} \omega_{\text{in}} & \text{if } r = s, \\ \omega_{\text{out}} & \text{if } r \neq s. \end{cases} \quad (11)$$

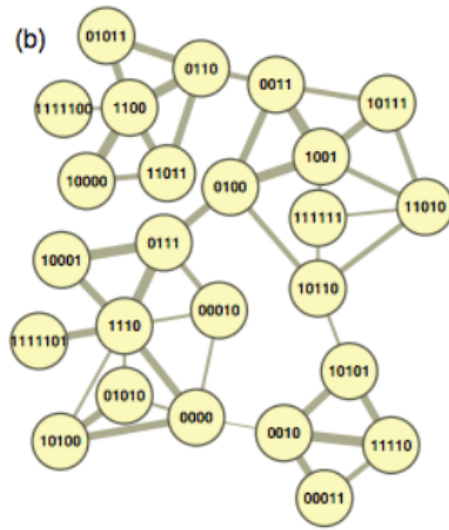
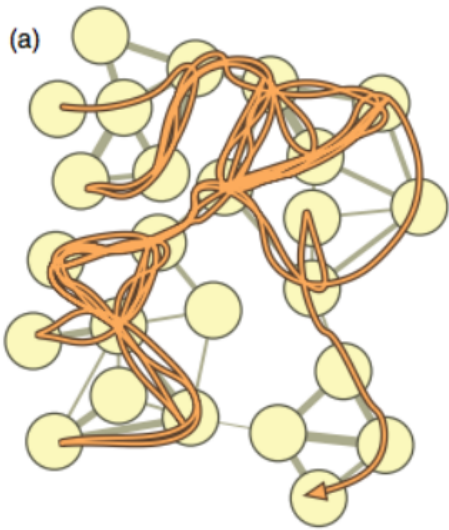
Optimal partition of the planted partition model = optimal partition of modularity:-)

Community detection in networks: Modularity optimization and maximum likelihood are equivalent, M. E. J. Newman

Dynamics as way to uncover communities

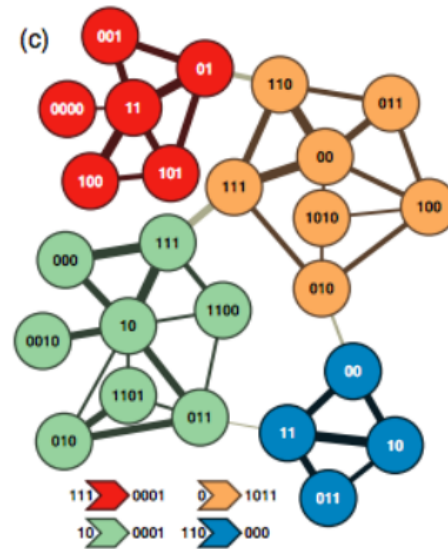


Dynamics as way to uncover communities



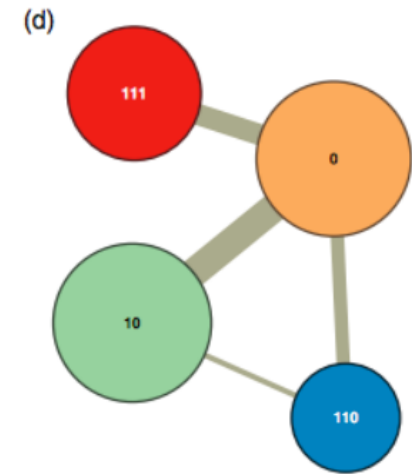
```

1111100 1100 0110 11011 10000 11011 0110 0011 10111 1001
0011 1001 0100 0111 10001 1110 0111 10001 0111 1110 0000
1110 10001 0111 1110 0111 1110 111101 1110 0000 10100 0000
1110 10001 0111 0100 10110 11010 10111 1001 0100 1001 10111
1001 0100 1001 0100 0011 0100 0011 0110 11011 0110 0011 0100
1001 10111 0011 0100 0111 10001 1110 10001 0111 0100 10110
111111 10110 10101 11110 00011
    
```



```

111 0000 11 01 101 100 101 01 0001 0 110 011 00 110 00 111
1011 10 111 000 10 111 000 111 10 011 10 000 111 10 111 10
0010 10 011 010 011 10 000 111 0001 0 111 010 100 011 00 111
00 011 00 111 00 111 110 111 110 1011 111 01 101 01 0001 0 110
111 00 011 110 111 1011 10 111 000 10 000 111 0001 0 111 010
1010 010 1011 110 00 10 011
    
```



```

111 0000 11 01 101 100 101 01 0001 0 110 011 00 110 00 111
1011 10 111 000 10 111 000 111 10 011 10 000 111 10 111 10
0010 10 011 010 011 10 000 111 0001 0 111 010 100 011 00 111
00 011 00 111 00 111 110 111 110 1011 111 01 101 01 0001 0 110
111 00 011 110 111 1011 10 111 000 10 000 111 0001 0 111 010
1010 010 1011 110 00 10 011
    
```

The Map Equation: coding trajectories

Imagine a random walk on a given network. If the network has community structure, the random walker would wander within a community for a long time before crossing a bridge to a different community. A straightforward way to describe the trajectory of the random walk is to write down the visited nodes in an ordered list, e.g., $v_1, v_4, v_1, v_7, v_3, \dots$. The amount of information required to express the trajectory is estimated as follows. We code each node into a finite binary sequence, i.e., a code word, and concatenate the code words. For example, if v_1, v_3, v_4 , and v_7 are coded into 000, 010, 011 and 110, the aforementioned trajectory is coded into 000011000110010 \dots . For unique decoding, the code has to be prefix-free. In other words, a code word must not be a prefix (i.e., initial segment) of another code word. For example, if v_1 and v_2 are coded into 000 and 0001, respectively, the code is not prefix-free because 000 is an initial segment of 0001.

The Map Equation

The **Huffman code** is a prefix-free code that encodes symbols separately and generally yields short binary sequences to represent trajectories of the random walk. It assigns **a short code word to a frequently visited node** and vice versa. The mean code word length per step of the random walk is given by $\sum_{i=1}^N p_i^* L(i)$, where p_i^* is the stationary density of the random walk at node v_i and $L(i)$ is the length of the code word for node v_i .

When the symbols (v_i in our case) appear **independently**, the Huffman code often yields a code length that is close to the theoretical lower bound obtained by the Shannon entropy, which is

$$H = - \sum_{i=1}^N p_i^* \log p_i^* \quad (3.85)$$

per step. However, the sequence of nodes is **correlated** in time because it is produced by the random walk. Then, an alternative coding scheme may lessen the mean code length. In particular, we can design a **two-layered** variant of the Huffman code to exploit the community structure of the network. Because there are less nodes in a community CM_i as compared to the entire network, we can express a trajectory within CM_i with a shorter, different Huffman code, which is local to CM_i . Based on this observation, we rebuild the Huffman code as follows.

The Map Equation

- (1) When the walker enters community CM_i , a code word to represent this entry event is issued.
- (2) The walker wanders within CM_i . The trajectory of the walker during this period is encoded by concatenating the code words corresponding to the sequence of the visited nodes. The sequence of these code words is simply placed after the code word produced in the previous step (i.e., entry to CM_i). It should be noted that the intra-community code words make sense only within CM_i . A different community $CM_{i'}$ ($i' \neq i$) may use the same code word as the one used within CM_i to represent a different node in $CM_{i'}$.
- (3) The walker exits CM_i . This event is represented by a special code word, which is concatenated after the sequence of code words produced so far.
- (4) The exit from CM_i implies that the walker immediately enters a different community, CM_j . Therefore, a code word to notify that the walker has entered CM_j is issued. Then, the code words local to CM_j are used until the walker exits CM_j . We repeat this procedure.

The Map Equation

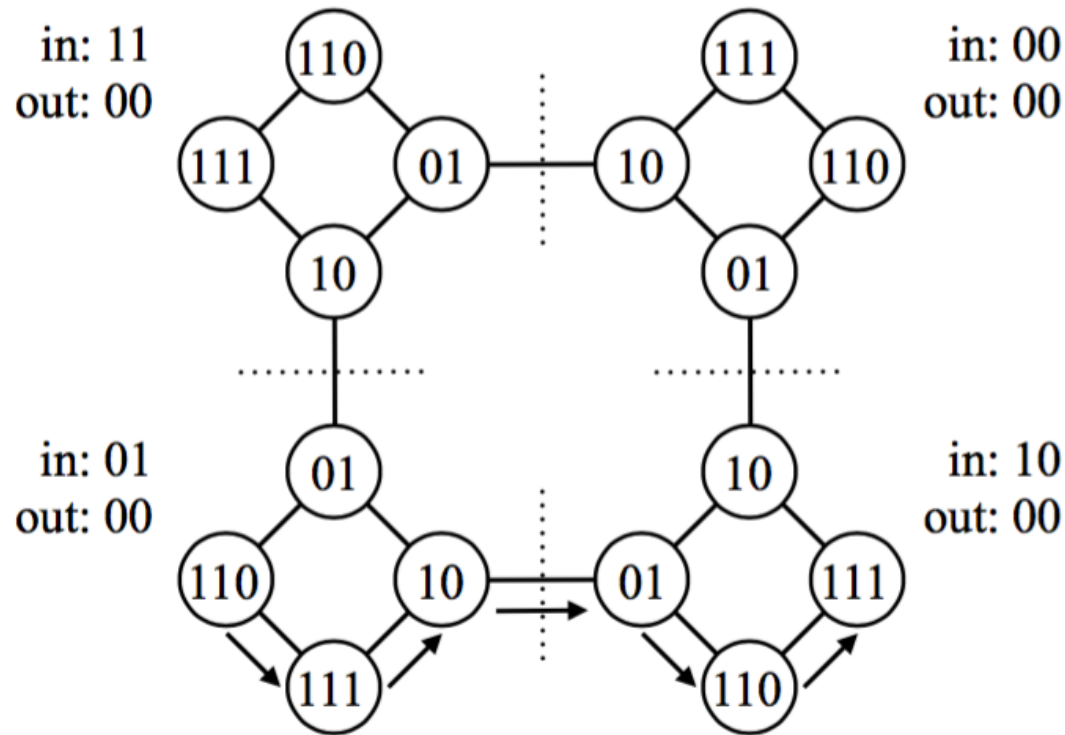


Fig. 3.6 Optimal partitioning according to Infomap and the resulting code words. This example is based on a demo applet available at Martin Rosvall's website <http://www.mapequation.org/apps/MapDemo.html>.

the trajectory shown by the arrows in the figure is encoded into 0111011110001001110111. The first 01 indicates that the walk starts in the left-bottom community, and the 110 that follows indicates that the walk starts at the 110 node in this community. 0010 in the middle indicates that the walk exits this community (by the code word 00) and immediately enters the community to the right (by the code word 10).

The Map Equation

In contrast to the original Huffman code, we have to invest $2N_{\text{CM}}$ code words to mark the entry to and exit from a community. However, we can save the code length when the walker wanders in a community, which occupies a majority of steps. Overall, the mean code length is expected to be smaller with the two-layer code in the presence of community structure. In order to detect communities in practice, there is no need for devising the optimal code of a given partition. Infomap instead proceeds by optimising a quality function, called the map equation, which generalises Eq. (3.85). The resulting quality function provides a theoretical limit of how concisely we can specify a walk in the network using a given partition. The optimisation is then performed by a greedy algorithm similar to the one used for maximising modularity (Section 3.10.1), with additional fine- and coarse-graining steps carried out for improving the partitioning.

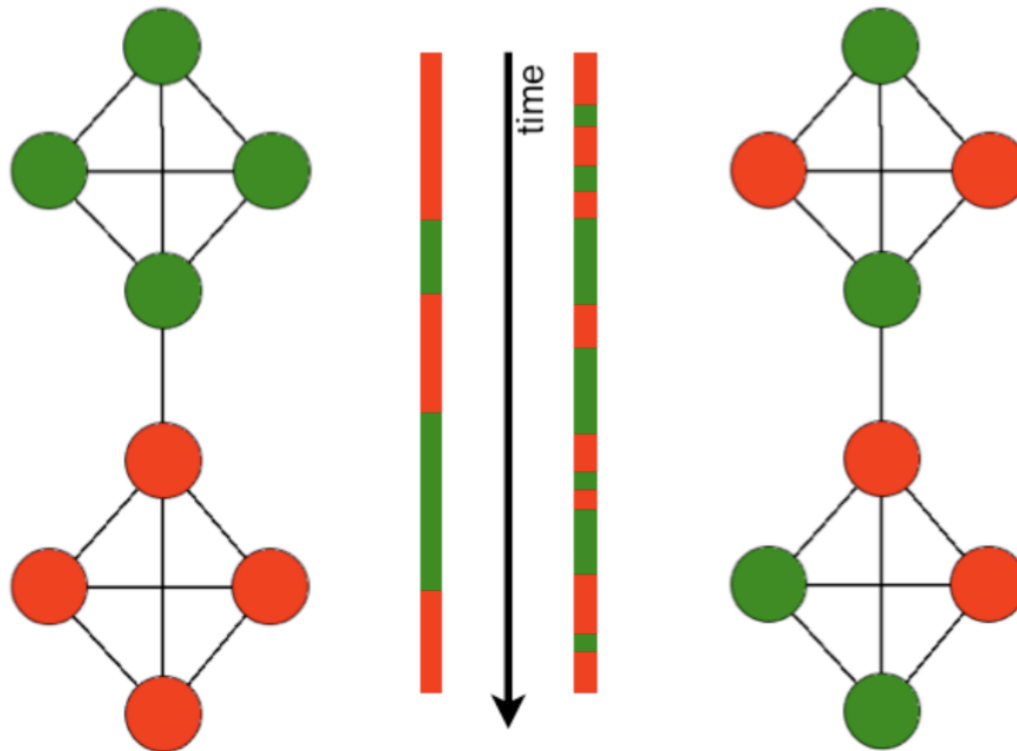
$$L(\mathbf{M}) = q_{\text{in}} H(\mathcal{Q}) + \sum_{\mathcal{C}} p_{\text{out}}^{\mathcal{C}} H(\mathcal{C}^{\mathcal{C}})$$

Minimizing the Map Equation provides the partition giving the best (most efficient) coding scheme

Markov stability

The quality of a partition is determined by the patterns of a flow within the network: a flow should be trapped for long time periods within a community before escaping it.

The stability of a partition is defined by the statistical properties of a random walker moving on the graph



Markov stability

The quality of a partition is determined by the patterns of a flow within the network: a flow should be trapped for long time periods within a community before escaping it.

The stability of a partition is defined by the statistical properties of a random walker moving on the graph

$$R(t) = \sum_{C \in \mathcal{P}} P(C, t_0, t_0 + t) - P(C, t_0, \infty)$$

$$P(C, t_0, t_0 + t)$$

probability for a walker to be in the same community at times t_0 and $t_0 + t$ when the system is at equilibrium

$$P(C, t_0, \infty)$$

probability for two independent walkers to be in C (ergodicity)

Markov stability versus Modularity

Let us consider a random walk on an undirected network:

$$p_{i;n+1} = \sum_j \frac{A_{ij}}{k_j} p_{j;n} \quad \xrightarrow{\text{equilibrium}} \quad p_i^* = k_i/2m$$

$$R(1) = \sum_{i,j} \left[\frac{A_{ij}}{k_j} \frac{k_j}{2m} - \frac{k_i k_j}{(2m)^2} \right] \delta(c_i, c_j)$$

Probability that a walker is in the same community initially and at time $t=1$

Same probability for independent walkers

$$R(1) = Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

Markov stability versus Modularity

Let us consider a random walk on an **directed** network:

$$p_{i;n+1} = \sum_j \frac{A_{ij}}{k_j^{\text{out}}} p_{j;n} \quad \xrightarrow{\text{equilibrium}} \quad p_i^* = \pi_i$$

$$R(1) = \sum_{i,j} \left[\frac{A_{ij}}{k_j^{\text{out}}} \pi_j - \pi_i \pi_j \right] \delta(c_i, c_j) \neq Q$$

Counting edges versus flows of probability

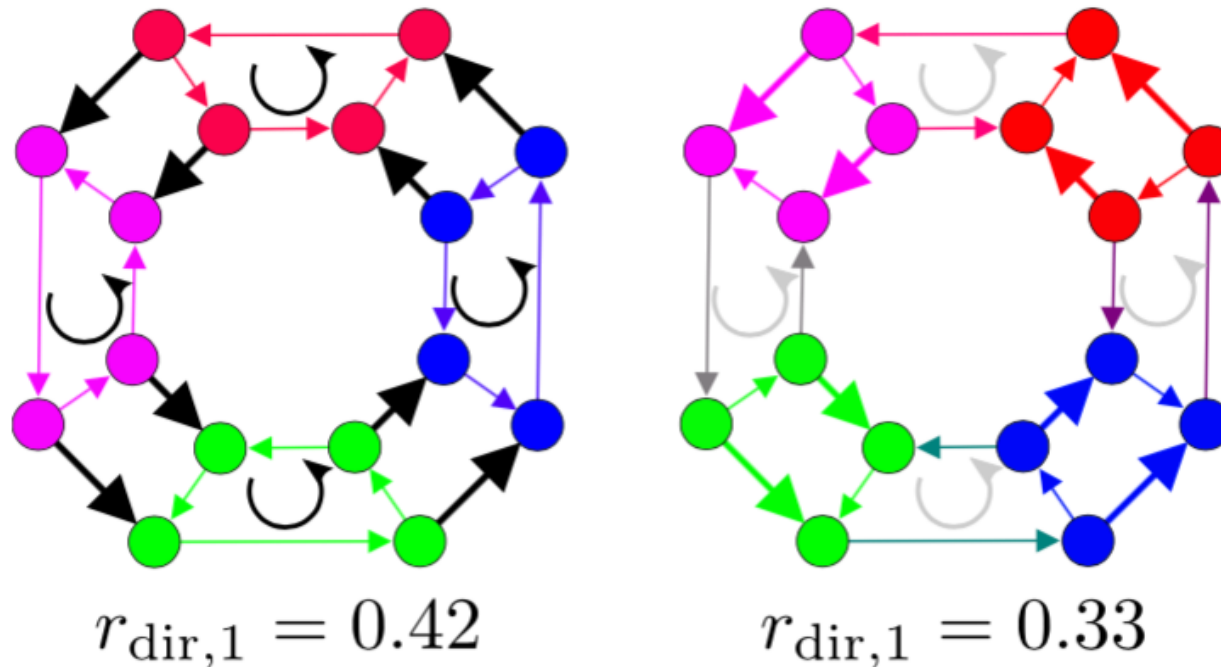
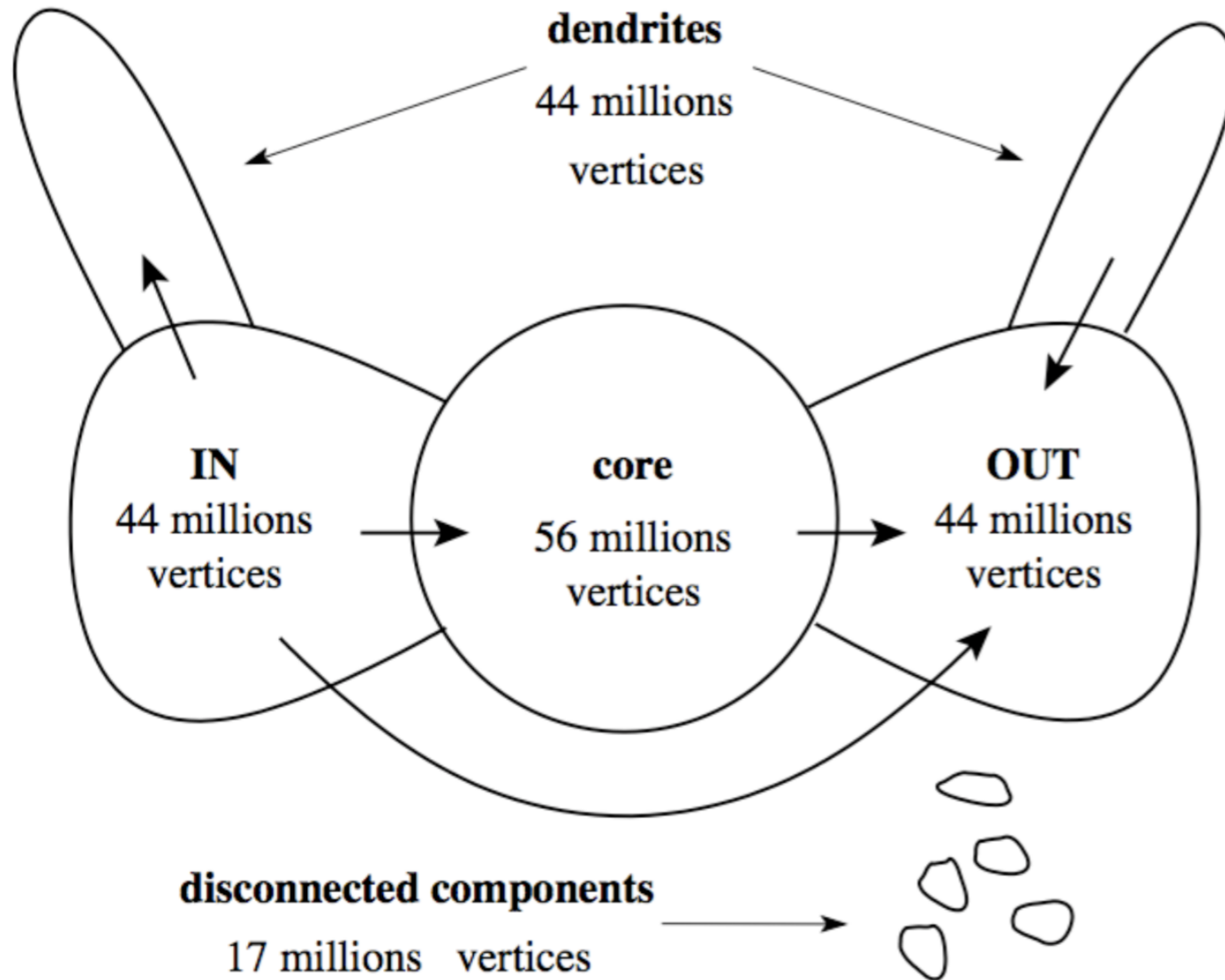


Fig. 4. Directed Markov Stability versus extensions of modularity. In this toy network [16], the weight of the bold links is twice the weight of the other links. The partition on the left (indicated by different colors) optimizes directed Markov Stability [34], which intrinsically contains the pagerank as a null model. The partition on the right instead optimizes an extension of modularity based on in- and out-degrees [64], [65]. Hence directed Markov Stability produces flow communities, whereas the extension of modularity ignores the effect of flows.

Counting versus flows



Time as a resolution parameter

Let us consider a continuous-time random walk with Poisson waiting times

$$\dot{p}_i = \sum_j \frac{A_{ij}}{k_j} p_j - p_i \quad \xrightarrow{\text{equilibrium}} \quad p_i^* = k_i / 2m$$

$$R(t) = \sum_{i,j} \left[\left(e^{t(B-I)} \right)_{ij} \frac{k_j}{2m} - \frac{k_i k_j}{(2m)^2} \right] \delta(c_i, c_j)$$

$$B_{ij} = A_{ij} / k_j$$

Probability that a walker is in the same community initially and at time t

Same probability for independent walkers

Time as a resolution parameter

Let us consider a continuous-time random walk with Poisson waiting times

$$R(0) = 1 - \sum_{i,j} \frac{k_i k_j}{(2m)^2} \delta(c_i, c_j) \quad \text{Communities = Single nodes}$$

$$R(t) \approx (1 - t)R(0) + tQ_C \equiv Q(t) \quad \text{Tuneable modularity of Reichart and Bornholdt}$$

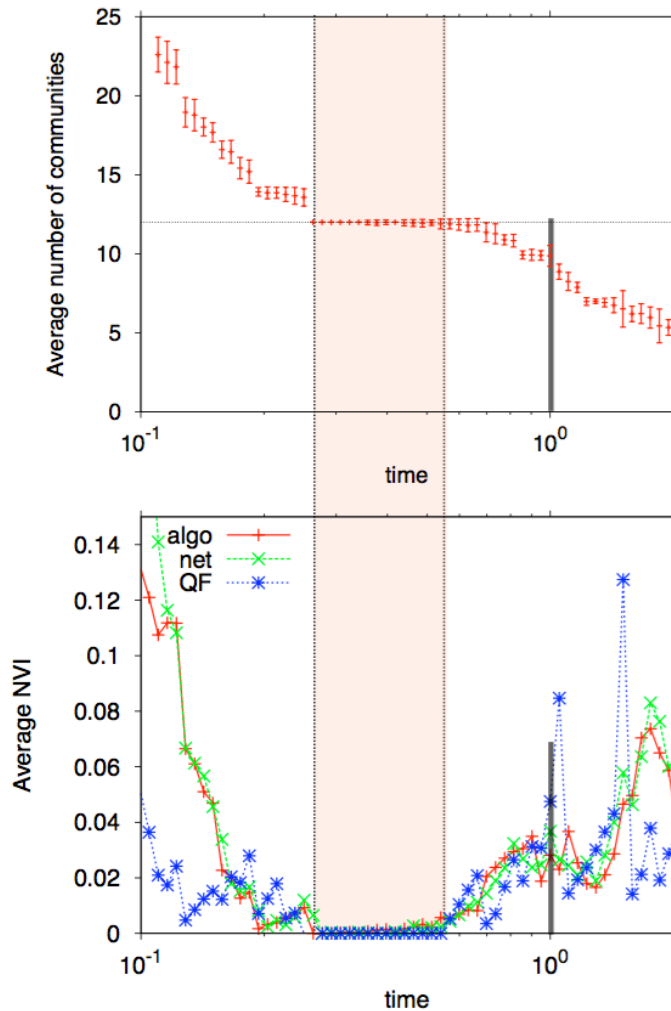
time



Asymptotically, two-way partition given by the Fiedler vector

In practice: selection of the significant scales?

football



algo: for each t , 100 optimizations of Louvain algorithm while changing the ordering of the nodes

$$\langle V \rangle_{\text{algo}}(t) = \frac{2}{T(T-1)} \sum_{i=1}^T \sum_{i'=i+1}^T \hat{V}(\mathcal{P}_i(t), \mathcal{P}_{i'}(t)).$$

net: for each t , 100 optimizations with a fixed algorithm but randomized modifications of the network

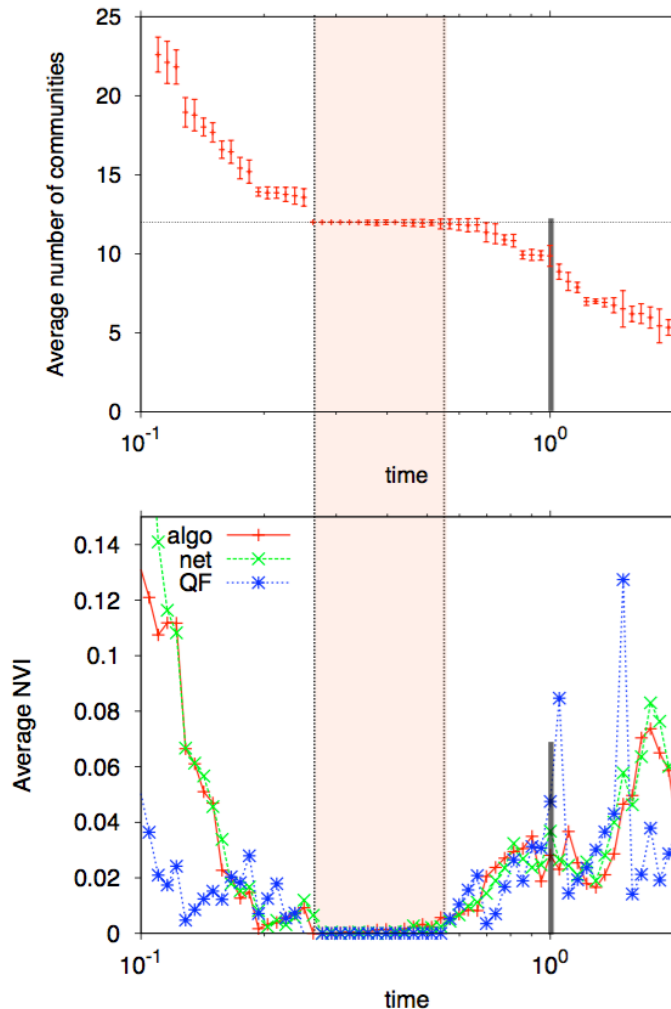
QF: for each t , one optimization. Partitions at 5 successive values of t are compared.

Compatible notions of robustness:

Lack of robustness \rightarrow high degeneracy in the landscape:
uncovered partitions are not to be trusted; wrong resolution

In practice: selection of the significant scales?

football



algo: for each t , 100 optimizations of Louvain algorithm while changing the ordering of the nodes

$$\langle V \rangle_{\text{algo}}(t) = \frac{2}{T(T-1)} \sum_{i=1}^T \sum_{i'=i+1}^T \hat{V}(\mathcal{P}_i(t), \mathcal{P}_{i'}(t)).$$

net: for each t , 100 optimizations with a fixed algorithm but randomized modifications of the network

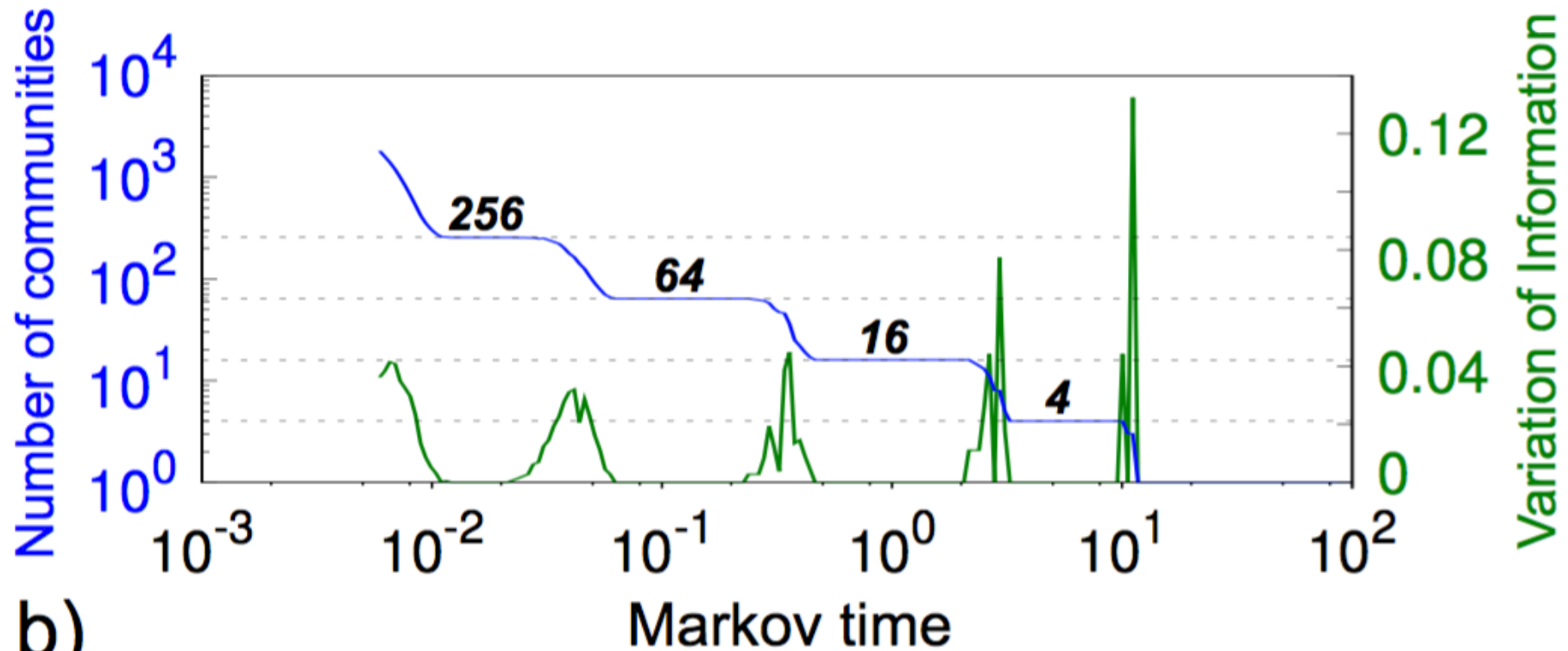
QF: for each t , one optimization. Partitions at 5 successive values of t are compared.

Compatible notions of robustness:

Lack of robustness \rightarrow high degeneracy in the landscape:
uncovered partitions are not to be trusted; wrong resolution

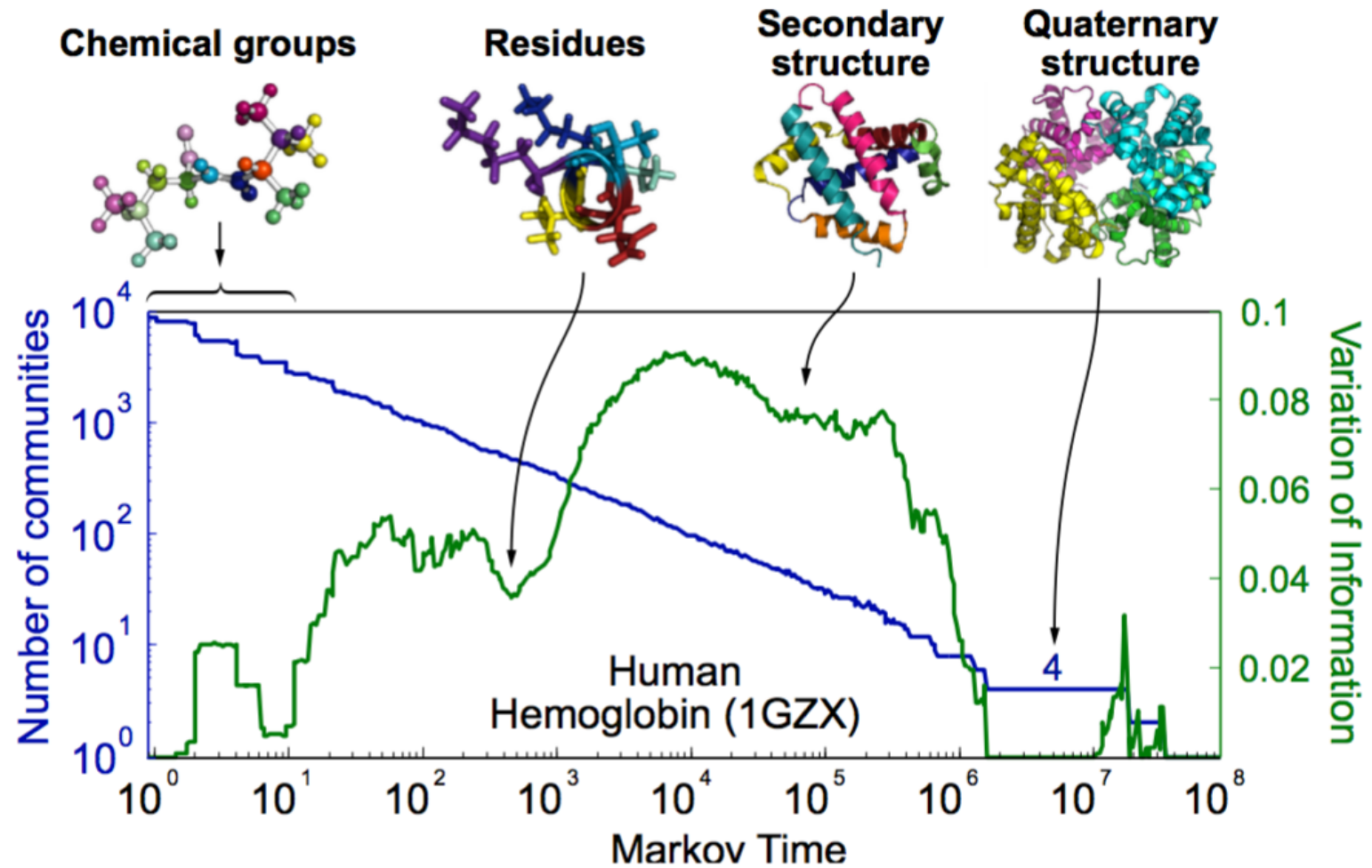
Time as resolution parameter

a)



b)

Time as resolution parameter



Time as resolution parameter

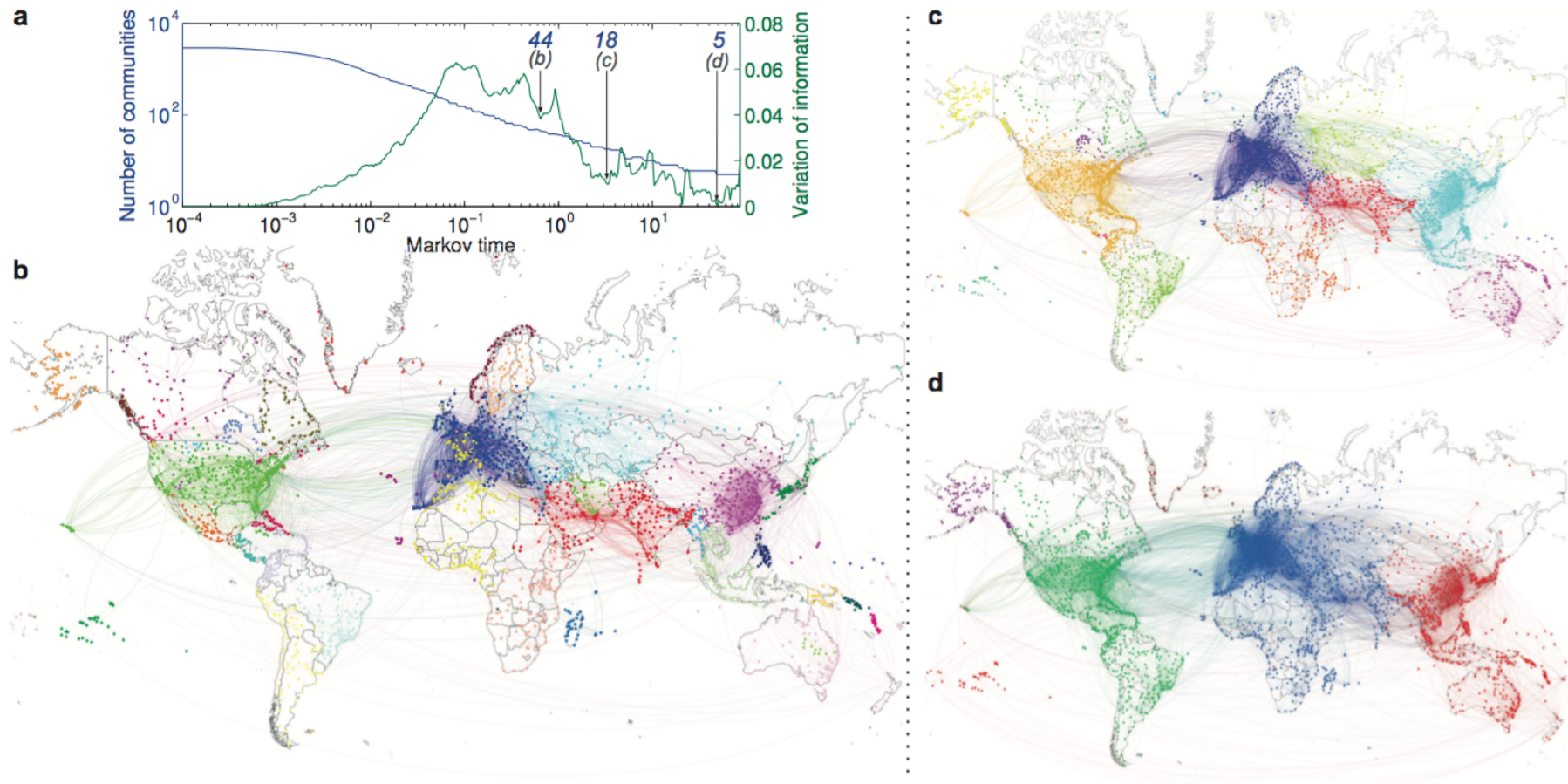
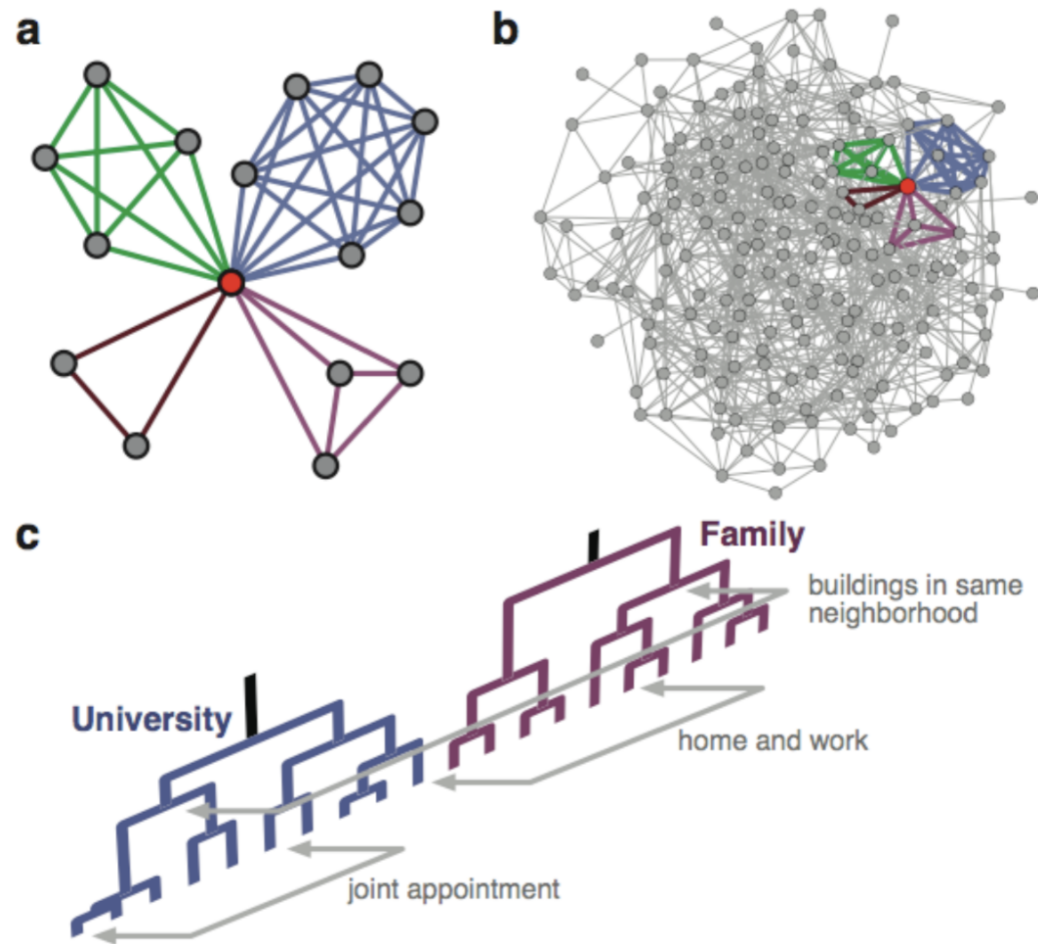


Fig. 8. Flow communities at multiple scales in an airport network. The airport network [82] contains $N = 2905$ nodes (airports) and 30442 weighted directed edges. The weights record the number of flights between airports (i.e., the network does not take into account passenger numbers, just the number of connections). Representative partitions at different levels of resolution with (b) 44, (c) 18 and (d) 5 communities are presented. The partitions correspond to dips in the normalized variation of information in (a) and show persistence across time (see Suppl. Info.).

Algorithms for memory networks: community detection

Edge partitions naturally provide overlapping communities.



Recent trends

Network inference: stochastic block models and model selection

Role detection and non-assortative communities

Non-backtracking random walks and detectability limit

Temporal communities

Reading list

General review

S. Fortunato. Community detection in graphs. *Phys. Rep.*, 486:75–174, 2010.

Non-backtracking random walks

F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborova, and P. Zhang. Spectral redemption in clustering sparse networks. *Proc. Natl. Acad. Sci. USA*, 110:20935–20940, 2013.

Algorithms for Higher Order Networks

M. Rosvall, A. V. Esquivel, A. Lancichinetti, J. D. West, and R. Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nat. Comm.*, 5:4630, 2014.

I. Scholtes, N. Wider, R. Pfitzner, A. Garas, C. J. Tessone, and F. Schweitzer. Causality-driven slow-down and speed-up of diffusion in non-Markovian temporal networks. *Nature Communications*, 5:5024, 2014.

Network Inference

Peixoto, T. P. and Rosvall, M. (2015). Modeling sequences and temporal networks with dynamic community structures, Preprint arXiv:1509.04740v1.

Generalized communities in networks, M. E. J. Newman, Tiago P. Peixoto, *Phys. Rev. Lett.* 115, 088701 (2015)

Detecting change points in the large-scale structure of evolving networks, L Peel, A Clauset, arXiv preprint arXiv:1403.0989

Stability

Stability of graph communities across time scales, JC Delvenne, SN Yaliraki, M Barahona, *Proceedings of the National Academy of Sciences* 107 (29), 12755-12760 (2010)

R. Lambiotte, J. C. Delvenne, and M. Barahona. Random walks, Markov processes and the multiscale modular organization of complex networks. *IEEE Trans. Netw. Sci. Eng.*, 1:76–90, 2014.

Codes

Louvain method in Python:

<http://perso.crans.org/aynaud/communities/>

The Map Equation (and its variations)

<http://www.mapequation.org/>

Markov stability

<https://github.com/michaelschaub/PartitionStability>

Network inference:

<https://graph-tool.skewed.de/static/doc/dev/community.html>