Newman-Girvan Modularity

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

Let us attribute each node i to a community ci

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

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

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

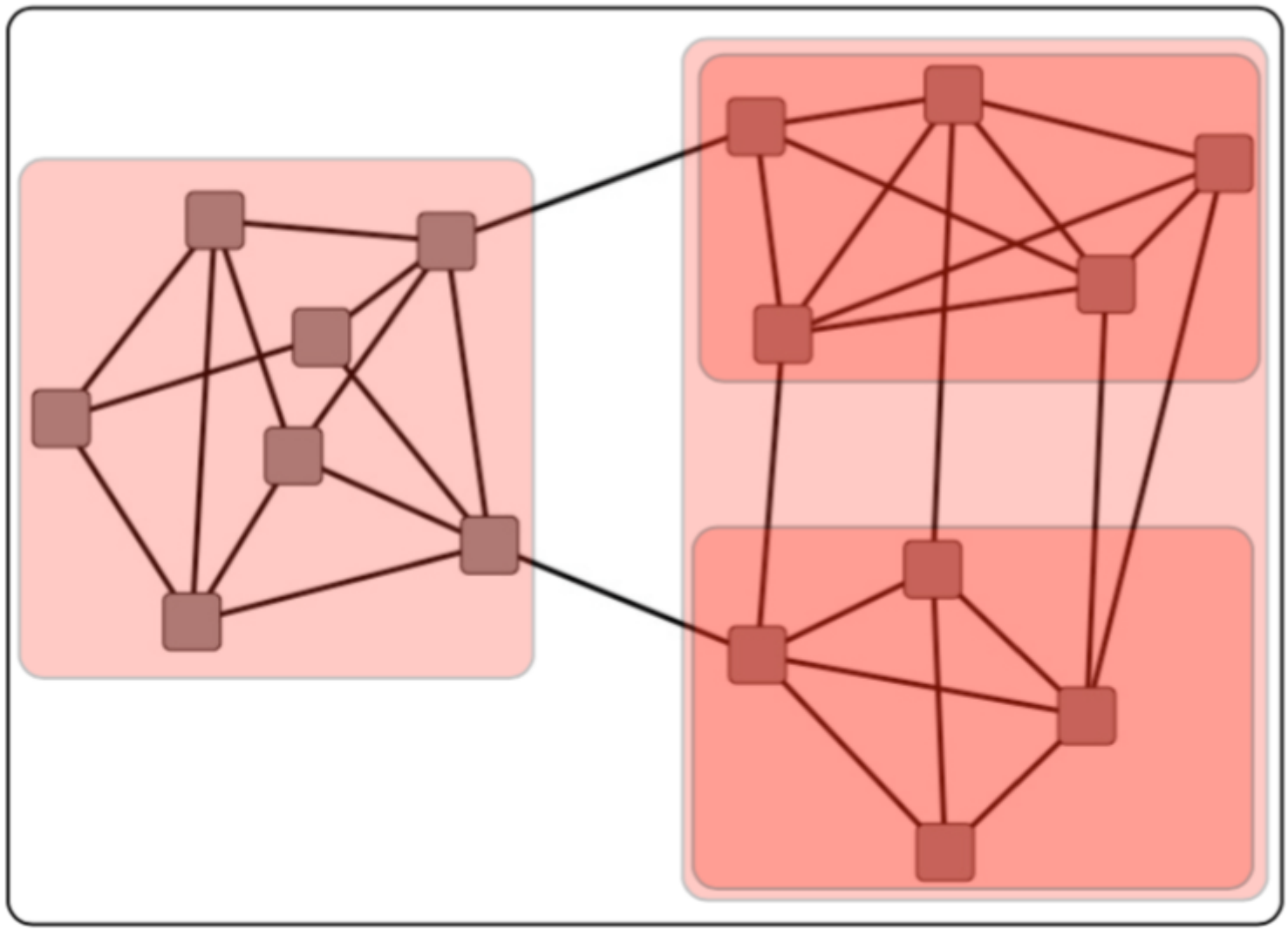
Allows to compare partitions made of different numbers of modules

M.E.J. Newman and M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E, 69, 026113, 2004.

Multi-level modularity

Resolution limit

What about sub (or hyper)-communities in a hierarchical network?



Multi-level modularity

Add a resolution parameter!

Reichardt & Bornholdt

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

Arenas et al.

$$Q(A_{ij} + r I_{ij})$$

Tuning parameters allow to uncover communities of different sizes
Reichardt & Bornholdt different of Arenas, except in the case of a regular graph where

$$\gamma = 1 + r / \langle k \rangle$$

J. Reichardt and S. Bornholdt, Phys. Rev. E 74, 016110 (2006). Statistical mechanics of community detection
A Arenas, A Fernandez, S Gomez, New J. Phys. 10, 053039 (2008). Analysis of the structure of complex networks at different resolution levels

Multi-level modularity

Add a resolution parameter!

Reichardt & Bornholdt

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

Corrected Arenas

$$Q(A_{ij} + r \frac{k_i}{\langle k \rangle} \delta_{ij})$$

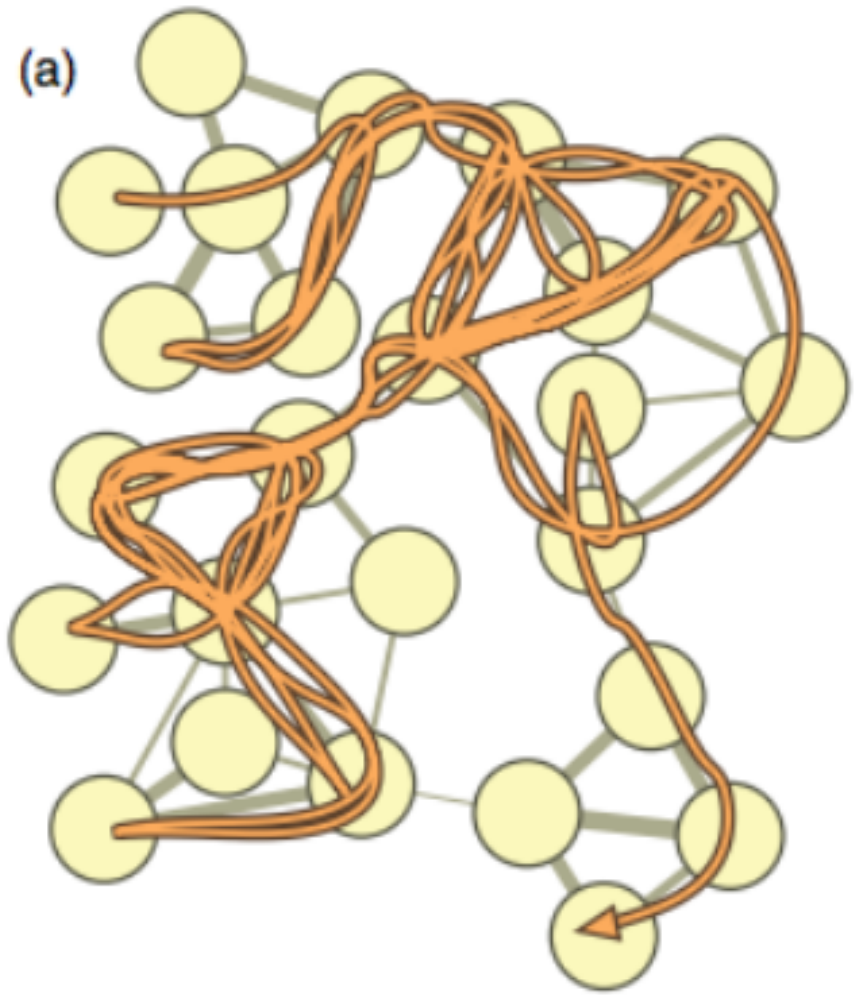
Preserves the eigenvectors of Laplacian (no A) and has a nice dynamical interpretation

Reichardt & Bornholdt = corrected Arenas for any graph

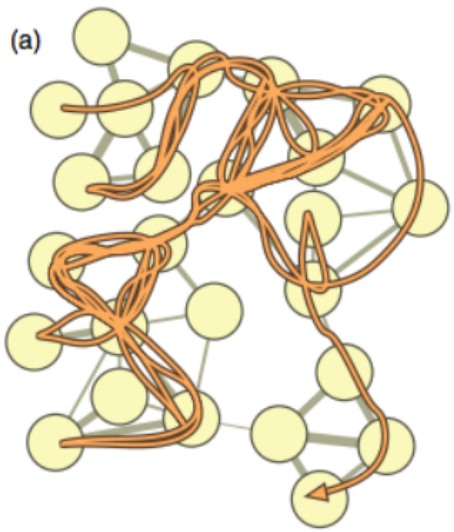
$$\gamma = 1 + r / \langle k \rangle$$

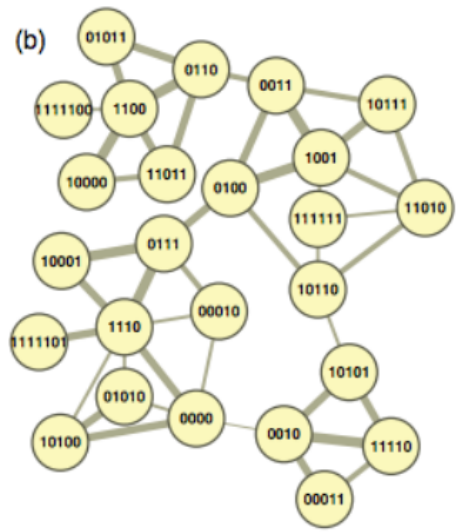
R. Lambiotte, Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on, 546-553 (2010)

Dynamics as way to uncover communities

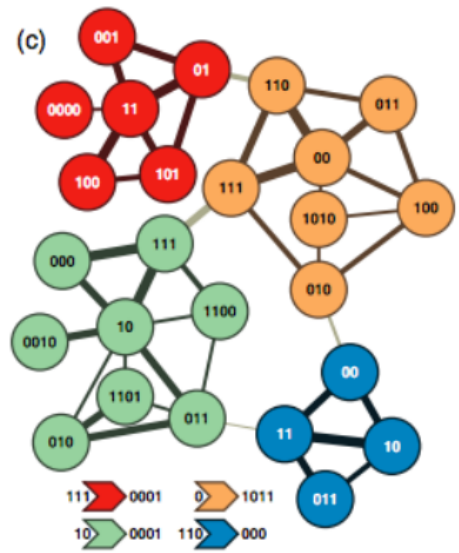


Dynamics as way to uncover communities

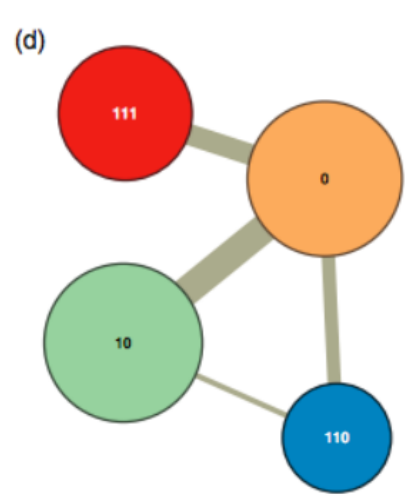




1111100 1100 0110 11011 10000 11011 0110 0011 10111 1001
0011 1001 0100 0111 10001 1110 0111 10001 0111 1110 0000
1110 10001 0111 1110 0111 1110 1111101 1110 0000 10100 0000
1110 10001 0111 0100 10110 11010 10111 1001 0100 1001 10111
1001 0100 1001 0100 0011 0100 0011 0110 11011 0110 0011 0100
1001 10111 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, ... 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$\cdots$. 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

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

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

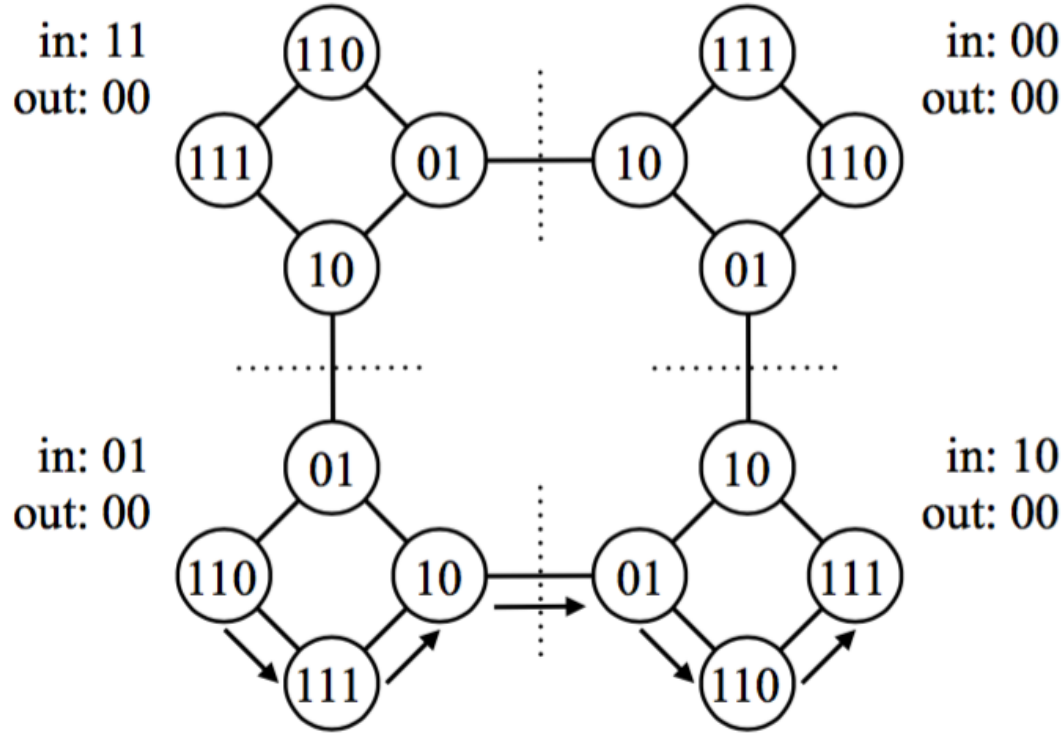


Fig. 3.6 Optimal partitioning according to Infomap and the resulting code words. This example is based on a demo applet available at Martin Rosvall's website http://www.mapequation.org/apps/MapDemo.html.

the trajectory shown by the arrows in the figure is encoded into 011101111000100111011 1. 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

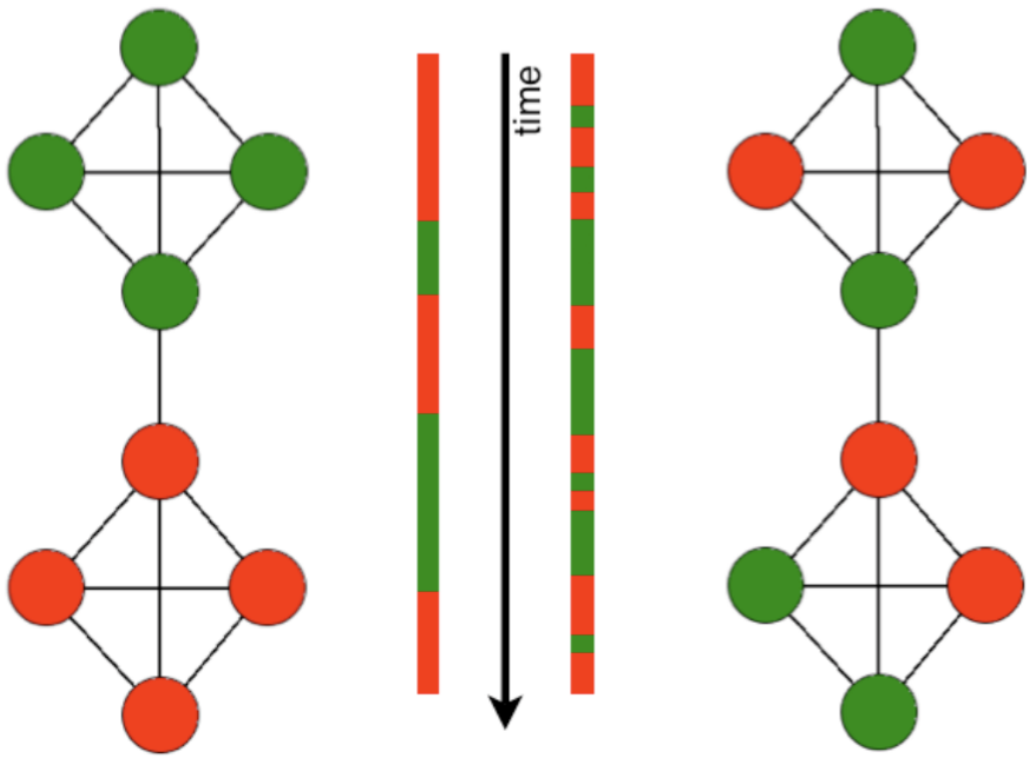
In contrast to the original Huffman code, we have to invest $2N_{\mathrm{CM}}$ code words to mark the entry to and exit from a community. However, we can save the code length when the walker wanders in a community, which occupies a majority of steps. Overall, the mean code length is expected to be smaller 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.

$$L(\mathrm{M}) = q_{\curvearrowright} H(\mathscr{Q}) + \sum_{C} p^{C}_{\circlearrowright} H(\mathscr{C}^{C})$$

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



J.-C. Delvenne, S. Yaliraki & M. Barahona, Stability of graph communities across time scales. arXiv:0812.1811.

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 t0 and t0 +t when the system is at equilibrium

$P(C, t_0, \infty)$

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

J.-C. Delvenne, S. Yaliraki & M. Barahona, Stability of graph communities across time scales. arXiv:0812.1811.

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:

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

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

Counting versus flows

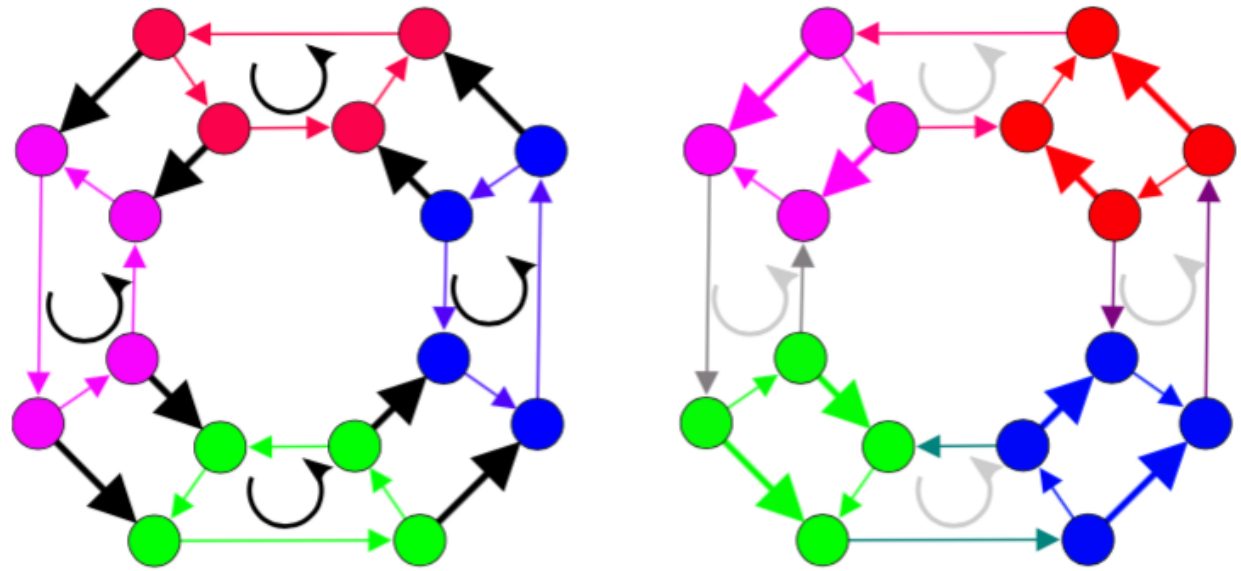
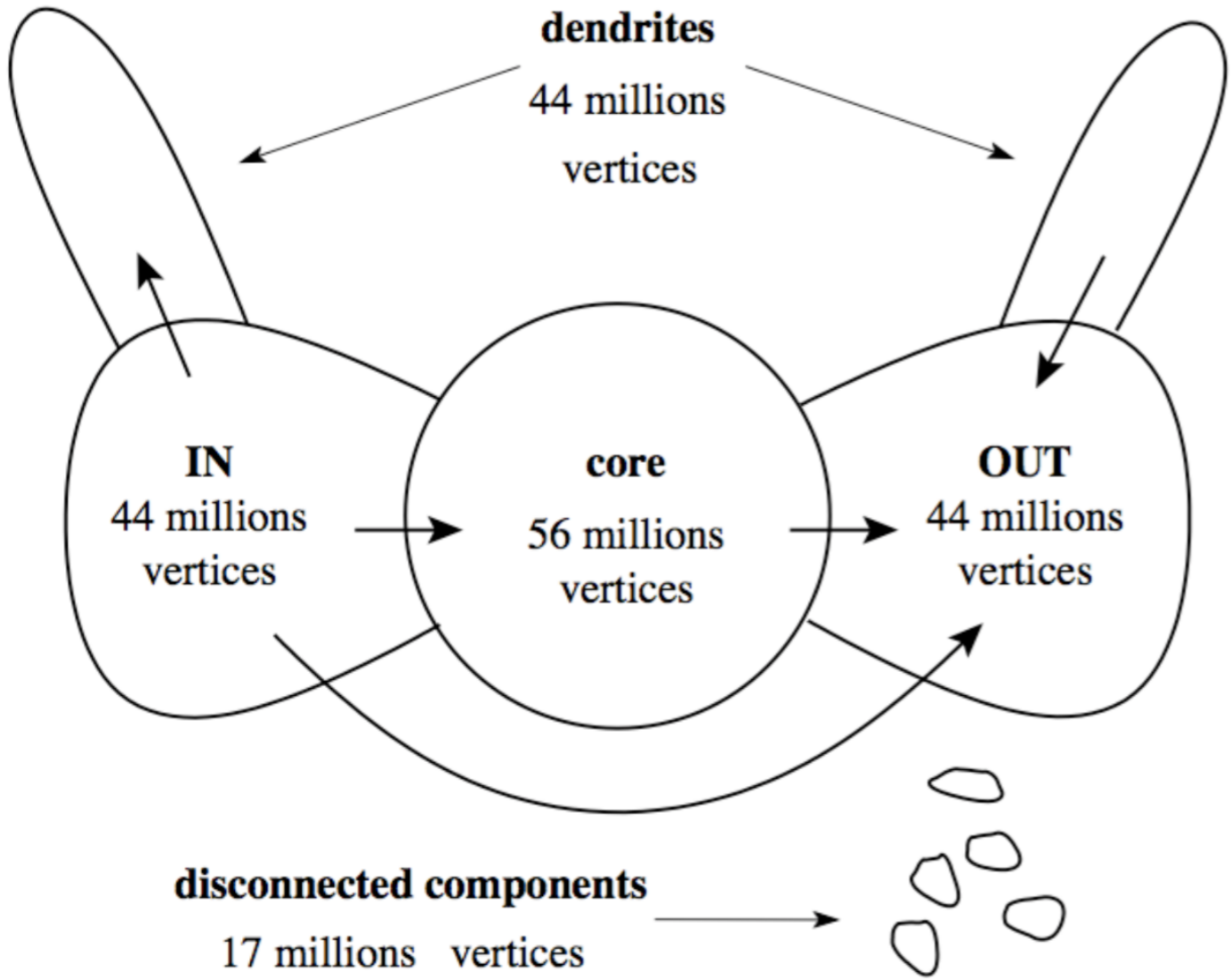


Fig. 4. **Directed Markov Stability versus extensions of modularity.** In this toy network [16], the weight of the bold links is twice the weight of the other links. The partition on the left (indicated by different colors) optimizes directed Markov Stability (34), which intrinsically contains the pagerank 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



Markov stability versus Modularity

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

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

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

$$R(1) \neq Q(A) \qquad \text{but} \qquad R(1) = Q(Y)$$

$$Y = \frac{X + X^T}{2} \qquad X_{ij} = \frac{A_{ij}}{k_j^{out}} \pi_j$$

Time as a resolution parameter

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

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

equilibrium

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

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

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

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

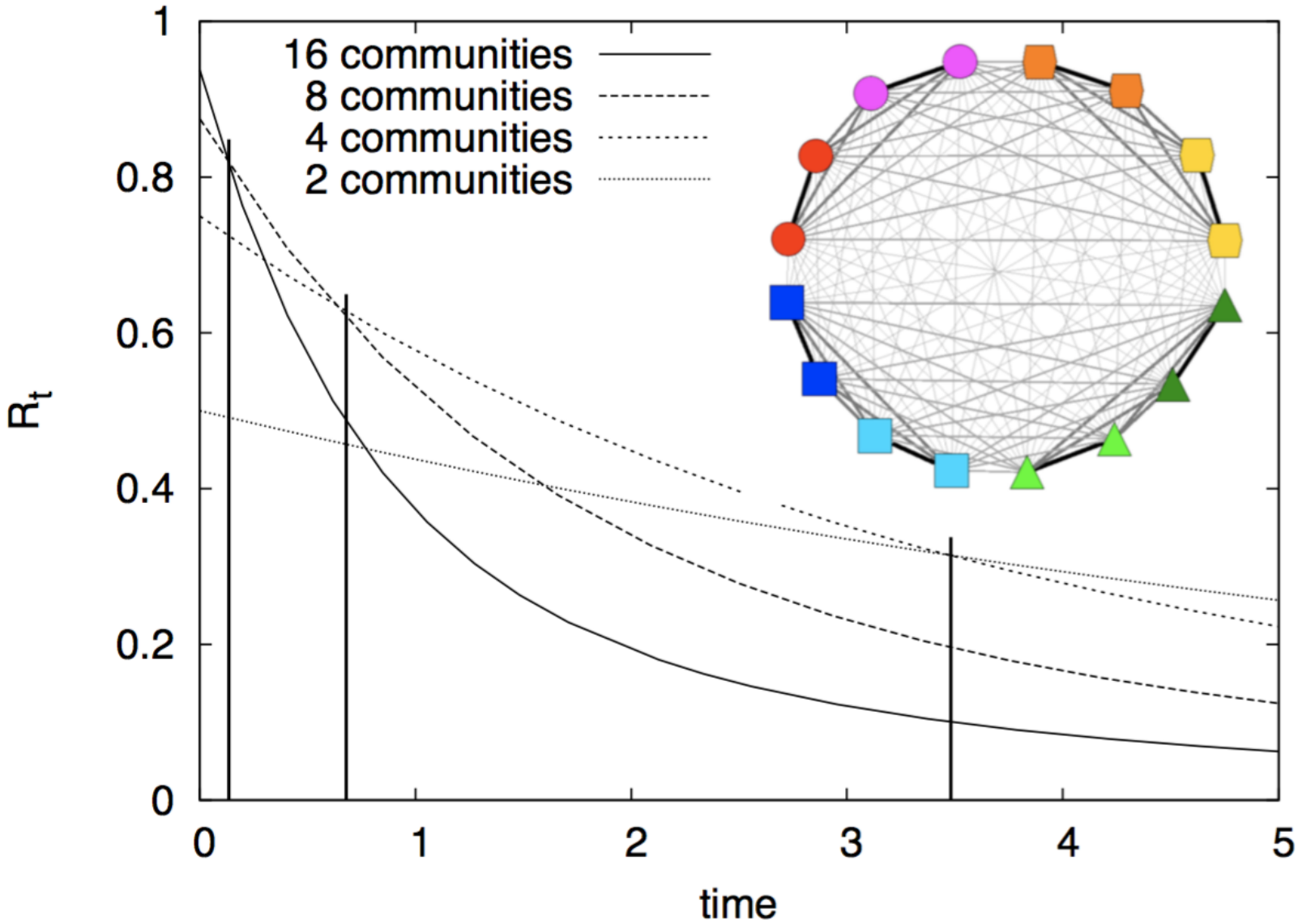
$$R(t) \approx (1-t)R(0) + tQ_C \equiv Q(t)$$ Tuneable modularity of Reichart and Bornholdt

time

Asymptotically, two-way partition given by the Fiedler vector

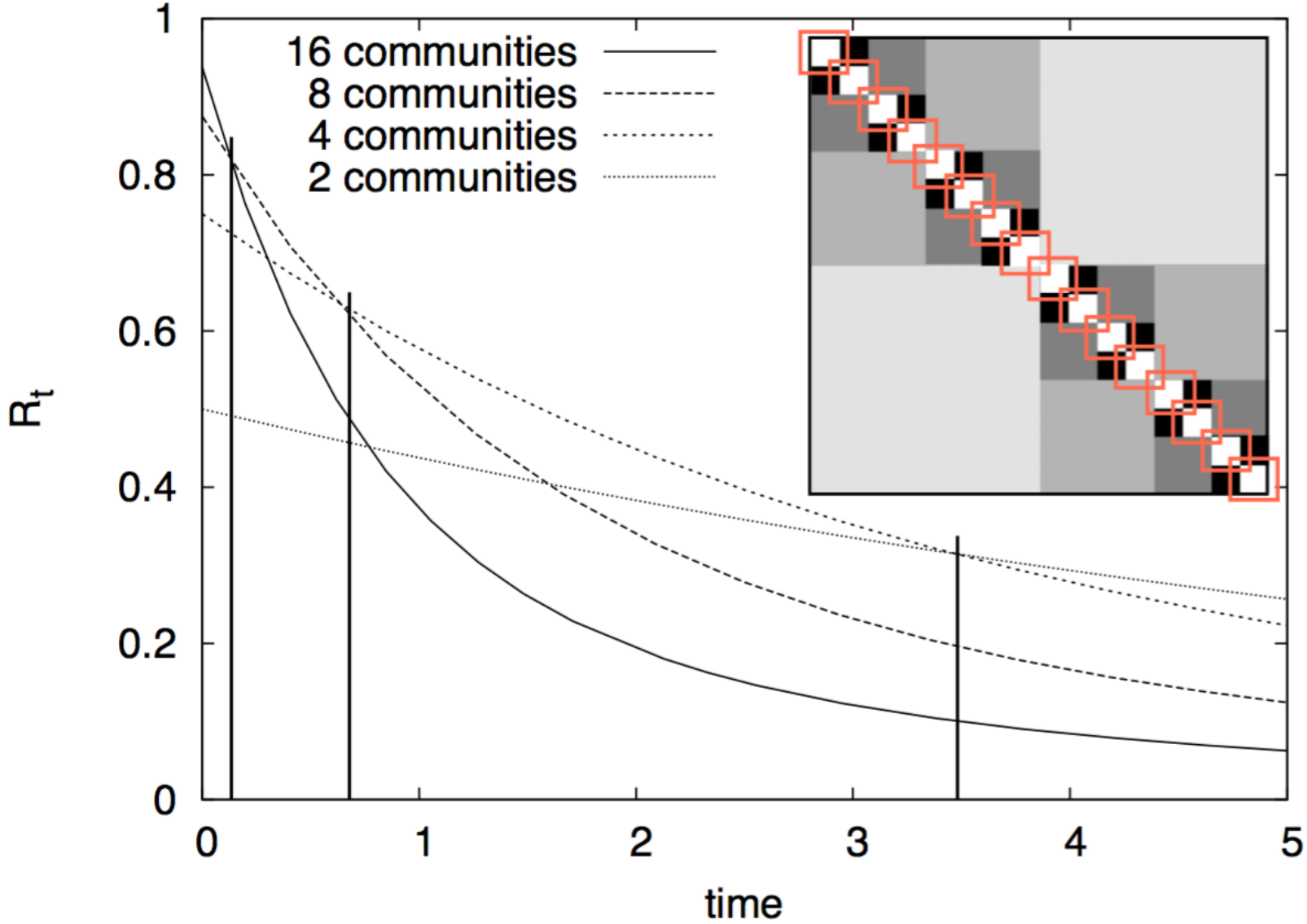
Time as a resolution parameter

Time is a “resolution parameter”: larger and larger communities when time is increased



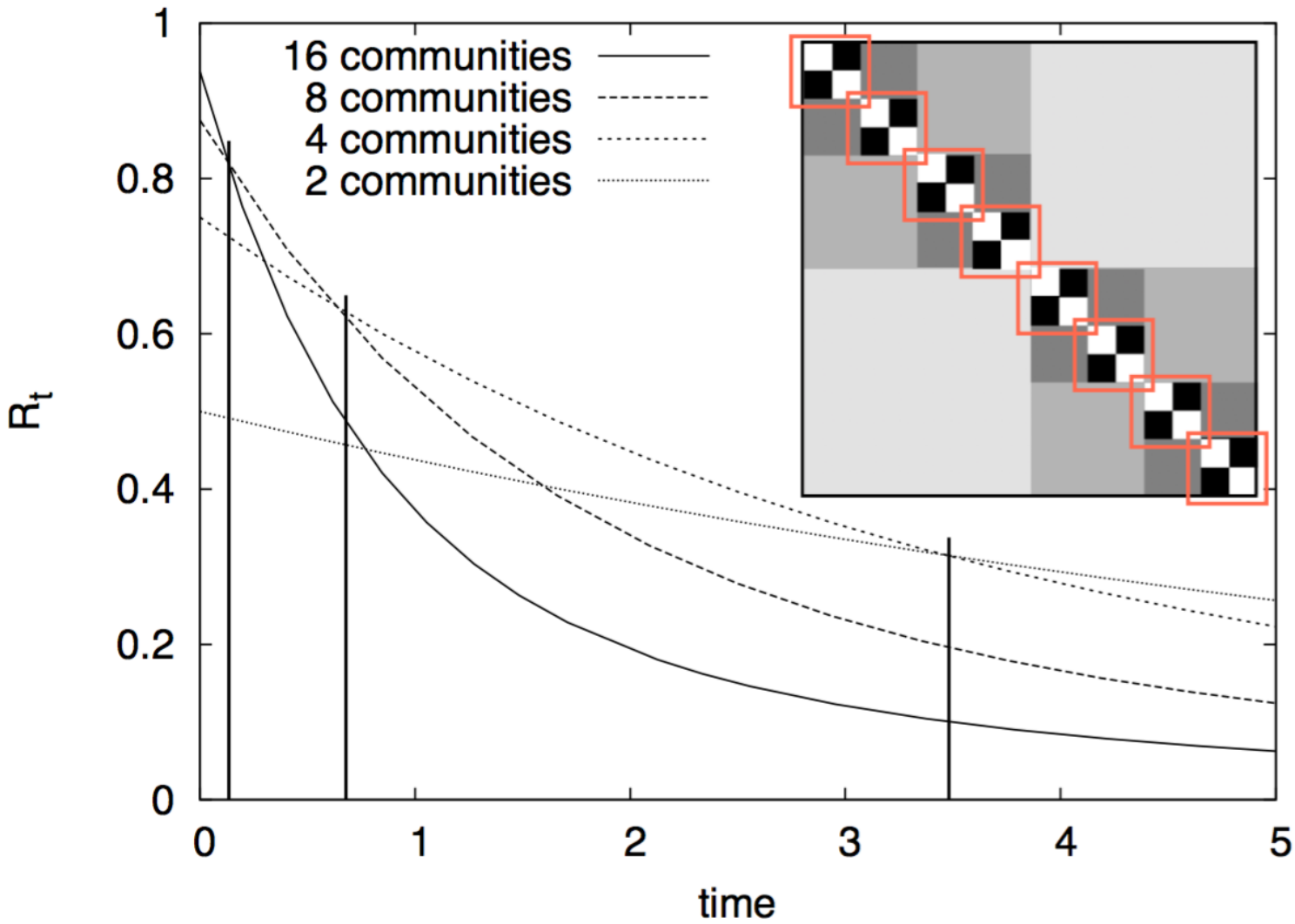
Time as a resolution parameter

Time is a “resolution parameter”: larger and larger communities when time is increased



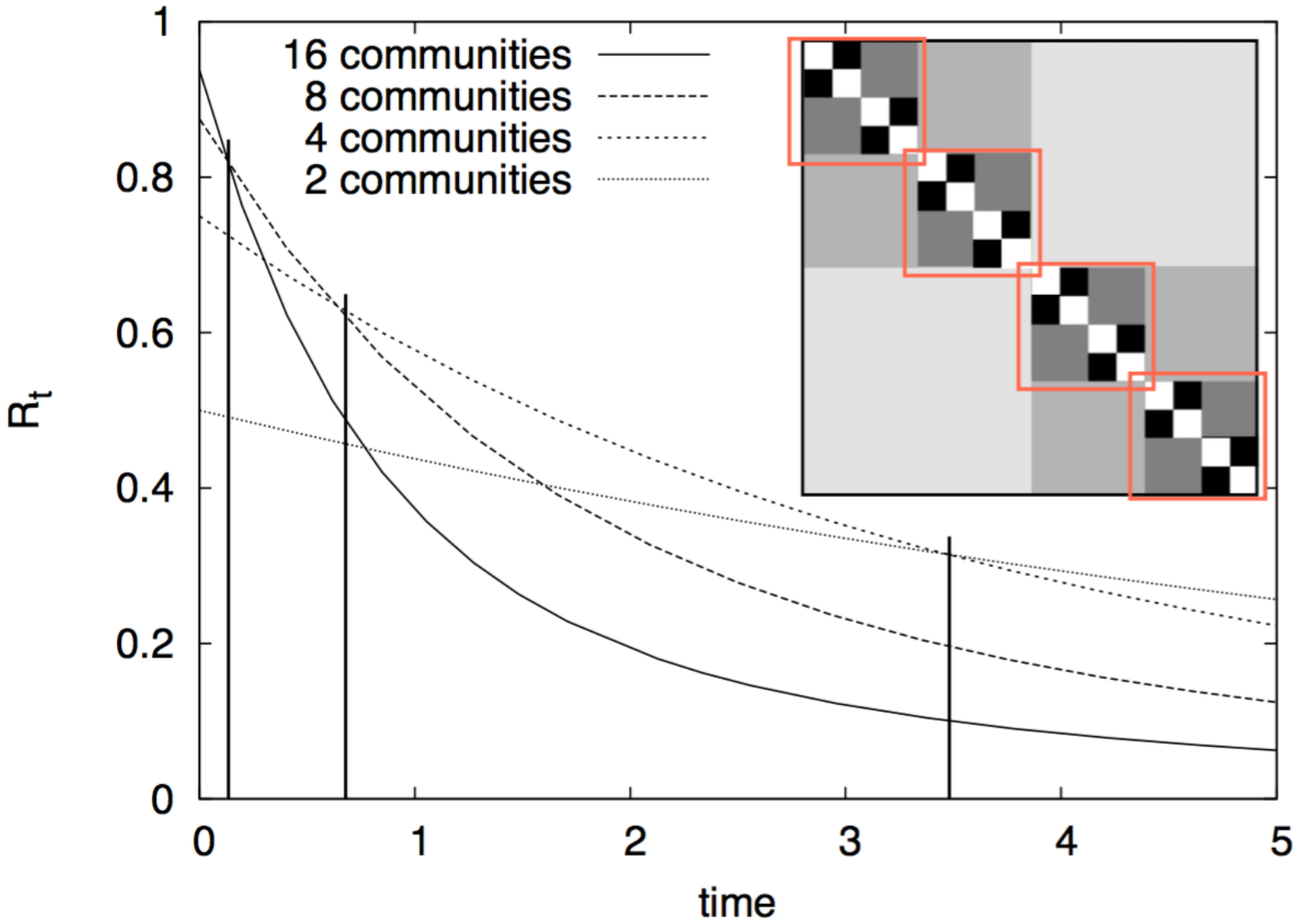
Time as a resolution parameter

Time is a “resolution parameter”: larger and larger communities when time is increased



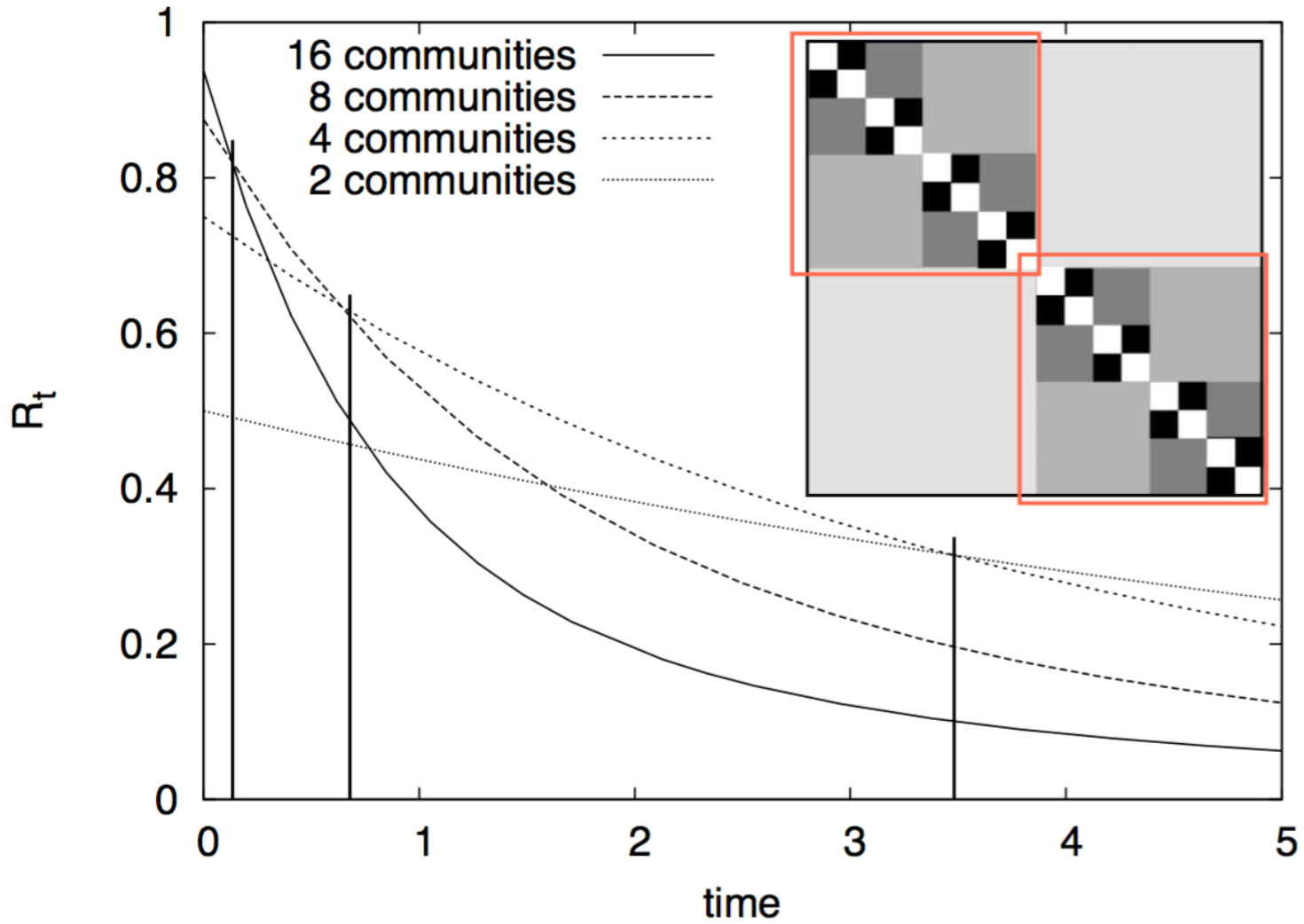
Time as a resolution parameter

Time is a “resolution parameter”: larger and larger communities when time is increased



Time as a resolution parameter

Time is a “resolution parameter”: larger and larger communities when time is increased



In practice: optimization?

The stability R(t) of the partition of a graph with adjacency matrix A is equivalent to the modularity Q of a time-dependent graph with adjacency matrix X(t)

$$X_{ij}(t) = \left(e^{t(B-I)}\right)_{ij} k_j \qquad X_{ij}(t) = X_{ji}(t)$$

which is the flux of probability between 2 nodes at equilibrium and whose generalised degree is

$$\sum_j X_{ij}(t) = k_i$$

$$R(t) = \sum_{i,j} X_{ij}(t)/2m - k_i k_j/(2m)^2 \ \delta(c_i, c_j) = Q(X(t))$$

For very large networks: $R(t) \approx (1-t)R(0) + tQ_C \equiv Q(t)$

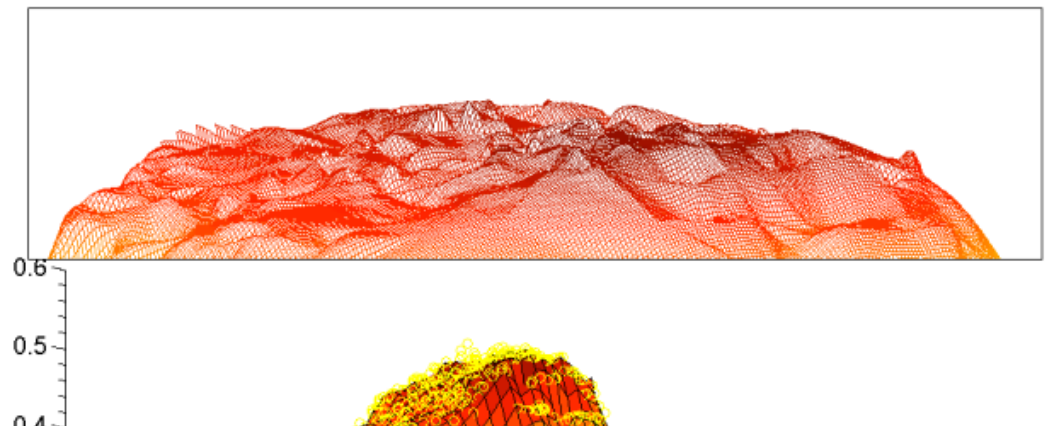
In practice: selection of the significant scales?

The optimization of R(t) over a period of time leads to a sequence of partitions that are optimal at different time scales.

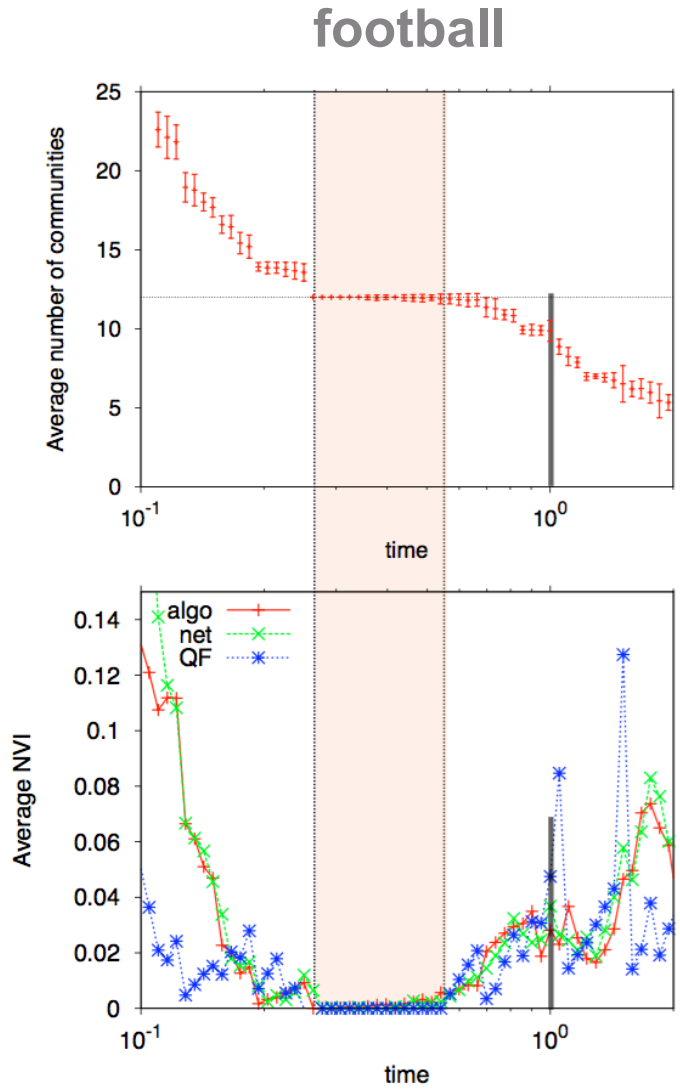
How to select the most relevant scales of description?
The significance of a particular scale is usually associated to a certain notion of robustness of the optimal partition. Here, robustness indicates that a small modification of the optimization algorithm, of the network, or of the quality function does not alter this partition.

We look for regions of time where the optimal partitions are very similar. The similarity between two partition is measured by the normalised variation of information.

Intuition: at a bad scale, several competing maxima make the lanscape more rugged, leading to a sensitivity in the outcome of the algorithm



In practice: selection of the significant scales?



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

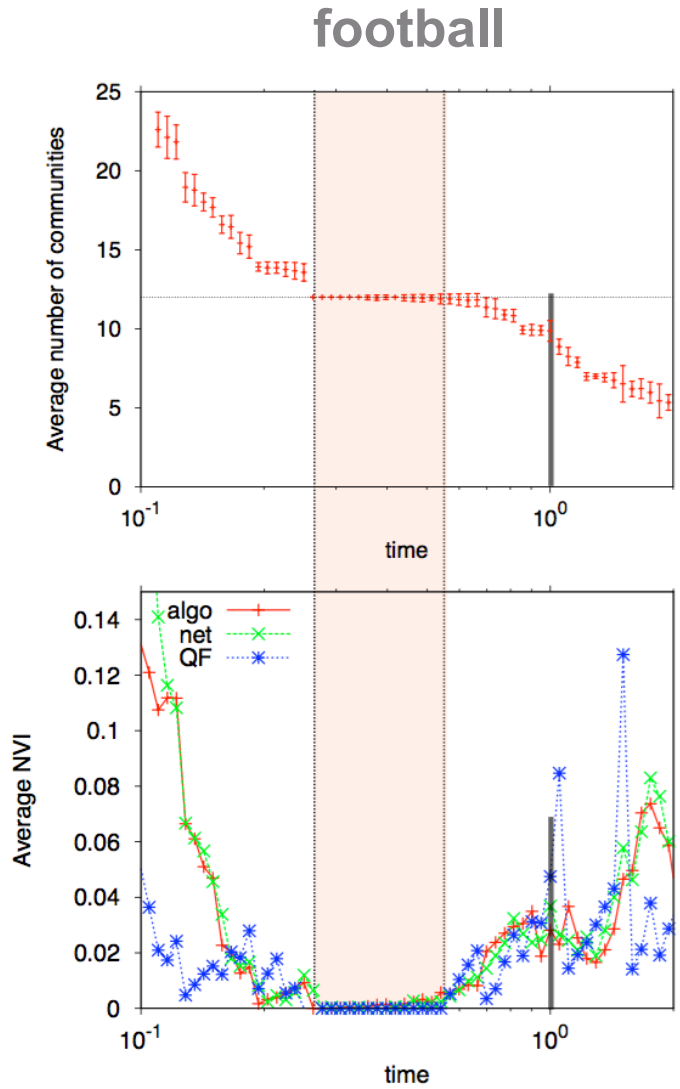
$$\langle V \rangle_{\text{algo}}(t) = \frac{2}{T(T-1)} \sum_{i=1}^{T} \sum_{i'=i+1}^{T} \hat{V}(\mathcal{P}_i(t), \mathcal{P}_{i'}(t))$$

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 -> high degeneracy in the landscape: uncovered partitions are not to be trusted; wrong resolution

In practice: selection of the significant scales?



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

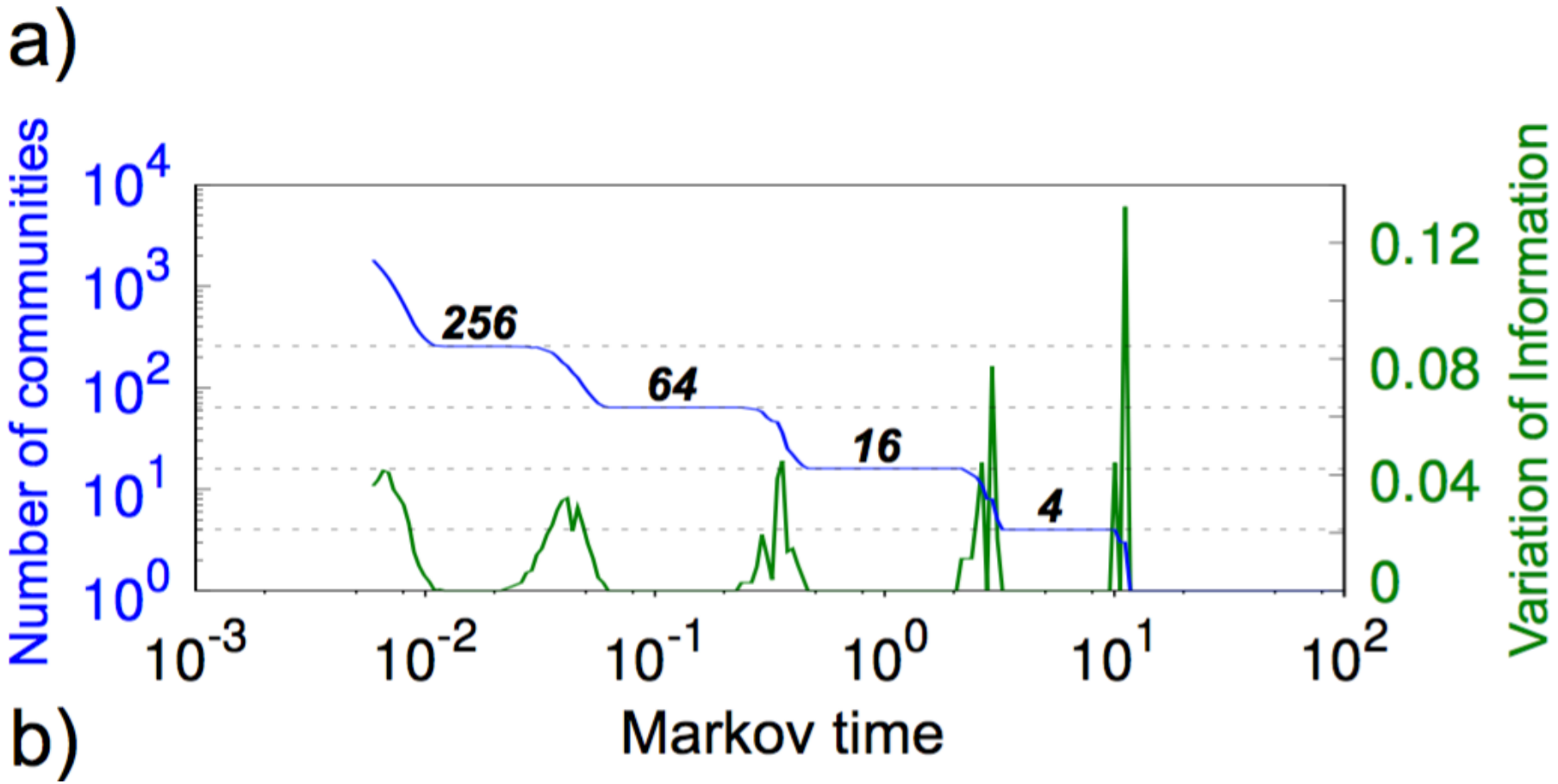
$$\langle V \rangle_{\text{algo}}(t) = \frac{2}{T(T-1)} \sum_{i=1}^{T} \sum_{i'=i+1}^{T} \hat{V}(\mathcal{P}_i(t), \mathcal{P}_{i'}(t))$$

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

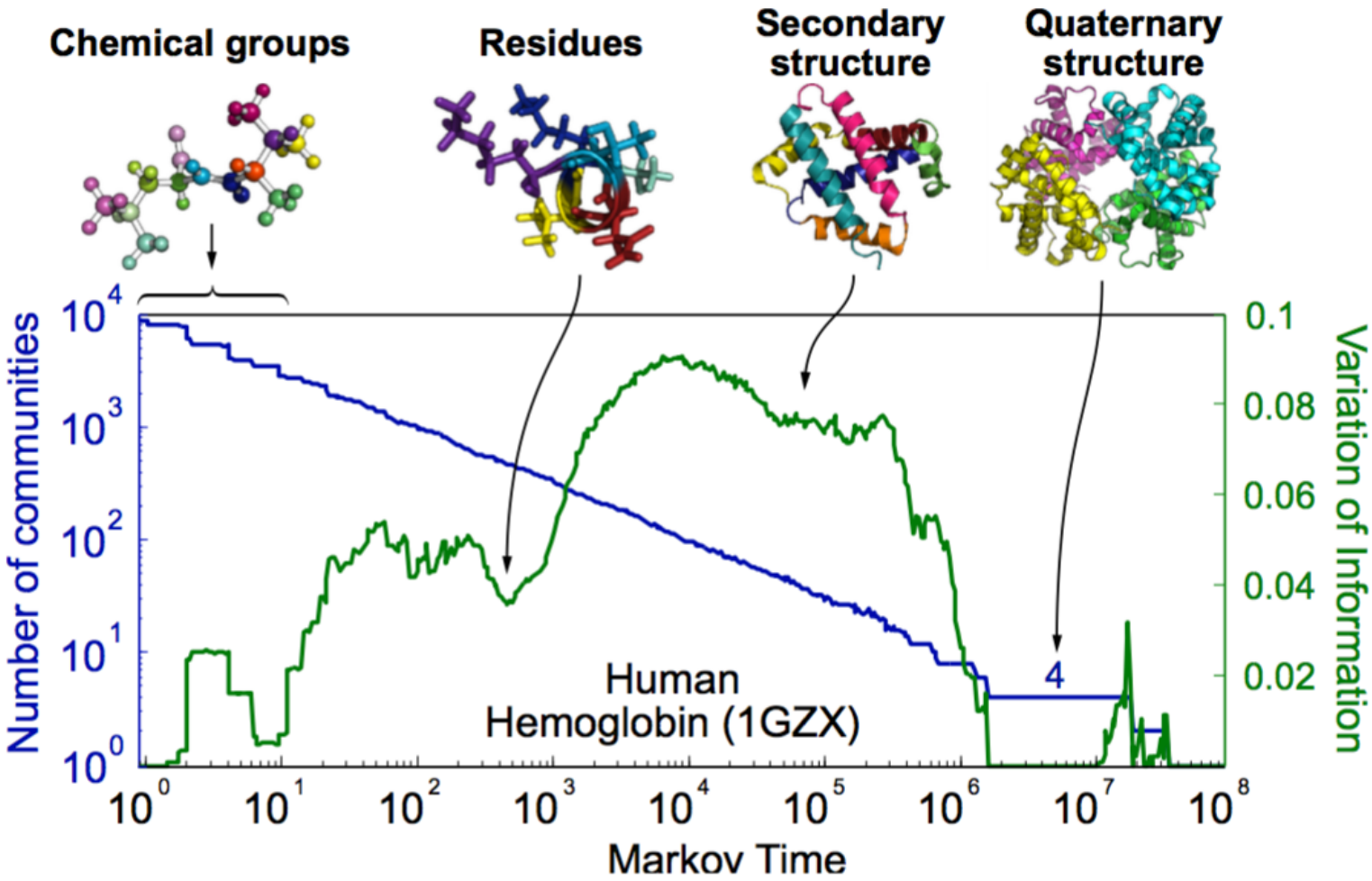
QF: for each t, one optimization. Partitions at 5 successive values of t are compared.

Compatible notions of robustness:
Lack of robustness -> high degeneracy in the landscape: uncovered partitions are not to be trusted; wrong resolution

Time as resolution parameter



Time as resolution parameter



Time as resolution parameter

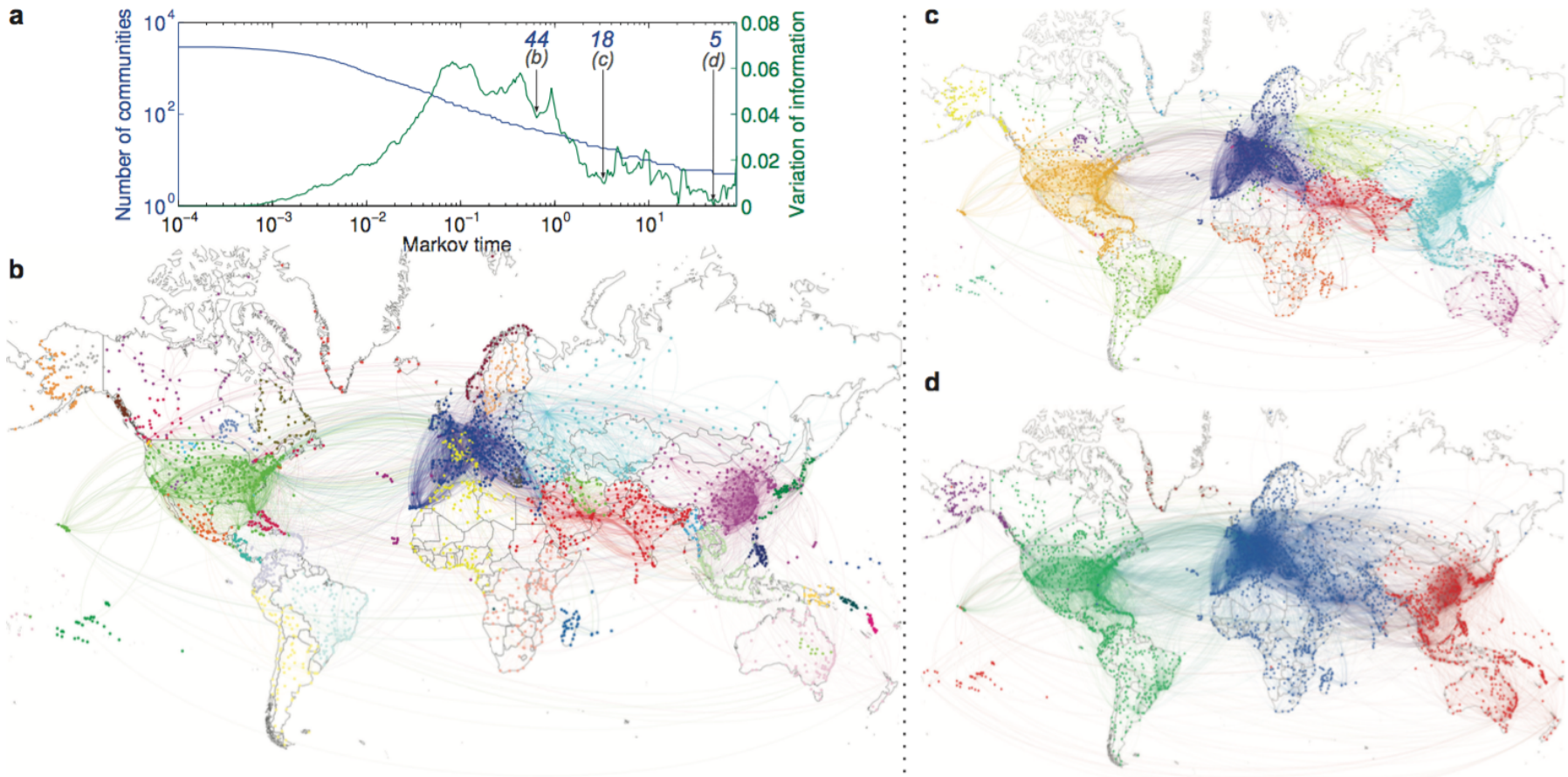


Fig. 8. **Flow communities at multiple scales in an airport network.** The airport network [82] contains $N = 2905$ nodes (airports) and 30442 weighted directed edges. The weights record the number of flights between airports (i.e., the network does not take into account passenger numbers, just the number 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.).

Similarity measures and kernels

When working with networks, many tasks can be simplified by defining a proper measure of distance, or similarity, between pairs of nodes

-> Node classification
-> Link prediction
-> Node clustering

Similarity matrix is N times N and encodes the similarity between nodes according to some principle.

Basic example: adjacency matrix (but very coarse-grained, 0 or 1)

-> Longer paths allow to define more refined measures allowing to rank pairs of nodes.

Walktrap

$$r_{ij} = \sqrt{\sum_{\ell=1}^{N} \frac{(T_{i\ell}^n - T_{j\ell}^n)^2}{k_\ell}}, \tag{211}$$

where n is the number of steps in a DTRW. The distance r_{ij} is small when a pair of random walkers — one starting from v_i and the other starting from v_j — visit each node with similar probabilities after n steps. The denominator k_ℓ discounts the fact that a walker visits v_ℓ with a probability proportional to k_ℓ at equilibrium.