Community detection in networks

Elsa Arcaute

Community detection algorithms

Last time:

→Networks present sometimes a natural division into groups: friendship, similarity measures, chemical interchange among molecules, etc.

There are many different algorithms to separate nodes within a network into different communities. These are based on the following assumptions:

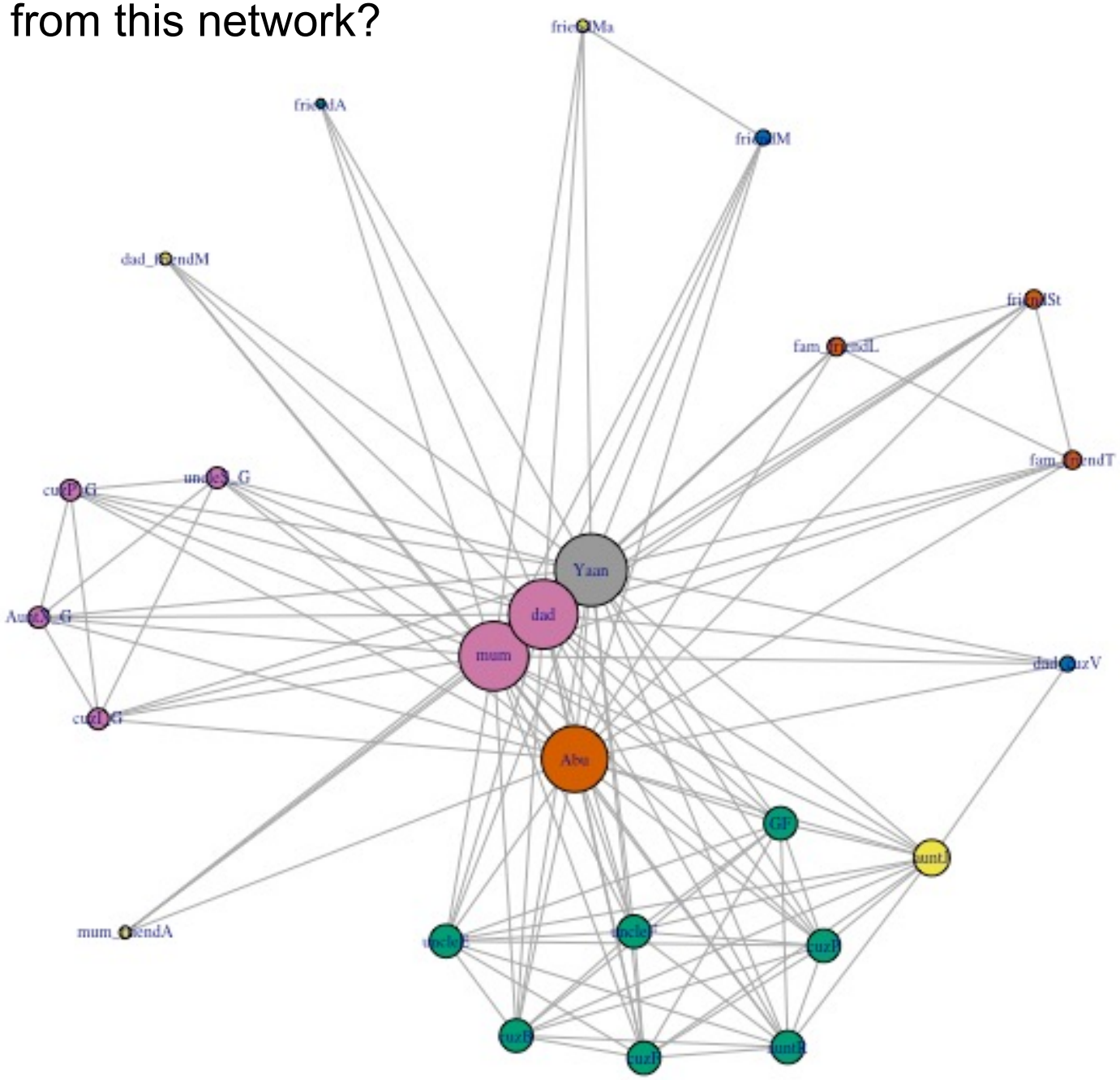
→ Communities can be understood in many different senses:
  → As a connected subgraph
  → As a dense neighbourhood of a network: e.g. random walker might spend a lot of time in a particular set of nodes

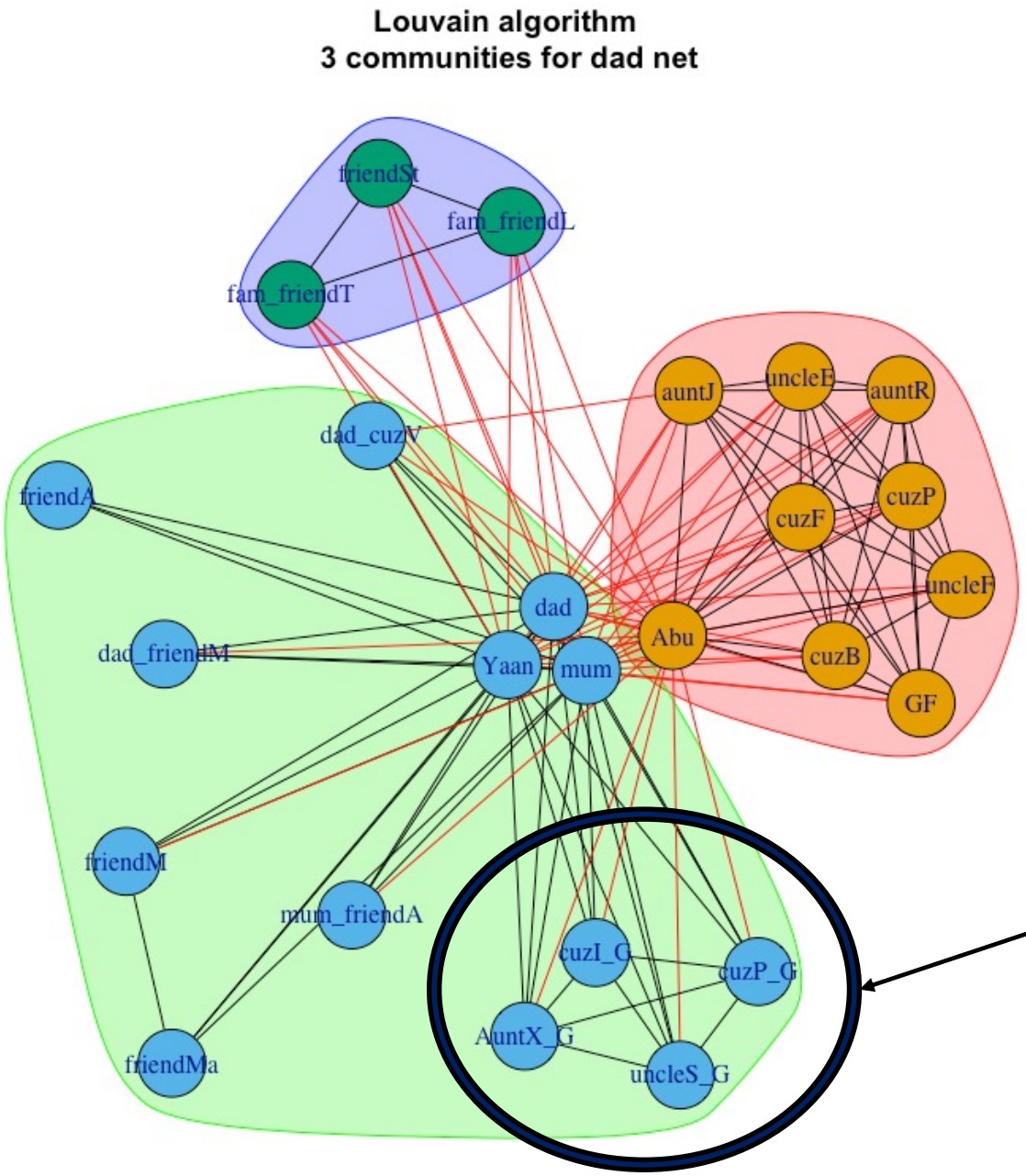
→ No community structure is expected from a random network

→ Many algorithms consider the best partition with no overlapping nodes. This is usually not the case in real life, and there are some algorithms that consider overlap.

Can we identify communities from this network?

Recall: Smallest social network of Yaan's dad, perceived by him





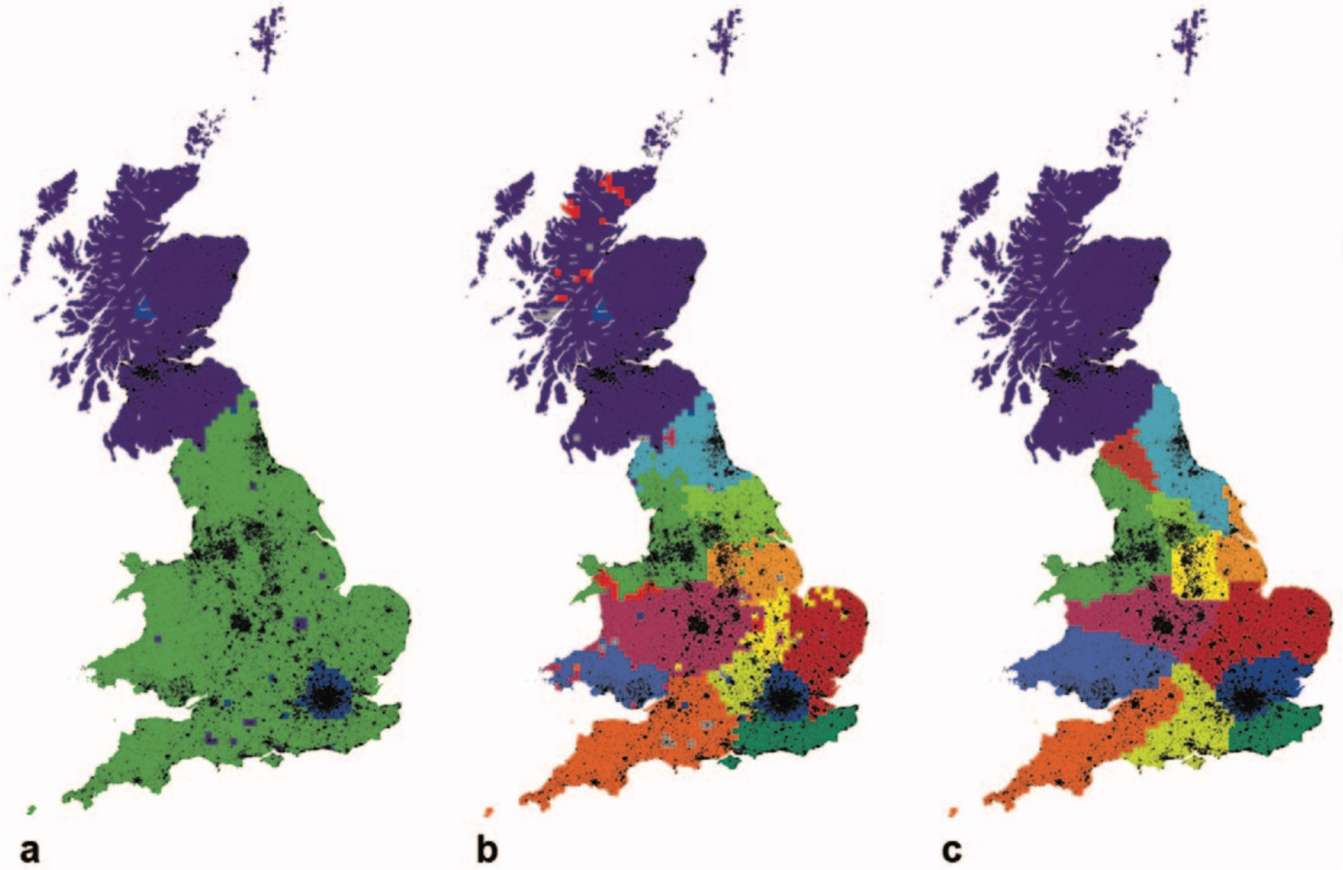
I was expecting to find these nodes as part of a separate community.

What happened?

→ Is it because the algorithm wasn't very good?

→ or because of the bias in the data?

Which is the best partition?



How do we find the best partition?

Figure 2. Defining regions through the spectral modularity optimization of telecommunications networks. a - even with just three regions we obtain a total modularity of 0.31, indicating a fairly good network partitioning. b - the final partitioning of Great Britain yields a modularity of 0.58. c - further fine tuning according to the process suggested by Newman [16] increases the modularity to 0.60.
doi:10.1371/journal.pone.0014248.g002

Ratti et al (Jon Reades among them), PLoS ONE (2010)

Community detection algorithms

→ There are A LOT of community detection algorithms: today only look at "descriptive"
→ Have a look at the attached reviews by Santo Fortunato
→ Main methods to look at today: existing algorithms in libraries in Python and R (NetworkX, cdlib, iGraph)
Girvan-Newman, Random walkers, Louvain method, Infomap, etc.

Not covered today: to test methods need benchmark networks
See A. Lancichinetti and S. Fortunato, Phys. Rev. E 80, 016118, 2009.
→ Traditionally used GN (Girvan-Newman) benchmark:
- all nodes have the same expected degree
- all communities have equal size

→ LFR benchmark (Lancichinetti-Fortunato-Radicchi):
- generalisation GN benchmark
- Tests can be performed on very large systems

→ communities can be overlapping!!! Need different similarity measures and benchmarks
→ Resolution problem: see Barthelemy and Santo Fortunato ECCS12 plenary talk
→ Tiago Peixoto's position: these methods are not good, need inferential methods
https://skewed.de/tiago/blog/descriptive-inferential#fig-generation

Datasets

Can start exploring the algorithms through existing data:

http://www-personal.umich.edu/~mejn/netdata/
http://www.cc.gatech.edu/dimacs10/downloads.shtml
http://snap.stanford.edu/data/index.html#communities

The packages in Python and R (NetworkX, iGraph), have libraries with “famous” networks, as shown in the first practical.

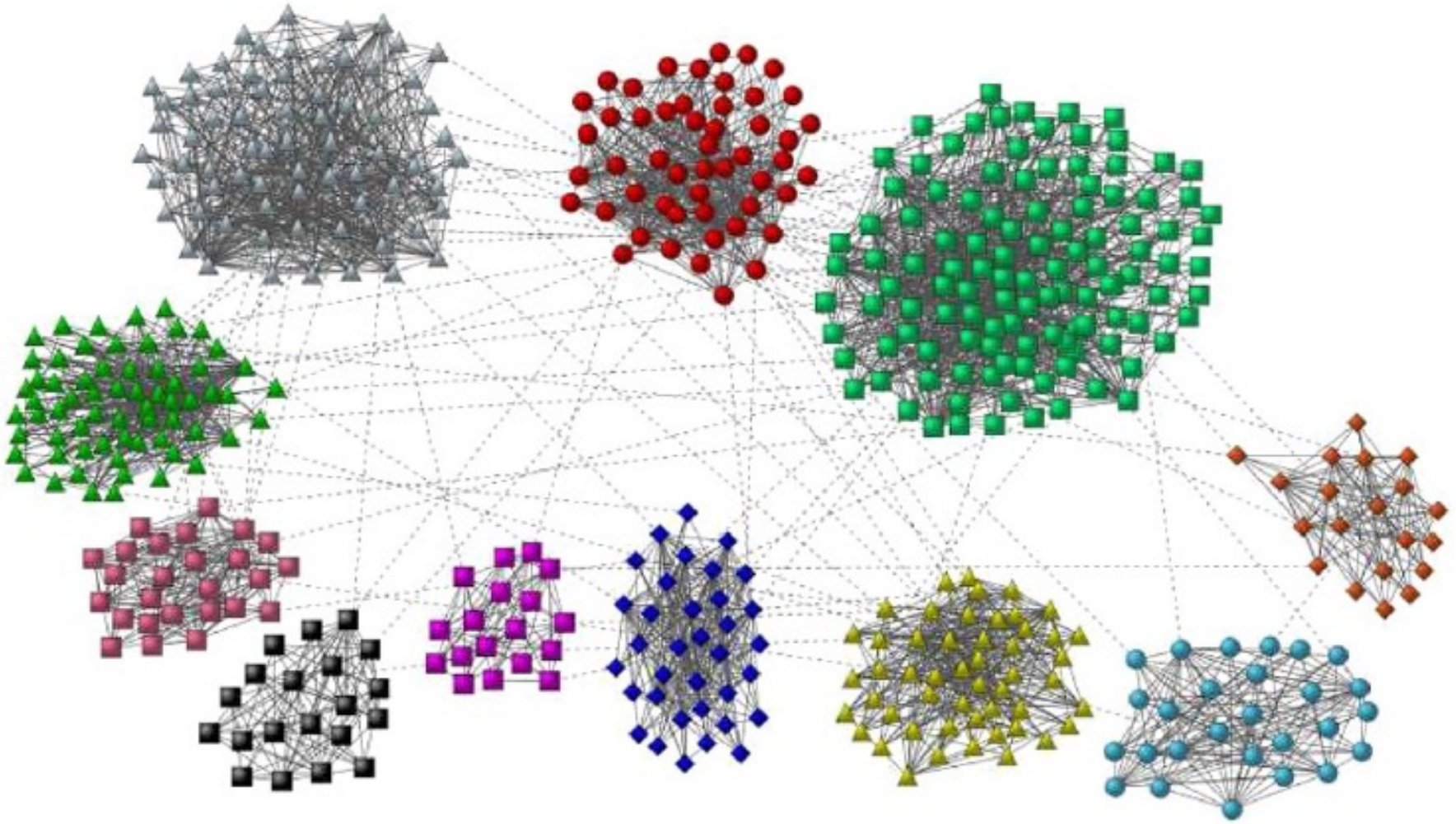


Fig. 31. A realization of the LFR benchmark graphs [326], with 500 vertices. The distributions of the vertex degree and of the community size are both power laws. Such a benchmark is a more faithful approximation of real-world networks with community structure than simpler benchmarks like, e.g., that by Girvan and Newman [12]. Reprinted figure with permission from Ref. [326].

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering

1. Build a similarity matrix for the network

2. *Similarity matrix*: how similar two nodes are to each other → we need to determine from the adjacency matrix

3. Hierarchical clustering iteratively identifies groups of nodes with high similarity, following one of two distinct strategies:

   Agglomerative algorithms merge nodes and communities with high similarity.

   Divisive algorithms split communities by removing links that connect nodes with low similarity.

4. *Hierarchical tree* or *dendrogram*: visualize the history of the merging or splitting process the algorithm follows. Horizontal cuts of this tree offer various community partitions.

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering

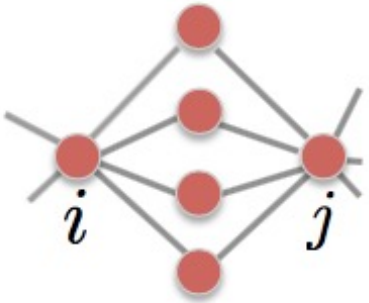
Agglomerative algorithms merge nodes and communities with high similarity.

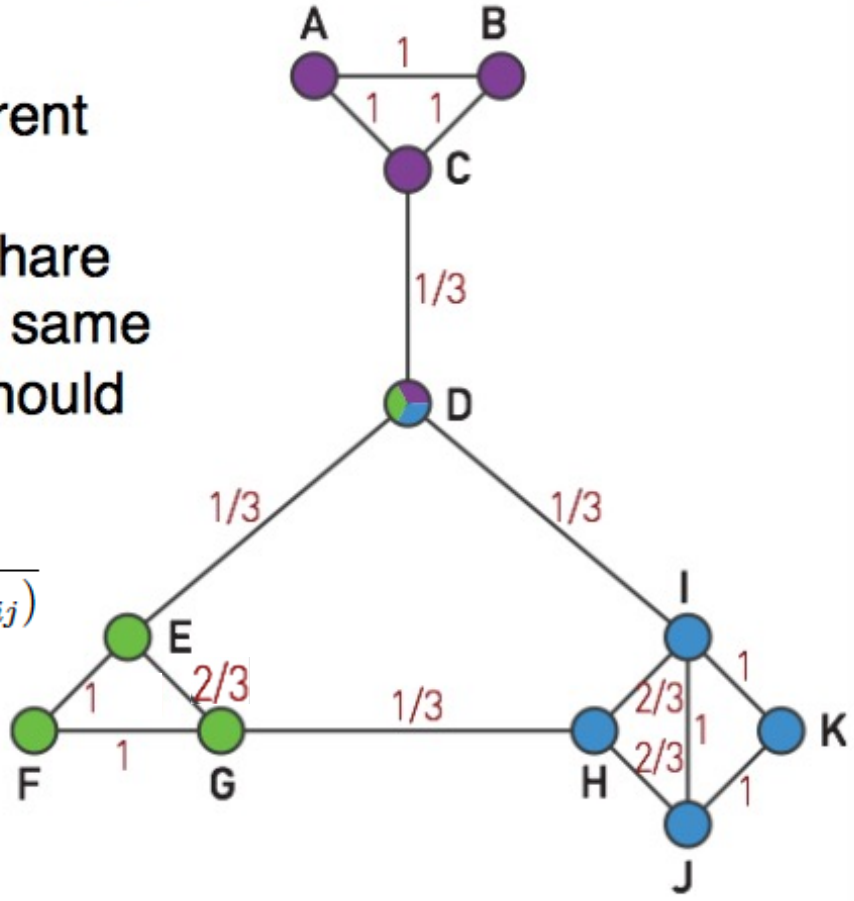
Step 1: Define the Similarity Matrix (Ravasz algorithm)

- High for node pairs that likely belong to the same community, low for those that likely belong to different communities.
- Nodes that connect directly to each other and/or share multiple neighbors are more likely to belong to the same dense local neighborhood, hence their similarity should be large.

Topological overlap matrix: $x_{ij}^0 = \dfrac{J(i,j)}{\min(k_i, k_j) + 1 - \Theta(A_{ij})}$

$J_N(i,j)$: number of common neighbors of node i and j; (+1) if there is a direct link between i and j;



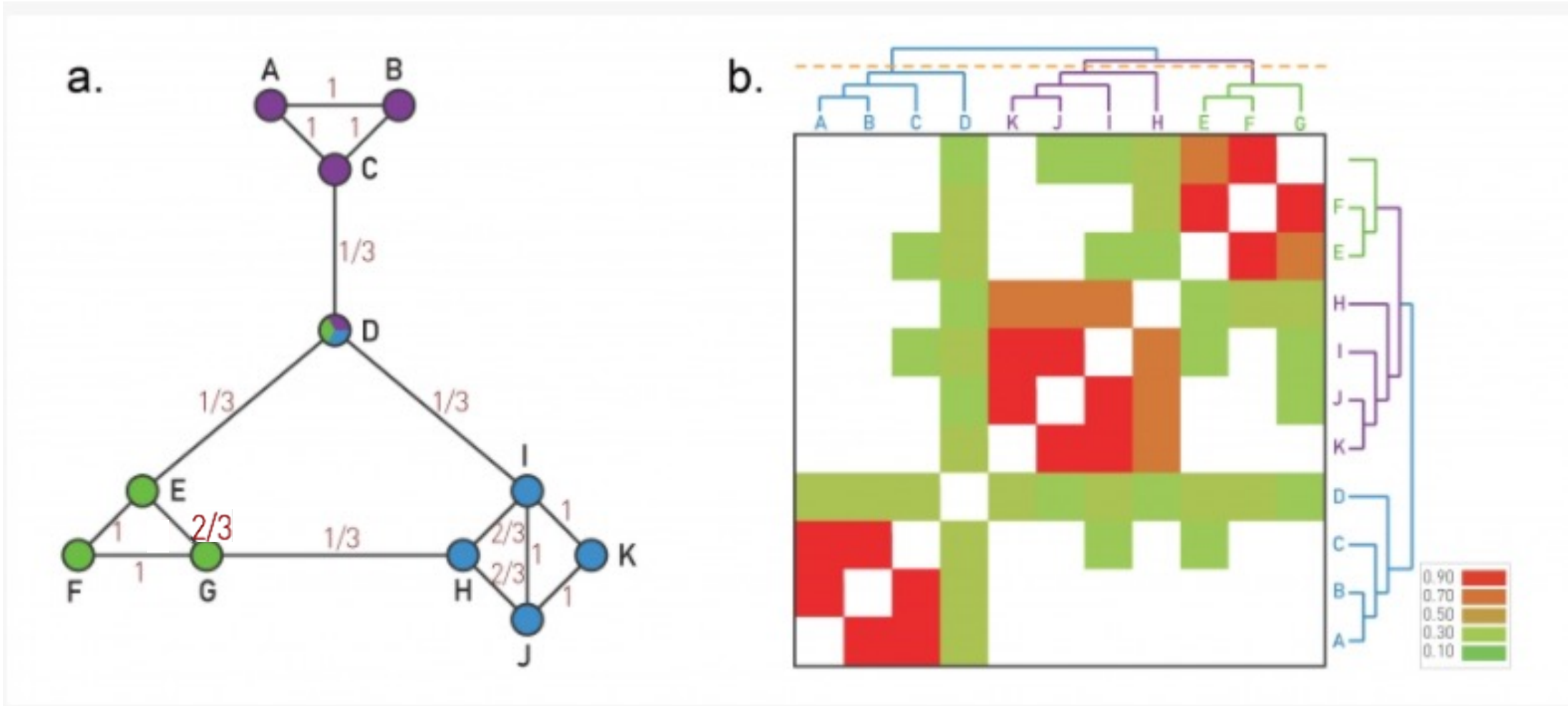


E. Ravasz et al., *Science* 297 (2002).

A.-L. Barabási, *Network Science: Communities*

$\Theta(A_{ij})$ is the Heaviside function, it is 1 if $A_{ij} > 0$ and 0 otherwise.

Slide from Barábasi's book
http://networksciencebook.com/chapter/9#hierarchical



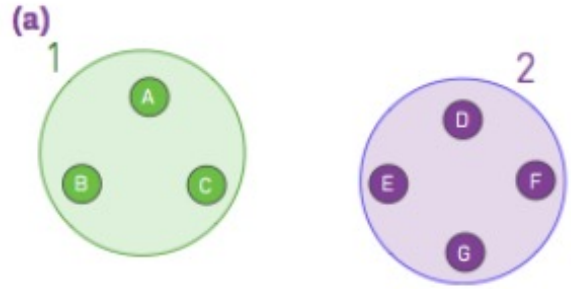
Visualisation of the topological overlap matrix.
→ Cutting the dendogram in the orange line gives rise to three communities.

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering

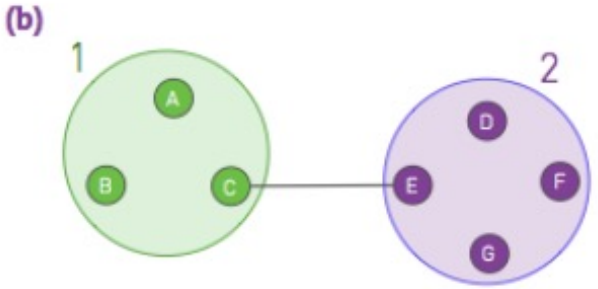
Step 2: Decide Group Similarity

- Groups are merged based on their mutual similarity through *single*, *complete* or *average cluster linkage*

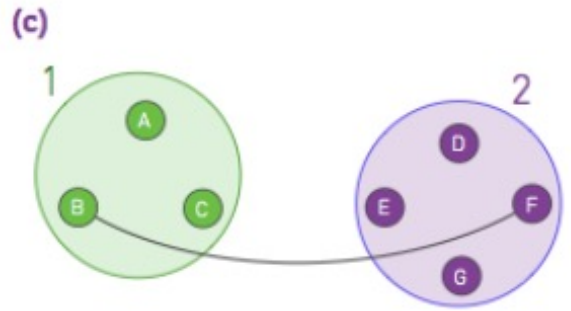


$$\frac{1}{x_{ij}} = r_{ij} =$$

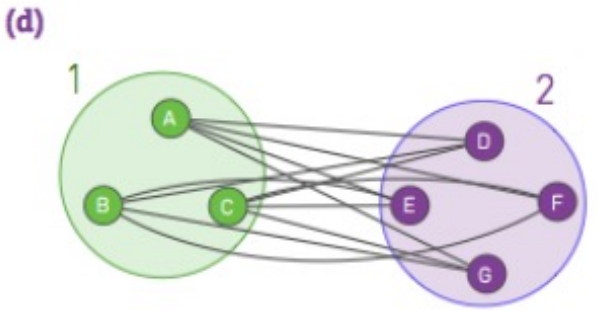
	D	E	F	G
A	2.75	2.22	3.46	3.08
B	3.38	2.68	3.97	3.40
C	2.31	1.59	2.88	2.34



Single Linkage: $r_{12} = 1.59$ → Similarity between groups given by most similar pair



Similarity between groups given by least similar pair ← Complete Linkage: $r_{12} = 3.97$



Average Linkage: $r_{12} = 2.84$ → Similarity between groups given by mean similarity of all pairs

E. Ravasz et al., *Science* 297 (2002).
A.-L. Barabási, *Network Science: Communities.*

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering

Step 3: Apply Hierarchical Clustering

- Assign each node to a community of its own and evaluate the similarity for all node pairs. The initial similarities between these "communities" are simply the node similarities.
- Find the community pair with the highest similarity and merge them to form a single community.
- Calculate the similarity between the new community and all other communities.
- Repeat from Step 2 until all nodes are merged into a single community.

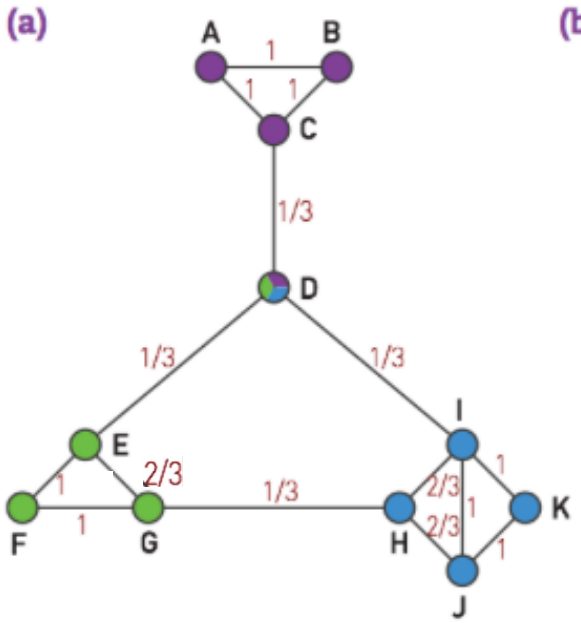
Step 4: Build Dendrogram

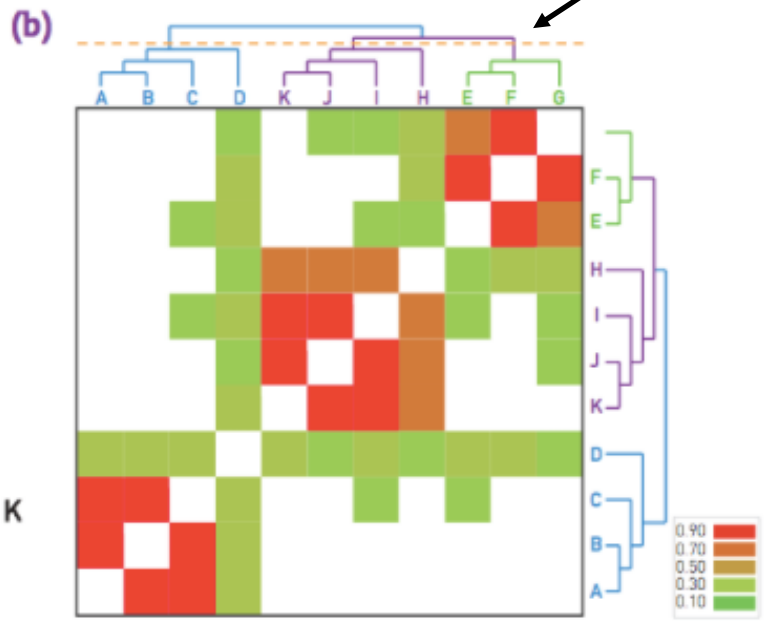
- Describes the precise order in which the nodes are assigned to communities.

E. Ravasz et al., *Science* 297 (2002).

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering





Note that we still haven't seen how to find the cut, other than by visual inspection.

Computational complexity:

- Step 1 (calculation similarity matrix): $O\left(N^2\right)$
- Step 2-3 (group similarity): $O\left(N^3\right)$
- Step 4 (dendrogram): $O\left(N \log N\right)$

$\Rightarrow O\left(N^3\right)$

E. Ravasz et al., *Science* 297 (2002).
A.-L. Barabási, *Network Science: Communities.*

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering

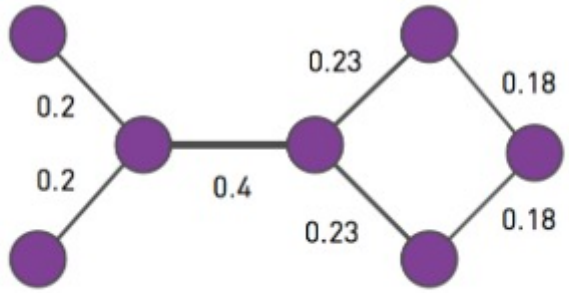
Divisive algorithms split communities by removing links that connect nodes with low similarity.

Step 1: Define a Centrality Measure (Girvan-Newman algorithm)

- *Link betweenness* is the number of shortest paths between all node pairs that run along a link.
- *Random-walk betweenness*. A pair of nodes m and n are chosen at random. A walker starts at m, following each adjacent link with equal probability until it reaches n. Random walk betweenness x_{ij} is the probability that the link $i \rightarrow j$ was crossed by the walker after averaging over all possible choices for the starting nodes m and n

(a)

(b)



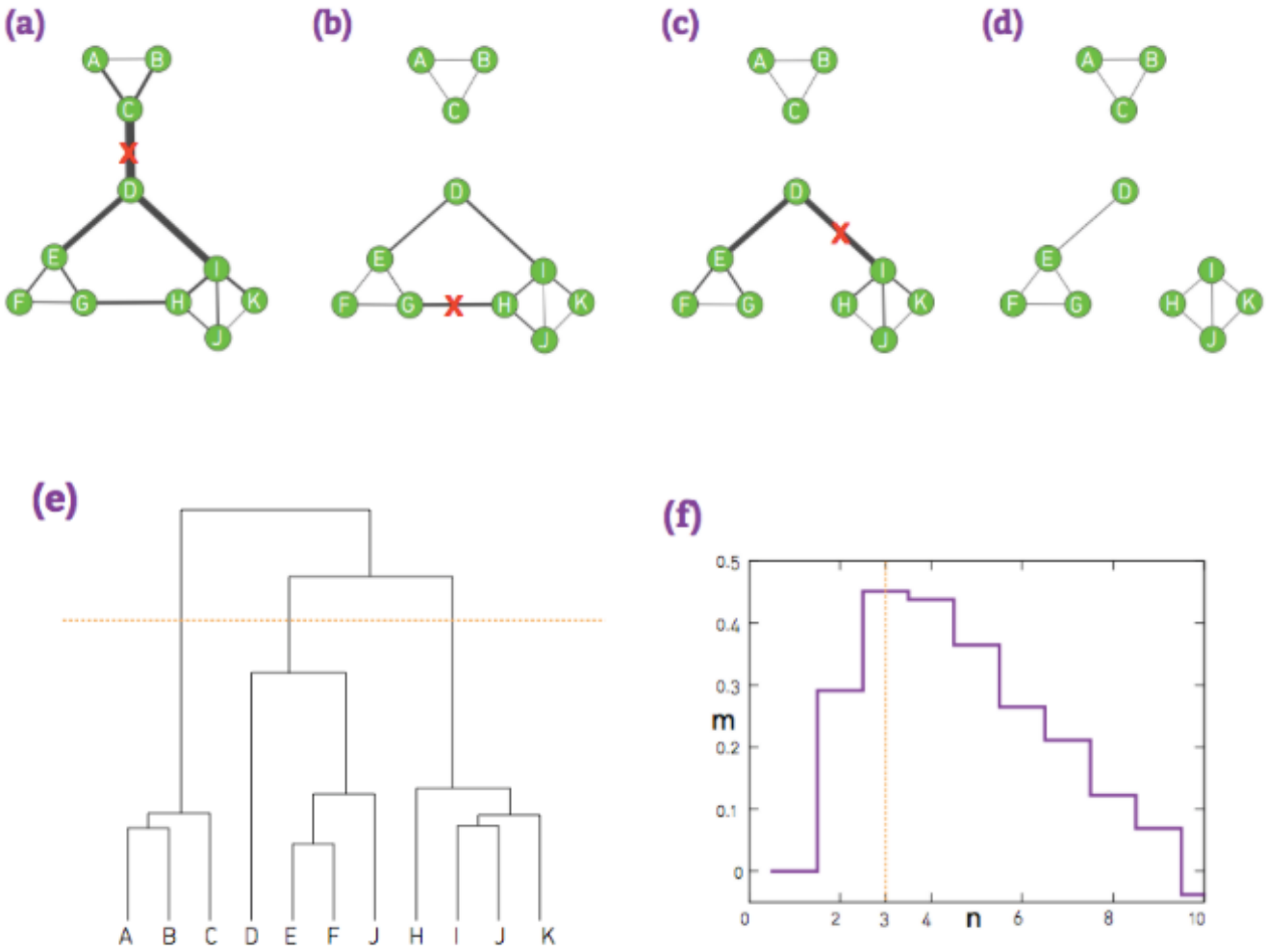
M. Girvan & M.E.J. Newman, *PNAS* 99 (2002).
A.-L. Barabási, *Network Science: Communities.*

Slide from Barábasi's lectures
https://www.barabasilab.com/course

Principles of hierarchical clustering

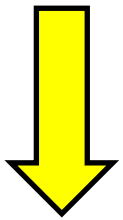
Step 2: Hierarchical Clustering

a) Compute of the centrality of each link.
b) Remove the link with the largest centrality; in case of a tie, choose one randomly.
c) Recalculate the centrality of each link for the altered network.
d) Repeat until all links are removed (yields a dendrogram).



M. Girvan & M.E.J. Newman, *PNAS* 99 (2002).
A.-L. Barabási, *Network Science: Communities.*

The role of modularity



Find the cut in the dendogram
→ Find best partition

Recall: modularity

Modularity: discrete case

Prob. of links in network

Expected prob of links
If network random

- Modularity matrix

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

- Modularity

$$Q = \frac{1}{2m} \sum_{ij} B_{ij} \delta(c_i, c_j)$$

If Q>0 assortative mixing
If Q<0 disassortative mixing

- Max modularity

$$Q_{max} = \frac{1}{2m} \left(2m - \sum_{ij} \frac{k_i k_j}{2m} \delta(c_i, c_j) \right)$$

Recall:

$$m = \frac{1}{2} \sum_{i=1}^{n} k_i = \frac{1}{2} \sum_{ij} A_{ij}$$

- Normalised modularity

or

assortativity coefficient

$$M = \frac{Q}{Q_{max}}$$

M.E.J. Newman *PNAS* 103 (2006)

Example of different community detection algorithms applied to the Karate club network

An Information Flow Model for Conflict and Fission in Small Groups

Wayne W. Zachary

Journal of Anthropological Research, Vol. 33, No. 4 (Winter, 1977), pp. 452-473 (22 pages)

https://www.jstor.org/stable/3629752

→ The following provides the functions for the algorithms in iGraph for R and in NetworkX or cdlib for Python. Note that for Python, we need to call cdlib instead of NetworkX, since some of the algorithms are not developed for NetworkX (see most updated version in practical).

Girvan Newman method

M. Girvan and M. E. Newman, Proc. Natl. Acad. Sci. U.S.A., 99, 782 (2002)
M. E. Newman and M. Girvan, Physical Review E 69, 026113 (2004)

First community detection algorithm

iGraph: edge.betweenness.community
NetworkX: girvan_newman

How does it work?

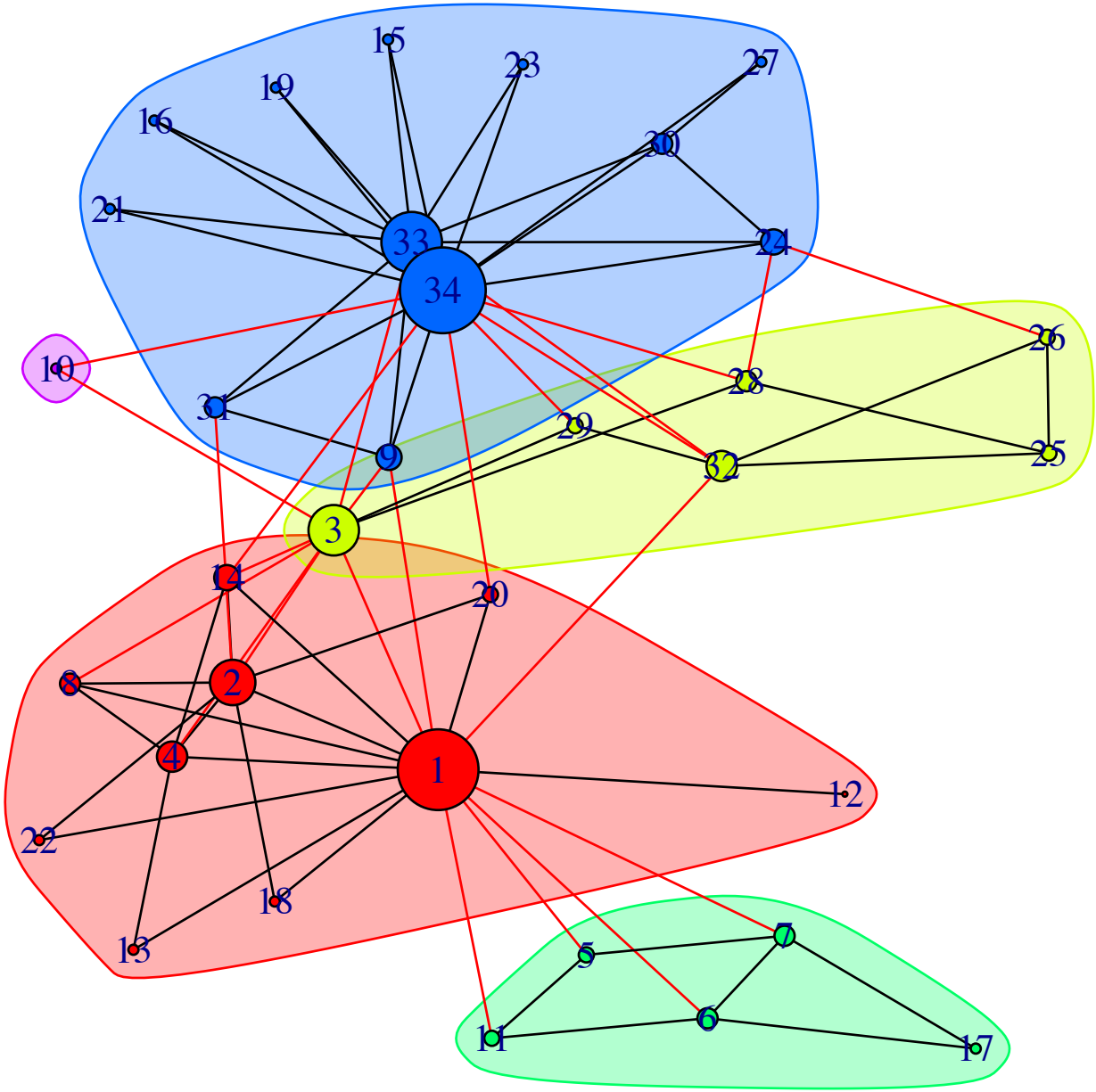
→ Hierarchical divisive algorithm: remove edges (links) according to their edge betweenness values (in decreasing order).
→ Removal stops when modularity of the partition is maximised.
➢ Edges with high betweenness will be most likely the ones connecting and hence belonging to different groups.

Performance

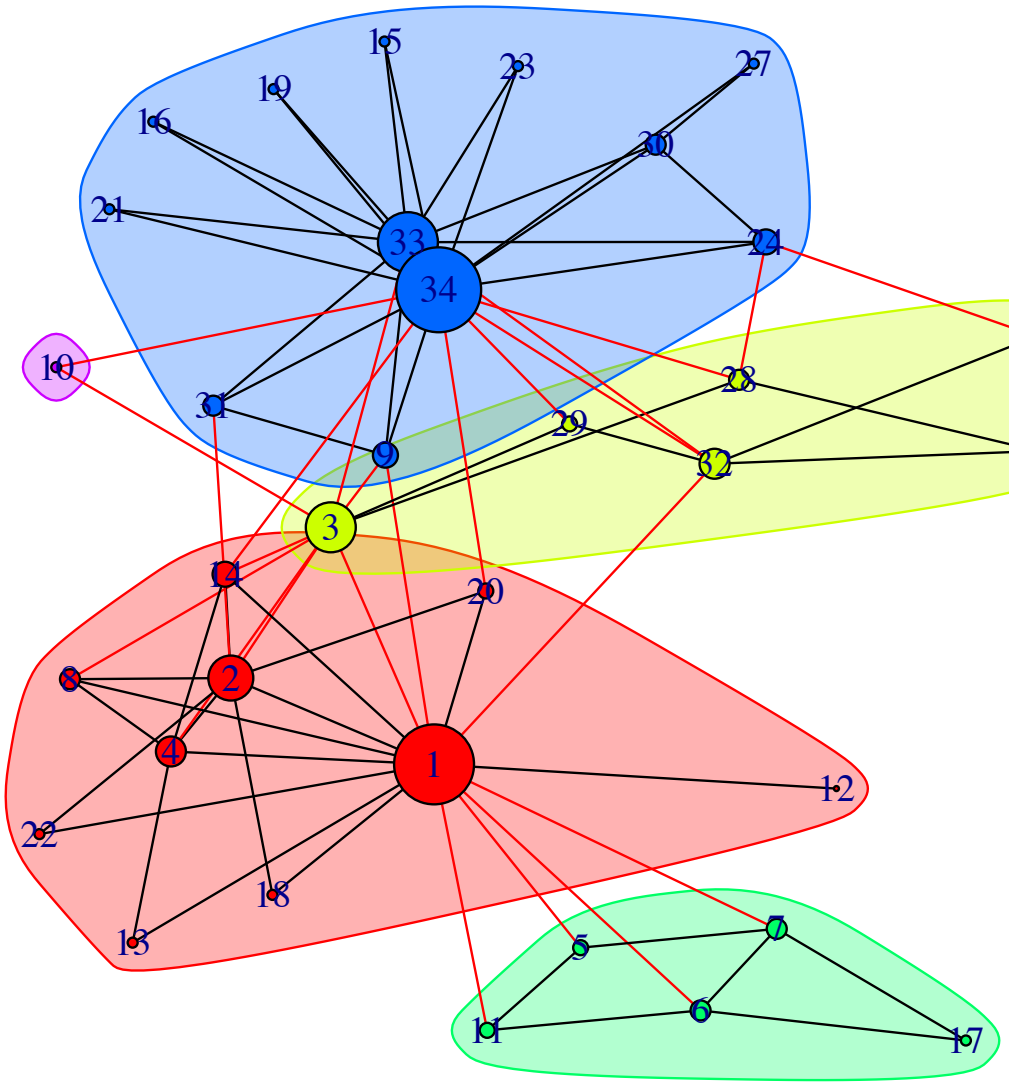
→ Good
→ slow: $O(N^3)$: betweenness needs to be re-calculated each time that an edge is removed

Can be used for weighted and directed graphs only in iGraph

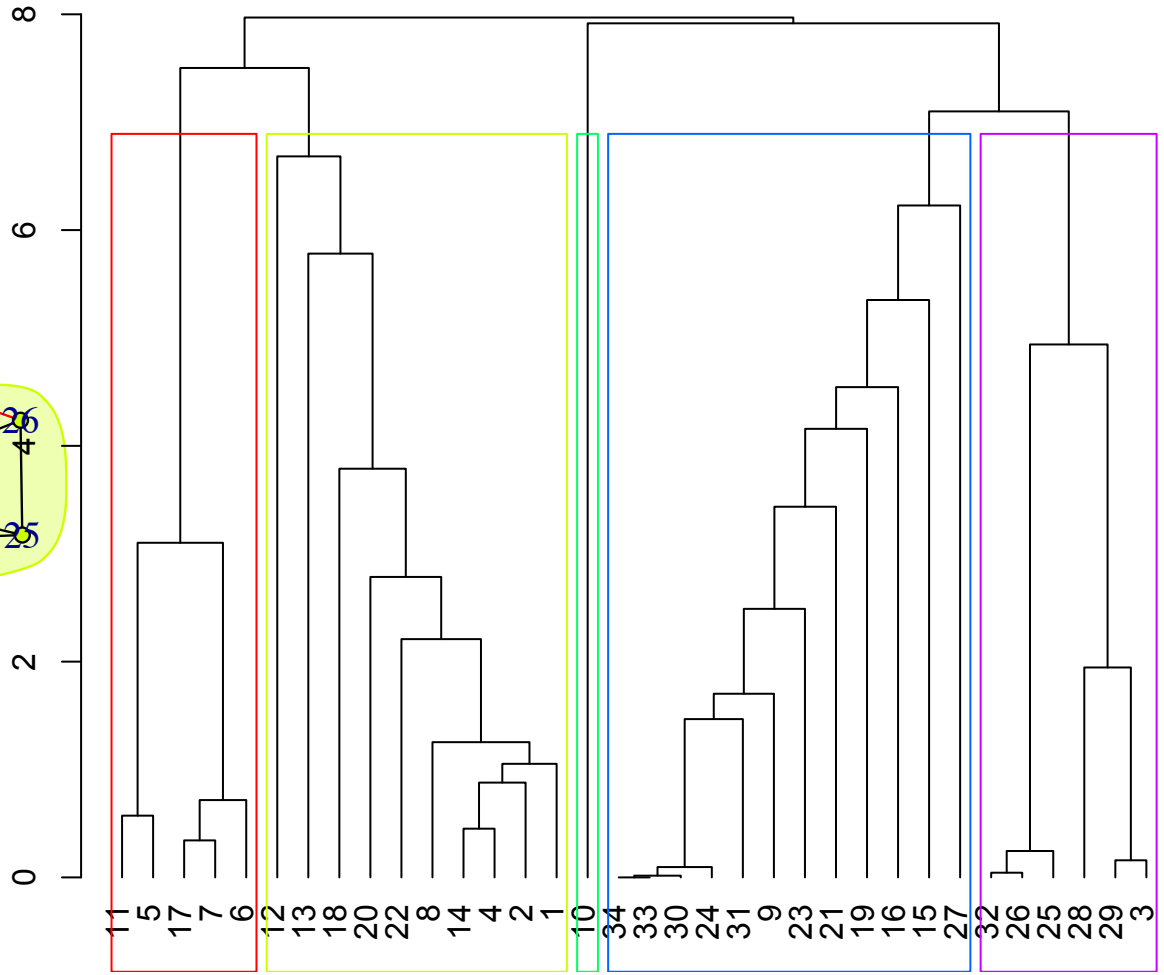
Girvan-Newman algorithm
5 communities for the karate club



Girvan-Newman algorithm
5 communities for the karate club



Community structure dendrogram for Girvan-Newman method
5 communities for the karate club



Fast Greedy modularity optimization: Clauset, Newman and Moore

A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E 70, 066111 (2004)
http://www.arxiv.org/abs/cond-mat/0408187

iGraph: fastgreedy.community
NetworkX: greedy_modularity_communities

How does it work?

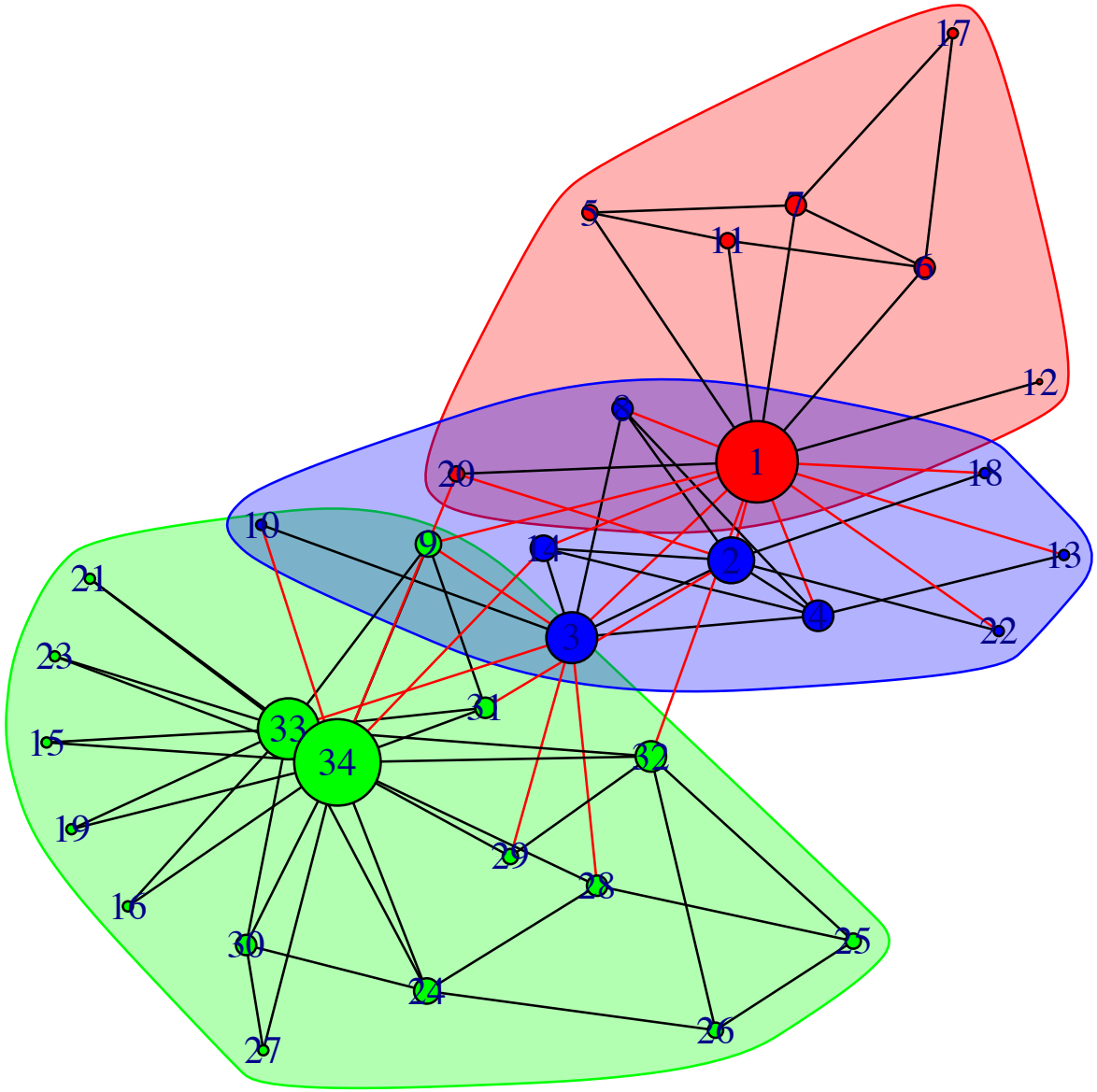
→ Assign each node to its own community, no links between communities
→ Inspect each community pair connected by a link, and assign them to the same community if the link increases maximally the Girvan-Newman modularity.
→ Bottom-up instead of top-down: communities are merged iteratively such that each merge is locally optimal

Performance

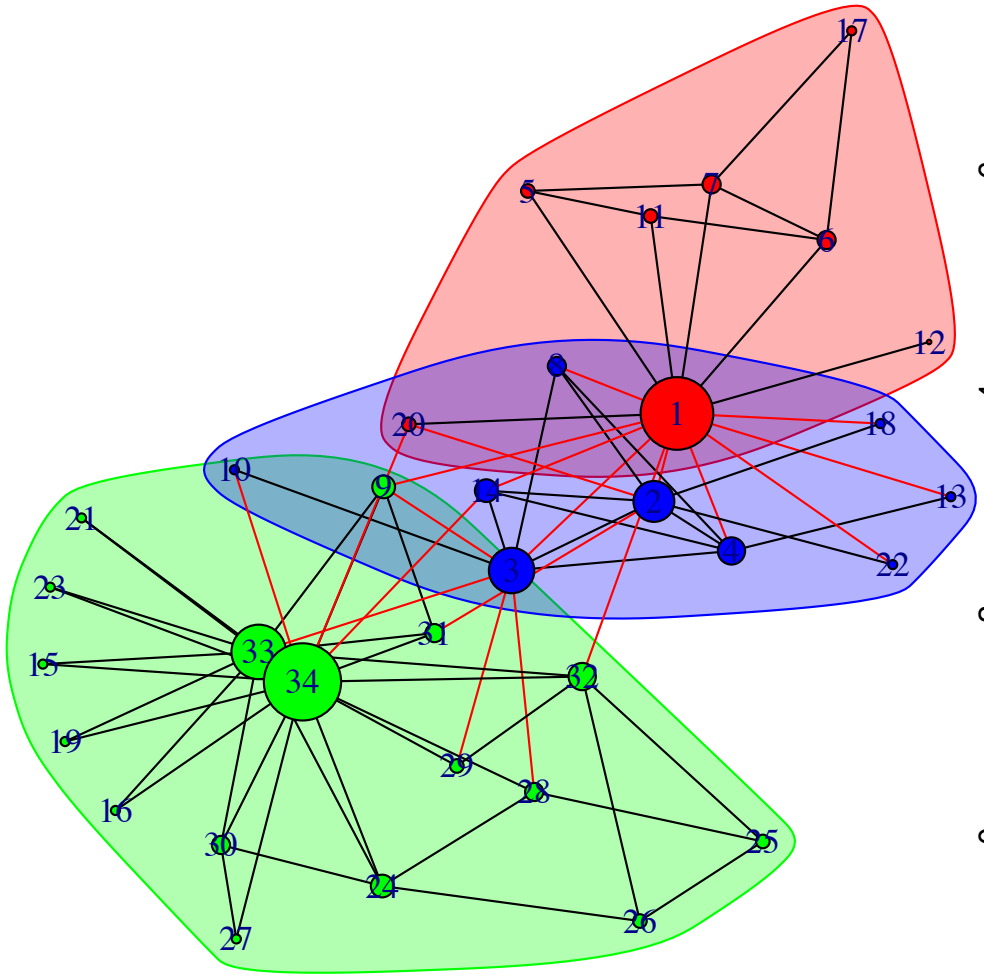
→ Not as good as Girvan-Newman
→ fast: $O(NLog^2N)$
→ Resolution limit

Can be used for weighted but UNDIRECTED graphs only

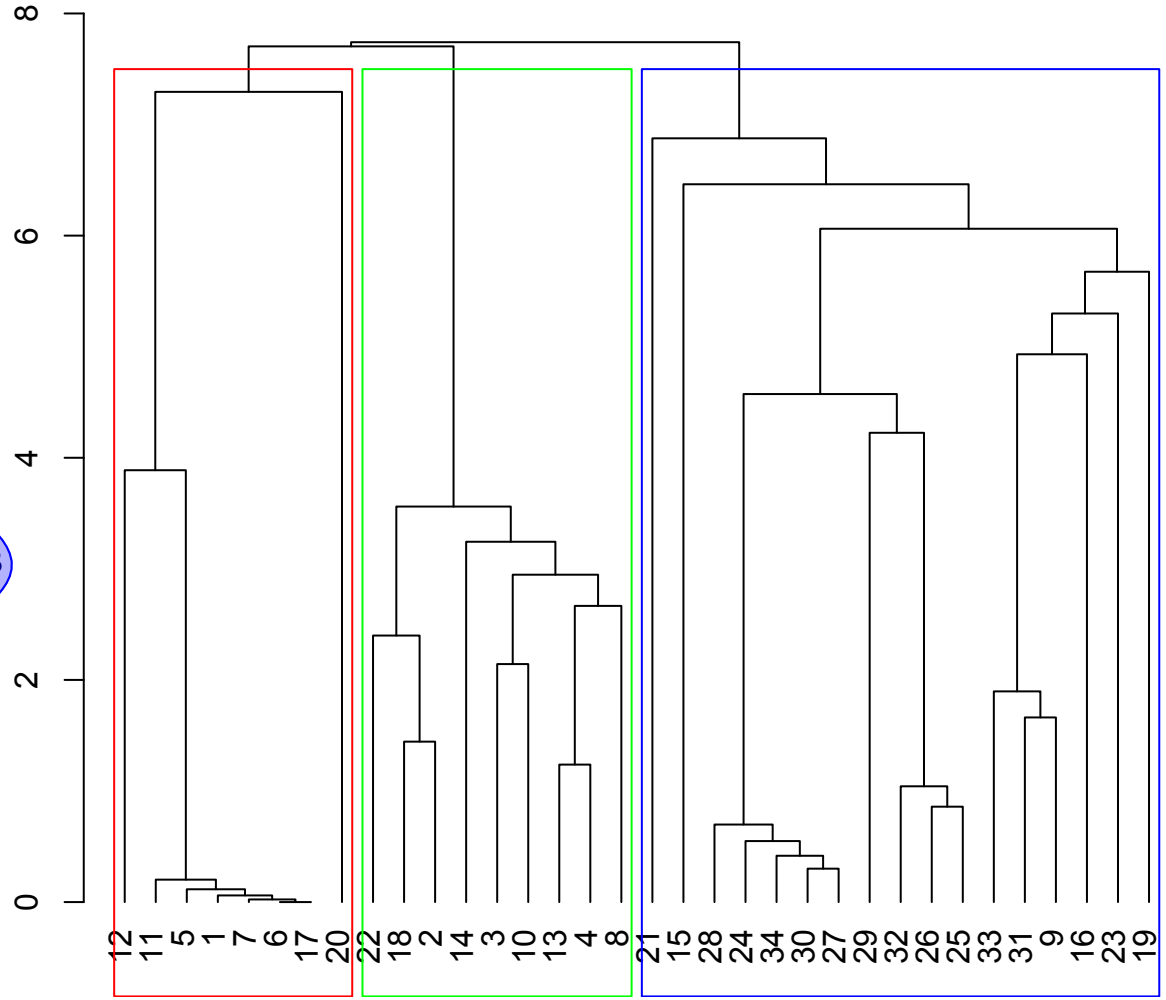
Clauset-Newman-Moore algorithm
3 communities for the karate club



Clauset-Newman-Moore algorithm
3 communities for the karate club



Community structure dendrogram for Clauset-Newman-Moore method
3 communities for the karate club



Random walk: Pons & Latapy

Pons P., Latapy M. (2005) Computing Communities in Large Networks Using Random Walks. In: Yolum ., Güngör T., Gürgen F., Özturan C. (eds) Computer and Information Sciences - ISCIS 2005. ISCIS 2005. Lecture Notes in Computer Science, vol 3733. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11569596_31
http://arxiv.org/abs/physics/0512106

iGraph: walktrap.community
cdlib: algorithms.walktrap

How does it work?

→ This function tries to find densely connected subgraphs
→ short random walks tend to stay in the same community
→ merge separate communities in a bottom-up manner
→ can use the modularity score to select where to cut the dendrogram

Performance

→ Slower than greedy
→ More accurate than greedy

Can be used for weighted networks in iGraph BUT directed graphs are considered undirected

Random walk: Pons & Latapy

Pons P., Latapy M. (2005) Computing Communities in Large Networks Using Random Walks. In: Yolum ., Güngör T., Gürgen F., Özturan C. (eds) Computer and Information Sciences - ISCIS 2005. ISCIS 2005. Lecture Notes in Computer Science, vol 3733. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11569596_31

We start from a partition $\mathcal{P}_1 = \{\{v\}, v \in V\}$ of the graph into n communities reduced to a single vertex. We first compute the distances between all adjacent vertices. Then this partition evolves by repeating the following operations. At each step k:

- Choose two communities C_1 and C_2 in $\mathcal{P}_k$ on a criterion based on the distance between the communities that we detail later.
- Merge these two communities into a new community $C_3 = C_1 \cup C_2$ and create the new partition: $\mathcal{P}_{k+1} = (\mathcal{P}_k \setminus \{C_1, C_2\}) \cup \{C_3\}$.
- Update the distances between communities (we will see later that we actually only do this for *adjacent* communities).

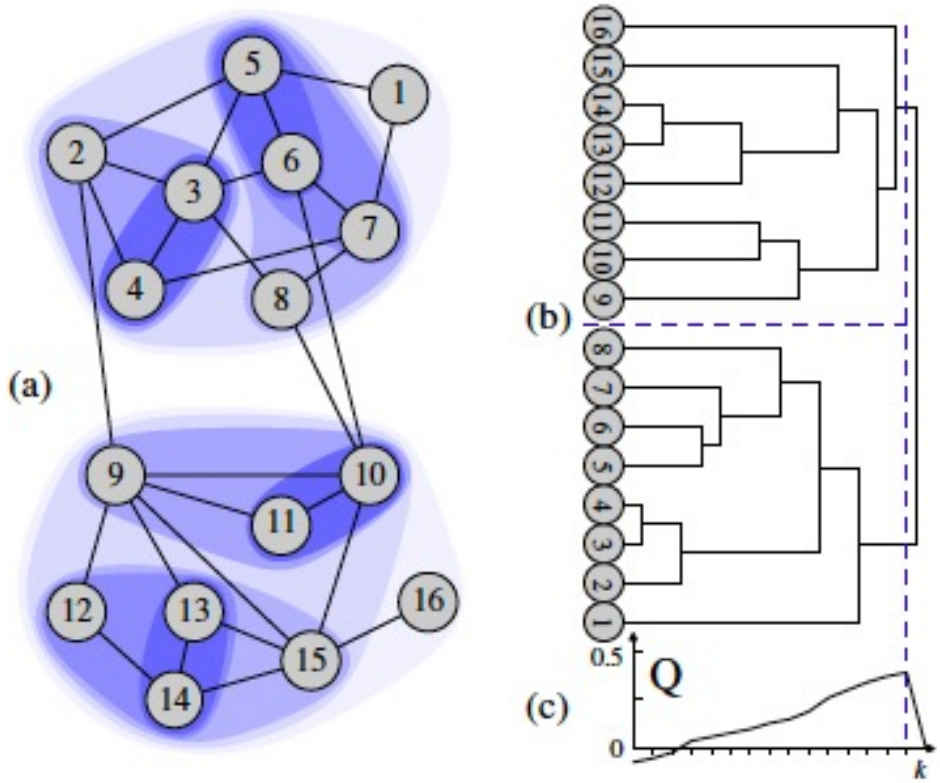
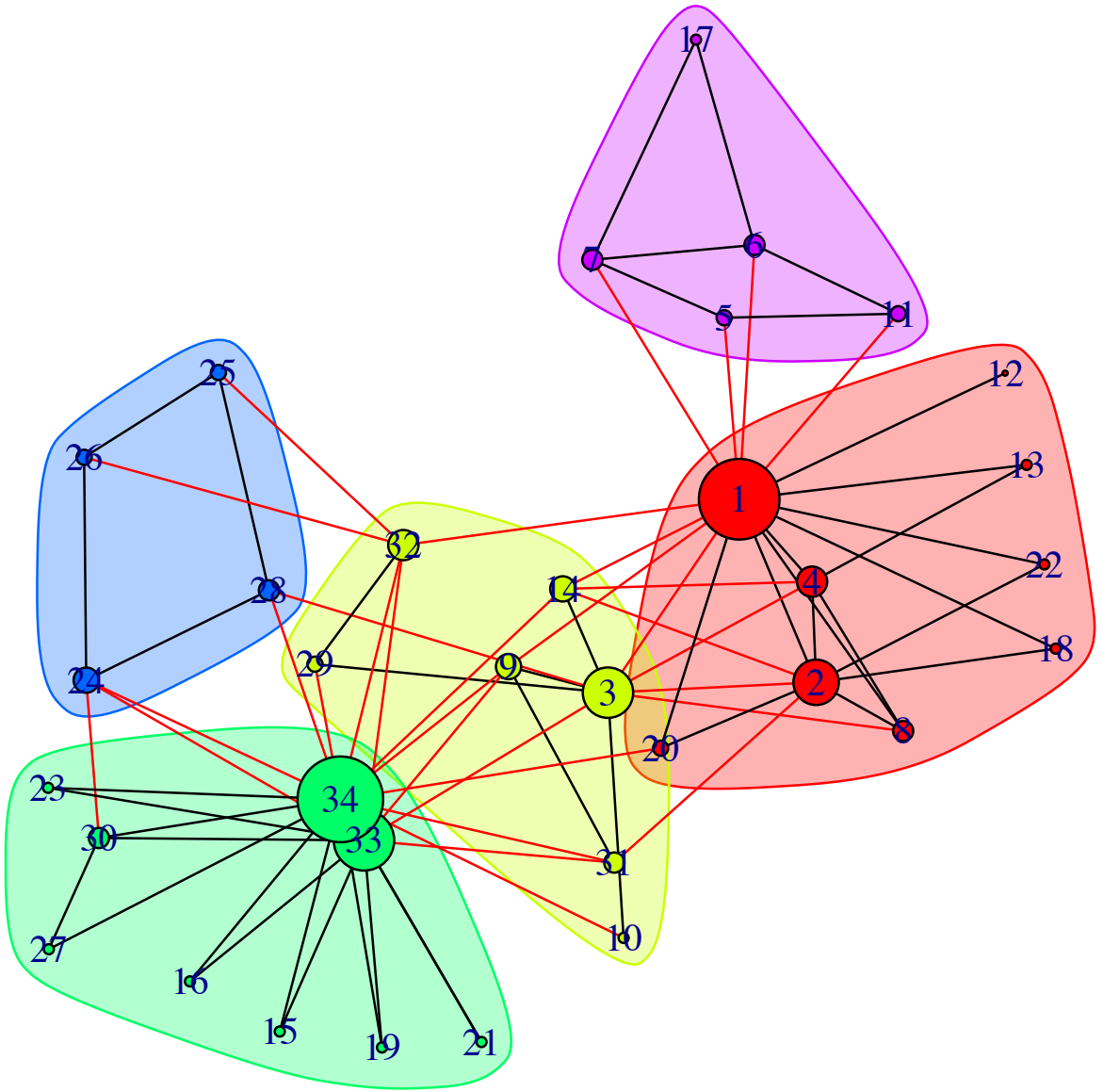
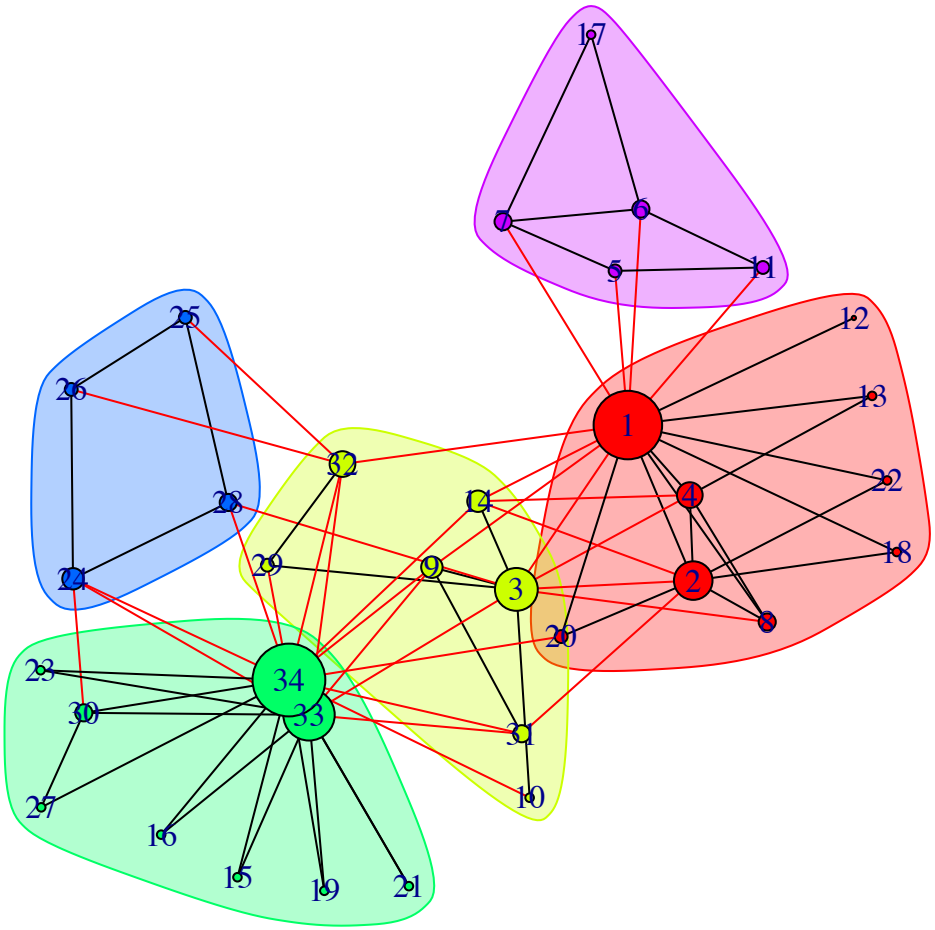


Fig. 1. (a) An example of community structure found by our algorithm using random walks of length $t = 3$. (b) The stages of the algorithm encoded as a tree (*dendrogram*). The maximum of Q, plotted in (c), shows that the best partition consists in two communities.

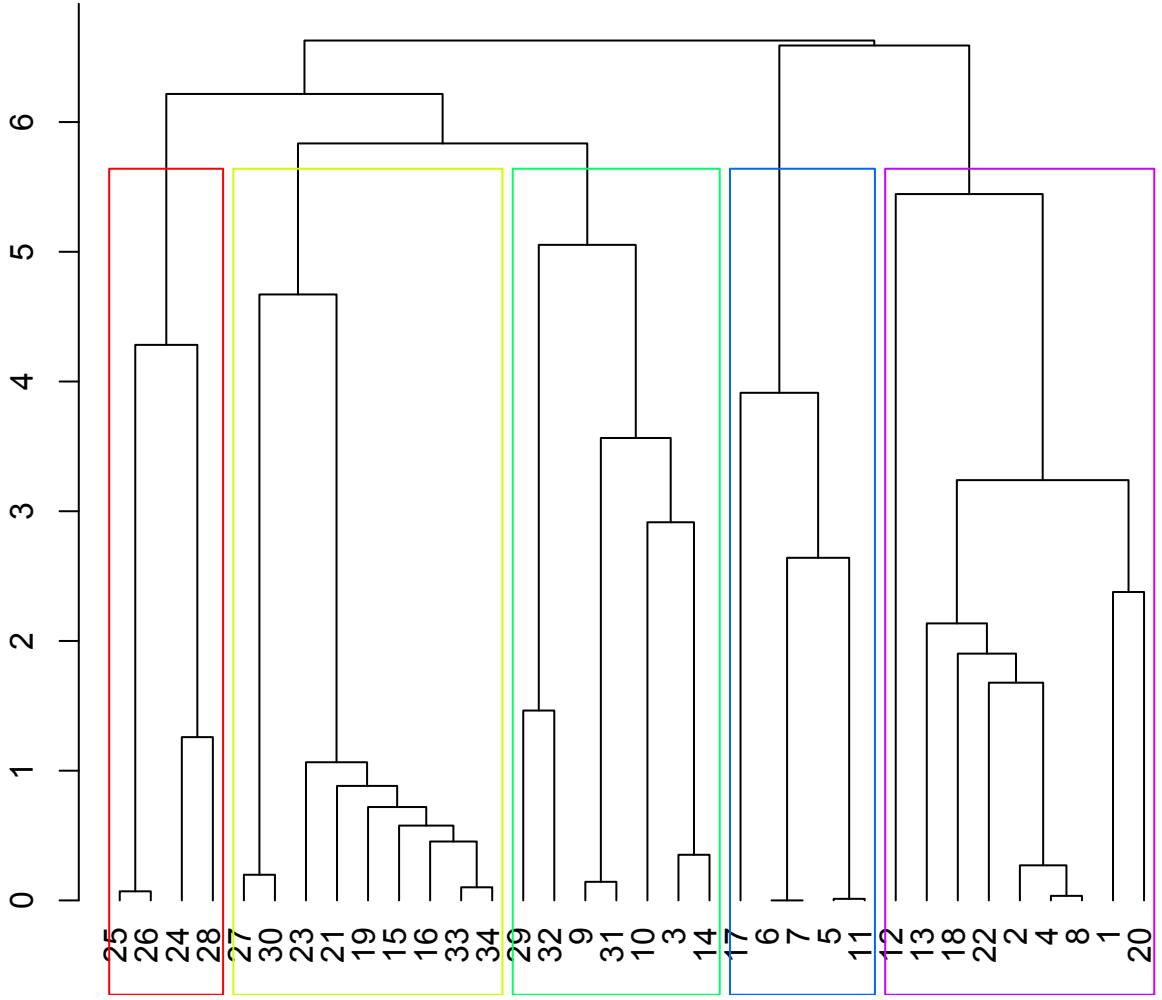
Random walks algorithm
5 communities for the karate club



Random walks algorithm
5 communities for the karate club



Community structure dendrogram for Random walks method
5 communities for the karate club



Leading eigenvector: Newman spectral approach

MEJ Newman: Finding community structure using the eigenvectors of matrices, Physical Review E 74 036104, (2006)

iGraph: leading.eigenvector.community
cdlib: algorithms.eigenvector

How does it work?

→ Method based on the spectral properties of graphs: IF communities well identified, e-vector components for nodes in same communities will have similar values
→ compute **eigenvector of modularity matrix** for the largest positive eigenvalue.
→ separate vertices into two community based on the sign of the corresponding element in the eigenvector. If all elements in the eigenvector are of the same sign that means that the network has no underlying community structure.
→ Choose partition that maximises the G-N modularity

Performance

→ Fast if select only a few e-vectors to compute, although slower than greedy
→ Good results

ONLY for undirected and un-weighted graphs!!!

Leading eigenvector: Newman spectral approach

MEJ Newman, PRE **74** 036104, (2006)

PHYSICAL REVIEW E **74**, 036104 (2006)

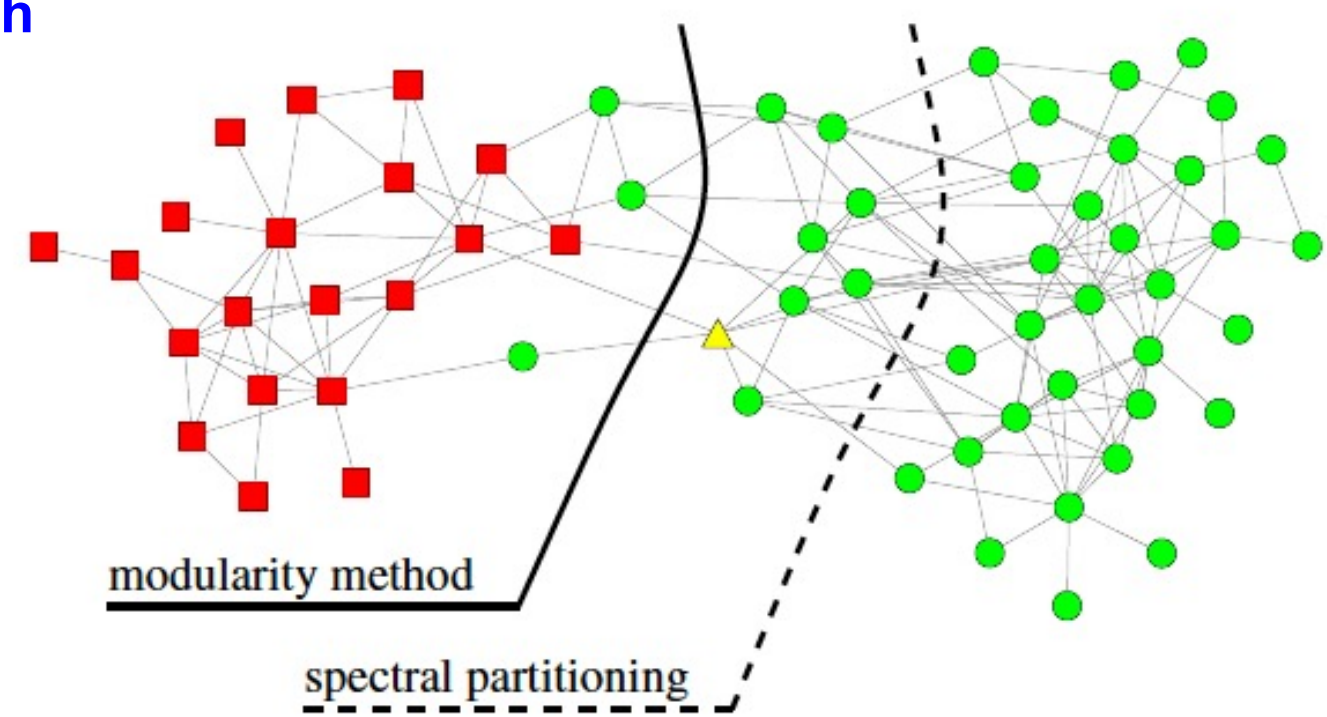
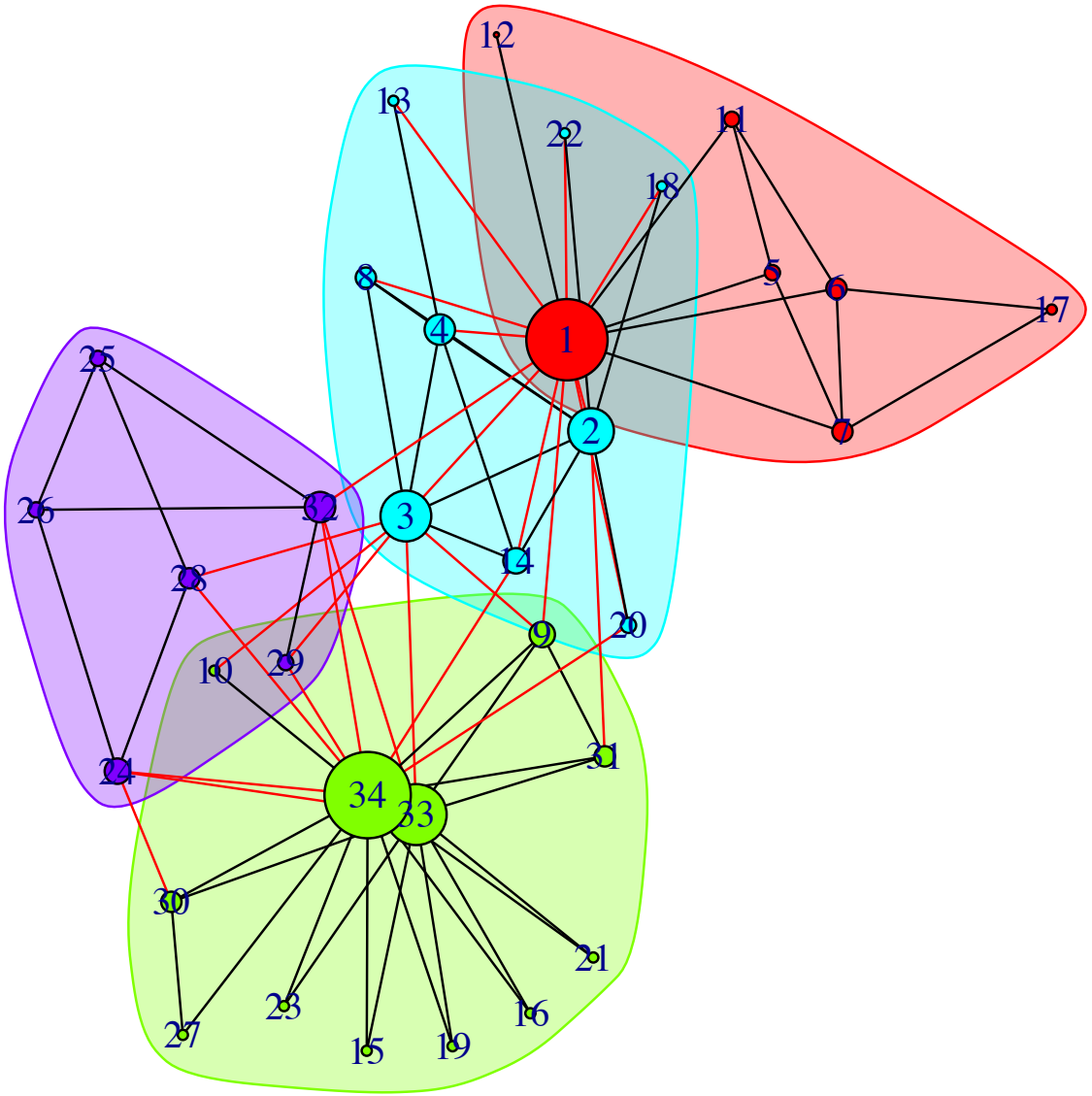


FIG. 2. (Color online) The dolphin social network of Lusseau *et al.* [70]. The dashed curve represents the division into two equally sized parts found by a standard spectral partitioning calculation (Sec. II). The solid curve represents the division found by the modularity-based method of this section. And the squares and circles represent the actual division of the network observed when the dolphin community split into two as a result of the departure of a keystone individual. (The individual who departed is represented by the triangle.)

Spectral algorithm
4 communities for the karate club



Louvain: Blondel et al, modularity optimization

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. 2008, P10008.
http://arxiv.org/abs/arXiv:0803.0476

iGraph: multilevel.community
cdlib: algorithms.Louvain

How does it work?

→ Local optimization of the G-N modularity in the neighbourhood of each node
→ Start from the whole network, identify a community, replace the community by a super-node → get smaller weighted and repeat
→ Iterate until modularity stable

Performance

→ Excellent
→ Computational linear complexity w.r.t. n. nodes

Can be used for weighted but UNDIRECTED graphs only

Fast unfolding of communities in large networks

Louvain: Blondel et al, modularity optimization

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. 2008, P10008.

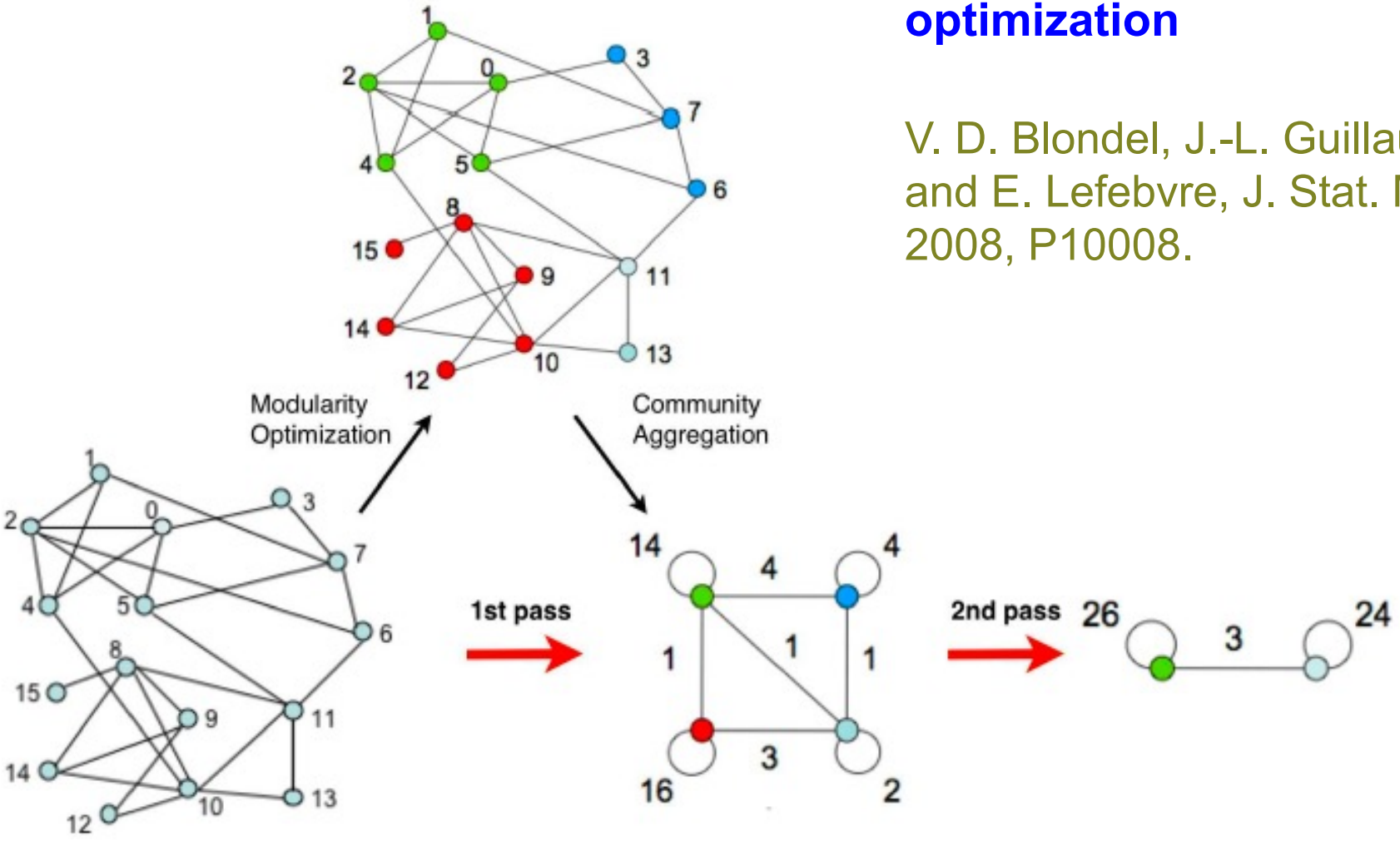
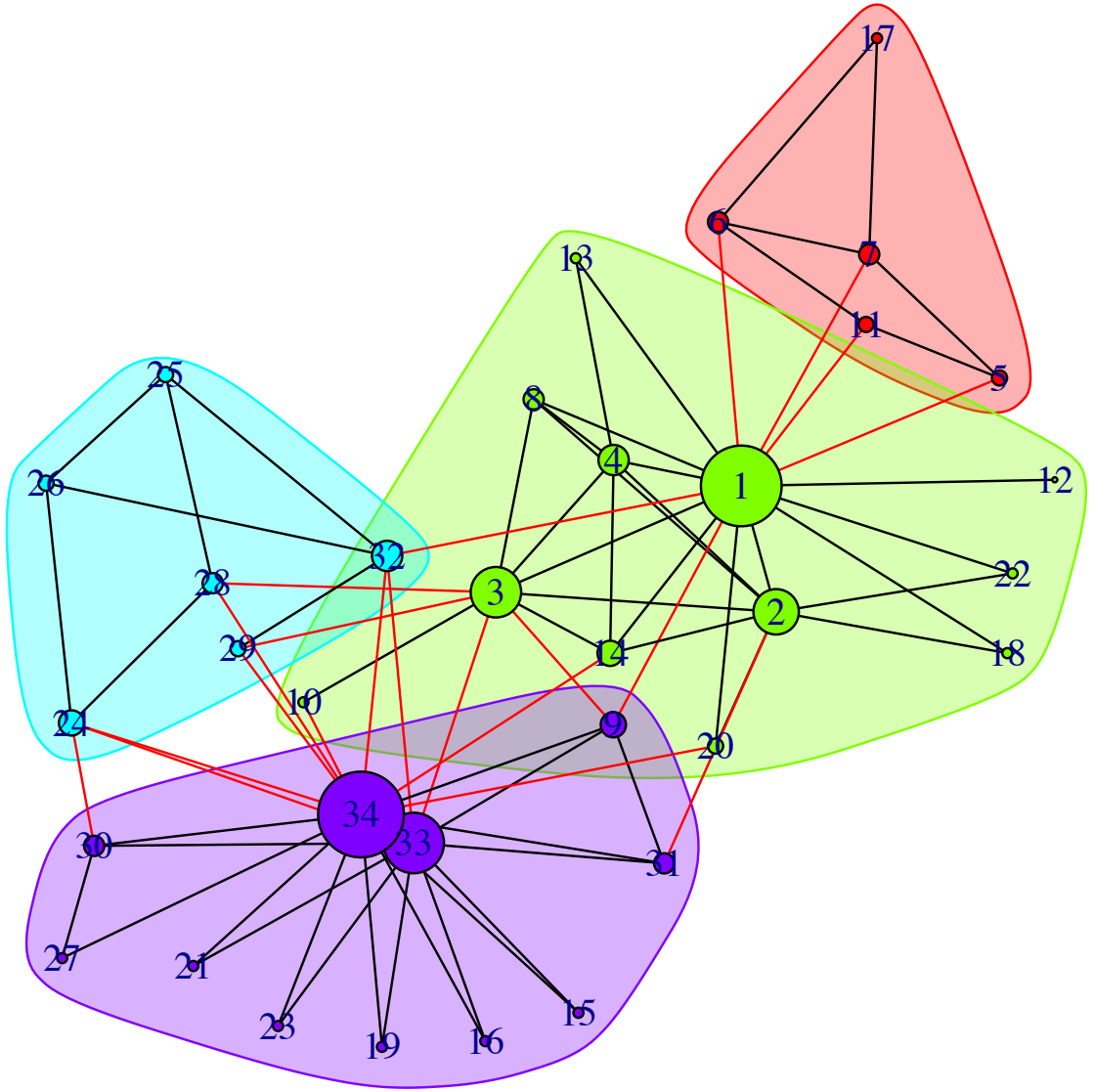


Figure 1. Visualization of the steps of our algorithm. Each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the found communities are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible.

Louvain algorithm
4 communities for the karate club



Infomap: Rosvall & Bergstrom

M. Rosvall and C. T. Bergstrom, Maps of information flow reveal community structure in complex networks, PNAS 105, 1118 (2008)
http://dx.doi.org/10.1073/pnas.0706851105, http://arxiv.org/abs/0707.0609
M. Rosvall, D. Axelsson, and C. T. Bergstrom, The map equation, Eur. Phys. J. Special Topics 178, 13 (2009).
http://dx.doi.org/10.1140/epjst/e2010-01179-1, http://arxiv.org/abs/0906.1405.

iGraph: infomap.community
Cdlib: algorithms.infomap

How does it work?

→ Find community structure that minimizes the expected description length of a random walker trajectory

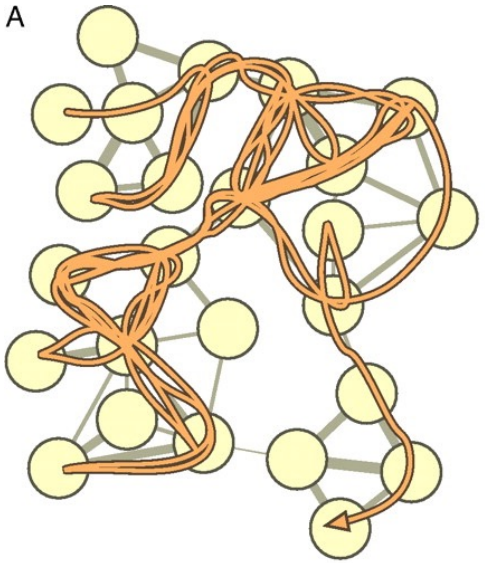
Performance

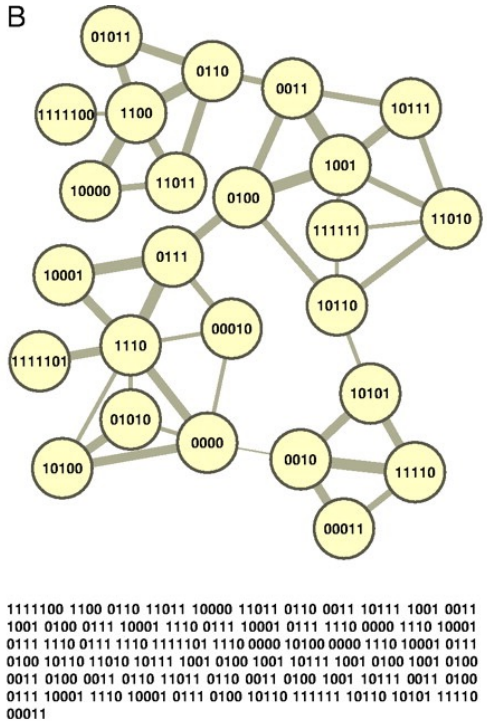
→ Fast
→ Excellent, best performance

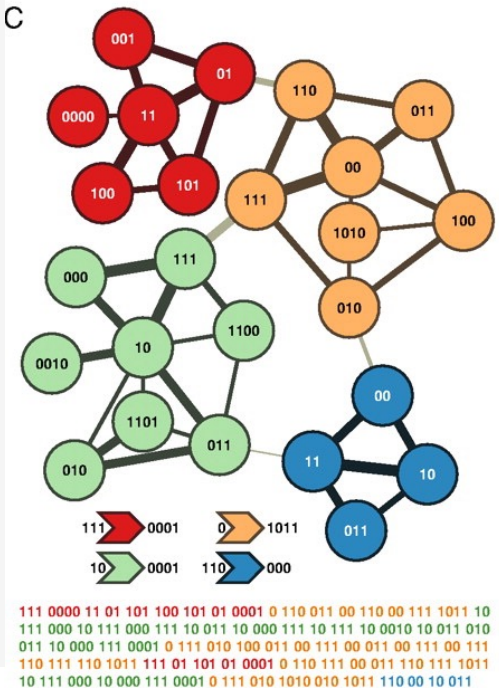
Can be used for weighted and directed graphs

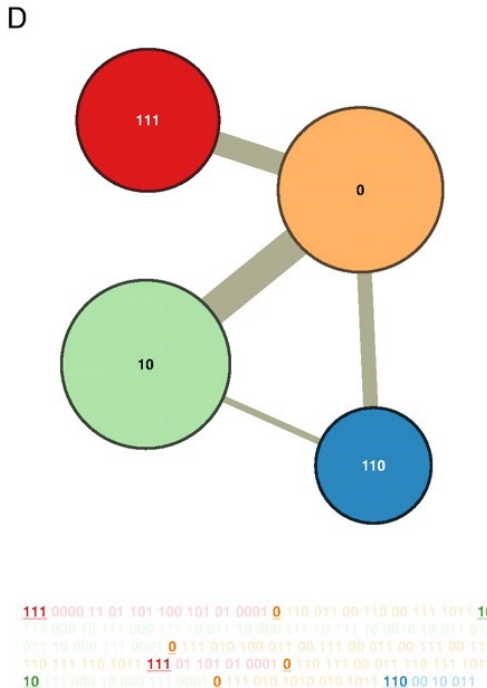
Infomap: Rosvall & Bergstrom

M. Rosvall and C. T. Bergstrom, PNAS 105, 1118 (2008)



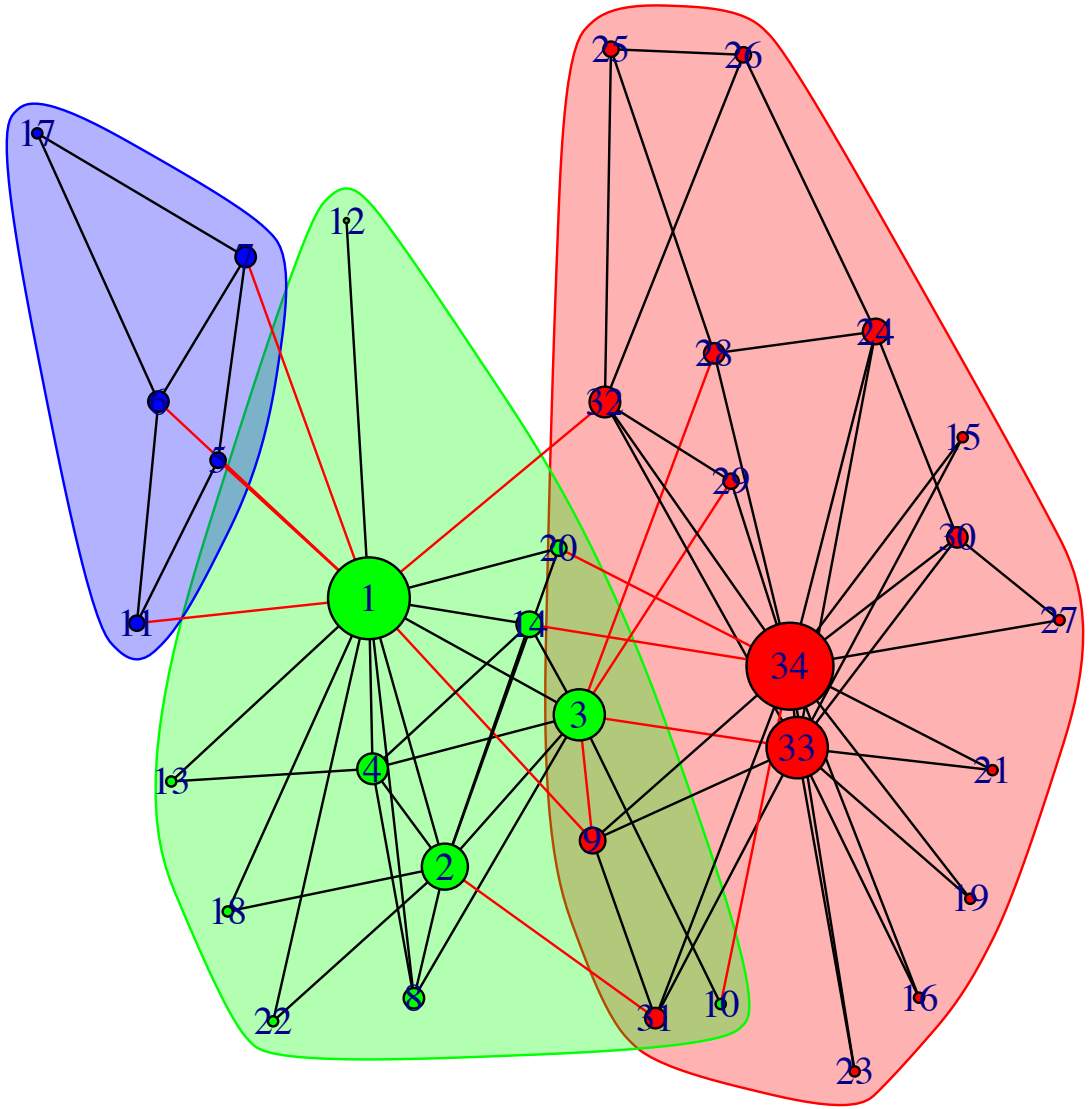






Detecting communities by compressing the description of information flows on networks. (*A*) We want to describe the trajectory of a random walk on the network such that important structures have unique names. The orange line shows one sample trajectory. (*B*) A basic approach is to give a unique name to every node in the network. The Huffman code illustrated here is an efficient way to do so. The 314 bits shown under the network describe the sample trajectory in *A*, starting with 1111100 for the first node on the walk in the upper left corner, 1100 for the second node, etc., and ending with 00011 for the last node on the walk in the lower right corner. (*C*) A two-level description of the random walk, in which major clusters receive unique names, but the names of nodes within clusters are reused, yields on average a 32% shorter description for this network. The codes naming the modules and the codes used to indicate an exit from each module are shown to the left and the right of the arrows under the network, respectively. Using this code, we can describe the walk in *A* by the 243 bits shown under the network in *C*. The first three bits 111 indicate that the walk begins in the red module, the code 0000 specifies the first node on the walk, etc. (*D*) Reporting only the module names, and not the locations within the modules, provides an efficient coarse graining of the network.

Infomap algorithm
3 communities for the karate club



So which one do we choose???

→ Select the method that gives rise to the highest modularity

→ Need insight from what you expect to find, and might need to change to weights of the links to some different link attribute, or might need to select a different node attribute that is representative of system

Method	Modularity	n. communities
Girvan-Newman	0.4012985	5
Fast greedy modularity: Clause-Newman-Moore	0.3806706	3
Random walk: Pons&Latapy	0.3532216	5
Spectral: leading e-vector	0.3934089	4
Louvain: Blondel et al	0.4188034	4
Infomap: Rosvall&Bergstrom	0.4020381	3

→ For the Karate club the best partition is 4 given by the highest modularity, in this case the Louvain method.

Other relevant methods

Radicchi et al

F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. 101, 2658 2004.

How does it work?

→ Hierarchical divisive algorithm: remove edges (links) according to their edge clustering coefficient (in decreasing order).

→ NO optimization of the modularity

→ Look at two types of communities:

Strong: for each node k_internal > k_external

weak: sum of k_internal for all nodes > sum k_external

Other relevant methods

Expert et al : community detection for spatial networks

Paul Expert, Tim S. Evan, Vincent D. Blondel, and Renaud Lambiotte, PNAS, 108, 19, p. 7663-7668 (2011). http://www.pnas.org/content/108/19/7663.abstract

How does it work?

→ Modify the modularity function for spatial networks: the null hypothesis takes into account the spatial features

→ “We show that it is possible to factor out the effect of space in order to reveal more clearly hidden structural similarities between the nodes.”

Community detection algorithms

Expert et al : community detection for spatial networks

Paul Expert, Tim S. Evan, Vincent D. Blondel, and Renaud Lambiotte, PNAS, 108, 19, p. 7663-7668 (2011). http://www.pnas.org/content/108/19/7663.abstract

$$Q = (\text{fraction of links within communities}) - (\text{expected fraction of such links}). \quad [2]$$

In a mathematical expression, modularity reads

$$Q = \frac{1}{2m} \sum_{C \in \mathcal{P}} \sum_{i,j \in C} [A_{ij} - P_{ij}], \quad [3]$$

where $i, j \in C$ is a summation over pairs of nodes i and j belonging to the same community C of $\mathcal{P}$ and therefore counts links between nodes within the same community.

(57–59). The most popular choice, proposed by Newman and Girvan (NG) (55) is

$$P_{ij}^{\mathrm{NG}} = k_i k_j / 2m, \quad \text{then } Q = Q_{\mathrm{NG}}, \quad [4]$$

→ Expected probability for randomized graph

Expert et al : community detection for spatial networks

PNAS, 108, 19, p. 7663-7668 (2011). http://www.pnas.org/content/108/19/7663.abstract

The NG null model only uses the basic structural information encoded in the adjacency matrix. Therefore, it is appropriate when no additional information on the nodes is available but not when additional constraints are known. In networks where distance strongly affects the probability for two nodes to be connected, a natural choice for the null model is inspired by the aforementioned gravity models

$$P_{ij}^{\mathrm{Spa}} = N_i N_j f(d_{ij}) \qquad [5]$$

where N_i is, as in Eq. 1, a notion of importance of node i and where the deterrence function

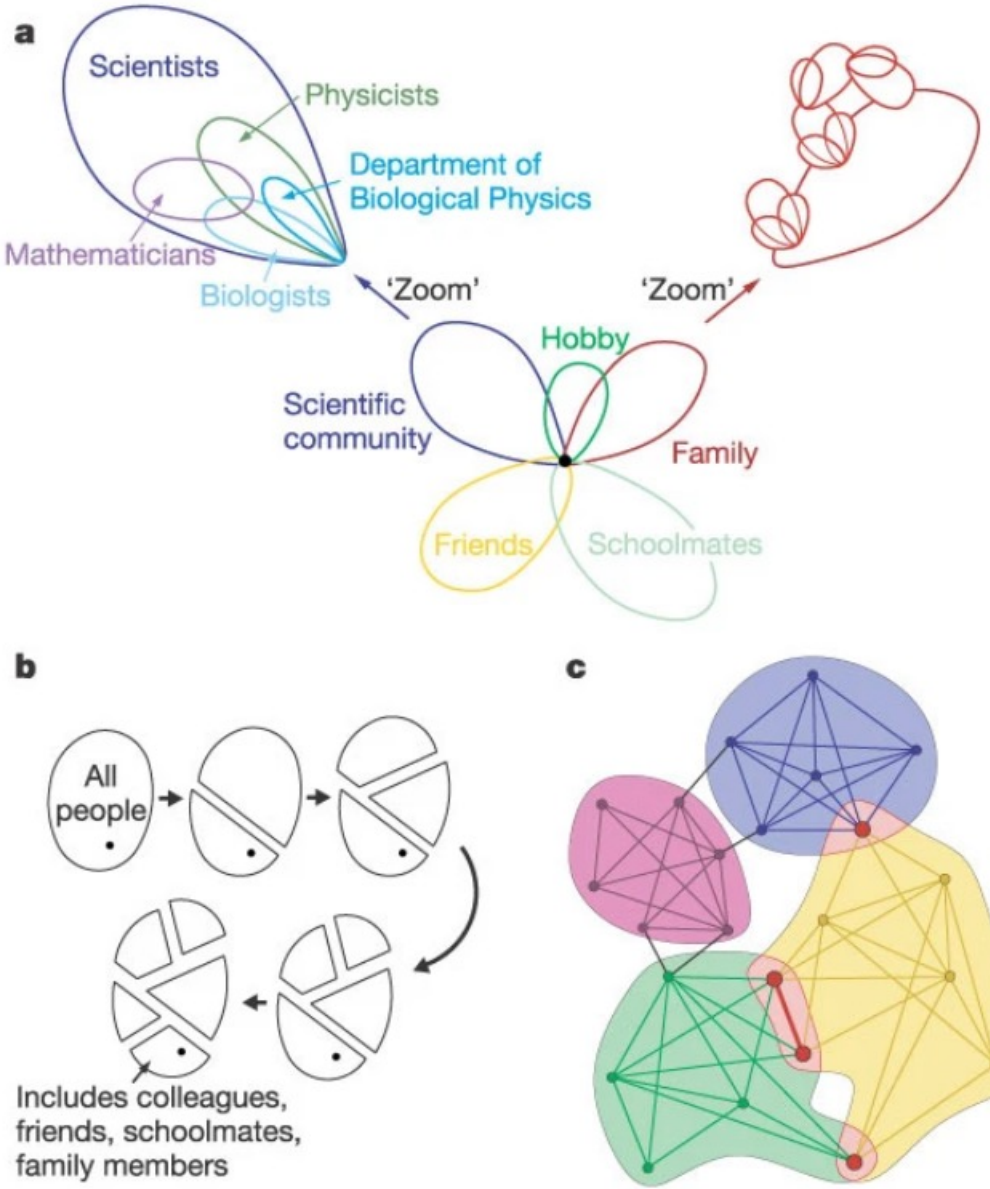
$$f(d) = \frac{\sum_{i,j|d_{ij}=d} A_{ij}}{\sum_{i,j|d_{ij}=d} N_i N_j}, \qquad [6]$$

is the weighted average of the probability $A_{ij}/(N_iN_j)$ for a link to exist at distance d. It is thus directly measured from the data † and not fitted by a determined functional dependence, as is often the case (15). By construction, the total weight of the network is conserved as required. Depending on the system under scrutiny, N_i may be the number of inhabitants in a city or the degree of a node when it corresponds to a single person in a social network. It is worth mentioning that in the latter case and if the embedding in space does not play a role—i.e., where $f(d)$ is flat—the standard NG model is exactly recovered (*SI Text*).

Uncovering the overlapping community structure of complex networks in nature and society

Gergely Palla, Imre Derényi, Illés Farkas & Tamás Vicsek

Nature **435**, 814–818(2005)



Science Advances 24 Jan 2020:
Vol. 6, no. 4, eaav1478
DOI: 10.1126/sciadv.aav1478

SCIENCE ADVANCES | RESEARCH ARTICLE

NETWORK SCIENCE

Community detection in networks without observing edges

Till Hoffmann[1], Leto Peel[2], Renaud Lambiotte[3]*, Nick S. Jones[1,4]*

We develop a Bayesian hierarchical model to identify communities of time series. Fitting the model provides an end-to-end community detection algorithm that does not extract information as a sequence of point estimates but propagates uncertainties from the raw data to the community labels. Our approach naturally supports multiscale community detection and the selection of an optimal scale using model comparison. We study the properties of the algorithm using synthetic data and apply it to daily returns of constituents of the S&P100 index and climate data from U.S. cities.