Université catholique de Louvain
Ecole Polytechnique de Louvain
Institute of Information and Communication
Technologies, Electronics and
Applied Mathematics

Algorithms for community and role detection in networks

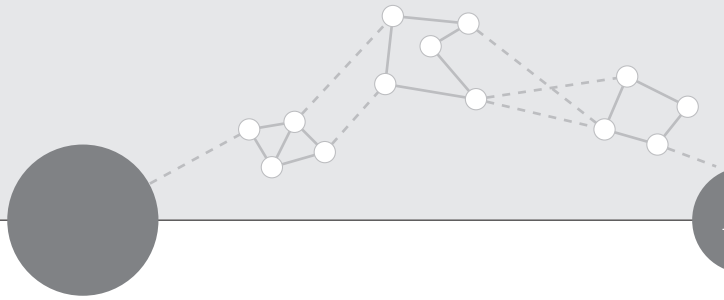
Arnaud Browet

Thesis submitted in partial fulfillment of the requirements for the Ph.D. Degree in Engineering Sciences

Thesis Committee

Advisors:	Paul Van Dooren, UCL
	Pierre-Antoine Absil, UCL
Jury:	Mauricio Barahona, Imperial College
	Renaud Lambiotte, University of Namur
	Christophe De Vleeschouwer, UCL
Chairman:	Raphaël Jungers, UCL

Louvain-la-Neuve, September 30, 2014.



Acknowledgment

Doing a Ph.D. has certainly been the biggest professional challenge that I have faced so far. But when I came back from the private sector to academia, I certainly did not expect to live such a great experience.

First of all, I would like to thank my advisors Paul Van Dooren and Pierre-Antoine Absil. They have offered me the opportunity to follow my own research interests and have always been of sound advice. I will always keep excellent memories of the time spent together, whether it was for science or pleasure.

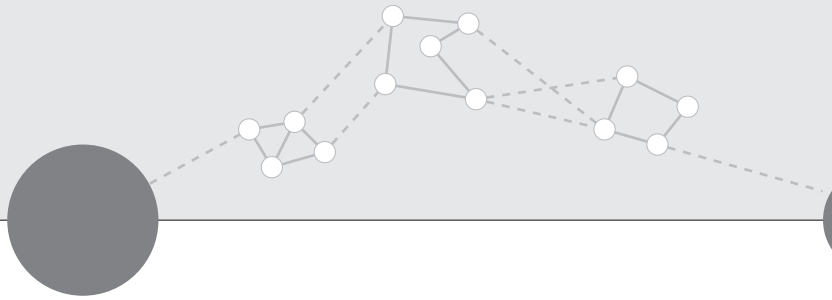
I am grateful to all the members of my thesis committee - Raphaël Jungers, Mauricio Barahona, Renaud Lambiotte and Christophe De Vleeschouwer - and to the additional members of my advisory committee - Benoit Macq and Marc Van Barel - for their time and commitment. Your constructive comments and remarks have greatly improved the quality of my thesis.

During my thesis, I was funded as a teaching assistant at the Pole of Mathematical engineering of the Institute of Information and Communication Technologies, Electronics and Applied Mathematics at the Université catholique de Louvain. I would like to express my gratitude for the administrative and financial support of the university. In particular, I warmly acknowledge the amazing assistance of the secretaries of the department - Isabelle Hisette and Nathalie Ponet. I am also thankful to the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

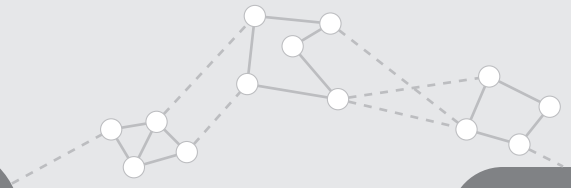
I have had the chance to meet and collaborate with talented researchers within our Large Graphs and Networks research group - Vincent Blondel, Yurii Nesterov, Jean-Charles Delvenne and Julien Hendricks. But my fellow Ph.D. students (and office mates) have also contributed a lot to enrich this amazing experience - Vincent Traag, Pierre Borckmans, Nicolas Boumal, Romain Hollanders, Francois-Xavier Orban, Adeline De Cuyper,... I simply can not name them all but none will be forgotten. I also deeply enjoyed the time spent in the Euler building, at the coffee breaks

or around a drink, in company of Etienne Huens, Robert "Bob" David and Nicolas Hudon.

Finally, I would never have been able to achieve my thesis without the encouragements and patience of my family. In particular, my father who has always been there for me and constantly helped me to reach my goals. You are the best dad ever. Last but definitely not least, I can not find words strong enough to express my feelings about my life partner. Sophie, those last few years have been filled of emotions, pain or happiness, doubts and joys, and you have always been my first and best support. I am blessed to live with you and you gave me the best baby girl I could ever dream of. I will always love you.



Contents



Symbols and Notations

A — adjacency matrix (p. 17)

B — adjacency matrix of the reduced graph (p. 145)

C_n — cycle graph (p. 20)

D — diagonal matrix of degree (p. 29)

E — set of edges (p. 17)

$G(V, E)$ — graph associated with its vertices and edges (p. 17)

$H(\sigma)$ — energy of a partition (p. 37)

I — identity matrix (p. 155)

K_n — complete graph (p. 21)

K_{n_1,n_2} — complete bipartite graph (p. 22)

L — Laplacian matrix (p. 29)

N_{ij} — null model (p. 38)

$P(.)$ — probability distribution (p. 25)

Q_w — modularity function (p. 43)

S — similarity matrix (p. 151)

V — set of vertices (p. 17)

W — weighted adjacency matrix (p. 19)

$\Delta H(c_1 \to i \to c_2)$ — variation of energy switching i from c_1 to c_2 (p. 63)

$\Delta H\left(\sigma^{(1)}, \sigma^{(2)}\right)$ — difference of energy between partitions (p. 61)

$\Delta H\left(c \to \{c_1, c_2\}\right)$ — variation of energy splitting c (p. 63)

$\Delta H(c_1 \cup c_2)$ — variation of energy merging c_1 and c_2 (p. 63)

$\Pi^{(r)}$ — best low-rank projector (p. 164)

$\delta(.,.)$ — Kronecker delta (p. 37)

$[. \mid .]$ — horizontal concatenation (p. 151)

$\langle . \rangle$ — expected value (p. 25)

μ_T — topological mixing parameter (p. 101)

μ_W — weight mixing parameter (p. 104)

$\|.\|_F$ — Frobenius norm (p. 151)

$\otimes$ — Kronecker product (p. 152)

$\rho(.)$ — spectral radius (p. 152)

σ — partition (p. 37)

σ_i — community index of node i (p. 36)

$\mathbf{1}$ — vector of all 1 (p. 18)

e_{int}, e_{ext} — internal and external density (p. 102)

k, k^{out}, k^{in} — undirected, outgoing and incoming degree (p. 18)

k_{int}, k_{ext} — internal and external degree (p. 101)

m — number of edges (p. 18)

m_w — sum of edge weight (p. 43)

n — number of nodes (p. 17)

n_c — number of nodes in community c (p. 45)

s, s^{out}, s^{in} — undirected, outgoing and incoming strength (p. 19)

Acronyms

CNM — Clauset-Newman-Moore (p. 108)

CPM — constant Potts model (p. 113)

ER — Erdős-Rényi (p. 102)

FCE — fast community extraction (p. 108)

LFR — Lancichinetti-Fortunato-Radicchi model (p. 99)

LM — Louvain method (p. 108)

LP — label propagation (p. 108)

MOD — modularity (p. 113)

NMI — normalized mutual information (p. 107)

SC — Schuetz-Caflish (p. 108)

SCC — strongly connected component (p. 24)

WCC — weakly connected component (p. 24)

Information theory

π — stationary node visit frequency (p. 53)

$H(X)$ — Shannon entropy of X (p. 50)

$H(X, Y)$ — joint entropy (p. 106)

$H(X \mid Y)$ — conditional entropy (p. 106)

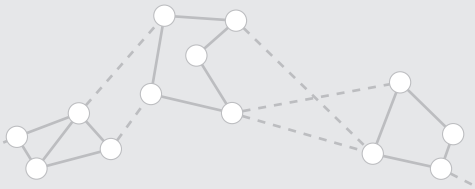
$I(x)$ — information of event x (p. 50)

$I(X, Y)$ — mutual information (p. 107)

$p(x)$ — probability to observe x (p. 50)

$p(x, y)$ — joint probability (p. 106)

$p(x \mid y)$ — conditional probability (p. 106)



1 Introduction

Back in 1735, Euler was facing a puzzle regarding the seven bridges of Königsberg. The problem was to find a path through the city that would cross every bridge exactly once. While this problem had ultimately no solution, the seminal paper of Euler (1741) laid the foundation of what we know today as graph theory or network science. Since then, the scientific efforts to comprehend this branch of discrete mathematics have led to a deep understanding of many complex systems. Indeed, graphs can be used to represent any type of relation between interacting agents. In computer science, graph theory has been applied to analyze the World Wide web [Albert, Jeong, and Barabási (1999); Barabási and Albert (1999)], to evaluate the resilience of peer-to-peer networks [Holme, Kim, Yoon et al. (2002)] or to rank web pages in search engines [Page, Brin, Motwani et al. (1999)]. Since the 1930s, graph theoretical tools have also proven to be very useful in sociology [Wasserman and Faust (1994); Scott (2000); Lazer, Pentland, Adamic et al. (2009)]. Researchers have studied human interactions using e-mail correspondence [Anteneodo, Malmgren, and Chialvo (2010)], mobility from mobile telecommunication data [Barabási (2005); González and Barabási (2007); Gonzalez, Hidalgo, and Barabasi (2008)] or the structure of scientific collaborations [Newman (2001)]. In biology, graphs have been widely used to model protein-protein interactions [Kim, Krapivsky, Kahng et al. (2002)], metabolic networks [Guimera and Amaral (2005)] or food webs [Guimerà, Stouffer, Sales-Pardo et al. (2010)]. The literature about graph theory applications is prolific and reveals the contemporary interests in analyzing the structural properties of networks.

This thesis aims at presenting recent progresses on network clustering, also known as graph partitioning. Those two problems are directly related although being conceptually slightly different in their objective. When considering network clustering, one is interested in identifying coherent groups of agents in the graph. For example, everybody knows a married couple who, one day, struggled with the organization of their wedding seating. In this case, one wants to identify groups of people who at least know each others, while also satisfying additional constraints, like

keeping family members together or potentially keeping divorcee apart from each others. This can in fact be translated into what is called the k-coloring problem over the graph of acquaintances, and while everybody knows how difficult this problem can become in real life, its graph theoretical counterpart has helped to prove that, indeed, it is extremely difficult. On the other hand, the graph partitioning problem consists in identifying a subset of the interactions between the agents of the graph such that the removal or the inhibition of those interactions will effectively partition the graph in a set of disconnected smaller components, while also ensuring additional constraints like minimizing the number of removed interactions. For instance, one may consider the problem of a water distribution network in which the flow at one end would be insufficient to satisfy the demand. Using graph theory, one can prove that identifying the minimal set of water pipes that if obstructed would prevent the water to reach the end of the network and would therefore bisect the graph in 2 disconnected components, allows to determine the bottlenecks of the water supply network and to compute the maximal flow one can send over the network (this is called the min cut/max flow problem). Somehow, network clustering is more focused on defining coherent groups of agents while graph partitioning is more focused on identifying weak interactions between agents. But in the end, both problems are essentially equivalent since removing interactions eventually defines disconnected groups of agents while defining coherent groups of agents implies that the interactions between the different groups could essentially be removed.

The first network clustering problem that we will analyze is known as community detection and consists in identifying groups of agents with many interactions between each other. This structural distribution of the interactions arises naturally in many systems, social networks being one perfect example. One will always be more confident in creating interactions with the friend of a friend rather than with a perfect stranger. Therefore, groups of friends will often have a high density of connections and only a few connections with the rest of the network, and that defines a community. The identification of those densely connected groups is a main concern in many applications that will be presented in Chapter 3. However, if one could potentially analyze without too much difficulty the networks of interest a few decades ago, at least for community detection, modern network science has to deal with millions of agents and billions of interactions between them. Therefore, network scientists have progressively become computer and data scientists and their main purpose is

now to define appropriate algorithms to extract relevant information contained in those large graphs. Many community detection algorithms have been defined over the years and we will present some of them in what follows. However, in computer science, it is now common to define parallel algorithms, meaning that instead of using a single machine, one can use multiple processors at the same time which allows to solve much larger problems than before. Our first contribution has been to define a fast, parallel and efficient algorithm to extract community structure out of very large graphs which will be the main topic of Chapter 3.

It is easy to show that an algorithm can be run in parallel because one just needs to show that it is designed mostly with independent operations. It is also easy to claim that an algorithm is fast by measuring the time it requires to extract a solution to a dedicated problem. However, it is more difficult to assess the accuracy of a community detection algorithm because the quality of a partition often depends on the particular application it is used for. Therefore, it is now common to compare the efficiency of different algorithms by applying them to computer generated benchmarks for which the exact solution can not be argued. This will be addressed in Chapter 4, where we will compare the efficiency of our algorithm with other popular methods, and we will also present some practical applications in which it was successfully applied.

Community detection is not the only way to handle network partitioning and indeed, some networks do not exhibit a community structure, while being still strongly structured. This situation may arise when the similarity between agents is somehow hidden in the graph. For instance, the internet topology consists in different routers that redirect the web queries, using specific protocols, to the correct servers. However, the internet topology is organized in different layers: the data layer which is the final destination of the query, a local layer which is controlled by the internet service providers (ISP), a regional layer which connects different ISP local networks and then a continental layer which in turn connects different regional layers, and so on. In this case, one does not expect to find many connections between similar kind of routers that are in the same layer. Instead, most of the links are indeed connecting different types of routers in a sort of top-to-bottom architecture. But one may be interested in identifying those similar types of routers which somehow serve the same purpose in the graph. In this context, we speak about role identification which will be the main concern of Chapter 5.

To extract a role structure, we will show that, in general, one needs

to define a similarity measure between each pair of agents in the graph. We will introduce some of the existing similarity measures and define a new robust similarity measure, based on the connectivity patterns of the agents, that presents interesting properties. Unfortunately, computing the exact similarity between every pair of agents can be computationally challenging in large graphs, so we will also propose a low-rank approximation of our similarity measure. In Chapter 6, we will compare our similarity measure with its low-rank approximation on benchmark graphs and show that the low-rank approximation achieves similar accuracy than the original measure. We will also apply our similarity measure to some real graphs and show that the extracted role structures are in general relevant to represent the underlying structure of the graphs.

Selected publications

Most of the work presented in the thesis has been submitted or published in journals and conference proceedings. We hereafter detail our main contributions.

Our first contribution was a fast and highly parallelizable algorithm for community detection, which allows to analyze massive networks. The content of this paper is the main focus of Chapter 3, and more precisely Section 3.4, and covers the definition of the algorithm, its convergence proof and a comparison of accuracy with other methods as presented in Chapter 4. The paper has been submitted to *Physical Review E* and is currently in revision before resubmission.

- Browet, Absil, and Van Dooren (2013) - Fast community detection using local neighborhood search - arXiv:1308.6276.

Our second contribution was to apply our community detection algorithm to the specific problems of image segmentation and video tracking. We showed that, even using a simple quality function, our method can extract coherent regions in an input picture and that it could be easily extended to track the evolution of the position of different objects simultaneously. This has been presented in 2 conferences and published in their proceedings

- Browet, Absil, and Van Dooren (2011) - Community Detection for Hierarchical Image Segmentation - 14^{th} International Workshop on Combinatorial Image.

- Browet (2011) - Community detection applied to video tracking - 10^{th} International Symposium on Iterative Methods in Scientific Computing.

The publication of those papers has attracted the attention of a research group working on mathematical characterization of materials, and opened a collaboration to apply our segmentation algorithm to images of microstructures in order to model the deformation of the material under stress constraints. This paper is currently in preparation for submission

- Dancette, Willemet, Browet, Martin and Delannay - Image-based meshing procedure to compute the mechanical response of materials - In preparation.

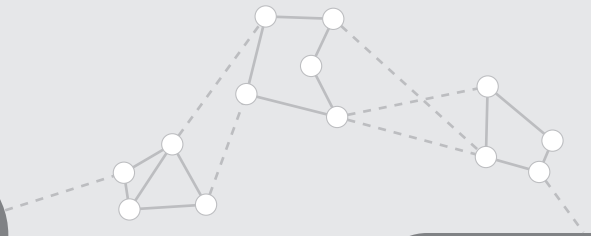
Our last important contribution is related to the role extraction problem discussed in Chapter 5. In this paper, we introduced our pairwise node similarity measure and its low-rank approximation, we demonstrated that both measures have similar accuracy on benchmark graphs and we analyzed the evolution of the low-rank similarity matrix for increasing value of the rank, which reveals the number of roles in the network. This paper has been accepted for publication in the peer-reviewed conference proceedings of the MTNS 2014

- Browet and Van Dooren (2013) - Low-rank Similarity Measure for Role Model Extraction - to appear in the proceedings of the 14^{th} International symposium on Mathematical Theory of Networks and Systems.

Finally, we also worked on a completely different topic that is the analysis of mobile phone telecommunication data which is not covered in the thesis. In a first paper, we analyzed different features of mobile phone users and proposed a model to identify their frequent positions. The paper has been published in *Physica A*. In a second paper, we developed a framework to detect unexpected events, like riots, based on mobile phone data. This paper has been published in the proceedings of the IEEE Third International Conference on Social Computing.

- Csáji, Browet, Traag et al. (2013) - Exploring the mobility of mobile phone users - Physica A: Statistical Mechanics and its Applications

- Traag, Browet, Calabrese et al. (2011) - Social Event Detection in Massive Mobile Phone Data Using Probabilistic Location Inference - SocialCom 2011



2 Elements of Graph Theory

Network theory and applications have attracted scientists from many different research areas. The available literature is abundant and goes far beyond the scope of this thesis. Without being exhaustive, this preliminary chapter introduces some fundamentals of graph theory that will be used throughout the thesis, as well as our notations which follow the formalism of [West (2001); Newman (2003)]. We refer the reader to [Newman (2010); Albert and Barabási (2002); Kolaczyk (2009); Diestel (2005)] for additional resources regarding the topic.

In Section 2.1, we will introduce different notations and give some basic definitions used to characterize graph properties. Then, in Section 2.2, we will present some specific structural properties of graphs. In Section 2.3, we will describe the notion of random graphs which allow to measure how much representative is a particular feature of a graph based on what is observed on average. Finally, in Section 2.4, we will formalize the problem of graph partitioning, which is the main interest of this thesis, and present some early developments in this field.

2.1 Fundamentals

A *graph* $G(V, E)$ is a mathematical representation of the pairwise interactions between $n \in \mathbb{R}$ individual agents and is defined by a set of *vertices* (or nodes) $V = \{1, \dots, n\}$ and a set of *edges* (or links) $E = \{(i, j) \mid i, j \in V\}$. A pair (i, j) belongs to E if there is an interaction between the agents i and j and the cardinality of the set E, i.e. the number of edges in the graph, is denoted by $|E| = m$. An edge (i, j) is called *incident* to both the nodes i and j which, in this case, are termed *neighbors*.

A graph can be univocally represented by its *adjacency matrix* $A \in \{0, 1\}^{n \times n}$ such that

$$A(i, j) = \begin{cases} 1 \text{ if } (i, j) \in E, \\ 0 \text{ otherwise.} \end{cases}$$

The matrix representation of a graph is particularly useful because it allows to analyze the graph using matrix theory and linear algebra. This

is known as algebraic graph theory or spectral graph theory. In this thesis, we will only consider graphs with *non-negative* adjacency matrices, i.e. $A(i,j) \geq 0 \ \forall i,j$. However, there exist graphs with negative links, called *signed graphs*, see [Zaslavsky (1982); Traag, Van Dooren, and De Leenheer (2013)].

A graph is called *undirected* if the edges have no orientation, i.e. each pair $(i,j) \in E$ is considered unordered and $(i,j) = (j,i)$, such as in friendship networks (where each relationship is considered reciprocal). The adjacency matrix associated to an undirected graph is *symmetric*, $A = A^T$. If the orientation of the edges matters, the graph is called *directed* and an edge (i,j) has a source i and a destination j. Telecommunication networks are a good example of directed network where one makes a distinction between the caller and the callee. In the remainder of the thesis, unless specifically stated otherwise, we will always consider graphs to be directed and assume that if the graph is undirected both (i,j) and (j,i) will belong to E. In directed graphs, a neighbor j of a node i is called a *child* when $(i,j) \in E$ and a *parent* when $(j,i) \in E$. Note that a neighbor can be both a child and a parent at the same time. An edge that connects a node to itself, $(i,i) \in E$, is called a *self-loop*.

The number of neighbors of a node i, or equivalently the number of its incident edges, is called the *degree* and is denoted by k_i. In directed networks, it is natural to make a distinction between the *in-degree* k_i^{in} and the *out-degree* k_i^{out} as the number of parents and the number of children, respectively. Using the adjacency matrix, the vectors of out and in-degrees can be computed as

$$k^{out} = A\,\mathbf{1} \quad , \quad k^{in} = A^T\,\mathbf{1},$$

where $\mathbf{1}$ is the vector of all 1's of appropriate dimension. A graph is called *regular* if all its nodes have the same degree, $k_i = k \ \forall i$, and therefore the number of edges is given by

$$m = \sum_i \sum_j A(i,j) = \sum_i k_i = n\,k.$$

For a directed graph, being regular implies that the in-degree and the out-degree are constant, which in turn forces the in-degree and the out-degree

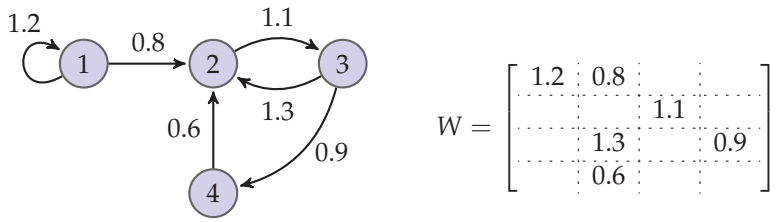


Fig. 2.1 A weighted directed graph and its weighted adjacency matrix.

to be equal since

$$n\,k^{out} = \sum_i k_i^{out} = m = \sum_i k_i^{in} = n\,k^{in},$$

so $k^{out} = k^{in} = k$.

In addition to its orientation, one can associate a weight to each edge representing the intensity of the interaction between the incident nodes, i.e. (i, j, w_{ij}), and the graph is then called *weighted*. In what follows, we will always consider graphs to be weighted. Indeed, if a graph is *unweighted*, one can consider that all its edges carry an equal unitary weight $(i, j, 1)$. Weighted graphs are univocally represented by their weighted adjacency matrix $W \in \mathbb{R}^{n\times n}$ such that $W(i, j) \neq 0$ if and only if $A(i, j) = 1$. Similarly to the degree, we define the strength s_i of node i as the sum of the weights of its incident edges, and, when the graph is directed, we distinguish between the *in-strength* and the *out-strength* as the sum of the weights of the incoming and outgoing edges, respectively. The in-strength and out-strength vectors can be computed as

$$s^{in} = W^T\,\mathbf{1} \quad , \quad s^{out} = W\,\mathbf{1}.$$

A small example of weighted directed graph is presented in Fig. 2.1 along with its weighted adjacency matrix. One can see that, for instance, the node labeled (2) has an in-degree $k_2^{in} = 3$, an out-degree $k_2^{out} = 1$, an in-strength $s_2^{in} = 2.7$ and out-strength $s_2^{out} = 1.1$. Let us mention that even if a graph is undirected, i.e. $A = A^T$, its weighted adjacency matrix is not necessarily symmetric since one can have that $w(i, j) \neq w(j, i)$.

The *density* of a graph is defined as the ratio between the actual number of edges in the graph and the maximal number of possible edges, i.e. $\frac{m}{n^2}$ for a directed graph with self-loops. A graph is termed *sparse* if its

density is low, which implies that its adjacency matrix is also sparse. The maximum number of edges being quadratic in the number of nodes in the graph, it is generally assumed that the number of edges in sparse graphs should grow linearly with the number of nodes, $m = \mathcal{O}(n)$, or in other words, that the average degree, defined by

$$\bar{k} = \frac{1}{n} \sum_{i=1}^{n} k_i,$$

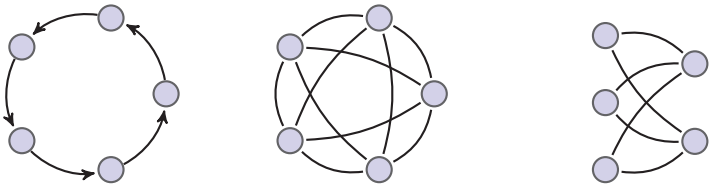
should be much smaller than the number of nodes $\bar{k} \ll n$.

In the following, we will only consider constant graphs, i.e. graphs that do not evolve through time neither in terms of their nodes, nor their edges. However, in some practical applications, the temporality of the connections is of main importance, for example when analyzing the spread of infectious diseases. We recommend [Nowak (2006); Clauset and Eagle (2007)] for readers interested in evolving graphs. Furthermore, we will only consider finite graphs, i.e. when the set of nodes V and the set of edges E are finite. The study of infinite graphs has shed light on interesting asymptotic behaviors of random graphs but is beyond the scope of this thesis. The reader will find interesting results and references in [Newman, Strogatz, and Watts (2001); Rath (2010)].

Let us conclude this section by defining some notions of distance on graphs. We define a *path* $p(i, j)$, also called a *walk*, between two nodes i and j as a sequence of edges

$$p(i, j) = \{(i, v_1); (v_1, v_2); \ldots; (v_{k-1}, v_k); (v_k, j)\},$$

such that the destination of each edge in the sequence (except for the last one) is always the source of the following edge. Note that there might exist multiple paths between a pair of nodes (i, j). A *simple* path is a path in which each edge is used at most once and a *cycle* is a simple path where the origin and the destination of the path are identical. We will sometimes refer to cycle graphs, denoted as C_n, which are graphs composed of exactly one cycle connecting n nodes as depicted in Fig. 2.2a. The *length* of a path is defined as the number of edges used in the path and the number of paths of length l between two nodes i and j can be computed as $A^l(i, j)$, where A^l is the l-th power of A. The distance $d(i, j)$ between two nodes i and j is defined as the length of the shortest



(a) Cycle graph C_5 **(b)** Complete graph K_5 **(c)** Complete bipartite $K_{3,2}$

Fig. 2.2 Example of structured graphs.

path between them,

$$d(i,j) = \min\left\{ l \mid A^l(i,j) \neq 0 \right\}.$$

Finally, the diameter of a graph is the largest distance between any pair of nodes in the graph. The diameter of a graph can be computed as

$$D(G) = \min\left\{ l \;\middle|\; \sum_{q=1}^{l} A^q(i,j) \neq 0,\ \forall i,j \in V \right\},$$

and is computationally expensive to measure in practice. It is interesting to note that the diameter of many real networks is very small compared to the size of the graphs [Watts and Strogatz (1998)]. This is known as the small world phenomenon. For example, the diameter of the World Wide Web has been estimated to be between 15 and 20 [Albert, Jeong, and Barabási (1999); Kang, Tsourakakis, Appel et al. (2008)], even though the number of web pages indexed by Google in 2010 was approximately 5×10^{10}*. This means that a target web page can be reached from any starting page using at most 20 clicks and only following existing hyperlinks.

2.2 Graph structures

In this section, we will define some structural properties of graphs that will be extensively used in the following chapters.

A graph is said *complete* if the set of edges E contains all the possible edges (potentially excluding self-loops), i.e. if its density is 1. The complete directed graph over n nodes is denoted K_n and contains $n(n-1)$ edges (or n^2 if one considers self-loops). The complete graph over 5 nodes,

*http://www.worldwidewebsize.com/

K_5, is presented in Fig. 2.2b where each arc represents a bidirectional edge (arrow heads are not displayed for readability).

Bipartite graphs form another class of structured graphs where the nodes are distributed among 2 independent sets. An independent set is a set of nodes such that any edge in the graph is incident to at most one node in the set, i.e. none of the nodes in an independent set are neighbors. Therefore, in bipartite networks, every edge is incident to one node within each of the independent sets. Hence, a graph is bipartite if and only if it does not contain odd cycles. This kind of graphs is often used to represent the assignments of different elements to specific tasks. For example, a bipartite graph is well suited to model the attendance of students to lectures in a university. One can test if a graph is bipartite in linear time using breadth-first-search or depth-first-search algorithms. Those algorithms are simply designed to iterate over all the edges of a graph and to sequentially assign each node to one of the two sets until either all the edges have been considered and the bipartite nature of the graph has been discovered, or until an edge is found to connect two nodes within the same set and the graph is not bipartite. A *complete bipartite* graph with two node sets of size n_1 and n_2 is denoted K_{n_1,n_2} and has $2n_1n_2$ edges. The complete bipartite graph $K_{3,2}$ is illustrated in Fig. 2.2c.

In order to apply classical analysis methods, like partitioning that will be introduced in Section 2.4, bipartite graphs are often *projected* onto one of their two independent sets. In the projected graphs, two nodes are connected if they share a common neighbor in the original bipartite graph. The projections on each of the two node sets of a bipartite graph are presented in Fig. 2.3. Note that the projection on each node set is unique (up to the definition of the weight of the edges in the projected graph), however it is in general not possible to infer the bipartite graph from one of its projections as depicted in the figure for the projected graph on the ○ set. Hence, the projections of a bipartite graph may hide some important features of the network, which will be illustrated in Section 6.1.2. Therefore, it is of main interest to avoid the projection of bipartite graphs, in particular when one wants to extract an appropriate partition for such a graph, which is one of the motivations of our results presented in chapter 5.

Similarly to bipartite networks, we define a N-partite network as a graph where nodes are organized among N independent sets. The scheduling problem is a typical application of N-partite graph. For example, one may want to find an optimal schedule for lectures that would minimize

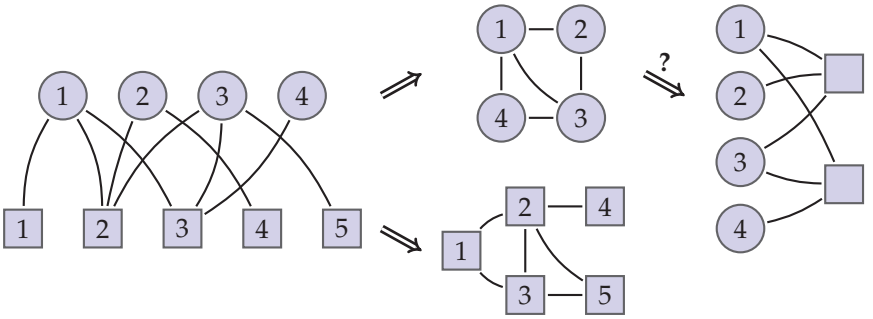


Fig. 2.3 Projections of a bipartite graph and backward inference.

the number of conflicts for the attending students; the graph could be defined as the set of students connected to the lectures they want to attend, themselves connected to the available time slots, with potentially other sets of nodes like the teachers able to give the lectures or the vacant auditoriums. Every graph is trivially n-partite, however, finding the smallest N such that a graph is N-partite is a NP-hard problem (although one can check in polynomial time if $N = 1$ (empty graph) or $N = 2$ (bipartite graph) are valid solutions) known as the minimal vertex coloring problem. In this context, N is termed the *chromatic number* of the graph.

A *subgraph* $H(V_H, E_H)$ of a graph $G(V, E)$ is a graph whose vertices are a subset of the vertices of G, $V_H \subset V$, and whose edges are a subset of the edges of G incident only to nodes in V_H,

$$E_H \subset \{(i,j) \mid i,j \in V_H, (i,j) \in E\}.$$

A subgraph will be called *induced* if it contains all the edges incident to two nodes in V_H, $E_H = \{(i,j) \mid i,j \in V_H, (i,j) \in E\}$. Lastly, a *spanning* graph is a subgraph that has the same node set as the original graph, $V = V_H$. Hence, there is only one spanning induced subgraph which is the graph itself.

An undirected graph is said to be *connected* if there exists a path between any pair of nodes. If a graph is not connected, then one can divide it in multiple induced subgraphs $H_i(V_{H_i}, E_{H_i})$ such that every subgraph is connected and all the edges of the original graph are contained in one of the subgraphs, i.e. there is no edge between any of the induced subgraphs,

$$V = \cup_i V_{H_i} \quad , \quad E = \cup_i E_{H_i}.$$

Each of those induced subgraphs H_i is called a *connected component* of the original graph. If the graph is directed, then we differentiate between two types of connected components. First, we define a strongly connected component as a maximal set of nodes such that there exists a directed path between every pair of vertices, *maximal* meaning that one can not add any node in the set while keeping the property valid. Strongly connected components are simply connected components in an undirected graph. We also define a weakly connected component as a maximal set of nodes such that there exists a path between every pair of vertices in the associated undirected induced graph. That is to say that there is a path between every pair of vertices if we do not take into account the direction of the edges. For example, in Fig. 2.1, the strongly connected component is defined by $SCC = \{②, ③, ④\}$ and the weakly connected component is $WCC = SCC \cup \{①\}$ since there is path from ① to the nodes in the SCC but not the other way around.

A *clique* is a complete induced subgraph. In Fig. 2.1, the set $\{②, ③\}$ defines a directed clique of size 2. A clique is called *maximal* if it can not be extended by adding additional nodes and *maximum* if there exists no larger clique in the graph. Hence, the maximum clique defines the largest set of fully connected nodes and may sometimes reveal important properties of the network. For example, this structure may induce some hierarchy in the organization of the vertices by defining a core and periphery around it which helps to explain some spreading processes over the network. This will be more detailed in the following chapter. However, finding the maximum clique in a given network is NP-complete and even an approximation may be hard to uncover.

Finally, let us mention that the complement of a graph $G(V, E)$ is a graph $G^{\perp}(V, E^{\perp})$ defined over the same set of nodes but such that $E^{\perp}$ contains all the edges not existing in G, i.e. $E^{\perp} = \{(i, j) \mid (i, j) \notin E\}$. For example, the complement of a clique is an independent set and vice versa.

2.3 Random graphs

Throughout the following chapters, we will extensively use random graphs to assess the quality of algorithms and measures. A graph is called random if either its node set, its edge set or both are generated using a random process. Random graphs are particularly useful to model typical properties of networks or to quantify how similar or dissimilar is a given

network from what one would observe on average in networks having similar properties. We will present some models to build random graphs in what follows but we refer the reader to [Janson, Luczak, and Kolchin (2000); Bollobás (2001)] for a thorough analysis on random graphs.

The most common random graph model is due to Erdős and Rényi (1960). To build such a random network, one starts with a fixed set of vertices and no edges. Then, every of the $n(n-1)$ possible edges (excluding self-loops) is added with a constant probability $p \in [0,1]$. One can prove that an Erdős-Rényi graph is asymptotically almost surely connected when $p > O(\frac{\log n}{n})$.

Since the probability to have an edge between any pair of nodes is p, the expected number of edges in an undirected Erdős-Rényi graph is

$$\langle m \rangle = \binom{n}{2} p$$

where $\binom{n}{2}$ denotes the binomial coefficient, i.e. the number of possible ways to select 2 nodes out of a set of n nodes, or equivalently the number of pairs of nodes in the network, $\binom{n}{2} = \frac{n(n-1)}{2}$. The expected number of edges is simply multiplied by 2 for directed Erdős-Rényi graphs.

The degree distribution of a network is the probability distribution of the degree of the nodes in the network. More precisely, the degree distribution $P(k)$ is the probability that a node taken at random has a degree k and is given by the expected proportion of nodes of degree k in a random network

$$P(k) = \frac{\langle |\{i \mid k_i = k\}| \rangle}{n}.$$

In Erdős-Rényi graphs, the degree distribution is binomial

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k},$$

which can be well approximated by a Poisson distribution for large n and small p,

$$P(k) \approx \frac{(np)^k e^{-np}}{k!}.$$

The Erdős-Rényi model has been widely used to create random graphs because the construction of the network is fairly easy and the model can be easily generalized to include additional properties. For example, we will

present in Section 6.1.1 a model inspired from the Erdős-Rényi model but with 2 distinct probability parameters p_1 and p_2 which allow to impose a structural distribution for the nodes. Erdős-Rényi graphs are known to be small-world networks [Watts and Strogatz (1998)], i.e. the diameter of the graph is small compared to the number of nodes and $D(G) \approx O(\log n)$, which has been often observed for real networks. However, many real networks do not have uniformly distributed edges and nodes tend to form clusters. This can be measured by the *clustering coefficient* which computes for each node the ratio between the actual number of triangles and the maximum possible number of triangles in the graph

$$C_i = \frac{|\{(j,k) \mid (i,j) \in E\,,\, (i,k) \in E\}|}{k_i(k_i-1)}.$$

In many real networks, the average clustering coefficient is high and unfortunately, this is not the case for Erdős-Rényi graphs. Therefore, Watts & Strogatz have proposed another model to better fit this property of real networks.

The random model of Watts & Strogatz starts from a fixed number of nodes n and an average degree k supposed to be even. Then, the network is built on a regular cycle such that every node is labeled from 1 to n and a node i is connected to its k closer neighbors in terms of labels, i.e. $k/2$ on the left of the cycle and $k/2$ on the right (node 1 is connected to node n to close the cycle). Then, every edge is selected exactly once and is rewired with a probability $\alpha \in [0,1]$ to any other node in the network, avoiding self-loops and multiple edges.

The regular cycle induces a high average clustering coefficient while the rewiring probability reduces the diameter of the graph. One can easily prove that when α tends to 1, a random graph created using this procedure tends to an Erdős-Rényi graph with $p = \frac{nk}{\binom{n}{2}}$, however the graphs are very different for small value of α.

The expected degree distribution of random graphs created using the model of Watts & Strogatz is much more complicated to derive, however it has been shown to be relatively similar to the degree distribution of Erdős-Rényi graphs, i.e. the distribution peaks at the average degree k and decays exponentially for smaller or larger degrees. This model is more closely representing some real networks with a high clustering coefficient, however many real networks, while being small-world, have been observed to be also scale-free which means that their degree distribution

follows a power-law [Clauset, Shalizi, and Newman (2009)], $P(k) \sim k^{-\gamma}$, with γ often observed between 2 and 3. This means that the degree distribution is highly heterogeneous in the sense that very high and very low degree nodes coexist in the network.

For this reason, another model of random graphs has been introduced by Barabási and Albert (1999) based on the *preferential attachment* hypothesis. This mechanism of preferential attachment mimics what has been observed in social networks, in the internet topology or in citation networks. All those networks tend to grow with time and as a new node joins the network, there is a high probability that it creates links with other highly connected nodes already present in the network. For example, a scientist writing a paper will often cite other well known papers in the topic which are already often cited, or a newcomer in a social network will tend to follow famous people, rather than unknowns, which are already followed by many.

A random network created using the Barabasi-Albert model starts with a small set of n_0 connected nodes with uniform degree $k \geq n_0$. Then, new nodes are added to the network and connected to some of the already existing nodes. More precisely, a new node will create a link with a node i with probability

$$p(i) = \frac{k_i}{\sum_j k_j}.$$

Therefore, a node with a high degree has a higher chance to keep increasing its degree for each new node joining the network. This will naturally create a network with very high and very low degree nodes and indeed the expected degree distribution follows a power-law with $\gamma = 3$. The expected diameter of such a network is small and tends to grow as $D(G) \sim O\big(\log n / \log\log n\big)$.

The Barabasi-Albert model successfully reproduces some typical properties of real networks. However, it has often been observed that while nodes tend to be locally and densely clustered, as represented by a high clustering coefficient, they also tend to be sparsely connected with the rest of the network. This means that often, one will observe many dense clusters of nodes, only sparsely connected with one another. This structural distribution of nodes in a network is called a community structure and will be the main focus of Chapter 3 and 4. Unfortunately, the random models described in this section have been shown to be free of any structural distribution of their nodes, however they introduced the ba-

sic requirements to build a more advanced model of small-world and scale-free networks containing community structure that will be detailed in Section 4.1.

2.4 Graph partitioning

Graph partitioning, or equivalently clustering, is the main topic of the thesis and, while it can be defined in different ways, as we will show in the following chapters, the basic principle is always the same and consists in identifying coherent groups of nodes that are on average more similar one with each other than with the rest of the network. Therefore, graph partitioning is always tied to a fitness measure or a cost function to optimize. Let us here introduce the fundamentals of graph partitioning while we will refine them later.

We already presented some applications of graph partitioning in the introduction, with the wedding seating problem or the water distribution network and the identification of its bottlenecks, but the scope of applications is extremely wide and goes from load balancing problems to image segmentation or from the mobile phone physical networks design to parallel jobs scheduling. Therefore, a common and general framework to analyze those problems should be defined.

The elementary formulation of the partitioning problem is to define a cut, i.e. a subset of the edges such that removing those edges creates 2 disconnected components in the graph. However, as we mentioned, one is interested in optimizing an objective criterion to define an appropriate cut. The objective function allows to discriminate the quality of different cuts. The first type of cut is the minimum cut and consists in identifying the minimal set of edges that bisects the graph. The minimum cut problem is equivalent to the maximum flow problem, i.e. by identifying the minimum cut, one can compute the maximum possible flow that can be sent through the network. Unfortunately, the minimum cut criterion tends to produce an unbalanced partition, i.e. for unweighted network, the optimal solution consists in identifying the node of minimum degree and to select its incident edges as the cut set. Other types of cuts have been defined like the maximal cut, which consists in finding the maximal set of edges that bisects the graph, or the sparsest cut which consist in identifying the cut that minimizes the ratio between the number of edges in the cut and the number of nodes in the smallest of the disconnected components.

However, unlike the minimum cut which can be solved in polynomial time, many cut problems like the maximum cut and the sparsest cut are NP-hard. In practice, one needs therefore to somehow find an approximation of the optimal cut. One possible way to define such an approximation is to use spectral partitioning which consists in defining a cut based on the eigenvectors and eigenvalues of an appropriate matrix. For example, let us define the Laplacian matrix as $L = D - A$ where D is the diagonal matrix of the degrees, $D = diag(k_i)$ and let us consider the ratio cut which is the ratio between the number of edges in the cut and the product of the size of the partitioned components. Note that $\lambda = 0$ is an eigenvalue of the Laplacian matrix, associated to the eigenvector $v = \mathbf{1}$, and since L is positive semidefinite, it is the minimal eigenvalue. In fact, one can prove that the algebraic multiplicity of the eigenvalue $\lambda = 0$ gives the number of connected components in the graph [Fiedler (1973)]. Moreover, the second smallest eigenvalue of the Laplacian gives a lower bound on the minimal value of the ratio cut [Hagen and Kahng (1992)] and the associated eigenvector v_2 can be used to partition the graph. Let us define a partitioning vector $x \in \{-1, 1\}^n$ such that nodes i and j are in the same partition if $x(i) = x(j)$. An approximation of the optimal ratio cut is given by the partitioning vector defined as

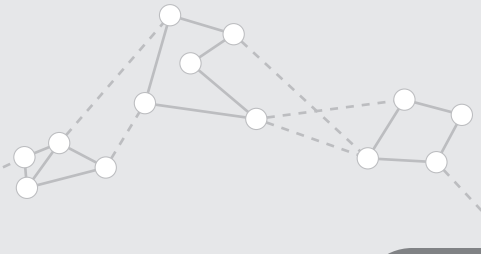
$$x(i) = \begin{cases} 1 & \text{if } v_2(i) > \kappa, \\ -1 & \text{otherwise.} \end{cases}$$

with $\kappa \in [\min(v_2), \max(v_2)]$. In practice, one needs to scan for different values of κ and select the one that produces the best value of the ratio cut. Different matrices should be used for other types of cuts although the principle is exactly the same.

Another approach to define graph partitioning is to somehow measure how central are each node or each edge in the network. This follows the idea of the works of [Granovetter (1983)] on "the strength of weak ties". It has been observed that most of the edges with large weight (the strong ties) are often lying in the middle of densely connected clusters of nodes and that it is in fact the edges with small weight (the weak ties) that maintain the global connectivity of the network and allow the exchange of information between clusters. Following this principle, different measures of centrality have been defined [Newman and Girvan (2004); Klein (2010)] like the edge betweenness centrality which is, for each edge, the number of pairs of nodes (i, j) such that at least one of the shortest path between them

contains the edge. One would expect that the weak ties, which are the edges that should define the cut, have a high betweenness centrality since they will often be involved in the shortest paths between nodes of different clusters. Therefore one can define a partition by sequentially removing the edge with the higher betweenness centrality until an appropriate number of clusters has been found.

Unfortunately, those formulations have a major drawback which is that one should estimate beforehand the number of clusters to compute. For spectral partitioning, one needs to recursively bisect the network and it is not clear how many of such bisections should be performed. For centrality measure, one would need to choose the number of edges to remove, and again, it is not clear how to decide this number. Therefore, other algorithms and measures which can somehow derive an appropriate number of clusters have been developed and we will present them in the following chapters.



3 Community detection

Over the years, the evolution of information and communication technology in science and industry has had a significant impact on how collected data are being used to understand and analyze complex phenomena [Halevy, Norvig, and Pereira (2009)]. With the increase in storage capacity and computational power, the amount of collected data has grown tremendously. Clever analysis of those so-called Big Data are at the core of many successful projects and suitable algorithms must be developed to deal with such data [Torralba, Fergus, and Freeman (2008); Skillicorn and Talia (2012)]. Indeed, contemporary networks contain millions or even billions of nodes and are too large to comprehend without appropriate tools. Even a simple visualization of the network is often virtually impossible.

As already mentioned, to establish some behavioral properties of the objects of interest represented in a graph, a popular technique is to cluster together highly similar nodes [Simon and Ando (1961)]. When the pairwise node similarity is encoded in the weight of the edges, this task is known in graph theory as community detection which has become widely popular after a publication by Girvan and Newman (2002). In this chapter, we will first introduce in Section 3.1 the problem of community detection and present different applications where the extraction of communities had proven to be of main interest. We will then present in Section 3.2 a general framework to quantify how representative is a community partition in a given network and derive different quality functions to infer the communities. Then, in Section 3.3, we will introduce some of the many existing algorithms and heuristics used for the extraction of community structures. Those first sections serve as an outline of the literature but we recommend [Porter, Onnela, and Mucha (2009); Fortunato (2010)] for in-depth reviews of community detection in graph. Finally, in Section 3.4, we will describe a new algorithm for community detection specifically designed to analyze very large networks.

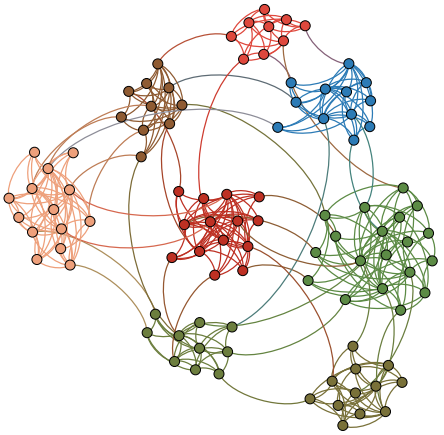


Fig. 3.1 Communities in a graph.

3.1 Communities in networks

Real graphs are undoubtedly not homogeneous and largely differ from uniform random graphs like in the Erdős-Rényi model for example. The degree distribution in real networks often follows a power law, with a fat-tail, rather than a binomial or a Poissonian distribution. This implies that nodes with very high and very low degree coexist in real networks. Vertices do not only differ globally in their degree distribution but also locally in their edges distribution. Indeed, it has often been observed that the local concentration of edges in specific group of nodes is much higher than the global density of the network. These groups of nodes found in real networks are known as community structures [Girvan and Newman (2002)] also termed nodes clusters. An example of community structure is illustrated in Fig. 3.1 on a computer generated graph using the LFR benchmark model that will be detailed in Section 4.1. One can see that each community, represented by a specific color, forms a cohesive group of highly interconnected nodes. Communities play a crucial role in understanding a network: the nodes within these dense clusters are expected to share many common properties or to fill a similar purpose in the graph. Furthermore, extracting communities can also reveal some behavioral or structural properties of the nodes. In social and metabolic networks, a core/periphery structure is often encountered within the different communities [Granovetter (1973); Burt (1976); Jeong, Tombor, Al-

bert et al. (2000); Guimera and Amaral (2005)]. The core nodes provide the stability of the clusters and are the most internally connected vertices. If such nodes are removed or inhibited, the community ceases to exist as a cohesive group. On the other hand, the peripheral nodes act as mediators between the different communities and allow the exchange of information. This fundamental distinction between the roles of the nodes in a community has been particularly studied in epidemiology. It has been shown that the community structures may largely decrease the spread of infectious diseases because only the peripheral nodes can disseminate the infection between clusters [Balcan, Colizza, Gonçalves et al. (2009)]. Furthermore, it is crucial to identify the core nodes as the target individuals for vaccination policies [Shaw and Tunc (2012); Tizzoni, Bajardi, Decuyper et al. (2013)].

Communities arise naturally in most, if not all, kinds of social networks where people form spontaneously groups sharing common characteristics depending on different types of interactions like family, fiends, work, sports and so on. The extraction of relevant clusters in large social networks is a real challenge but provides accurate insights about the influence of individuals and spread of information [Wu and Liu (2008); Liu and Hu (2005)]. It has been shown that the latter has similar dynamics than the spread of infectious diseases and highly interests advertisement companies: the term *viral marketing* has not been chosen at random [Leskovec, Adamic, and Huberman (2006)]. Communities can also reveal unexpected behaviors. For instance, Belgium is known to be mainly separated in 2 distinct regions using different languages, either french or dutch. However, the extraction of communities in a mobile telecommunication network has highlighted the essential role of the capital, Brussels, for the communication between the 2 large communities [Blondel, Krings, and Thomas (2010)]. Surprising results have also been obtained in the analysis of mobile telecommunication networks of France and Great-Britain [Blondel, Deville, Morlot et al. (2011); Ratti, Sobolevsky, Calabrese et al. (2010)] where the geographical borders of the departments can be recovered as the boundary of communities, even though the creation of those geographical borders goes back to several decades and should not influence how people interact nowadays. Other types of social interactions have been studied using community detection. Institutional and geographical constraints have proven to control the patterns of scientific collaborations [Evans, Lambiotte, and Panzarasa (2011)], criminal networks have been studied to reveal the structure of large organizations of offenders [Krings,

Dabin, and Blondel (2011)] and a network of bottlenose dolphins has been investigated to uncover their social strata [Arenas, Fernández, and Gómez (2008)].

Furthermore, the potential applications of community detection go far beyond the analysis of social behaviors. Scientists have given evidence, using a goods exchange network, that trading communities reduce the probability of international conflicts [Lupu and Traag (2012)]. The world-wide air transportation network has been investigated to explain some anomalies in the airports connectivity, i.e. the most connected cities are not the most central ones, which reveals that geopolitical considerations have to be taken into account to understand the airports connectivity [Guimerà, Mossa, Turtschi et al. (2005)]. Community detection has also yielded functional cartographic representations of the different roles of nodes in metabolic networks, known to be extremely difficult to study [Guimera and Amaral (2005)]. It has been used to better understand the dispersal of coral larvae in the Great Barrier reef which is essential to prescribe better management and control policies of the coral reef water [Thomas, Lambrechts, Wolanski et al. (2014)]. Community extraction has also been applied in image processing to reveal the contour of objects and to track moving bodies in video [Hu, Ronhovde, and Nussinov (2012)] which will be illustrated in Section 4.2.

In the following sections, we will explain how communities can be algorithmically extracted from a given network and we will present in Section 3.4 a new fast algorithm to unfold *hierarchical* community structures. In many complex systems, the hierarchical nature of the communities is evident, one of the best examples being the human body where atoms form molecules, multiple molecules compose the cells, which in turn form tissues that constitute the different organs, etc. We believe that it is essential to provide potentially different hierarchical levels of clustering when identifying community structures. This allows to analyze the network at various resolution levels depending on the application. In some circumstances, it might also be relevant to consider *overlapping* communities which entails that each node may have multiple community assignments. For instance, in a word association graph constructed based on a dictionary, a word like "bright" may belong to communities of words associated to either *Intelligence*, *Astronomy*, *Light* or *Colors*. Dedicated algorithms have been developed to disclose overlapping communities like the clique percolation method [Palla, Derényi, Farkas et al. (2005)], the Order Statistics Local Optimization Method (OSLOM) [Lancichinetti, Radicchi, Ramasco

et al. (2011)] or the link communities (rather than node communities) introduced by [Ahn, Bagrow, and Lehmann (2010)]. However, in this work, we will not consider the problem of overlapping communities, which often require an ad hoc definition, and we recommend [Farkas, Ábel, Palla et al. (2007); Kumpula, Kivelä, Kaski et al. (2008)] for additional sources.

3.2 Quality functions

There is no universally accepted definition of communities. As mentioned previously, communities are informally defined as sets of nodes with a high internal density, either in the number of internal edges or their weight, and a low external density with the rest of the network. This implies that the subgraph induced by a community (see Section 2.2, page 23) should be at least weakly connected, otherwise the total internal density could be increased by partitioning the community into its weakly connected components without increasing the external density. However a more explicit definition must be stated in order to compute what would be the optimal (or at least a good) community partition in a given network.

The most straightforward description of communities is to assume that they are spanned by the densest possible clusters, i.e. maximal complete subgraphs or cliques [Palla, Derényi, Farkas et al. (2005)]. Nevertheless, this characterization is quite restrictive and does not fit for real networks. It is not clear that imposing the maximum clique (which is already NP-hard to compute) as a community is appropriate since this can mask smaller cliques for non-overlapping communities. Moreover, the notion of clique is ambiguous in weighted networks and the core/periphery structure often encountered within communities in real networks can not be represented by cliques.

A large internal density is not the only defining property of a community which also requires a low external density to be accurately characterized. This leads to the definitions of *strong* and *weak* communities [Radicchi, Castellano, Cecconi et al. (2004); Hu, Chen, Zhang et al. (2008)]. A community is said strong if the internal degree of all its vertices is higher than its external degree. This definition is also quite restrictive and barely found in real networks. Hence, communities have also been defined in a weak sense as clusters with a total internal degree (the sum of the internal degree of its vertices) larger than its total external degree. This is the fundamental hypothesis behind the planted partition model [Condon and Karp (2001)].

While those characterizations of communities are more appropriate for real networks, they lack the formalism to create an algorithm to actually extract community structure from a given network. Therefore, it is often more convenient to establish a global definition of communities by using fitness measures or quality functions over the entire graph. As we will see in the following sections, this can be done by defining a *null model*, i.e. a graph which corresponds to the original graph at study for a specific set of features, e.g. the degree distribution or the expected number of triangles, but is essentially random in nature. Since a random graph is assumed to exclude community structure, the null model serves as a comparison to assess that the graph contains communities. Hereafter, we present a general formulation proposed by [Reichardt and Bornholdt (2006)] which defines the energy of a partition based on a chosen null model and provides a quantitative criterion to optimize the community assignment of each node. An particularly popular cost function that we describe in Section 3.2.2, namely the modularity, has been introduced by [Newman and Girvan (2004)] and can be expressed within this general framework. This cost function was one of the first to have the considerable advantage to not require to know beforehand the number of clusters to extract. Let us mention that, although here we first present the formalism of Reichardt & Bornholdt, the modularity of Newman & Girvan was historically developed before but can be better interpreted within this general framework.

3.2.1 Reichardt & Bornholdt: energy of a partition

Reichardt and Bornholdt have introduced a model inspired from statistical mechanics in which they interpret the problem of community detection as finding the ground state of a spin glass i.e. a disordered magnet [Reichardt and Bornholdt (2006)]. Each node i in the graph is labeled by a spin variable $\sigma_i \in \{1, \dots, c\}$. This spin state represents the community assignment of the node. In the optimal spin configuration, edges should ideally connect nodes in the same spin state, i.e. in the same community, while vertices in different spin state should be as sparsely connected as possible. This leads to the definition of an energy function that favors edges between nodes in the same spin state and penalize other existing edges. More precisely, for any existing edges, $A(i,j) \neq 0$, the partition is

- rewarded by $a_{ij} > 0$ if the nodes are in the same spin state $\sigma_i = \sigma_j$;
- penalized by $c_{ij} > 0$ if the nodes are in different spin states $\sigma_i \neq \sigma_j$.

Conversely, the missing edges in the graph must also be considered to evaluate the quality of a partition, otherwise the optimal community structure would be to assign an identical spin variable to all the nodes. That is, for any non present edges, $A(i,j) = 0$, the partition should be

- penalized by $b_{ij} > 0$ if the nodes are in the same spin state $\sigma_i = \sigma_j$;
- rewarded by $d_{ij} > 0$ if the nodes are in different spin states $\sigma_i \neq \sigma_j$.

If we denote by $\delta(\sigma_i, \sigma_j)$ the Kronecker delta, i.e.

$$\delta(\sigma_i, \sigma_j) = \begin{cases} 1 & \text{if } \sigma_i = \sigma_j \\ 0 & \text{if } \sigma_i \neq \sigma_j \end{cases},$$

we can write the energy of a specific partition σ as

$$\begin{aligned} H(\sigma) = -\sum_{i,j \in V} & \overbrace{\Big[a_{ij}A(i,j) - b_{ij}\left(1 - A(i,j)\right)\Big]}^{\text{internal edges}} \delta\left(\sigma_i, \sigma_j\right) \\ & - \underbrace{\Big[c_{ij}A(i,j) - d_{ij}\left(1 - A(i,j)\right)\Big]}_{\text{external edges}} \left(1 - \delta\left(\sigma_i, \sigma_j\right)\right) \end{aligned} \tag{3.1}$$

where the minus sign before the summation symbol is a convention such that the optimal partition is defined by the spin configuration of minimum energy, i.e. the ground state of the infinite range spin glass. One can rearrange the terms in Eq. (3.1) such that

$$H(\sigma) = -H_0 - \sum_{i,j \in V} \Big[\alpha_{ij}A(i,j) - \beta_{ij}\Big] \delta\left(\sigma_i, \sigma_j\right) \tag{3.2}$$

where $H_0 = \sum_{i,j \in V} \Big[-\left(c_{ij} + d_{ij}\right) A(i,j) + d_{ij}\Big]$ is independent of the partition and therefore often discarded from the fitness measure. The values of the parameters α_{ij} and β_{ij}, defined by

$$\alpha_{ij} = a_{ij} + b_{ij} + c_{ij} + d_{ij} \quad , \quad \beta_{ij} = b_{ij} + d_{ij},$$

depend on the null model one would like to compare the graph with, i.e. a graph which matches the network under study for a specific set of chosen features but otherwise random in nature and therefore without community structure. For instance, one can assume that a sufficiently dense

community should contain a lot of triangles [Arenas, Fernández, Fortunato et al. (2008)]. This hypothesis is often expected within "balanced" social networks [Traag, Van Dooren, and De Leenheer (2013)]: the friends of my friends are often my friends too. In this framework, the parameters α_{ij} and β_{ij} might be computed as

$$\alpha_{ij} = \sum_k A(j,k)\, A(k,i)\, \delta(\sigma_i, \sigma_k) \quad , \quad \beta_{ij} = \sum_{i,j,k} N_{ij}\, N_{jk}\, N_{ki}\, \delta(\sigma_i, \sigma_k)$$

where N_{ij} is the probability that there exists a directed edge between the nodes i and j in the null model. This triangle pattern model is well suited to analyze social networks, however it illustrates a potential issue for the choice of α_{ij} and β_{ij}. Finding the optimal partition for a cost function like Eq. (3.2) is in general a NP-hard problem [Brandes, Delling, Gaertler et al. (2006)], hence one will be interested in finding a good approximation of the optimal partition using greedy algorithms. This means that it is often desirable to keep the parameters α and β simple (in term of computation complexity) and independent of the partition which is not the case here.

Another natural choice is to give rewards and penalties for the presence of the edges proportional to the weight of the edges. That is, the parameter α is chosen such that $\alpha_{ij} = w(i,j)$ which imposes to use a weighted null model solely defined by the parameter β,

$$H_W(\sigma) = -\sum_{i,j \in V} \Big[W(i,j) - \beta_{ij} \Big] \delta\left(\sigma_i, \sigma_j\right). \tag{3.3}$$

One can observe that the energy of a partition, as defined by Eq. (3.2) or Eq. (3.3), only depends on the internal edges in the different communities. Hence, to get more insights about the null model and the optimal partition, we can rewrite those cost functions such that the summation index runs over the community labels rather than the nodes. That is to say,

$$H_W(\sigma) = -\sum_{r=1}^{c} l_r - \langle l_r \rangle = -\sum_{r=1}^{c} h_w(\sigma_r), \tag{3.4}$$

where l_r is to the total weight of the edges inside community r and $\langle l_r \rangle$ is the corresponding expected value in the null model,

$$l_r = \sum_{i,j \in V} W(i,j)\delta(\sigma_i, r)\delta(\sigma_j, r), \tag{3.5}$$

$$\langle l_r \rangle = \sum_{i,j \in V} \beta_{ij}\delta(\sigma_i, r)\delta(\sigma_j, r). \tag{3.6}$$

This shows that this kind of fitness measures for the partition σ is *additive*, meaning that the value of the cost function can be computed as the sum of the individual fitness values of each community. As we will see later, this property is very useful when one wants to find the optimal partition using a greedy algorithm.

The formulation of Eq. (3.3) for the energy of a partition brings us back to the original characterization of communities, defined as highly connected clusters of nodes (large values of l_r), but with the additional assumption that the actual internal density of the clusters must be larger than the one expected in a random network ($\langle l_r \rangle < l_r$ for most r) which is still to be defined. Reichardt & Borhnoldt proposed to define the parameter β as

$$\beta_{ij} = \gamma_{RB} p_{ij} \tag{3.7}$$

where $\gamma_{RB} \geq 0$ is a scaling parameter and p_{ij} is the expected number of edges or the expected weight of an edge between nodes i and j, which leads to

$$H_{RB}(\sigma) = -\sum_{i,j \in V} \Big[A(i,j) - \gamma_{RB} p_{ij} \Big] \delta\left(\sigma_i, \sigma_j\right) \tag{3.8}$$

$$= -\sum_{r=1}^{c} l_r - \gamma_{RB} \left\langle l_r \right\rangle_p \tag{3.9}$$

The scaling parameter γ_{RB} allows to analyze the network at different resolution levels, independently of the chosen null model. Indeed, if $\gamma_{RB} = 0$, the optimal partition consists of a single community with all the nodes, which is the most coarse-grained representation of a network. On the other hand, if $\gamma_{RB} \to \infty$, then the optimal partition consists of n clusters, each composed of a single vertex, which is the most fine-grained representation of a network. The value of the resolution parameter γ_{RB} can be chosen according to prior knowledge about the expected size of the communities. However, there are ongoing research interests to charac-

terize stable partitions, i.e. partitions that are optimal for a large range of values of γ_{RB} [Delvenne, Yaliraki, and Barahona (2010); Traag, Krings, and Van Dooren (2013); Delvenne, Schaub, Yaliraki et al. (2013)]. Those stable partitions are supposed to be the ones that provide the most significant clusters to represent the network [Lancichinetti, Radicchi, Ramasco et al. (2011)]. However, the stable partitions extracted for different resolution levels are often independent, i.e. one can not build one of the stable coarse-grained partitions based on one of the stable fine-grained partitions because the structure of the communities tends to be completely reorganized. Hence, what will be considered the "best" partition for a given network will always depend on the applications.

In their original work, Reichardt & Borhnoldt have suggested to use an Erdős-Rényi graph as the null model. This means that the parameter p_{ij} in Eq. (3.7) is chosen constant

$$p_{ij} = p,$$

such that every edge in the null model exists with a constant probability p. If we denote by n_r the number of nodes in community r, one can compute the expected number of edges inside community r as

$$\langle l_r \rangle_p = p n_r^2, \tag{3.10}$$

hence self loops are allowed in this null model. This leads to

$$H_{RB}(\sigma) = -\sum_{r=1}^{c} l_r - \gamma_{RB} p n_r^2 \tag{3.11}$$

which shows that a set of nodes is sufficiently connected to be a community if its internal density is larger than $\gamma_{RB} p$. As mentioned previously, the null model should match the original graph for some of its structural features, therefore the parameter p was chosen such that both graphs have on average the same number of edges. This can be done by choosing

$$p = \frac{m}{n^2}$$

where m is the total number of edges, $m = \sum_{i,j \in V} A(i,j)$.

The selection of an appropriate null model is a major concern. With the increasing interest for the detection of community partitions, many

different expressions of null models have been proposed [Perry and Wolfe (2012); Mondragón (2014)]. Some null models have also been designed for specific applications like the analysis of social networks [Milo, Shen-Orr, Itzkovitz et al. (2002)] or biological networks [Milenković, Filippis, Lappe et al. (2009)]. While the selection of a suited null model is often crucial, we will keep here the discussion focused on generic null models. In particular, the choice of an Erdős-Rényi graph for the null model has two major drawbacks. First, it is not straightforward to extend the model to weighted graphs. While $p_{ij} = p = \frac{m}{n^2}$ gives the probability to connect any pair of nodes such that the expected number of edges in the null model is equal to m, defining the weight of such edges would require additional and specific knowledge about the network. For example, it has been shown that choosing a geometric distribution for the weights is not appropriate to reveal community structure [Garlaschelli (2009)]. Moreover, it is well known that Erdős-Rényi graphs exhibit a Poissonian degree distribution for large values of n, though in real networks, the degree distribution often follows a fat-tail power law [Albert and Barabási (2002)]. This indicates that the number of edges might not be the best feature to match in the original graph. This observation leads to the definition of a null model with, on average, the same degree distribution as the input network, which is called the configuration null model. This is at the basis of the fitness measure introduced by Newman & Girvan called the *modularity*.

3.2.2 Newman & Girvan: Modularity

Newman & Girvan have introduced a fitness measure called the modularity [Newman and Girvan (2004); Newman (2004a); Leicht and Newman (2008)], based on the configuration model that produces a random graph with the same degree distribution as the original network [Luczak (1989); Molloy and Reed (1995)]. The modularity has rapidly become one of the most popular cost functions for community detection.

Let us first consider an unweighted directed network and explain how one can build a random graph using the configuration model. As we will see, there is a straightforward extension for weighted networks, however the intuition behind the model might be more clear using an unweighted graph.

As illustrated in Fig. 3.2, from an *input graph*, one can build two sequences of nodes representing the edge list. The $i^{(out)}$ sequence corresponds to nodes having an outgoing edge, while conversely the $j^{(in)}$

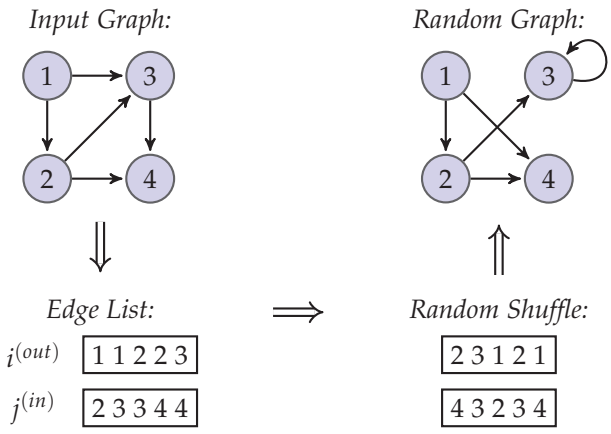


Fig. 3.2 Configuration null model

sequence corresponds to nodes having an incoming edge. This implies that each node is present in the $(out)/(in)$ sequence as many times as its out/in degree respectively. Hence, the k^{th} edge can be recovered by taking the k^{th} element of the $i^{(out)}$ sequence as source and the k^{th} element of the $j^{(in)}$ sequence as destination.

Then, a random shuffle is applied to the outgoing and incoming nodes sequences and the random graph is created by reading sequentially the nodes in the shuffled sequences. This procedure rewires all the edges of the graph, although maintaining the exact same degree sequence as the original graph.

Based on this configuration null model, it is possible to define the expected number of edges p_{ij} between 2 nodes i and j in the random graph. Each outgoing stub, i.e. a half-edge represented by a single element of the $i^{(out)}$ sequence, is randomly matched to one of the incoming stubs, i.e. a half-edge from $j^{(in)}$ sequence. The probability that an outgoing stub from node i is rewired to one of the incoming stubs of node j is $\frac{k_j^{in}}{m}$ where k_j^{in} is the number of incoming stubs attached to node j out of $m = \sum_{i,j \in V} A(i,j)$ incoming stubs in total. If k_i^{out} is the number of outgoing stubs from node i, the expected number of edges from i to j can be approximated for large m by

$$p_{ij} \cong k_i^{out} \frac{k_j^{in}}{m}. \tag{3.12}$$

One can observe that the configuration null model matches, on average, both the number of edges in the graph and the degree sequence. Indeed, we have that

$$\sum_{i \in V} k_i^{in} = \sum_{i \in V} k_i^{out} = \sum_{i \in V} \sum_{j \in V} A(i,j) = m,$$

so, the expected outgoing degree of a node i in the random graph $k_{i,r}^{out}$ is the sum of the expected number of edges from i to each node in the graph and is given by

$$\langle k_{i,r}^{out} \rangle = \sum_{j \in V} p_{ij} = \frac{k_i^{out}}{m} \sum_{j \in V} k_j^{in} = k_i^{out},$$

and the expected number of edges in the random graph m_r is given by

$$\langle m_r \rangle = \sum_{i,j \in V} p_{ij} = \frac{1}{m} \sum_{i \in V} k_i^{out} \sum_{j \in V} k_j^{in} = m.$$

Based on a community partition, the *modularity cost function*, as introduced by [Newman and Girvan (2004)], compares the actual edge density within each community to the expected edge density in a random network using the configuration null model. That is,

$$Q(\sigma) = \sum_{i,j \in V} \left[\frac{A(i,j)}{m} - \frac{k_i^{out}}{m} \frac{k_j^{in}}{m} \right] \delta(\sigma_i, \sigma_j) \tag{3.13}$$

which, up to a scaling $\frac{1}{m}$ that does not change the optimal partition, can be expressed in the framework of Reichardt & Borhnoldt, given by Eq. (3.2), with $\alpha_{i,j} = 1$ and $\beta_{i,j} = p_{ij}$ as defined in Eq. (3.12), so $\gamma_{RB} = 1$ in Eq. (3.7).

As mentioned, the modularity has a straightforward extension to weighted graphs [Newman (2004a)], using the outgoing and incoming strengths instead of the degrees for the null model

$$Q_w(\sigma) = \frac{1}{m_w} \sum_{i,j \in V} \left[W(i,j) - s_i^{out} \frac{s_j^{in}}{m_w} \right] \delta(\sigma_i, \sigma_j) \tag{3.14}$$

where m_w is the total weight of the input graph

$$m_w = \sum_{i,j \in V} W(i,j) = \sum_{i \in V} s_i^{out} = \sum_{j \in V} s_j^{in}.$$

The modularity score of a partition ranges in $[-1, 1]$. It equals 0 for a partition with only 1 community

$$\begin{aligned} Q_w(\mathbf{1}) &= \frac{1}{m_w} \sum_{i,j \in V} W(i,j) - \frac{1}{m_w} \sum_{i,j \in V} s_i^{out} \frac{s_j^{in}}{m_w} \\ &= \frac{1}{m_w} \sum_{i,j \in V} W(i,j) - \frac{1}{m_w} \sum_{i \in V} s_i^{out} \frac{1}{m_w} \sum_{j \in V} s_j^{in} = 1 - 1 = 0, \end{aligned}$$

and is in general negative for a partition with n communities, i.e. every node defining its own community, if there are not many self loops in the graph. Hence, community partitions associated to high positive values of the modularity score are supposed to accurately represent the modular structure of the network. It was originally assumed by Newman & Girvan that a modularity score larger than 0.3 for a given partition indicates that the network under study has a community structure accurately represented by the aforementioned partition. However, this hypothesis has been questioned in recent studies. It has been shown that the optimal modularity score tends to naturally increase with the size of the networks and that some particular graph classes can achieve arbitrary large value of modularity like trees which intuitively do not contain community structure [de Montgolfier, Soto, and Viennot (2011); Bagrow (2012)]. Moreover, the modularity suffers from local degeneracies and exhibits a plateau landscape around its optimum [Good, de Montjoye, and Clauset (2010)]. This means that there exists a large number of alternative partitions that, while being structurally different than the optimal partition, achieve roughly a similar modularity score. This problem occurs for different versions of the modularity, e.g. unweighted in Eq. (3.13) or weighted in Eq. (3.14), and is unfortunately more severe for networks with a modular structure. Therefore, one should avoid using the modularity score to compare the partitions of different networks and even more so if the networks are of different sizes. Additionally, one should not use the modularity score of a partition to assess the quality of that partition regarding the modular structure of the network. Instead, additional knowledge and thorough analysis of the extracted communities should be used to infer the quality of the clusters regarding the application.

In addition, the communities extracted using modularity optimization are subject to resolution limits, i.e. the size of the clusters is constrained by the size of the network. The resolution limit phenomenon has been

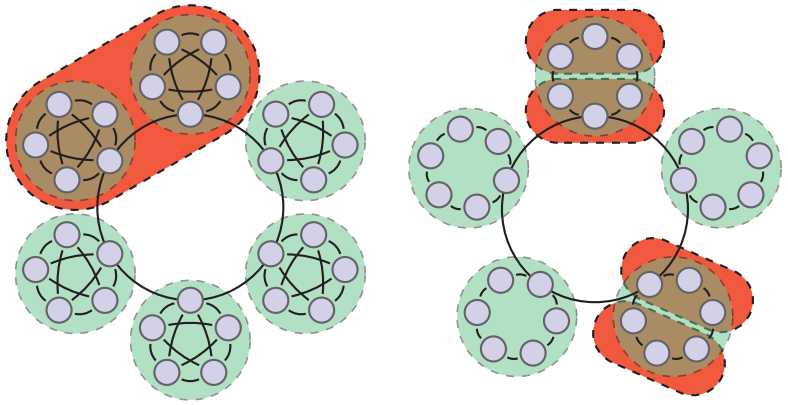


Fig. 3.3 Resolution limits on the ring of cliques and the ring of rings networks

introduced by [Fortunato and Barthélemy (2007)] using a ring of cliques as represented on the left hand side in Fig. 3.3. In such network, the natural communities are intuitively induced by each of the cliques, as represented in light green in the figure. However, one can show that, if n_c is the number of nodes in each of the k cliques, then the modularity of a partition that merges two consecutive cliques on the ring, as represented in red in the figure, is higher than the natural partition if

$$n_c\,(n_c - 1) + 2 < k.$$

In other words, if the network is large and contains many small cliques, then the communities induced by the cliques are too small to be optimal according to the modularity, even if they are the densest clusters. More formally, it has been proved that modularity maximization will in general fail to extract communities with less than $\sqrt{m/2}$ edges. This resolution limit also occurs when one considers the weighted modularity given by Eq. (3.14) [Berry, Hendrickson, LaViolette et al. (2011)]. In this case, it may be impossible to recover communities with a total internal weight less than $\sqrt{m_w \epsilon/2}$ where ϵ is the maximum weight of the inter-communities edges.

Furthermore, the size of the communities extracted using modularity optimization is not only lower-bounded but may also be constrained by an upper bound in some situations. This has been termed the "field of

view limit" and has been highlighted by [Schaub, Delvenne, Yaliraki et al. (2012)]. Though, in this work, the authors consider a slightly different definition of the communities, i.e. communities act as local traps for a random walker that navigates over the network following the existing edges. Once the random walker enters a community, he should need a large number of steps to exit from it if the community is well defined. Hence, one can consider a "ring of rings" network, as represented on the right hand side of Fig. 3.3. When a random walker enters one of the external rings, the probability that he will stay within the same ring is quite large and becomes even larger as the random walker goes more deeply within the ring. This leads to a natural definition of communities as the subgraphs induced by each of the external rings, as depicted in light green in the figure. However, one can show that, if the number of external rings k is such that

$$k < \frac{n_c}{4}\frac{(n_c+2)}{(n_c+1)} \approx \frac{n_c}{4}$$

then a partition that splits each ring in two separate components, as represented in red in the figure, has a larger modularity score than the natural partition. This "field of view limit" arises when communities are formed by long chains of nodes which may happen when one considers graphs of images or DNA sequences for example. We will have to consider this field of view limit to define an appropriate null model for image segmentation in Section 4.2.

Despite those theoretical limitations, the modularity cost function, as given by Eq. (3.14), has been widely applied in many different fields and practical contexts [Guimerà, Mossa, Turtschi et al. (2005); Kashtan and Alon (2005); Mucha, Richardson, Macon et al. (2010); Conover, Davis, Ferrara et al. (2013)], but it is not entirely known why modularity optimization is able to achieve good clustering results in real networks. The problem to find the community structure that maximizes the modularity score, has been proven to be a NP-hard problem [Brandes, Delling, Gaertler et al. (2008)]. Hence, a lot of different heuristics and greedy algorithms have been developed to approximate the optimal partition and we will present some of them in Section 3.3.

3.2.3 Resolution limit free models

More recently, different models for the spin interaction weights that do not include any null model have been proposed. Hence, under some assumptions, one can prove that such models do not suffer from resolution limits and provide in general a better network clustering. However, the optimization of this kind of cost functions requires some additional parameters tuning to extract the most significant partitions [Lambiotte, Delvenne, and Barahona (2008); Delvenne, Yaliraki, and Barahona (2010); Traag, Krings, and Van Dooren (2013)] which may increase significantly the computational cost of community detection algorithms.

Constant Potts model

The Constant Potts Model (CPM) has been introduced by [Traag, Van Dooren, and Nesterov (2011)] and is defined in the framework of Reichardt & Borhnoldt of Eq. (3.2) by

$$\alpha_{ij} = w(i,j),$$
$$\beta_{ij} = \gamma,$$

which leads to

$$H_{CPM}(\sigma) = -\sum_{i,j\in V}\Big[W(i,j) - \gamma_{CPM}\Big]\delta\left(\sigma_i,\sigma_j\right). \tag{3.15}$$

This model is essentially equivalent to the Erdős-Rényi null model if one considers

$$\gamma_{CPM} = \gamma_{RB} p.$$

In this work, the authors give a precise definition of what it means for an objective function to be free of resolution limits.

Let $\sigma = \{\sigma_1, \dots \sigma_c\}$ be an optimal partition of a graph G for a fitness measure $H_G(\sigma)$. Then, the fitness measure $H(\sigma)$ is *resolution-limit-free* if any sub-partition $\varsigma = \{\varsigma_1, \dots, \varsigma_k\}$, $\varsigma_i \in \sigma$, is also optimal for the graph G_ς induced by the nodes in ς, that is $H_{G_\varsigma}(\varsigma) \leq H_{G_\varsigma}(\theta)$ for any partition θ of G_ς. In particular, this implies that if one looks at each subgraph G_i induced by a community σ_i, the optimal partitioning of G_i according to $H_{G_i}(\sigma)$ must be to cluster all the nodes together if $H_G(\sigma)$ is resolution-limit-free. Hence, each community does not depend on the rest of the

network and is both locally and globally optimal. Clearly, the modularity is not resolution-limit-free in this context since one can expect that the partitioning obtained for different subgraphs will be completely independent of the partitioning of the entire graph.

Moreover, Traag et al. give a sufficient condition for an objective function to be resolution-limit-free. They define the weight functions $\alpha_{ij}(G)$ and $\beta_{ij}(G)$, associated to a graph G, as *local* if for any subgraph $g \subset G$,

$$\alpha_{ij}(G) = \lambda(g)\alpha_{ij}(g),$$
$$\beta_{ij}(G) = \lambda(g)\beta_{ij}(g),$$

where λ is a scaling that may depend on the subgraph g and they show that any fitness measure that has local weights is resolution-limit-free.

Ronhovde & Nussinov

Ronhovde & Nussinov proposed a model [Ronhovde and Nussinov (2009, 2010)] with

$$\alpha_{ij} = w(i,j) + \gamma_{RN},$$
$$\beta_{ij} = \gamma_{RN},$$

which leads to

$$H_{RN}(\sigma) = -\sum_{i,j\in V}\Big[(w(i,j)+\gamma_{RN})A(i,j) - \gamma_{RN}\Big]\delta\left(\sigma_i,\sigma_j\right). \tag{3.16}$$

For unweighted graph, this cost function is equivalent, up to a scaling factor, to the Erdős-Rényi null model with

$$\gamma_{RN} = \frac{\gamma_{RB}p}{1-\gamma_{RB}p}.$$

However, the optimal partition for weighted network might be different.

Label Propagation

Raghavan et al. introduced an algorithm for community detection called Label Propagation [Raghavan, Albert, and Kumara (2007)]. As we will see in Section 3.3.5, this algorithm searches for a partition where every node has an internal degree or an internal strength within its community larger

than in any other community.

Even though in the original paper the authors do not optimize a quality criterion and rather define a spreading pattern of the labels over the network, it has been shown by [Tibély and Kertész (2008)] that the Label Propagation algorithm is equivalent to finding the local minima of a Potts model with

$$\alpha_{ij} = w(i,j),$$
$$\beta_{ij} = 0,$$

hence

$$H_{LP}(\sigma) = -\sum_{i,j \in V} W(i,j)\delta\left(\sigma_i, \sigma_j\right). \tag{3.17}$$

The optimal partition for this model is meaningless since it is trivially composed of a single community with all the nodes. However, Tibéli & Kertész have demonstrated that the number of local minima is in general much larger than the number of nodes and that some of those local minima may be interesting.

3.2.4 The Map equation

The Map equation has been introduced by [Rosvall and Bergstrom (2008, 2011)] and has been shown to be an efficient objective function for community detection. It is inspired by information and optimal coding theory. As it does not fit in the framework of Reichardt & Bornholdt, we will first review some basic elements of information theory and then present the fitness measure.

Information theory

Information theory refers to a branch of applied mathematics that quantifies how information can be optimally represented, compressed, stored or exchanged. For example, one may want to send a picture through a communication channel, and for the sake of simplicity, assume that the picture contains 1000 white pixels, so each pixel takes the value 255. One naive solution is to send every single pixel, and since each pixel requires 8 bits (to represent a number between 0 and 255), this solution requires to send 8000 bits of data. On the other hand, one could send only the value of

the first pixel (8 bits) and, for any consecutive pixels, send the difference between the current pixel and the previous one, which in this case would be always 0 (1 bits). Hence, this coding strategy is much more effective and only requires to transfer of 1007 bits of data. This kind of compression is called lossless, i.e. one can recover the original picture without loss of quality, and is the basic principle of the BMP file format. Information theory was originally developed by [Shannon (1948)] and now finds applications in many different problems of signal processing, e.g. natural language processing, cryptography, image processing, etc. [MacKay (2003); Cover and Thomas (2012)].

Let us assume that x is an event that may occur with a probability $p(x)$. The *information* contained in the event x is defined by

$$I(x) = -\log\Big(p(x)\Big).$$

One can observe that the information of the event x is inversely proportional to its probability to occur $p(x)$. In the limit, if the event x happens with probability $p(x) \to 1$, observing x does not bring much information since we knew it would happen with high probability. On the other hand, if the probability $p(x) \to 0$, observing the event x is rare enough such that when it happens, it gives a lot of information about the system. When the logarithm is taken in base 2 (which will always be the case here), the information of x is measured in bits and corresponds the number of bits required to represent the event x. Suppose that we flip a coin and that the event x is "the coin flipped to head", which happens with probability $p(x) = \frac{1}{2}$. Then we need $I(x) = 1$ bit to represent that x happens (or not),

$$\textit{code}: \quad \underbrace{1}_{\text{success}} \quad \underbrace{0}_{\text{failure}}$$

If the event x corresponds to obtaining a specific face for a roll of an octahedral dice, we need $I(x) = 3$ bits to represent x since $p(x) = \frac{1}{8}$,

$$\textit{code}: \quad \underbrace{111}_{\text{success}} \quad \underbrace{000\ 001\ 010\ 100\ 011\ 101\ 110}_{\text{failure}}$$

Given a random variable X, the Shannon entropy $H(X)$ is a measure of the uncertainty of the distribution of X. It is defined as the expected

information of a single realization of X, i.e.

$$H(X) = \langle I(X) \rangle = -\sum_{X=x} p(x) \log\Big(p(x)\Big).$$

The Shannon's noiseless coding theorem demonstrates that the entropy of X is the optimal lower bound on the average code length to represent all the states of X without loss of information. In other words, for any code c, such that the codeword c_i is associated to $X = x_i$,

$$H(X) \leq -\sum_i p(x_i) \, |c_i| \, ,$$

where $|c_i|$ is the length of the codeword c_i.

A code that attains the minimal average code length is called a Huffman code and an algorithm to extract such a code has been developed by [Huffman (1952)]. For example, let us consider a random variable X that can reach 4 different states with equal probability. Since $\forall i \; p(x_i) = \frac{1}{4}$, the optimal coding strategy requires at least

$$H(X) = -\sum_{i=1}^{4} \frac{1}{4} \log\Big(\frac{1}{4}\Big) = 2 \textit{ bits per code},$$

and one can show that there exists a Huffman code defined by

x_i	$p(x_i)$	*code*
1	1/4	00
2	1/4	01
3	1/4	10
4	1/4	11

If the 4 distinct states x_i are not equiprobable, then one can adopt a different strategy and give shorter codewords for the states that occur more often, hence reducing the expected codeword length:

x_i	$p(x_i)$	*code*
1	1/2	0
2	1/4	10
3	1/8	110
4	1/8	111

In this case, the entropy of the X is $H(X) = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4}$ and one can check that the above code is a Huffman code since the expected length of a codeword is also $\frac{7}{4}$. It is easy to show that there is not always a Huffman code attaining the Shannon entropy. Consider a random variable with 2 states that occurs with probability $p_1 > p_2$. The entropy of this random variable is always smaller than 1, however it is impossible to encode it with an expected codeword length smaller than 1 since we need at least 1 bit per state. However, it has been proven that there exists always a Huffman code with an expected codeword length larger of at most 1 bit than the Shannon entropy [Cover and Thomas (2012)].

In the following, we will need to encode a sequence of events, which means that we draw multiple realizations of the random variable X. For example, consider the following sequence according to the previous 4 states variable and the associated coding sequences,

$$\begin{aligned} 1-4-2-1-3 \quad &\Rightarrow \quad 0011010010 \quad \textit{equiprobable} \\ &\Rightarrow \quad 0111100110 \quad \textit{not equiprobable} \end{aligned}$$

where the colors are only present to help the reader to match the original sequence with the encoded sequences. When the equiprobable encoder is used, the decoding of the sequence is unambiguous because we know that we need to read 2 bits per state. However, when the not equiprobable encoder is applied, the code needs to be prefix-free to be decoded unambiguously, which means that no code can appear as a prefix of another code. Since we do not know the length of each code when the encoded sequence is received, if a code appears as prefix of another code, then the decoder can not know, when reading the bits consecutively, which of the 2 states is in the original sequence. This is not the case here and the proposed code is prefix-free. First the decoder reads 0 and knows that the associated state is 1. Then, the decoder reads 1 and knows that it needs at least 1 more bit to decode the state. So, it reads 11 and again requires another bit of data to decode the state. Finally, it reads 111 and knows that the associated state is 4, and so on.

The Map Equation

The map equation introduced by Rosvall & Bergstrom is based on the idea we previously mentioned: a random walker, following each edge proportionally to its weight, should spend most of its time inside communities

and barely use edges across communities [Rosvall and Bergstrom (2008)]. One way to encode each step of a random walk is then to give a specific codeword to each node of the graph. In this case, one can compute the expected length of a codeword as the entropy of the stationary frequency distribution to visit each node. If we defined the transition probability matrix L for a graph $G(W, E)$,

$$L = S^{-1}W$$

where S is the diagonal matrix of the node strength, $S = diag\,(W\mathbf{1})$, then the stationary frequency distribution to visit each node is given, according to the Perron-Frobenius theorem, by π, the dominant left eigenvector of L,

$$L^T\pi = \pi,$$

which can be computed using a power method. The expected codeword length of a single step of the random walker is then bounded by the Shannon entropy of π

$$H(\pi) = -\sum \pi_i \log\Big(\pi_i\Big).$$

However, we can do better here because there is an underlying network structure such that only the transitions between connected states are possible. The actual lower bound is given by the Shannon entropy of the associated Markov process [Cover and Thomas (2012)] which is given by

$$H(G) = \sum \pi_i \; L(i,j) \; \log\Big(L(i,j)\Big).$$

Finding a code that reaches this lower bound might be really complicated or even impossible. The clever idea of the map equation is then to assume that the steps of the random walker can be efficiently represented by a two levels encoder (or with even more levels as presented in [Rosvall and Bergstrom (2011)]). The first level encodes the different communities while the second level encodes the individual nodes. This allows to reuse the codewords for the nodes in a community to encode the nodes in another community since those can be differentiated knowing in which community is the random walker. This efficiently reduces the expected length of each step of the random walker. So, the problem of finding an accurate community partition becomes to find an optimal code to minimize the expected description length of each step of a random walker which is given by the map equation. Given a partition σ, one can com-

pute the probability that the random walker exits its current community as

$$q_{out} = \sum_{k=1}^{m} q_{k,out} \tag{3.18}$$

where $q_{k,out}$ is the per step probability that the random walker exits community k, and is given by

$$q_{k,out} = (1-\tau) \sum_{i \in k} \sum_{j \notin k} \pi_i \, L(i,j) + \tau \frac{n - n_k}{n-1} \sum_{i \in k} \pi_i \tag{3.19}$$

where τ is a teleportation probability (hence, the random walker is a random surfer). This teleportation parameter must be included to analyze directed networks otherwise the random walker would end up being stuck in stationary classes of the Markov process represented by L and not explore the entire network. Though, the clustering results are highly robust to the choice of the value of τ and Rosvall & Bergstrom specify $\tau = 0.15$ as a typical choice. The first term of Eq. (3.19) corresponds to the probability that the random walker is in community k, does not get teleported and exits the community by following one of the outgoing edges, while the second term corresponds to the probability for the random walker to be teleported to one node outside community k. The entropy of the movements between communities is then given by

$$H(q_{out}) = \sum_{k=1}^{m} \frac{q_{i,out}}{q_{out}} \log\Big(\frac{q_{i,out}}{q_{out}}\Big) \tag{3.20}$$

and gives a lower bound on the expected length of the codeword associated to communities. Conversely, the probability to move within community k is given by

$$q_{k,in} = q_{k,out} + \sum_{i \in k} \pi_i \tag{3.21}$$

where $q_{k,out}$ corresponds to the exit codeword, associated to community k, which specifies that the random walker will exit the community in the next step. The entropy of movements within module k is then given by

$$H(q_{k,in}) = \frac{q_{k,out}}{q_{k,in}} \log\Big(\frac{q_{k,out}}{q_{k,in}}\Big) + \sum_{i \in k} \frac{\pi_i}{q_{k,in}} \log\Big(\frac{\pi_i}{q_{k,in}}\Big). \tag{3.22}$$

Finally, the map equation reads

$$H_M(\sigma) = q_{out} H(q_{out}) + \sum_{k=1}^{m} q_{k,in} H(q_{k,in}). \tag{3.23}$$

We will present in Section 3.3.7 the Infomap,which is a collection of greedy algorithms proposed by Rosvall & Bergstrom to optimize the map equation.

3.2.5 Surprise

For the sake of completeness, we conclude this section about fitness measures for community detection by introducing a very recent quality function called *Surprise* [Aldecoa and Marín (2011)], which is inspired by the work of [Arnau, Mars, and Marín (2005)].

Based on a community partition, the Surprise computes the probability to (surprisingly) observe at least as many internal edges as within the proposed partition in a uniform random graph. Let us derive this probability by first denoting by M the maximum possible number of edges in an undirected network,

$$M = \frac{n(n-1)}{2},$$

and, as previously, by m the actual number of edges in the graph

$$m = \frac{1}{2} \sum_{i,j \in V} A(i,j).$$

Furthermore, we denote by F the maximum possible number of intracommunity edges for the given partition

$$F = \frac{1}{2} \sum_{c} n_c (n_c - 1) = \frac{1}{2} \sum_{i,j \in V} \delta\left(\sigma_i, \sigma_j\right),$$

where n_c is the number of nodes in community c, and similarly we denote by f the actual number of intracommunity edges

$$f = \frac{1}{2} \sum_{i,j \in V} A(i,j) \delta\left(\sigma_i, \sigma_j\right).$$

Then, the probability to observe f internal edges, out of the F possible

internal edges in a graph with m edges is given by a hypergeometric distribution

$$P(X = f \mid F, m, M) = \frac{\binom{F}{f}\binom{M-F}{m-f}}{\binom{M}{m}}.$$

The Surprise is defined by a cumulative hypergeometric distribution since it computes the probability to observe at least f intracommunity links over a maximal number of intracommunity links in the network given by $\min(F, m)$, hence

$$H_S(\sigma) = -\log \sum_{j=f}^{\min(F,m)} \frac{\binom{F}{j}\binom{M-F}{m-j}}{\binom{M}{m}} \tag{3.24}$$

Aldecoa & Marin have recently shown that the Surprise qualitatively outperforms other commonly used criteria for community detection [Aldecoa and Marín (2011)]. Furthermore, the maximization of $H_S(\sigma)$ provides accurate and natural partitions [Aldecoa and Marín (2013)] although it is not entirely known how the measure is affected by resolution limits.

Even though the performances of the Surprise function are excellent, the fact that the measure is not additive renders the extraction of significant community partitions computationally more intensive. Moreover, this cost function still misses generalization for weighted or directed networks which are our main concern. Hence, we will not evaluate the performance of this measure in details, even though we believe that it will become increasingly popular in the future.

3.2.6 Summary

In this section, we introduced different cost functions for community detection which should be optimized over the community assignment of each node. Reichardt & Bornholdt introduced a general framework in which most of them can be expressed as a particular case of the choice of the parameters. The energy of a partition, that should be maximized in

	Unweighted network	Weighted network
Reichardt & Bornholdt	$\alpha_{ij} = 1$	$\alpha_{ij} = w(i,j)$
	$\beta_{ij} = \gamma_{RB} p_{ij}$ $\quad p_{ij} = \frac{mn_c^2}{n^2}$	
Newman & Girvan (modularity)	$\alpha_{ij} = 1$ $\quad \beta_{ij} = \frac{k_i^{out} k_j^{in}}{m}$	$\alpha_{ij} = w(i,j)$ $\quad \beta_{ij} = \frac{s_i^{out} s_j^{in}}{m_w}$
Traag et al. (CPM)	$\alpha_{ij} = 1$	$\alpha_{ij} = w(i,j)$
	$\beta_{ij} = \gamma_{CPM}$	
Ronhovde & Nussinov	$\alpha_{ij} = 1 + \gamma_{RN}$	$\alpha_{ij} = w(i,j) + \gamma_{RN}$
	$\beta_{ij} = \gamma_{RN}$	
Raghavan et al. (label propagation)	$\alpha_{ij} = w(i,j)$ $\quad \beta_{ij} = 0$	

Table 3.1 Summary of cost functions for community detection.

order to extract relevant communities, is given, up to a scaling factor, by

$$H(\sigma) = \sum_{i,j \in V} \Big[\alpha_{ij} A(i,j) - \beta_{ij} \Big] \delta\left(\sigma_i, \sigma_j\right).$$

Many researchers have proposed values for the parameters α_{ij} and β_{ij} and we presented some of the most popular choices, summarized in Table 3.1.

However, not all criteria can be expressed in the framework of Reichardt & Bornholdt. For example, we presented the map equation, introduced by Rosvall & Bergstrom and based on information theory, that quantifies the entropy of a random walker over the network. The optimal community partition should entail the maximal compression of the network and therefore minimize the average description length of a single step of the random walker given by

$$H_M(\sigma) = q_{out} H(q_{out}) + \sum_{k=1}^{m} q_{k,in} H(q_{k,in}).$$

Finally, we described another approach based on the probability to observed surprisingly as many edges within the communities of a partition than within the same communities but in a uniform random graph. This

probability is given by an hypergeometric distribution and the so-called Surprise reads

$$H_S(\sigma) = -\log \sum_{j=f}^{\min(F,m)} \frac{\binom{F}{j}\binom{M-F}{m-j}}{\binom{M}{m}}.$$

The optimization of those cost functions is in general NP-hard. Moreover, the Infomap and the Surprise are not additive in the sense that one can not compute the value of those objective functions as the sum of the individual fitness value of each community (some global considerations have to be taken into account), which is often an additional challenge. In the following section, we will present some of the existing algorithms that greedily find an approximation of the optimal community partition and then present a fast, efficient and highly parallelizable algorithm that we developed.

3.3 Algorithms for Community Detection

Finding the optimal partition for a given cost function is in general out of reach due to the NP-hardness of the underlying optimization problem. Hence, multiple greedy algorithms and heuristics have been developed to approximate the optimal community partition of a network. All those algorithms have to choose a balance between their time complexity and the expected quality of the extracted partitions. For example, one can easily implement an exhaustive search through all the possible partitions, however the associated complexity would grow faster than exponentially since the number of partitions is given by the Bell number, and one should not expect to get an accurate community partition for real networks, even with only a few thousands nodes, in less than a lifetime.

In this section, we will describe some of the most popular algorithms for community detection, either due to historical reasons or due to their performance. Although, we will only present the algorithms that are applicable to weighted and directed graphs. Unless specifically stated otherwise, we will always consider the maximization of a general fitness measure $H(\sigma)$ knowing that any of the previously mentioned cost functions can be used.

3.3.1 Spectral optimization

Spectral optimization refers to the optimization of the fitness measure $H(\sigma)$ using the eigenvectors and eigenvalues of the associated matrix [Newman (2006)] and is similar to the classical spectral partitioning. One can consider a general formulation

$$H(\sigma) = \sum_{i,j \in V} B(i,j)\delta(\sigma_i, \sigma_j) \tag{3.25}$$

where the matrix B depends on the chosen cost function. For instance, using the CPM yields $B = W - \gamma \mathbf{1}\mathbf{1}^T$, while modularity, with the configuration null model, yields $B = W - \frac{W\mathbf{1}\mathbf{1}^T W}{m}$. Let us first assume that we are looking for an optimal bisection of the network which will be defined by a partitioning vector $\mathbf{s}$ such that $s_i = 1$ if node i is in the first community, and $s_i = -1$ if node i is in the second community. One can write that

$$\begin{aligned} H(\sigma) &= \sum_{i,j \in V} B(i,j)\delta(\sigma_i, \sigma_j) = \sum_{i,j \in V} B(i,j)\frac{s_i s_j + 1}{2} \\ &= K + \frac{1}{2}\sum_{i,j \in V} B(i,j) s_i s_j = K + \frac{1}{2}\mathbf{s}^T B \mathbf{s} \end{aligned}$$

where $K = \frac{1}{2}\sum_{i,j \in V} B(i,j)$ is a constant independent of the partition σ. So, the problem of maximizing $H(\sigma)$ can be written as

$$\begin{aligned} \max_{\mathbf{s}} \quad & \mathbf{s}^T B \mathbf{s}, \\ s.t. \quad & \mathbf{s} = \{-1, 1\}^n. \end{aligned}$$

which is strictly equivalent to

$$\begin{aligned} \max_{\mathbf{s}} \quad & \mathbf{s}^T \, \widetilde{B} \, \mathbf{s}, \\ s.t. \quad & \mathbf{s} = \{-1, 1\}^n. \end{aligned}$$

where $\widetilde{B}$ is the symmetric part of B, $\widetilde{B} = \frac{B + B^T}{2}$, since for the antisymmetric part

$$\mathbf{s}^T \left(\frac{B - B^T}{2} \right) \mathbf{s} = \frac{1}{2}\left(\mathbf{s}^T B \mathbf{s} - \mathbf{s}^T B^T \mathbf{s} \right) = \frac{1}{2}\left(\mathbf{s}^T B \mathbf{s} - \mathbf{s}^T B \mathbf{s} \right) = 0.$$

This optimization problem is NP-hard, however if we relax the constraint $\mathbf{s} = \{-1, 1\}^n$ and only impose $\|\mathbf{s}\|_2^2 = n$, the problem becomes similar to optimizing a Rayleigh quotient for which the solution is known to be given by the dominant eigenvector of $\widetilde{B}$. Since $\widetilde{B}$ is symmetric, $\mathbf{s}$ can be decomposed on the basis of the eigenvectors $\mathbf{v_i}$ of $\widetilde{B}$,

$$\mathbf{s} = \sum_{i=1}^{n} \alpha_i \mathbf{v_i} \quad , \quad \alpha_i = \mathbf{v_i}^T \mathbf{s}.$$

Then, one can see that

$$\begin{aligned} 4\left(H(\sigma) - K\right) &= \sum_i \alpha_i \mathbf{v_i}^T \widetilde{B} \sum_j \alpha_j \mathbf{v_j} \\ &= \sum_{i,j} \alpha_i \alpha_j \mathbf{v_i}^T \widetilde{B}\, \mathbf{v_j} = \sum_i \alpha_i^2 \lambda_i \end{aligned}$$

where λ_i is the eigenvalue associated to v_i. This shows that, as a first approximation, the partitioning vector can be chosen, according to the dominant eigenvector v_1, such that

$$s_i = \begin{cases} 1 & \text{if } v_1(i) > 0 \\ -1 & \text{if } v_1(i) < 0 \end{cases}$$

One can then reapply the same procedure to each of the already extracted communities in order to obtain sub-communities, and do so as long as the fitness measure increases. However, one should not bisect the subgraph induced by the community, but rather compute the dominant eigenvector of the submatrix spanned by the set of nodes in the subgraph, to ensure an increment of the cost function defined over the whole graph. Unlike in the classical graph partitioning problem, the number of communities does not need to be specified in advance and the algorithm stops when no more positive gain of the fitness measure can be achieved by bipartitioning any of the communities.

This algorithm has been shown to work very well for graph bisection, however its performance declines when the graph contains more than 2 clusters, i.e. in the early stages of the algorithm, spectral partitioning tends to cut some of the existing clusters which become impossible to recover afterwards. Additionally, if some elements of the dominant eigenvector v_1 are close to 0, then the assignments of the associated nodes may

be equivocal. The cut value of 0 that defines the partitioning vector $\mathbf{s}$ is somehow arbitrary and a slight variation of this threshold can lead to very different community partitions. Some authors [Wang, Shen, and Ouyang (2008); Richardson, Mucha, and Porter (2009)] have proposed to use more than a single eigenvector to define the partitioning vector. In this case, the p dominant eigenvectors are computed and form an indicator matrix $X \in \mathbb{R}^{n \times p}$, i.e. each row of X is an indicator vector of dimension p for the corresponding node. The community partition of the graph is then obtained by clustering the nodes in the induced p-dimensional space. Unfortunately, one needs to find an appropriate value for p, neither too large to reduce the computational complexity, nor too small to have sufficient information in the indicator vector to achieve good performances.

The dominant eigenvectors of $\widetilde{B}$ can be computed using a power method in $O(n)$ iterations and, even if the matrix $\widetilde{B}$ is full, its peculiar form allows a faster computation of the multiplication with a vector in $O(m+n)$. So, the complexity of the algorithm is $O(n\,(m+n))$ or $O(n^2)$ if the graph is sparse, which is acceptable.

3.3.2 Simulated Annealing

Simulated annealing (SA) is a general discrete optimization technique proposed by [Kirkpatrick, Gelatt, and Vecchi (1983)]. This algorithm is designed to allow, at the beginning, the exploration of a large proportion of the admissible set and to progressively narrow its search space. Let us assume that we have 2 different states $\sigma^{(1)}$ and $\sigma^{(2)}$, that in our context will end up being community partitions, and that the current state is $\sigma^{(1)}$. Furthermore, we will denote by ΔH the variation of the quality function to switch from state (1) to state (2),

$$\Delta H\left(\sigma^{(1)}, \sigma^{(2)}\right) = H\left(\sigma^{(2)}\right) - H\left(\sigma^{(1)}\right).$$

The algorithm accepts to switch to state $\sigma^{(2)}$ with probability

$$Pr\left(\sigma^{(1)} \rightsquigarrow \sigma^{(2)}\right) = \frac{1}{k} e^{\beta \Delta H\left(\sigma^{(1)}, \sigma^{(2)}\right)}$$

where k is a scaling parameter over all the possible transitions from state $\sigma^{(1)}$,

$$k = \sum_{\sigma^{(i)} \in \Theta\left(\sigma^{(1)}\right)} e^{\beta \Delta H\left(\sigma^{(1)}, \sigma^{(i)}\right)},$$
$$\Theta\left(\sigma^{(1)}\right) = \left\{\sigma \mid \sigma^{(1)} \rightsquigarrow \sigma\right\},$$

and β is a noise parameter that can be considered as an inverse temperature $\beta = \frac{1}{T}$. One can see that transitions that reduce the cost function might be accepted which serves the objective to explore the search space and reduces the risk that the algorithm gets trapped in local minima. Hence, the algorithm starts with a high temperature, or a small β, such that all the transitions are accepted with an almost uniform probability distribution. Then, the temperature is slowly decreased, and the transitions with a positive variation of the cost function start to have a higher probability to be accepted. Finally, $\beta \to \infty$ and the algorithm only accepts transitions with a strictly positive variation of the cost function until no such transition exists. The system will always converge to a stable state that can be an arbitrarily good approximation of the true optimum depending on the chosen parameters.

This algorithm has been applied for community detection by [Guimerà, Sales-Pardo, and Amaral (2004); Guimera and Amaral (2005)] by defining two types of transitions, though in this version, the probability to accept a transition is slightly different

$$Pr\left(\sigma^{(1)} \rightsquigarrow \sigma^{(2)}\right) = \begin{cases} 1 & \text{if } \Delta H > 0, \\ \frac{1}{k} e^{\beta \Delta H\left(\sigma^{(1)}, \sigma^{(2)}\right)} & \text{if } \Delta H \leq 0. \end{cases}$$

The first type of accepted transition is a local vertex switch, i.e. a random node is taken out of its current community and assigned to another neighboring community. The gain ΔH depends on the chosen cost function, but for example, the effect of a switch of node i from community c_1 to community c_2, denoted $c_1 \to i \to c_2$, for the modularity is given by

$$\begin{aligned} \Delta H_Q(c_1 \to i \to c_2) = {} & W(i, c_2) + W(c_2, i) - W(i, c_1) - W(c_1, i) \\ & - \frac{s_i^{out}}{m}\left(S_{c_2}^{in} - S_{c_1}^{in} + s_i^{in}\right) - \frac{s_i^{in}}{m}\left(S_{c_2}^{out} - S_{c_1}^{out} + s_i^{out}\right) \end{aligned} \tag{3.26}$$

where $W(i,c)$ represents the sum of the edges from node i to nodes in community c (apart from the self loop of node i) and $S_c^{out/in}$ is the sum of the outgoing/incoming strengths of nodes in community c,

$$W(i,c) = \sum_{\substack{j \in c \\ j \neq i}} W(i,j), \tag{3.27}$$

$$S_c^{out/in} = \sum_{j \in c} s_j^{out/in}. \tag{3.28}$$

The second type of transition is more global and consists of either a merge of 2 communities or a split of a community into 2 distinct sets. Let us again illustrate those transitions using modularity for 2 communities c_1 and c_2. The gain of a merge, denoted $c_1 \cup c_2$, can be computed as

$$\Delta H_Q(c_1 \cup c_2) = W(c_1,c_2) + W(c_2,c_1) - \frac{1}{m} S_{c_1}^{out} S_{c_2}^{in} - \frac{1}{m} S_{c_1}^{in} S_{c_2}^{out} \tag{3.29}$$

where

$$W(c_1,c_2) = \sum_{\substack{i \in c_1 \\ j \in c_2}} W(i,j).$$

The splits can be carried in different manners, for example via spectral bisectioning or constrained simulated annealing of the subgraph induced by the community. The gain for a split of a community c into 2 communities c_1 and c_2, denoted $c \to \{c_1, c_2\}$, can be computed as the opposite of the gain to merge c_1 and c_2

$$\Delta H\left(c \to \{c_1,c_2\}\right) = -\Delta H(c_1 \cup c_2) \tag{3.30}$$

It has been shown that applying global transitions leads to better approximations of the optimum than using only local moves. Typically, in practical applications, one considers n^2 local moves and n global moves before updating the temperature of the system. However, the complexity of the algorithm can not be estimated and depends on the initial temperature and the cooling process. Though, the algorithm is typically very slow and should not be applied to graphs with more than 10^4 nodes.

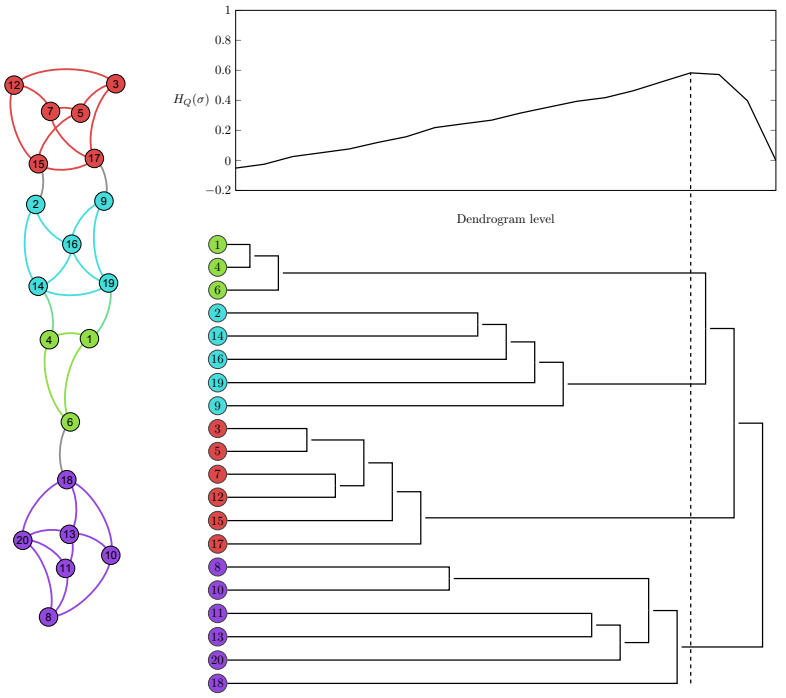


Fig. 3.4 Input graph and dendrogram created by Newman's algorithm

3.3.3 Newman

After introducing the modularity as a quantitative criterion for the quality of a community partition, Newman proposed to optimize this measure using a greedy hierarchical agglomerative algorithm [Newman (2004b)], although this algorithm can also be applied to optimize other additive cost functions.

Newman's algorithm starts with an initial partition with as many communities as nodes in the network, i.e. each node defines its own community. Then, communities are repeatedly joined in pairs. At each step, the "join" that provides that largest increase or the smaller decrease of the cost function is chosen among all possible joins. Since the network contains initially n communities and at each step 2 communities are merged, reducing the number of communities by 1, the maximum number of joins is $n - 1$. This can be represented by a dendrogram, i.e. a tree showing the order of each join, as illustrated in Fig. 3.4. A cut through this den-

drogram defines a community partition for which one can compute the associated value of the fitness measure. Therefore, the best partition can be selected among all the levels of the dendrogram, which is represented by the dashed vertical line in Fig. 3.4.

At each iteration, the algorithm must choose the best possible join. Yet merging a pair of communities between which there are no edges can never improve the value of the cost function. Hence, the algorithm only considers pairs of communities that are connected by at least one edge, so in the worst case, it needs to check for $O(m)$ possible joins. After a join, the algorithm must update a matrix of the fraction of edges between each communities by summing the corresponding rows and columns, which can be done in $O(n)$ flops. Finally, each join requires $O(m+n)$ flops in the worst case. Since there are at most $n-1$ iterations, the complexity of the Newman's algorithm is $O((m+n)n)$ or $O(n^2)$ for a sparse graph, which is similar to the complexity of spectral optimization.

Fast modularity

Clauset et al. have observed that the implementation of Newman's algorithm is very inefficient for sparse graphs and often wastes both time and memory space for the storage and the operations on the communities connectivity matrix whose elements are 0 in the vast majority, at least during the first iterations [Clauset, Newman, and Moore (2004)]. They did not describe a new algorithm but rather improved the implementation of Newman's original algorithm by using much more efficient data structures called max-heap. A heap is a tree-based data structure with a key for each node in the tree and an order relation on those keys. A heap is a max-heap if the key of a parent node is always greater than the keys of its children. Obviously, the largest key in a max-heap is always the root node but note that there is no specific ordering between siblings or cousins in the tree.

Instead of using the adjacency matrix, the algorithm of Clauset et al. is based on 3 data structures that are maintained during the whole process:

1. a masked matrix M containing the variation of the cost function when 2 communities (or nodes during the first iterations) i and j are joined:

$$M(i,j) = \begin{cases} \Delta H(i \cup j) & \text{if } A(i,j) \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

Each row of this matrix is stored as a balanced binary max-heap, such that an element can be found, deleted or inserted in $O(\log n)$ time and the maximum element can be found in constant time.

2. a max-heap L containing the largest element of each row of M as well as the indices of the corresponding communities.
3. a vector K containing the scaled degrees $K_i = \frac{k_i}{2m}$, or the scaled strengths for weighted networks. This is required to optimize the modularity, though for a directed network one would need 2 vectors $K^{(out)}$ and $K^{(in)}$. Moreover, one needs to store the community sizes in the vector K to optimize one of the resolution free models that we introduced in Section 3.2.3.

The algorithm can then be summarized as follows

1. Compute the initial matrix M and store it as a balanced binary max-heap for each row. Compute the elements of the vector K for each node. Populate the max-heap L with the largest element on each row of M. Set H to its initial value.
2. Select the largest element of M from L, join the corresponding communities and update M, L and K. Update $H = H + M(i, j)$.
3. Repeat step 2 until the largest element of L is negative. The value of H is the approximation of the maximum value of the cost function.

Note that the update rule imposes that as soon as the largest element of L is negative, all the elements of M are negative which in turn imposes that H can only decrease in consecutive iterations. So, the algorithm can stop earlier than when there is no more than 1 community in step 3. One can check that the update of the matrix M and the max-heap L takes $O((|i| + |j|) \log n)$ flops where i and j are the joined communities and $|i|$ denotes the number of neighboring communities of i, read its degree. Knowing that the total degree of the communities is $\sum_i |i| = 2m$, the worst case complexity is $O(md \log n)$ where d is the depth of the dendrogram. Typically, the observed depth of the dendrogram is $O(\log n)$ and the complexity of the algorithm is $O(m \log^2 n)$ or $O(n \log^2 n)$ for sparse network. This represents a huge improvement over the original algorithm of Newman and this algorithm was the first to allow the analysis of large networks (around 10^6 nodes). Let us mention that some authors have pointed out that this algorithm might be biased and form large communities during the early steps at the expense of small existing communities.

In successive papers [Danon, Díaz-Guilera, and Arenas (2006); Wakita and Tsurumi (2007)], different modifications of this algorithm have been proposed, though, up to our knowledge, those specific algorithms have not been extended to weighted or directed networks and we will not discuss them further.

3.3.4 Schuetz & Caflisch

Schuetz & Caflisch have also observed the bias of the algorithm of Clauset et al. towards the formation of large communities during the early stages [Schuetz and Caflisch (2008)]. They proposed an alteration of the algorithm of Clauset et al. to promote the creation of multiple clusters at the same time, hence reducing the risk of condensation of the nodes into a few large communities. This task is carried out by allowing at most l independent joins in each level of the dendrogram. The qualifier "independent" for the joins refers to a "Touched-Community-Exclusion-Rule" (TCER): a join between 2 communities is accepted only if none of the 2 communities have been involved in a previous join at the current dendrogram level. Each level of the algorithm can be summarized as follows:

0. Initialize: mark all the communities as "not touched",
1. Select the best join $\Delta H(i \cup j)$ over all remaining connected pairs of communities (i, j) where neither i nor j is marked as "touched",
2. If the gain $\Delta H(i \cup j) > 0$, apply the join and tag both communities i and j as touched. Otherwise, discard the pair (i, j),
3. Repeat steps 1 and 2 until l joins have been applied, or the best gain is negative $\Delta H(i \cup j) < 0$, or all the communities are tagged as touched.

While multiple clusters grow at the same time, the TCER imposes that each community can not be part of more than one join per level, hence this rule prevents the aggregation into a few large communities. Furthermore, it guarantees that the variation of the cost function can be computed as the sum of the gains of each accepted join because those are independent. Finally, the TCER ensures that the cost function can only increase at each level of aggregation since all the accepted joins have an associated positive gain $\Delta H > 0$. Note that even if $\Delta H(i \cup j) > 0$ and $\Delta H(j \cup k) > 0$, there is no guarantee that $\Delta H(i \cup j \cup k) > 0$.

Additionally, the algorithm of Schuetz & Caflisch further improves the quality of the partition with a refinement step called "vertex mover". The refinement step is applied on the partition obtained after convergence of the joining steps. All the nodes are sorted by increasing degree and sequentially reassigned to a neighboring community when that produces a positive gain, $\Delta H(c_1 \to i \to c_2) > 0$. Those reassignments occur until any individual vertex movement would decrease the quality of the partition.

The complexity of this algorithm is $O(md \log n)$ which is identical to the complexity of the implementation introduced by Clauset et al. Indeed, both algorithms are based on the same data structures and the same kind of operations. However, the algorithm of Schuetz & Caflisch does not require the creation and maintenance of the max-heap L, but on the contrary it needs to compute $O(l)$ pairwise gains other than the maximum which leads in general to a larger computational cost in practice. The complexity of a single vertex mover step is $O(m)$ since all edges have to be considered to compute the different neighboring gains. However, the overall complexity can not be estimated since it depends on the number of vertex mover steps required to reach local convergence. But, the authors claim that the running time of the vertex mover steps was always at least one order of magnitude smaller than the running time of the joining steps in all the example tested.

3.3.5 Label propagation

The label propagation algorithm has been introduced by [Raghavan, Albert, and Kumara (2007)] and even though it is not designed to optimize a particular cost function, it has been shown by [Tibély and Kertész (2008)] that this algorithm extracts a stable partition for the resolution-limit-free model defined by Eq. (3.17). In this framework, a partition is stable if there is no individual vertex switch from one community to another that increases the quality of the partition.

The label propagation algorithm starts in a configuration where each node has a distinct label. Communities are defined during the whole process as sets of nodes with the same label, hence the algorithm is initialized with a partition in which each node defines its own community. Then, the labels propagate through the network using a simple updating rule: a node is selected and its label is updated to the label shared by most of its neighbors. When there is a tie, one label is chosen uniformly at random. As the labels propagate, some of them will disappear while others will

dominate inside groups of highly connected nodes. The node selection is carried out by going through a randomly shuffled vector containing all the node indices. In one iteration, all the nodes have their labels updated at most once, then a new randomly shuffled index vector is created for the following iteration. The updating of the node labels can be performed either

- synchronously: at iteration t, a node takes the label shared by most of its neighbors at time $t-1$,
- asynchronously: at iteration t, a node takes the label shared by most of its neighbors at time t for the nodes that have been already updated, and at time $t-1$ for the others.

The algorithm stops when each node has a label shared by most of its neighbors. Note that due to the random selection of labels when ties occur, it is often not possible to reach a stable labels configuration. The stopping criterion is similar to the definition of strong communities introduced by [Radicchi, Castellano, Cecconi et al. (2004)] as presented in Section 3.2. However, strong communities should contain nodes with more neighbors inside the community than outside it, while here communities contain nodes with more neighbors inside their community than in any other community.

Obviously, due to the randomness involved both in the nodes selection and in the tie breaking processes, multiple stable label configurations may be obtained from the same initial configuration (when some prior knowledge is available on the communities, one can choose to not label some nodes initially to favor the aggregation around specific center nodes). However, Raghavan et al. showed that different community partitions obtained for a single network are in general similar. Though, one can not choose easily a specific partition as the best one among all available partitions. Therefore, the proposed solutions are to either consider overlapping communities or to aggregate the different partitions to form a new overall partition. Assume that a node i takes the labels l_1 and l_2 in 2 different runs of the algorithm. Then, one way to aggregate the different partitions is to give a new label to node i as (l_1, l_2) and to identify the community of node i as the set of nodes with the same label (l_1, l_2). However, this solution tends to over-partition the network in many communities that become smaller as the number of individual runs aggregated is large.

Each iteration of the label propagation algorithm takes a linear time

$O(m)$, however the overall complexity of the method depends on the number of iterations which can not be estimated but has been observed to stay small in many examples.

3.3.6 Louvain Method

The Louvain method is a greedy hierarchical clustering algorithm introduced by [Blondel, Guillaume, Lambiotte et al. (2008)]. This algorithm has become extremely popular in recent years because it is fast, allowing to analyze huge networks with billions of edges [Haynes and Perisic (2009)], and produces significant partitions [Lancichinetti and Fortunato (2009b)]. For example, this algorithm is now implemented within the famous business social network *LinkedIn*[1] and allows everyone to visualize the relationships between colleagues or customers and to better comprehend its professional network.

The algorithm is divided in two phases applied recursively as depicted in Fig. 3.5. The first phase is the *optimization* phase which looks for a locally optimal partition by considering only individual vertex swaps. This phase of the algorithm is similar to the vertex mover refinement step of the algorithm of Schuetz & Caflisch in Section 3.3.4. As in many other greedy algorithms, the community partition is initialized with a single node per community and as many communities as nodes in the input network. Then, based on the chosen cost function, individual nodes are removed from their current community and swapped to the neighboring community which produces the largest positive gain of the cost function. For example, if we consider for a potential switch a node i in a community c_1 then its new community c_2 is chosen such that

$$c_2 = \arg\max_c \Delta H\big(c_1 \to i \to c\big) \tag{3.32}$$

where $\Delta H\big(c_1 \to i \to c\big)$ is given by Eq. (3.26) for the modularity and by

$$\begin{aligned}\Delta H_{CPM}\big(c_1 \to i \to c\big) =& W(i,c) + W(c,i) - W(i,c_1) \\ & - W(c_1,i) - 2\gamma_{CPM}\left(n_c - n_{c_1}\right)\end{aligned} \tag{3.33}$$

for the constant Potts model where n_c is the number of nodes in community c. Similar formulations can be obtained for the other cost functions

[1] http://inmaps.linkedinlabs.com/

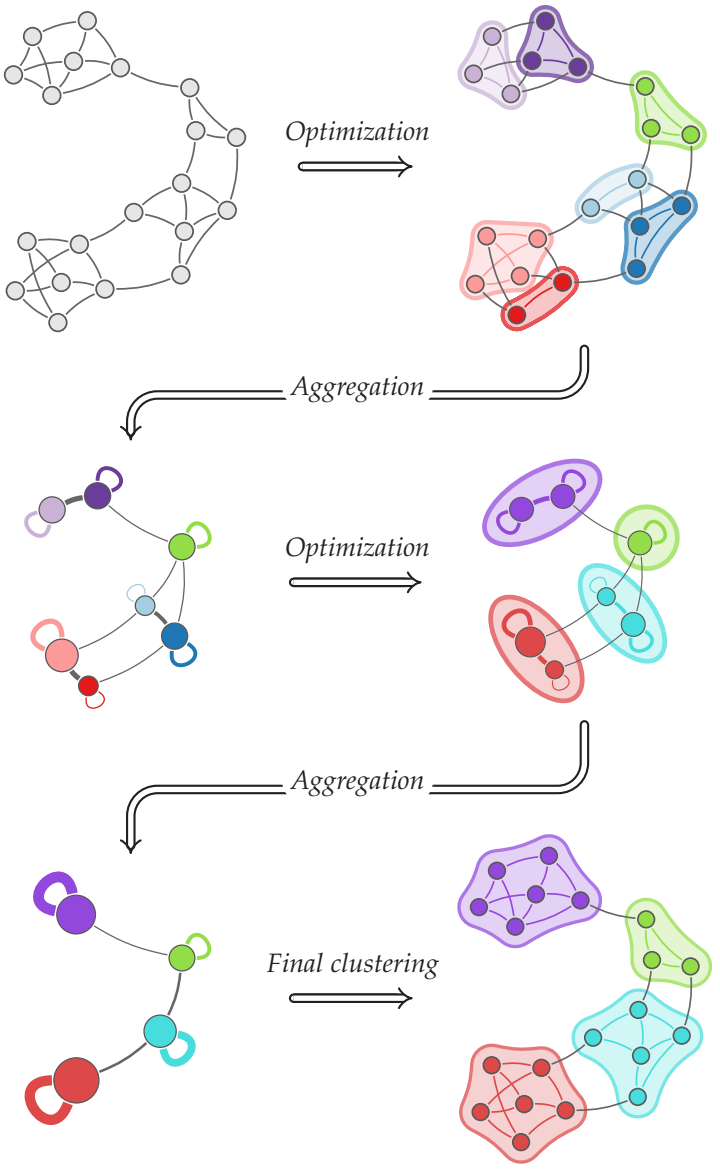


Fig. 3.5 Optimization and aggregation steps of the Louvain method.

presented in the thesis.

If none of the potential switches to a neighboring community produces

a positive gain, that is $c_2 = c_1$ and $\Delta H\big(c_1 \to i \to c_2\big) = 0$, then the node either stays in community c_1 if $\Delta H\big(c_1 \to i \to \{\}\big) < 0$, or otherwise is removed from c_1 and creates a new community.

The nodes are potentially switched in a random order. Each node is considered once before a new random order of the nodes is created. When no more local correction with a non negative gain can be applied to increase the value of the cost function, the algorithm initiates the second phase which is the innovative idea introduced by Blondel et al.

The second phase of the algorithm is the *aggregation* phase in which the input graph is collapsed according to the communities found in the first phase. This creates a new network where each single node corresponds to a community in the previous graph as depicted between two consecutive rows of Fig. 3.5. The purpose of this aggregation phase is to reapply the optimization phase on the collapsed graph such that one can consider additional improvements of the cost function by clustering groups of nodes rather than individual nodes.

However, the cost function must still be optimized with respect to the original network. This imposes that the creation of a community of "super"-vertices in the aggregated graph should be equivalent to clustering all the nodes of the associated communities in the original network. Let us denote by c_1 and c_2 two communities in the original graph $G(V, E, W)$ which become "super"-vertices in the aggregated graph $G'(V', E', W')$. Then, for the modularity, one needs to impose that

$$\sum_{\substack{i\in c_1\\ j\in c_2}} \left[W(i,j) - \frac{s_i^{out} s_j^{in}}{m_w} \right] = W'(c1, c2) - \frac{s_{c_1}'^{out} s_{c_2}'^{in}}{m_w'} \tag{3.34}$$

which naturally leads to define an edge between 2 nodes in the aggregated graph as the sum of the edges between the 2 associated communities in the original graph,

$$W'(c_1, c_2) = \sum_{\substack{i\in c_1\\ j\in c_2}} W(i,j). \tag{3.35}$$

A nice property of the configuration null model is that by defining the "super"-edges as in Eq. (3.35), the aggregated graph can be analyzed independently of the original graph and Eq. (3.34) is automatically satisfied.

Indeed,

$$s_{c_1}^{\prime out} = \sum_{c \in V'} W'(c_1, c) = \sum_{c \in V'} \sum_{\substack{i \in c_1 \\ j \in c}} W(i,j),$$
$$= \sum_{i \in c_1} \sum_{c \in V'} \sum_{j \in c} W(i,j) = \sum_{i \in c_1} \sum_{j \in V} W(i,j) = \sum_{i \in c_1} s_i^{out},$$

the same property can be obtained for the internal strength s_j^{in} and since

$$m'_w = \sum_{c_1, c_2 \in V'} W'(c_1, c2) = \sum_{c_1, c_2 \in V'} \sum_{\substack{i \in c_1 \\ j \in c_2}} W(i,j),$$
$$= \sum_{c_1 \in V'} \sum_{i \in c_1} \sum_{c_2 \in V'} \sum_{j \in c_2} W(i,j) = \sum_{i \in V} \sum_{j \in V} W(i,j) = m_w,$$

this implies that Eq. (3.34) is satisfied.

Unfortunately, this does not hold for the constant Potts model since the condition of Eq. (3.34) becomes

$$\sum_{\substack{i \in c_1 \\ j \in c_2}} [W(i,j)] - \gamma_{CPM} n_1 n_2 = W'(c1, c2) - \gamma_{CPM} \tag{3.36}$$

where n_1 and n_2 are the number of nodes in communities c_1 and c_2 respectively. However, we can slightly modify the definition of the cost function as

$$H_{CPM} = \sum_{i,j \in V} \Big[W(i,j) - \gamma_{CPM} N_i N_j \Big] \tag{3.37}$$

where N_i is the node size of node i. This form is equivalent to the original definition if we initialize the node size to 1 for every node in the original network. Then, the node size of a node in the aggregated graph must be computed as the sum of the node size of the nodes in the associated community,

$$N'_c = \sum_{i \in c} N_i$$

and the condition becomes

$$\sum_{\substack{i \in c_1 \\ j \in c_2}} [W(i,j)] - \gamma_{CPM} n_1 n_2 = W'(c1, c2) - \gamma_{CPM} N_{c_1} N_{c_2}$$

which naturally holds.

The two phases of the Louvain method are recursively applied until no cluster increasing the value of the cost function can be found in the last aggregated graph. In other words, the stopping criterion is that the optimal partition of the last aggregated graph, on the left hand side of Fig. 3.5, is to keep each "super"-vertex (the nodes of the aggregated graph) alone in its own community. This algorithm produces hierarchical levels for the communities, i.e. the partition at a specific scale is defined by merging communities of the partition at a lower scale. This allows to analyze a network at various resolutions, from a coarse-grained resolution in the last aggregated network to a fine-grained resolution in the early steps.

This algorithm has proven to produce good community structures. Its complexity is expected to be $O(n \log n)$ but a precise complexity analysis is still lacking due to the hardness to describe a priori the number of corrections through all the nodes in the optimization phase. Although the algorithm is fast in practice, the sequential corrections slow it down and make it very hard to parallelize on a multiple core architecture which in turn reduces the possible applications to very large networks. There has been some attempts to parallelize the Louvain method, either using naive optimization strategies or complex data structures and shared memory [Staudt and Meyerhenke (2013); Bhowmick and Srinivasan (2013)], but the quality of the extracted partitions tends to be much smaller than using the original algorithm due to the lack of local information about the graph topology within some threads. In Section 3.4, we will present a brand new algorithm which is inspired by the Louvain method in the fact that it uses the same aggregation step, but is faster while producing communities of similar quality and being highly parallelizable.

3.3.7 Infomap

The Infomap is the optimization strategy introduced by Rosvall & Bergstrom to maximize the value of the map equation which is given by Eq. (3.23) [Rosvall and Bergstrom (2010)]. Rather than a dedicated algorithm, the Infomap consists in a collection of algorithms applied to optimize the measure.

First, one needs to compute the stationary node visit frequency π

which can be done by applying a power method to the matrix

$$L = (1 - \tau)S^{-1}W + \frac{\tau}{n}\mathbf{1}\mathbf{1}^{\mathbf{T}} \tag{3.38}$$

where τ is the teleportation probability. This teleportation probability guarantees the existence of a unique steady state distribution π. Indeed, the induced Markov chain is aperiodic and irreducible, hence by the Perron-Frobenius theorem, the steady state distribution is unique.

Then, to optimize the map equation, the Infomap applies the algorithm of Clauset et al. , presented in Section 3.3.3, by repeatedly merging pairs of communities which produce the largest decrease in the expected description length of the random walker. After convergence, the partition is refined using simulated annealing, as presented in Section 3.3.2. Using the same initial partition, the simulated annealing algorithm is initialized at different temperatures and the run that produces the shortest description length, i.e. the minimal value of the map equation, is selected.

In a subsequent paper [Rosvall and Bergstrom (2011)], a multilevel hierarchical extension of the map equation has been developed. In this case, Rosvall & Bergstrom observed that the Louvain method was more appropriate to optimize the measure and leads to better optima. Again, the partition obtained after convergence is refined by considering both submodules movements and individual node movements until the value of the cost function can not be improved.

The complexity of the Infomap is directly tied to the form of the cost function (two-levels or multilevel) and to the algorithms used. Moreover, the map equation is not an additive cost function. Hence, computing the variation when merging 2 communities is much more costly than for the other cost functions which increases the computational complexity of the method.

3.4 Fast community extraction

We will now present a new algorithm that we developed purposely for community detection in very large graphs. Our algorithm is somehow at the crossroads between the algorithm of Schuetz & Caflisch (SC), the Label propagation method (LP) and the Louvain method (LM). The general idea is to build several communities at the same time (SC), by updating the nodes assignments synchronously (LP), and to iterate on those nodes assignments until the cost function reaches a local optimum (LP-LM) be-

Algorithm 1 Fast community extraction

Input : a graph $G(V, E)$
Output : a community partition matrix $C \in \mathbb{R}^{k \times n}$

Initialize $C = I_n,\ C_t = 0,\ G_t = G$
▷ C final partition, C_t partition at level t for G_t
while $C_t \neq I$ **do**
  $C_t \leftarrow \textsc{Assign}(G_t)$ ▷ see Section 3.4.1
  $C_t \leftarrow \textsc{Positive}(C_t, G_t)$ ▷ see Section 3.4.2
  while $\exists\, i \in V_t, c \in C_t$ with $\Delta H\,(c_i \to i \to c) > 0$ **do**
    $C_t \leftarrow \textsc{Maximal}(C_t, G_t)$ ▷ see Section 3.4.4
    $C_t \leftarrow \textsc{Positive}(C_t, G_t)$
  $G_t \leftarrow \textsc{Aggregate}(G_t, C_t)$ ▷ Aggregation step (LM)
  $C = C_t C$ ▷ Aggregated communities from partition C_t

fore applying an aggregation of the network based on the community partition (LM).

But, we want to maintain as much as possible the key property of each of those algorithms. First, if one has access to several processors, it is convenient to use all the computational power available to speed up the execution, which is the main advantage of the synchronous LP. However this algorithm can not be used to optimize a specific cost function. On the other hand, LM can be applied to any cost function (preferably additive) but its updating rule makes it really hard to parallelize. Indeed, consider that all the nodes may switch of community at the same time and assume that a node i wants to plug into a community c, that is $\Delta H(i \to c) > 0$. Two problematic situations may arise. First, others nodes may join community c, which becomes $c \cup \{i_1, \dots i_k\}$, and this may render the switch of i suboptimal or even decrease the quality of the partition if $\Delta H\,(i \to c \cup \{i_1, \dots i_k\}) < 0$. What is even worse is that the community c can be split because some of its nodes were themselves switched, which we can denote with a slight abuse by $c \to \cup \left\{c_1, \dots, c_p\right\}$. In this case, the switch of node i can simply not be applied at all.

In the following sections, we will show how we can build communities while still allowing synchronous corrections of the nodes for any type of quality function. Our method can be summarized as presented in Algo. 1 for which each step will be detailed in a dedicated section, apart from the aggregation step which was explained in Section 3.3.6. Our approach to allow synchronized corrections is to assign each node to another node

Algorithm 2 Initial nodes assignments

function ASSIGN($G(V, E)$)
  for all $i \in V$ **do**
    $a(i) = \arg\max_j \Delta H(i \to j)$ ▷ Best neighbor assignment
  end for
  $T \leftarrow \text{graph}\big(V, \{(i, a(i))\ \forall i\}\big)$ ▷ Assignment graph
  $C_t \leftarrow WCC\,(T)$ ▷ Weakly Connected Components
  return C_t

rather than to a community, which is represented by the function ASSIGN(G). By doing so, we induce directed tree-like structures over the graph. In Section 3.4.1, we will show how those structures define communities and what their characteristics are. Then, we will present how we can apply corrections of the node assignments to improve the quality of the community partition. Two types of corrections are proposed. First, we will discussed in Section 3.4.2 the POSITIVE(C_t, G) correction step which imposes that the gain to assign each node to its community must be positive. Then in Section 3.4.4, we will present the MAXIMAL(C_t, G) correction step that imposes that the gain to assign each node to its community must be larger than the gain to assign it to any other community. Finally in Section 3.4.5, we will discuss the convergence of our algorithm.

3.4.1 Definition of communities

As we have just explained, when communities are defined as sets of vertices, one can not apply synchronous corrections of the nodes assignments because each community may cease to exist as a cohesive group. We propose to define communities by assigning each node to another node rather than to a community and, by doing so, we ensure that corrections can be applied synchronously. Indeed, if a node requires a switch of its assignment to a node, it will always be able to apply it whatever is the current state of the communities in the network since the nodes are permanent unlike the communities that are evolving.

To initialize the community structure, each node is assigned to its best neighbor based on a chosen cost function independently of the choice of the other nodes (see function ASSIGN(G) in Algo. 2), that is for all $i \in V$,

the assignment $a(i)$ is computed as

$$a(i) = \underset{j \in N(i)}{\arg\max}\, \Delta H\left(i \to \{j\}\right). \tag{3.39}$$

where $N(i)$ is the set of neighbors of i, i.e. $N(i) = \{j \mid (i,j) \in E\}$. For nodes having multiple neighbors that provide the best positive gain, one of them is chosen at random. It is easy to see that each node has a null gain to be assigned to itself since the community partition does not change in this case,

$$\Delta H\left(i \to \{i\}\right) = 0.$$

This implies that the gain to assign each node to its best neighbor must be positive and nodes without any neighbor that provides a positive gain will always be assigned to themselves, i.e.

$$\max_{j \in N(i)} \Delta H\left(i \to \{j\}\right) = 0 \quad \Rightarrow \quad a(i) = i.$$

As depicted in Fig. 3.6, the assignments of each node create a set of directed subgraphs, spanning the input graph, which we will call the assignment graph. We define the community partition as the set of weakly connected components of this assignment graph. For example, the assignment graph presented in Fig. 3.6 induces 3 communities highlighted by different colors. Throughout our algorithm, all the corrections will therefore be applied to the assignment graph which in turn will modify the community structure.

Each weakly connected component of the assignment graph is described by a directed tree-like structure. Indeed, one can see that a component with k nodes contains exactly k edges since every node has a single outgoing assignment. Therefore, each component is spanned by a directed tree with one additional edge, hence justifying the name "directed tree-like structure". Furthermore, it is easy to prove that such structure contains exactly 1 strongly connected component (SCC) in the form of a directed cycle.

Let us consider a community c induced by the weakly connected component of the assignment graph T_c. First, one needs to consider an undirected version of the connected component graph T_c^u, and as just stated this graph is a tree with exactly one additional edge. It is well known [West (2001)] that adding one edge to a tree creates exactly one cycle. Therefore, when one adds direction to the edges, the graph T_c can not

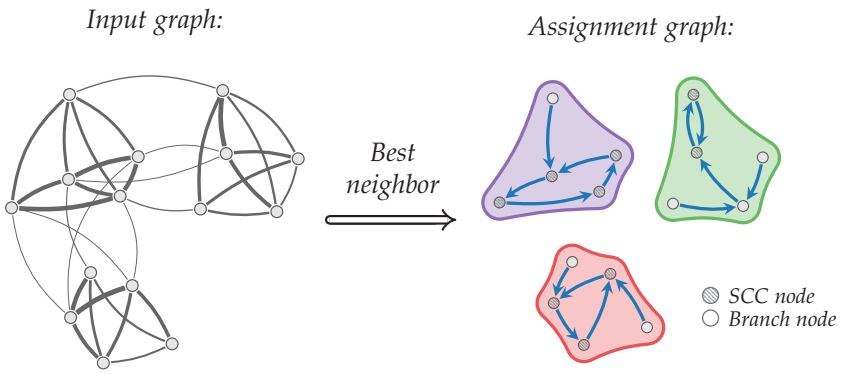


Fig. 3.6 Input graph & assignment graph defining the communities

contain more than 1 cycle. However this graph must also contain at least one cycle. Indeed, consider a sequence of nodes $s = \{i_1, \ldots, i_q\}$ such that i_{l+1} follows i_l if the assignment of i_l is i_{l+1}, $a(i_l) = i_{l+1}$. Now, consider the assignment of the node i_q. Either $a(i_q) \in s$ and our graph contains at least a cycle, or $a(i_q) \notin s$ and we can increase the sequence $s \to \{s, a(i_q)\}$. However, the number of nodes in T_c is finite and therefore the sequence can not become arbitrary long. Hence, after following a sufficient number of assignments, the sequence s can not be increased anymore and the last node q' must have its assignment in s which creates a cycle. Therefore, the directed tree-like structure spanning our communities contains exactly one strongly connected component (SCC) in the form of a directed cycle. The nodes composing those SCCs in each communities are represented by ◍ in Fig. 3.6 and in the subsequent figures.

The SCC can be seen as a set of connected roots defining the core of the community. The others nodes in T_c form what we will call the branches or the leaves for nodes with an incoming degree in T_c of 0. The nodes in the different branches are represented by ○ in Fig. 3.6 and the following. Obviously from the definition, there is a path from every branch node to any root node in the SCC and our directed tree-like structures are somehow reverse trees (for which paths are supposed to be from a root to the leaves). For a directed graph, the SCCs can be of any size but in an undirected graph, the SCCs are (in most cases) of size 1 (if $a(i) = i$) or 2. Indeed, consider a maximal sequence of nodes in a SCC $s = \{i_1, \ldots, i_q\}$ such that $a(i_l) = i_{l+1}$ and $a(i_q) = i_1$ and assume that $q > 2$. It is clear

from the assignment rule that

$$\Delta H\left(i_1 \to \{i_2\}\right) > 0,$$

and furthermore,

$$\Delta H\left(i_2 \to \{i_3\}\right) > \Delta H\left(i_1 \to \{i_2\}\right) > 0,$$

otherwise the assignment of i_2 would have been i_1 since G is undirected. Similarly,

$$\Delta H\left(i_q \to \{i_1\}\right) > \Delta H\left(i_{q-1} \to \{i_q\}\right) > \cdots > \Delta H\left(i_1 \to \{i_2\}\right) > 0$$

which is impossible because i_1 would have chosen i_q as its assignment. The only way to have a SCC of size larger than 2 for an undirected graph is that all the gains $\Delta H\left(i_p \to \{i_{p+1}\}\right)$ are equal for all p and the assignments, chosen at random in this case, are forming a loop as presented. This occurs very rarely and was never observed in our experiments for weighted networks.

The assignment of each node is clearly independent of the assignments of the other nodes and only depends on the graph G. Hence, we can easily parallelize this initialization of the communities and use as many processors as available. The assignment of each node can be computed in $O(k^{out})$, so the complexity of Assign(G) is $O(m)$.

Even if each node had chosen the neighbor that provides the largest local gain as defined by Eq. (3.39), the synchronicity of the assignments might produce communities in which the presence of some nodes decreases the value of the cost function. Formally, one can easily build a graph such that there exist 3 nodes i,j and k with $\Delta H\left(i \to \{j\}\right) > 0$ and $\Delta H\left(j \to \{k\}\right) > 0$, while $\Delta H\left(i \to \{j,k\}\right) < 0$. This implies that some correction steps should be applied to increase the overall quality of the community partition. Our algorithm considers two types of corrections applied recursively until a local optimum has been reached. We call them the Positive and the Maximal corrections as presented in Algo. 1 and we will discuss each of them in the next sections. We will also explain how the communities should be stored in memory based on the assignment graph such that the corrections can be applied efficiently.

Algorithm 3 Positive correction

function POSITIVE($C_t, G(V, E)$)
  for all $i \in V$ **do**
    $g(i) = -\Delta H\left(c_i \to i \to \{\}\right)$ ▷ Local gain
  while $\exists i \in c_i$ with $g(i) < 0$ **do**
    $c_1, c_2 \leftarrow$ SPLIT(c_i) ▷ Optimal assignment graph bisection
    for all $j \in c_1 \cup c_2$ **do**
      $g(j) = -\Delta H\left(c_j \to j \to \{\}\right).$ ▷ Update local gain
    $C_t = C_t \setminus \{c_i\} \cup \{c_1, c_2\}$ ▷ Update communities
  return C_t

3.4.2 Positive correction

The first type of correction is designed to split communities by assuming that every node should contribute positively to the fitness measure. We define the local gain $g(i)$ of a node i as the gain to assign the node to its community, which can be computed as the opposite of the gain to remove i from its community c_i,

$$g(i) = -\Delta H\left(c_i \to i \to \{\}\right). \tag{3.40}$$

For obvious computational reasons, one can not check for all the possible divisions of all the existing communities. First, our algorithm analyzes only the communities that contain at least 1 node with a non positive local gain. Those communities are the best candidates to find a split that increases the cost function. Furthermore, we want to maintain the assignment graph that defines the communities and therefore, the corrections can only be applied by updating some node assignments. Hence, only a small number of divisions are applicable and consist of removing existing assignments. So, the POSITIVE correction step, as described in Algo. 3, recursively searches for the optimal bisection of communities containing a node with a non positive local gain, $g(i) < 0$, by removing at most 2 assignments.

Within a community, two different kinds of bisection that preserves the rest of the assignment graph can be considered. First, we can remove any single assignment within one of the directed branches as depicted in Fig. 3.7. The node whose assignment was removed is then considered self-assigned. This effectively bisects the community since there exists 2

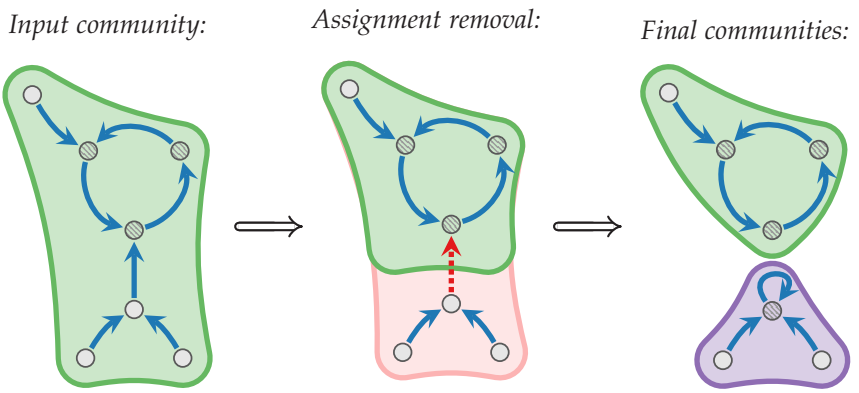


Fig. 3.7 Positive correction in a branch

SCCs after the removal of the assignment: the original SCC of the input community and a SCC of size 1 in the disconnected branch.

Another possibility is to remove assignments within the SCC of the input community. However, one can easily check that removing one assignment changes the structure of the SCC without actually splitting the community. Hence, we need to remove any pair of assignments within the SCC to bisect the community as depicted in Fig. 3.8. In this case, the 2 nodes, whose assignments were removed, are considered self-assigned, which eliminates the SCC of the input graph and creates 2 SCCs of size 1.

For every such bisection, we can easily compute the associated gain of the cost function as given by Eq. (3.30) and our algorithm simply selects the best among all the possible assignment(s) removal and applies the bisection. The local gain of each node within the input community is then updated based on the new community structure and the positive correction is reapplied if required (if one of the nodes still has a negative local gain).

A priori, the complexity of the positive correction step may seem pretty bad. To compute the bisection gain, one needs to know the total weight of the internal edges in the input community and the total weight of the edges between the 2 separated components. In the worst case scenario, if the input community has n_c nodes, one needs to check for $O(n_c)$ edges per node in one of the separated components. Since the positive correction analyzes the removal of every single assignment, the worst case is to have an assignment graph composed of one long chain of nodes, for

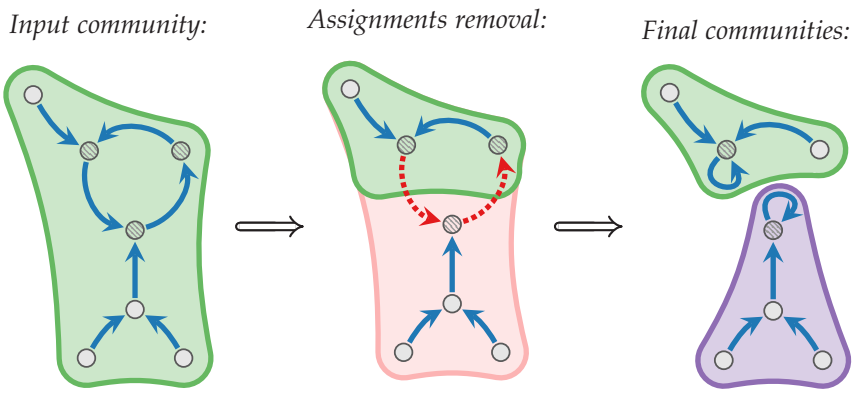


Fig. 3.8 Positive correction in a strongly connected component.

which the sum of the lengths of the branches is $O(n_c^2)$ and the complexity of the Postive function is $O(n_c^3)$. However, with an efficient storage of the communities based on the assignment graph, one can achieve a linear complexity in the number of edges within the community and a quadratic complexity in the number of nodes in the SCC, $O(m_c + n_{SCC}^2)$ or $O(n_c + n_{SCC}^2)$ for sparse graphs. The quadratic part does not increase dramatically the running time of the algorithm because the size of the SCCs is generally much smaller than the size of the communities. Finally, the positive corrections applied in a community are clearly independent of the state of all the other communities for additive cost functions, so each correction can be assigned to a different processor.

3.4.3 Storage of communities

To apply our corrections, we need to update efficiently the communities according to the modifications of the assignment graph. As we have just discussed for the positive correction step, we need to determine the set of nodes in a branch that remain connected to a particular node whose assignment is removed. Moreover, we need to remove those nodes from their community. Furthermore, in the following section, we will present the maximal correction step for which we will also need to insert branches in existing communities. This has led us to store the communities as doubly linked lists (DLL). Using this data structure, the insertion and the removal of a branch can be done in constant time and the computation of

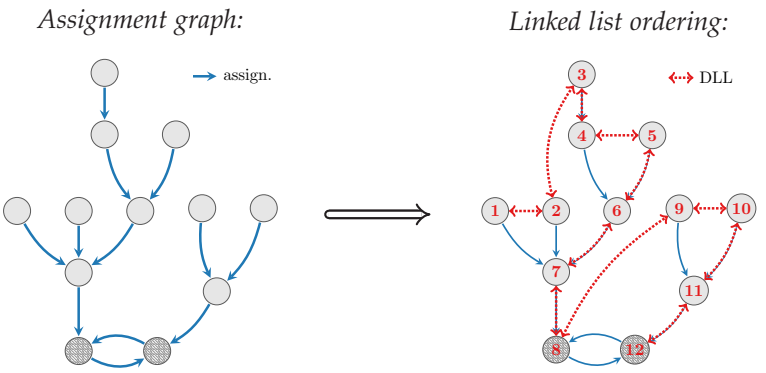


Fig. 3.9 Node ordering in the doubly linked list (DLL) to store a community.

the set of nodes in a branch can be done in linear time in the size of the branch.

An example of DLL representing a community is presented in Fig. 3.9 along with the assignment graph spanning the community. The purpose of the DLL is to explore entirely each branch while progressing through the assignment graph. More precisely, for any pair of nodes i and j in the same community, if $o(i)$ and $o(j)$ are the orders of the nodes in the DLL and $o(i) > o(j)$, then there must be no path from i to j in the assignment graph, unless i is in the SCC. Moreover, if k is the first node in the assignment graph such that there is a path from i to k

$$\mathcal{I} = \{i, i_1, \ldots, i_p, k\},$$

and from j to k

$$\mathcal{J} = \{j, j_1, \ldots, j_q, k\},$$

with $i_\alpha \neq j_\beta \; \forall \alpha, \beta$, then if $o(i) > o(j)$, the order of any nodes in $\mathcal{I}$ must be larger than the order of any nodes in $\mathcal{J}$,

$$o\left(i_\alpha\right) > o\left(i_\beta\right) \quad \forall \alpha, \beta.$$

Visually, the DLL starts the ordering from a leaf node (with an in-degree of 0 in the assignment graph). Then, we move down in the branch of this leaf by following the assignments until a node with strictly more than 1

branch is discovered (with an in-degree > 1 in the assignment graph). In this case, the DLL is pointed to another leaf in those branches and we iteratively explore all the branches with the same principle.

The insertion of a branch in an existing DLL can be done in constant time. Let $p(i)$ denote the predecessor of a node i in the list and $s(i)$ its successor and assume that we have to insert a branch $d_1 \leftrightarrow \cdots \leftrightarrow d_k$, also stored as a DLL, in an existing community because the node d_k has rewired its assignment to a node i of that community. The insertion can be done by

- updating the predecessor of node i such that it points to d_1

$$p(d_1) = p(i) \quad , \quad s(p(i)) = d_1,$$

- updating the links of node i such that its predecessor is node d_k

$$p(i) = d_k \quad , \quad s(d_k) = i,$$

which is done in constant time. The removal of a branch is done similarly by updating the predecessor of the first node in the branch and the successor of its last node.

Let us briefly illustrate how the DLL of Fig. 3.9 has been constructed. While the nodes labels in the figure correspond to the order of the nodes in the DLL, we will use them as regular labels for the sake of simplicity. Without loss of generality, we can assume that we start the DLL by the exploration of the branch of node ①. Following the edges in the assignment graph, we obtain a first DLL as

$$① \leftrightarrow ⑦ \leftrightarrow ⑧ \leftrightarrow ⑫$$

Then suppose that we find node ②. We observe that it has not been linked yet to any DLL. Since its assignment is node ⑦ which is already linked, using the update rule, the DLL becomes

$$① \leftrightarrow ② \leftrightarrow ⑦ \leftrightarrow ⑧ \leftrightarrow ⑫$$

Then for example, we may find node ⑤. By following the chain of assignments, we find the branch ⑤ $\leftrightarrow$ ⑥ and we end up again on node

⑦. The DLL becomes

$$① \leftrightarrow ② \leftrightarrow ⑤ \leftrightarrow ⑥ \leftrightarrow ⑦ \leftrightarrow ⑧ \leftrightarrow ⑫$$

and so on, until all the nodes have been assigned inside a DLL.

We already described different advantages of the DLLs to handle the modifications applied to the community structure through the assignment graph, however we did not mention that, in addition, it also allows to compute in linear time all the bisection gains when a positive correction step is required. As mentioned in Section 3.4.2, for each possible assignment removal, we need to compute the sum of the weights of the edges between the 2 disconnected sets. Using the ordering of the DLL and knowing the number of branches attached to each node, we only need to iterate once over each node in the community that undergoes a positive correction, as represented in Fig. 3.10. Each panel of the figure corresponds to the removal of 1 assignment in the community spanned by the assignment graph presented in Fig. 3.9. In each panel, we present the assignment graph of the community on the left hand side, with all assignments in blue, the removed assignment in red and the assignments of nodes in the removed branch in green. On the right hand side, we present a schematic view of the adjacency submatrix of the community. Even though we do not actually compute this submatrix, we believe that this representation adds clarity for the reader.

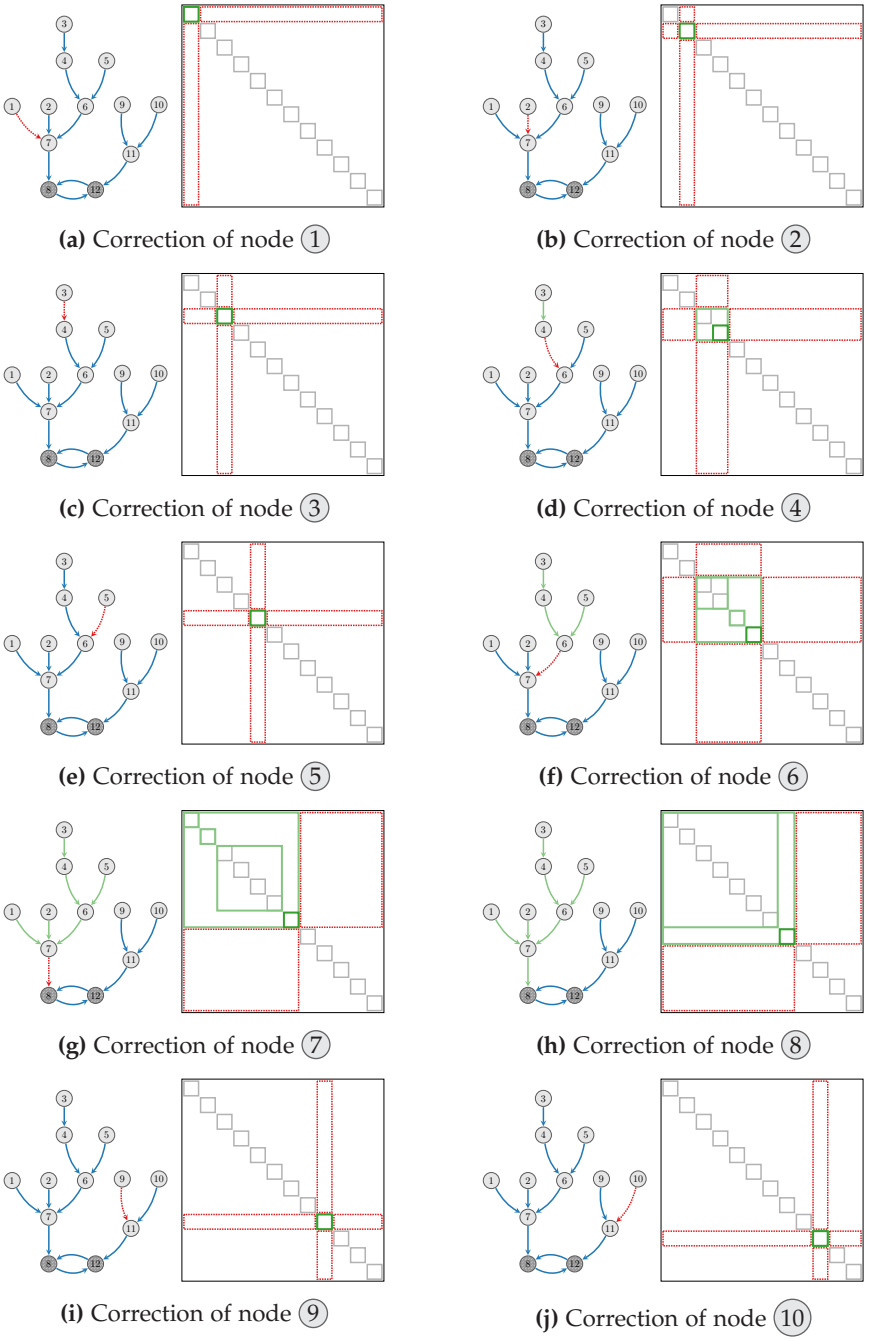
Panel (a) corresponds to the removal of the assignment of the first node. The gain of this removal is easily computed knowing the sum S_1 of the outgoing and incoming edges to all the other nodes in the community

$$S_1 = \sum_{j \in c} w(1, j) + w(j, 1),$$

as represented in red in the submatrix. Then, we store the stack S_1 to compute the effect of following removals and we define a first group $g_1 = \{1\}$. Similarly, panel (b) corresponds to the removal of the assignment of the second node for which we compute the associated gain using

$$S_2 = \sum_{j \in c} w(2, j) + w(j, 2),$$

and we define another group $g_2 = \{2\}$. However, thanks to the structure of the DLL, we know that the edges between node 1 and node 2 will never



(a) Correction of node 1

(b) Correction of node 2

(c) Correction of node 3

(d) Correction of node 4

(e) Correction of node 5

(f) Correction of node 6

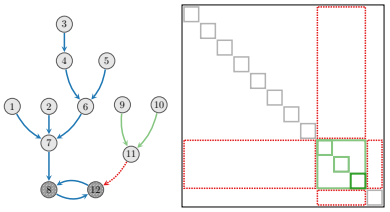
(g) Correction of node 7

(h) Correction of node 8

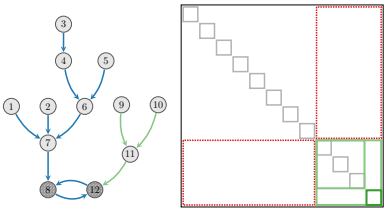
(i) Correction of node 9

(j) Correction of node 10

Fig. 3.10 Positive correction iterations through the assignment graph



(k) Correction of node (11)



(l) Correction of node (12)

Fig. 3.10(cont'd) Positive correction iterations through the assignment graph

connect separated components anymore. So, we can update the value of the stack S_1 accordingly

$$S_1 = S_1 - w(1,2) - w(2,1).$$

Likewise for panel (c) where the third node removes its assignment, we can compute the sum of the weight of the edges between the bisected sets, define a group and update the stacks of the previous nodes

$$S_3 = \sum_{j \in c} w(3,j) + w(j,3),$$

$$g_3 = \{3\},$$

$$S_1 = S_1 - w(1,3) - w(3,1) \quad , \quad S_2 = S_2 - w(2,3) - w(3,2).$$

Then, we reach node 4 that requires a peculiar approach as represented in panel (d). We know that, unlike the previous nodes, node 4 has 1 connected branch, so to compute the sum of the weight of the edges between the bisected sets, we do not compute a new stack and a new group but rather update the previous one. Since the last group was g_3, this leads to

$$\forall j < 4,\ S_j = S_j - w(j,4) - w(4,j),$$

$$S_3 = S_3 + \sum_{\substack{j \in c \\ j \neq 3}} w(4,j) + w(j,4),$$

$$g_3 = \{g_3, 4\} = \{3,4\}.$$

Note that the summation index does not consider the edges to node 3

since this node is part of the previous group. As previously, we create a stack S_4 and a group g_4 for node 5 which is a leaf, and update the previous stacks. When we reach node 6 in panel (f), knowing that it has 2 connected branches, we can compute the sum of the weight of the edges as the sum of the stacks of 2 previous groups (which were groups 4 and 3) and of its edges. In this case, all those nodes will form a cohesive group in the consecutive iterations, so we update the stack of the smallest group index in this branch and discard the other

$$\begin{gathered}\forall j < 6,\ S_j = S_j - w(j,6) - w(6,j),\\ S_3 = S_3 + S_4 + \sum_{\substack{j\in c\\ j\neq g(3),g(4)}} w(6,j) + w(j,6),\\ S_4 = 0, g_4 = \emptyset\\ g_3 = \{g_3, g_4, 6\} = \{3,4,5,6\}\end{gathered}$$

For a node i in the DLL, knowing the number b_i of branches attached to i and the number of groups n_g already created, the steps to compute the bisection gain can be summarized as follows

---

for all $j < i$ **do**
  $S_{g(j)} = S_{g(j)} - w(j,i) - w(i,j)$ ▷ update the previous stacks

$g_{n_g - b_i + 1} = g_{n_g - b_i + 1} \cup \cdots \cup g_{n_b} \cup \{i\}$ ▷ update the smaller index group

$S_{n_g - b_i + 1} = \sum_{l=n_g}^{n_g - b_i + 1} S_l + \sum_{\substack{j\in c\\ j\neq g_{n_g - b_i + 1}}} (w(i,j) + w(j,i))$

for $l = n_g$ **to** $n_g - b_i + 2$ **do**
  $S_l = 0, \quad g_l = \emptyset$

Using this iterative procedure for all the nodes in the community, we can compute all the bisection gains with a linear complexity in the number of edges in the community $O(m_c)$ since we use each edge exactly twice (once in each direction, outgoing and incoming).

Algorithm 4 Maximal correction

function MAXIMAL($C_t, G(V, E)$)
  $C = C_t$
  for all $i \in V$ **do**
    $c_i^* = \arg\max_c \Delta H\big(c_i \to i \to c\big)$ ▷ Best community for i
  for all $i \in V$, if $c_i^* \neq c_i$ **do**
    draw $p(i)$ uniform $\in [0, 1]$
    if $p(i) < p$ **then** ▷ See Section 3.4.5
      $b(i) = branch(i)$ ▷ Nodes in the branch of i
      if $\Delta H\big(c_i \to b(i) \to c_i^*\big) > 0$ **then**
        $a(i) = \arg\max_{j \in c_i^*} \Delta H(i \to j)$ ▷ Update assignment of i
        $C \leftarrow insert\ \big(b(i), c_i^*\big)$. ▷ Insert branch $b(i)$
  return C

3.4.4 Maximal correction

The positive correction described in Section 3.4.2 ensures that all the nodes have a positive contribution to the cost function, i.e.

$$g(i) \geq 0,\ \forall\ i \in V,$$

as given by Eq. (3.40). However, it might be possible to further improve the quality of the community partition by rewiring some of the node assignments, that is by switching some nodes from one community to another.

The second type of correction is designed to assign each node to the community which provides the largest gain based on the current partition. This MAXIMAL correction is summarized in Algo. 4. As in the Louvain method, one can compute the best community to allot a node i, currently in a community c_i, as

$$c_i^* = \arg\max_c \Delta H\big(c_i \to i \to c\big) \tag{3.41}$$

However, in our framework, communities are spanned by the assignment graph such that if a node switches its assignment to reach a new community, it imposes to all the other nodes in its connected branches to also change of community (without actually changing their assignments). This is depicted in Fig. 3.11 for a switch of a node in a branch and in Fig. 3.12 for a switch of a node in the SCC of a community. In the latter, one can

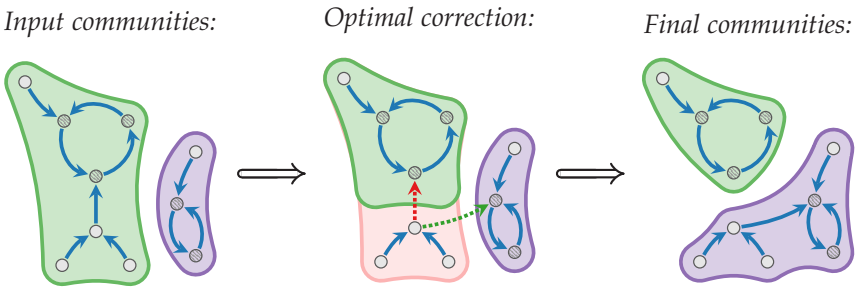


Fig. 3.11 Maximal correction in a branch

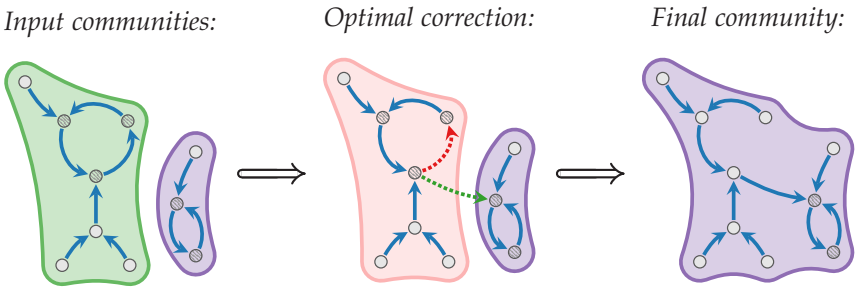


Fig. 3.12 Maximal correction in a strongly connected component.

observe that, since there exists a path in the assignment graph from any node in the community to any node in its SCC, when a node in the SCC switches its assignment, the SCC ceases to exist and the 2 communities are merged.

Hence, for every node with a positive switching gain, i.e.

$$\Delta H\left(c_i \to i \to c_i^*\right) > 0,$$

we first determine the set of nodes in its connected branches $b(i)$, i.e. the set of nodes with a directed path leading to i in the assignment graph. Then, we allow the switch only if the total gain $T(i)$ to switch all the nodes in $b(i)$ to c_i^* is positive. The total gain can be computed as

$$T(i) = \sum_{j \in b(i)} \Delta H\Big(c_i \to j \to \left\{c_i^* \cup b(i)\right\}\Big) \tag{3.42}$$

Finally, if the switch of a node is accepted, i.e. $T(i) > 0$, we update the assignment graph and insert the branch $b(i)$ in the DLL of c_i^*. The new assignment of the switching node is computed as the best single node assignment within the new community c_i^*,

$$a(i) = \arg\max_{j \in c_i^*} \Delta H(i \to j).$$

One can observe that the maximal correction does not require any other information than the current community partition to compute the potential switches of all the nodes. Furthermore, we apply all the maximal corrections synchronously, that is the best community for any node only depends on the input partition and is independent of the corrections applied to other nodes. The communities are updated after all the new assignments have been computed. This allows an efficient use of a parallel processor architecture where each core can handle a set of nodes independently.

The set of possible corrections for a node is constrained to its neighboring communities. Indeed, knowing that $\forall i\,, g(i) > 0$ from the positive correction, the maximum gain

$$g^*(i) = \max_c \Delta H\big(c_i \to i \to c\big)$$

can only be positive if the node shares at least one edge with the community c_i^*. Therefore, for each node, we only need to look for its neighboring communities and the complexity of the Maximal correction is linear in the number of edges $O\big(m\big)$.

Like for the initial assignment, the synchronicity of the maximal corrections can lead to communities with nodes having a negative local gain. Hence, after each maximal correction, we reapply a positive correction step to ensure the positivity of the gain of each node. It is worth noting that while the positive correction can only increase the number of communities, the maximal correction tends to decrease the number of communities by reassigning nodes in SCCs, e.g. Fig. 3.12. Hence, those 2 types of correction balance each other. This sequence of maximal correction followed by positive correction is repeatedly applied to the assignment graph until all the nodes are assigned to a community with a maximal non-negative gain.

When this stable partition is reached, the network is aggregated as in

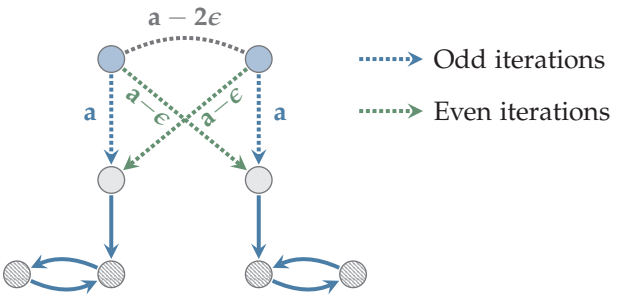


Fig. 3.13 Infinite maximal correction with mutually attractive nodes.

the Louvain method. Then, the exact same procedure is applied to this aggregated graph which provides another hierarchical level of clustering. The algorithm stops when it can not extract a community structure in the last aggregated graph, or in other words, when the optimal community matrix in the aggregated graph is the identity, i.e. the mutual assignment of any pair of "super"-nodes produces a negative gain. Unfortunately, the total complexity of the algorithm can not be evaluated since it depends on the number of maximal and positive corrections applied which can not be estimated.

3.4.5 Convergence

Let us conclude the discussion about our method by addressing some concerns about the convergence of the algorithm. One can observe that the positive correction step only increases the value of the cost function. Since the network is finite, the number of possible community partitions is also finite, hence the cost function is discrete and upper bounded, so the positive correction step converges. One can even bound the number of iterations per positive correction. We have seen that the positive correction removes existing assignment to bisect communities. This means that in a community with n_c nodes, we can apply at most $n_c - 1$ assignments removals. Hence, given a community partition, the number of iterations for one positive correction step is bounded by $\sum_c (n_c - 1)$.

However, we can not apply a similar reasoning for the maximal correction. Each individual modification of the assignments leads to a positive variation of the cost function because it may be accepted only if the total gain is positive $T(i) > 0$. However, all the assignments are corrected

at the same time and one can not guarantee that this leads to a positive variation of H. Indeed, the distribution of the weight of the edges might prevent this iterative procedure to converge. In some situations, mutually attractive or repulsive nodes would permanently switch their community assignments which produces an infinite sequence of maximal corrections. An example of such distribution of the weight of the edges is presented in Fig. 3.13 for which only the weights of the relevant edges are displayed. This assignment graph defines 2 communities for which we will analyze the assignments of the 2 top nodes ◯. Let us assume that the initial assignments of the nodes are the dashed blue edges, hence each node has an internal weight of a in its community. In this case, both ◯ nodes have a positive gain to switch community because each shares 2 edges with the other community (the gray edge of weight $a - 2\epsilon$ and the dashed green edge of weight $a - \epsilon$) and their total internal weight would become $2a - 3\epsilon > a$. Hence, the assignments of both nodes are rewired to the dashed green assignments since those are the largest single nodes assignment in the other community. But, due to the synchronicity of the rewiring step, the two ◯ nodes stay in different communities. In this configuration, both nodes have again a positive switching gain because their current internal weight has become $a - \epsilon$ (and not $2a - 3\epsilon$ as expected) while their total weight to the other community is $2a - 2\epsilon$ (from the gray and the dashed blue edges). Hence, both nodes switch back to the dashed blue assignments and the system will keep oscillating between those 2 states.

This kind of phenomenon also arises on consensus problem for synchronous multi-agent systems [Sarlette, Tuna, Blondel et al. (2008); Carli, Fagnani, Frasca et al. (2010)]. While various strategies can be applied, we designed our algorithm to accept each individual (maximal) correction with a probability $p < 1$ which ensures convergence. For instance, in the example provided in Fig. 3.13, if only one of the 2 assignment is swapped, both ◯ nodes would have a total internal weight ($2a - 3\epsilon$ and $2a - 2\epsilon$) larger than the external weight (a and $a - \epsilon$). This probability p somehow balances the quality of the clustering with the computational time required to meet a stable state and guarantees convergence. When p is small, only a few corrections are accepted and the cost function will mostly increase at each step. The algorithm will then converge quickly to a local optimum but will not explore much of the search space. On the other hand, when p is large, more corrections are accepted, potentially increasing the overall quality of the communities but also the number of

iterations required to reach convergence.

To observe why the introduction of this probability p ensures convergence, let each possible community partition be represented by one state. The partition at iteration $t+1$ only depends on the partition at iteration t, hence the sequence of corrections is a Markov process and correspond to a walk over the associated Markov chain of community partition states. If $p = 1$, all the maximal corrections are accepted and the Markov process is fully deterministic, i.e. from any starting state there is only one reachable state. We know that from any initial partition, after sufficiently many steps, we end up in a stationary class of the Markov chain, but one can not have any guarantee about the size of this class. If the stationary class is of size larger than 1, then the algorithm will not converge as presented in Fig. 3.13. When $p < 1$, the Markov process becomes stochastic. In this case, from any starting state, one could reach potentially many states, each corresponding to the acceptance of a subset of all the maximal corrections. Let us show that this guarantees that all the stationary classes are of size 1, thus the stationary classes are stationary states and the algorithm converges to one of those stable partitions.

To see this, first observe that there exist stationary states. Indeed, the community partition associated to the global optimum of the cost function can not be improved, hence no corrections could be applied to this partition and the associated state must be stationary. Moreover, for any cost function, there are often multiple local optima for which the associated states could be stationary.

Furthermore, from any initial state of the Markov chain, there exists a path leading to a stationary state. A valid strategy to identify such a path is to only accept one maximal correction per iteration. If we do so, the cost function increases at each iteration because the only corrections that can be accepted have a positive global gain $T(i) > 0$, as given by Eq. (3.42). Then, after a sufficient number of corrections, the cost function can not be further increased and reaches a local optimum. So, the state associated to this community partition is stationary.

This imposes that all the stationary classes of our Markov process are of size 1 as soon as $p < 1$. For any set of states of dimension larger than 1, there exists a path from each state in the set to a stationary state as we have just demonstrated. Hence the set can not be a stationary class.

Finally, it is well known that from any starting state, any random walk over a Markov chain always ends up in a stationary class with probability 1. This means that, if $p < 1$, from any initial partition, our sequence of

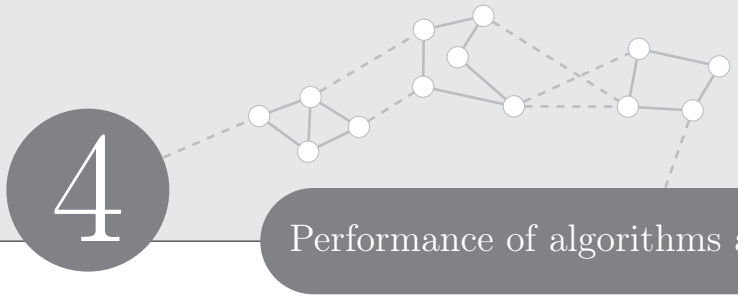
positive and maximal corrections converges with probability 1.

3.4.6 Concluding remarks

In this section, we have presented a new algorithm for community detection based on a particular structure that we called the assignment graph which is used to define the communities from its weakly connected components. We then introduced two types of local corrections of the assignment of the nodes (the positive and the maximal corrections) to greedily improve the value of the quality function and we demonstrated how those corrections can be efficiently applied to maintain the structure of the assignment graph. We showed that the corrections of the assignment of the nodes can be applied synchronously which makes the algorithm highly parallelizable. In the following chapter, we will see that another advantage of the assignment graph is that it allows our algorithm to converge faster (i.e. in less iterations over the entire set of nodes) than the Louvain method that is its main competitor in terms of computational time.

Nonetheless, other types of local (or potentially global) corrections could be introduced to enrich the algorithm but that would either increase the computational time to extract a community partition or prevent the algorithm to be efficiently run on a parallel architecture. Moreover, one could relax or even remove some of the checking rules to investigate a potential correction of a node assignment. For example, we only applied a positive correction step for communities containing a node with a negative local gain (Eq. (3.40)). Although, it might happen that the value of the cost function would still increase if some communities were split even if they do not contain such nodes. Allowing more potential corrections would naturally increase the computational cost of the algorithm but it could also increase its efficiency. Finally, we imposed our correction steps to result in strictly positive variation of the cost function. But, as for simulated annealing (see Section 3.3.2), one could consider additional explorations of the search space by allowing sometimes corrections leading to small negative variations of the cost function. Though, to guarantee the convergence, one will need to ensure at some point that all the corrections are strictly increasing the quality function and it is not sure that the assignment graph would not return to a previous state which will make the additional exploration useless. We let those investigations on possible modifications of the algorithm for future research and, in the following chapter, we will apply our community detection algorithm to synthetic

benchmark graphs to prove its efficiency and on graphs extracted from pictures or video frames as a segmentation method.



4 Performance of algorithms and applications

The popularity of an algorithm is often tied to two factors: the reputation of its authors and its performance. The former is rather arbitrary and, until recently, the efficiency of community detection algorithms was hardly compared besides their complexity. Scientists have never agreed on a set of benchmark networks with known community structures that could be shared and used as standard. The very few available real graphs, like the Zachary karate club or the bottlenose dolphins networks, are either too small or containing communities so strongly defined that all algorithms have a similar performance on those benchmark graphs. Obviously, there exist many large real networks that could be used to evaluate the quality of the different methods but, their "true" community partitions are not known a priori which makes any quantitative analysis difficult. Another problem is that, many real networks are built on data that belong to private companies and organizations, like mobile telecommunication exchanges between customers. Due to the sensitivity of those data, scientists can only acquire them under non-disclosure agreement, hence they cannot be shared between researchers.

This explains why so many greedy algorithms and heuristics have been developed, without being able to state which one was the best to use in a practical context. Hence, there have been many efforts recently to define appropriate models to create computer generated benchmark graphs designed to assess the quality of community detection algorithms. In the first section of this chapter, we will present the LFR model which has established itself as a reference in the field [Lancichinetti, Fortunato, and Radicchi (2008); Lancichinetti and Fortunato (2009a)]. We will thereafter apply all the algorithms that we presented in Chapter 3 to this model and thoroughly compare their results against various parameters settings. We will show that our method, described in Section 3.4, produces quantitatively similar results as some of the best available algorithms and that its main advantage is its computational speed which outperforms all the other methods. Hence, we strongly believe that, in the future, it will allow scientists to analyze huge networks or to process massive amounts of

data.

Then in Section 4.2, we will present the results of our algorithm applied to uncover the boundaries of objects of interest inside an input picture, which is known as image segmentation. We encountered this problem in the early development of our algorithm and it largely inspired the method by bringing out the needs for a fast and parallelizable algorithm for community detection. We will extend this framework for $3D$ segmentation and present some early results on video tracking in Section 4.2.3. Finally, in Section 4.2.4, we will show how our algorithm has been used in a practical application to extract the boundary of different materials in microstructural images. This allows to generate an accurate mesh over the polycristal microstructure to simulate its deformations.

4.1 LFR benchmark model

To analyze the efficiency of different community detection algorithms, one needs to apply them to networks for which the actual partitions are known and then compare their results with those true partitions. Since the true community partitions of real networks are rarely known a priori, it makes more sense to apply the different algorithms to computer generated benchmark graphs in which the exact partition can be defined beforehand. One can then quantitatively compare the outcome of each method and evaluate their performance. Naturally, if an algorithm failed to extract an accurate partition for computer generated benchmark graphs, it does not mean that this algorithm is bad in general but rather than it may fail to extract communities in networks that exhibit similar properties to the generating model of the applied benchmark. But, if the generating model is broad enough to simulate many real networks, one can reasonably extrapolate the results on computer generated benchmark graphs to real networks.

A particular class of random graphs has become popular after the works of Newman & Girvan [2002] and is based on the planted ℓ-partition model [Condon and Karp (2001)]. In this model, the graph is composed of l communities with v vertices each, so the total number of vertices is $n = l\ v$. The edges are added in the graph at random using two probability parameters. For any ordered pair of nodes (i, j), an edge from i to j is added with probability p_{in} if both nodes belong to the same community. Otherwise, if they belong to different communities, the edge is added with probability p_{out}. Therefore, the expected internal density of

each community is p_{in} and their expected external density is p_{out}. This implies that the communities are well defined as long as $p_{in} > p_{out}$ which defines a range of parameter values where the algorithms should be able to extract good community structures. In graphs created with the planted partition model, the expected internal and external degrees, respectively defined for each node as the expected number of incident edges directed within or outside its communities, are equal for all nodes and given by

$$\langle k_{int} \rangle = p_{in} (v - 1), \tag{4.1}$$
$$\langle k_{ext} \rangle = p_{out} (l - 1) v. \tag{4.2}$$

Newman & Girvan [2002] have considered a slightly different version of this model where they fixed the average degree to $\langle k \rangle$. Therefore, the probability parameters p_{in} and p_{out} are not independent and bound by the relation

$$\langle k \rangle = \langle k_{int} \rangle + \langle k_{ext} \rangle = p_{in} (v - 1) + p_{out} (l - 1) v. \tag{4.3}$$

It is then easier to work with a *topological mixing parameter* $\mu_T \in [0, 1]$ which sets, for each node, the expected proportion of edges directed outside of its community. Hence, this topological mixing parameter is just another way to control the strength of the communities in the random network. The expected internal and external degrees then read

$$\langle k_{int} \rangle = (1 - \mu_T) \langle k \rangle, \tag{4.4}$$
$$\langle k_{ext} \rangle = \mu_T \langle k \rangle. \tag{4.5}$$

from which the original probability parameters of the l-planted partition model can be recovered.

As in the original model, let us compute the range of values for the topological mixing parameter such that the communities are well defined. When $\mu_T = 0.5$, each node has on average as many neighbors inside and outside of its community, however this does not give the actual threshold. Indeed, if the network is large enough, the edges added outside of the communities will be widely spread across the graph and the predefined communities will remain the most densely connected clusters of nodes even for $\mu_T > 0.5$. More precisely, one can compute the expected internal

(resp. external) density $\langle e_{int} \rangle$ (resp. $\langle e_{ext} \rangle$) of each node as

$$\langle e_{int} \rangle = \frac{\langle k_{int} \rangle}{(v-1)} = \frac{(1-\mu_T)\langle k \rangle}{v-1},$$
$$\langle e_{ext} \rangle = \frac{\langle k_{ext} \rangle}{v\,(l-1)} = \frac{\mu_T \langle k \rangle}{v\,(l-1)}.$$

and the condition $p_{in} > p_{out}$ becomes

$$\mu_T < \frac{v\,(l-1)}{v\,l-1} \approx \frac{l-1}{l}.$$

Hence, as previously mentioned, the threshold for μ_T, such that the communities are well defined, grows with the number l of communities in the network, at least theoretically.

The model of Newman & Girvan has also been extended to weighted networks by [Fan, Li, Zhang et al. (2007)]. One can choose 2 values w_{int} and w_{ext} for the weight of the internal and external edges respectively, or choose the weight of the edges uniformly distributed around those values. We can again derive thresholds for the parameters μ_T and $R = \frac{\langle w_{ext} \rangle}{\langle w_{int} \rangle}$, the ratio between the expected weight of external and internal edges, and we obtain

$$\mu_T < \frac{v\,(l-1)}{R(v-1)+v\,(l-1)} \approx \frac{l-1}{l+(R-1)}. \tag{4.6}$$

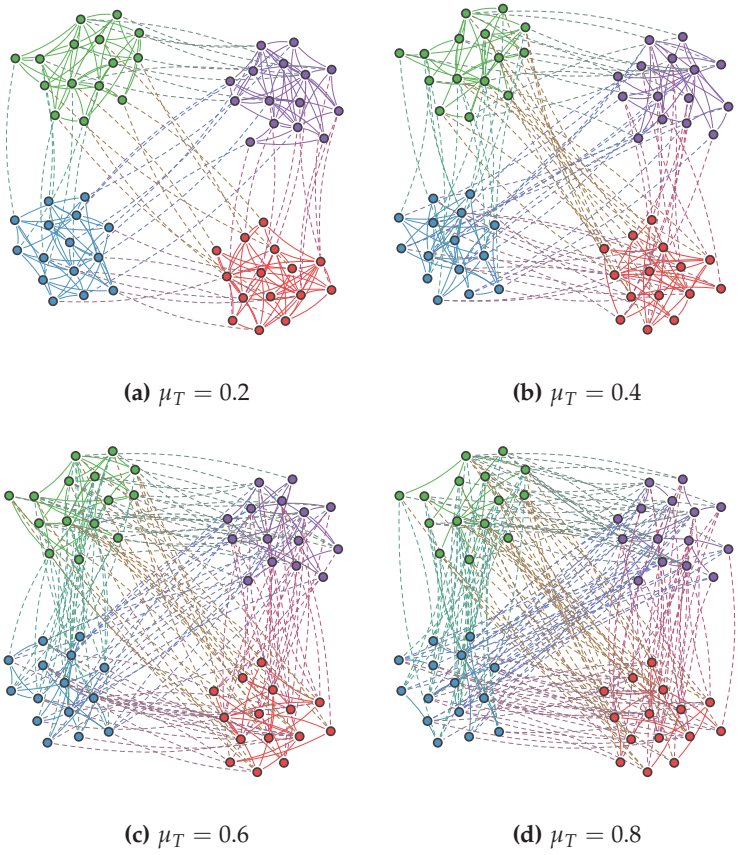
if $\frac{R}{v}$ is small. As expected, there is an interaction between μ_T and R. If $R \approx 1$, then $\langle w_{ext} \rangle \approx \langle w_{int} \rangle$ and the network is essentially unweighted so we recover the previous threshold value for μ_T. If $R \approx 0$, we get $\mu_T < 1$ which is always true of course but makes sense since the external edges carry almost no weight so whatever is the number of internal edges, their large weights maintain the cohesion of the community (however, we still need the communities to be connected). On the other hand, If $R \gg s1$, the communities are well defined only if there are many internal edges such that their total weight is large enough to compensate the weight of the external edges. However in this case, the approximation about $\frac{R}{v}$ might not hold so the threshold value is slightly different.

Although those models may be interesting to simulate some particular graphs, they are not appropriate to represent many real networks that contain a community structure. All the communities in the planted ℓ-partition model are spanned by Erdős-Rényi random graphs with pa-

rameter p_{in} and they are also interconnected by an Erdős-Rényi random graph with parameter p_{out}. Hence, the degree distribution is Poissonian with an average degree $\langle k \rangle$ as given by Eq. (4.3). However, as stated in Chapter 3.1, the degree distribution is often more heterogeneous in real networks and exhibits a power law distribution with nodes of very low and very high degrees coexisting in the graph. Moreover, the community size distribution tends to also follow a power law and the hypothesis of the planted partition model, that all communities are of equal size, is not realistic (though, the model have been extended by [Brandes, Gaertler, and Wagner (2003)] to create communities whose size is normally distributed). Finally, the edge weight distribution has also been observed to follow a power law [Barrat, Barthélemy, Pastor-Satorras et al. (2004)].

More recently, a model to create benchmark graphs containing those power law distributions has been proposed by [Lancichinetti, Fortunato, and Radicchi (2008)] and is now commonly called the LFR benchmark. First, each node is given a degree from a power law distribution with parameter τ_1, i.e. $k_i \backsim k^{-\tau_1}$, and with the average and maximal degree fixed. Then, the communities size is taken from a power law distribution with parameter τ_2, i.e. $n_c \backsim n^{-\tau_2}$, and with the minimal and maximal sizes fixed such that the total number of nodes is n. Again, the topological mixing parameter is used to control the expected internal and external densities of the nodes. Each node is assigned to a community, to match the drawn sizes sequence, and shares a fraction $(1 - \mu_T)$ of its edges with nodes in the same community, Eq. (4.4), using the configuration model described in Section 3.2.2 (if this is not possible, some nodes are shuffled between communities). Finally the outgoing edges are added such that each node shares on average a fraction μ_T of its edges with nodes in other communities, Eq. (4.5). Some realizations of the LFR benchmark are presented in Fig. 4.1 with $n = 60$ and $\langle k \rangle = 8$ for different value of μ_T (the exponent of the power laws where set by default to $\tau_1 = 2.5$ and $\tau_2 = 1.5$ which are often observed in the literature). All those graphs have the same expected number of edges. One can easily see how, when μ_T increases, the number of external edges grows and therefore the number of internal edges shrinks, hence the communities become loosely defined and obviously harder to extract for any algorithm.

The LFR benchmark has been extended to directed and weighted networks [Lancichinetti and Fortunato (2009a)]. For the latter, first an unweighted graph is created and then the weight of the edges is added to the network. To achieve this, one introduces a *weight mixing parameter*



(a) $\mu_T = 0.2$ **(b)** $\mu_T = 0.4$

(c) $\mu_T = 0.6$ **(d)** $\mu_T = 0.8$

Fig. 4.1 LFR benchmark graphs for different topological parameters μ_T

$\mu_W \in [0, 1]$ that sets the expected proportion of the strength of each node that is carried by its external edges. In the LFR model, the strength of the nodes is computed as a power of their degree, $s_i = k_i^\beta$, which was again chosen to match observations on real graphs [Barrat, Barthélemy, Pastor-Satorras et al. (2004)]. The internal and external strengths are given by

$$s_i^{int} = (1 - \mu_W)\, k_i^\beta,$$
$$s_i^{ext} = \mu_W k_i^\beta.$$

The actual weight of the edges is chosen to minimize the variance between

the expected and the observed strengths,

$$var\left[w_{ij}\right] = \sum_{i=1}^{n} \left(k_i^{\beta} - \rho_i\right)^2 + \left(\left(1-\mu_W\right) k_i^{\beta} - \rho_i^{int}\right)^2 + \left(\mu_W k_i^{\beta} - \rho_i^{ext}\right)^2$$

where ρ represents the actual strengths of the nodes,

$$\begin{aligned} \rho_i &= \sum W(i,j) \\ \rho_i^{int} &= \sum W(i,j)\delta\left(\sigma_i,\sigma_j\right), \\ \rho_i^{ext} &= \sum W(i,j)\left(1-\delta\left(\sigma_i,\sigma_j\right)\right). \end{aligned}$$

The optimization of $var\left[w_{ij}\right]$ is approximated by iteratively increasing or decreasing the weight of the edges. It has been observed that $var\left[w_{ij}\right]$ decreases exponentially with the number of iterations using this procedure. Hence, the complexity to create weighted networks with the LFR benchmark is equivalent to the complexity to create unweighted networks, which is $O(m)$. Unfortunately, the thresholds for the parameters μ_T and μ_W to have good communities are less clear in this model. Indeed, the notion of density is not well defined in weighted networks and, if we apply the same reasoning as in Eq. (4.6), then the internal and external densities are independent of μ_T and solely controlled by μ_W. However, one can still derive a weaker bound such that the expected weight of an internal edge is larger than the expected weight of an external edge. The expected edge weights are given by

$$\left\langle w^{int}\right\rangle = \frac{\left(1-\mu_W\right)\langle s\rangle}{\left(1-\mu_T\right)\langle k\rangle} = \frac{\left(1-\mu_W\right)}{\left(1-\mu_T\right)}\langle k\rangle^{\beta-1}, \tag{4.7}$$

$$\left\langle w^{ext}\right\rangle = \frac{\mu_W\langle s\rangle}{\mu_T\langle k\rangle} = \frac{\mu_W}{\mu_T}\langle k\rangle^{\beta-1}, \tag{4.8}$$

and the condition $\left\langle w^{int}\right\rangle \geq \left\langle w^{ext}\right\rangle$ simply becomes

$$\mu_T \geq \mu_W.$$

Measuring the quality of a partition

To compare the results of each algorithm, we need a quantitative criterion to measure how close are the extracted partitions to the true partition of the benchmark. Several measures have been proposed in the literature

[Meilă (2007)] based for example on the number of correctly classified vertices or edges, like the Rand index or the Jaccard index, but the most commonly used measures in community detection are based on information theory as introduced in Section 3.2.4. The idea is to quantify how much information about the true partition one can infer from the extracted partition. To do so, we need to introduce two additional measures of entropy. First, the joint entropy measures the uncertainty of the joint probability distribution $p(x,y)$ to observe $X = x$ and $Y = y$ and is given by

$$\begin{aligned} H(X,Y) &= -\sum_{X=x,Y=y} p(x,y) \log\Big(p(x,y)\Big), \\ &= H(Y,X). \end{aligned}$$

Then, if we observe a particular realization of the random variable $Y = y$, the entropy of the probability distribution of X is given by

$$H(X \mid Y = y) = -\sum_{X=x} p(x \mid y) \log\Big(p(x \mid y)\Big),$$

and the conditional entropy is defined as

$$\begin{aligned} H(X \mid Y) = E\Big[H(X \mid Y = y)\Big] &= \sum_{Y=y} p(y)\ H(X \mid Y = y) \\ &= -\sum_{Y=y} p(y) \sum_{X=x} p(x \mid y) \log\Big(p(x \mid y)\Big), \\ &= -\sum_{X=x,Y=y} p(x,y) \log\left(\frac{p(x,y)}{p(y)}\right), \end{aligned}$$

knowing that $p(x,y) = p(y)\ p(x \mid y) = p(x)\ p(y \mid x)$.

In our context, the random variables X and Y correspond to community partitions, so $p(x) = p(X = x)$ is the probability that a node taken at random belongs to community x in the partition X. Therefore

$$p(x) = \frac{n_x}{n},$$

where n_x is the number of nodes in community x in the partition X, and

$$p(x,y) = \frac{n_{xy}}{n},$$

where n_{xy} is the number of nodes that belong to community x in the partition X and to community y in the partition Y.

One can observe that

$$\begin{aligned} H(X \mid Y) &= -\sum_{X=x,Y=y} p(x,y)\log\Big(p(x,y)\Big) + \sum_{X=x,Y=y} p(x,y)\log\Big(p(y)\Big), \\ &= H(X,Y) + \sum_{Y=y} p(y)\log\Big(p(y)\Big)\left[\sum_{X=x} p(x \mid y)\right], \\ &= H(X,Y) - H(Y), \end{aligned}$$

since $\sum_{X=x} p(x \mid y) = 1$.

As previously mentioned, we want to measure how much information we have about the true partition X from an extracted partition Y. This is given by the mutual information $I(X,Y)$ between X and Y which is the difference between the entropy of X and the conditional entropy of X knowing Y,

$$I(X,Y) = H(X) - H(X \mid Y) = H(X) + H(Y) - H(X,Y). \tag{4.9}$$

If the two partitions are independent (in the limit of infinitely large graphs) $H(X \mid Y) = H(X)$ so the mutual information is 0 and, if the two partitions are exactly identical, then the conditional entropy of X knowing Y is 0 so $I(X,Y) = H(X)$. So, the larger is the value of the mutual information between the two partitions, the closer they are to each other. However, it has been shown that the mutual information can not accurately distinguish different sub-partitions. If X is a partition and Y_1 and Y_2 are two different partitions obtained by sub-partitioning some clusters of X, then $I(X,Y_1) = I(X,Y_2) = H(X)$, since the conditional entropy in this case is $H(X \mid Y_1) = H(X \mid Y_2) = 0$, although the partitions Y_1 and Y_2 may be very different. Therefore, [Danon, Díaz-Guilera, Duch et al. (2005)] introduced a normalized variant, called the normalized mutual information, which is able to distinguish between different sub-partitions

$$NMI(X,Y) = \frac{2\, I(X,Y)}{H(X) + H(Y)}. \tag{4.10}$$

The normalized mutual information has become one of the most common quantitative criteria to compare the output of community detection algorithms. Other measures based on information theory have been proposed,

like the variation of information, but we observed that most of them are highly correlated with the NMI, hence it will be used in what follows. The NMI ranges in $[0,1]$ where 1 indicates exactly identical partitions and 0 (in the limit) totally independent partitions. Finally, in the context of community partitions, the normalized mutual information is computed as

$$NMI(X,Y) = \frac{-2\sum_{x,y} n_{xy} \log\left(\frac{n_{xy}\, n}{n_x\, n_y}\right)}{\sum_x n_x \log\left(\frac{n_x}{n}\right) + \sum_y n_y \log\left(\frac{n_y}{n}\right)}. \tag{4.11}$$

4.1.1 Weighted networks

We extensively applied all the algorithms presented in Chapter 3 to the weighted LFR benchmark and measured their performance using the NMI. All the algorithms were implemented in C++, either directly by the authors of the method or within an open-source package. The Louvain method (LM, Section 3.3.6) was implemented by V. Traag and is available at `http://www.traag.net/code/`. We built the implementation of our fast community extraction (FCE, Section 3.4) based on that source code. The algorithm of Clauset et al. (CNM, Section 3.3.3) is available at `http://www.cs.unm.edu/~aaron/research/fastmodularity.htm` and the method of Schuetz & Caflisch (SC, Section 3.3.4) is available at `http://www.biochem-caflisch.uzh.ch/public/5/network-clusterization-algorithm.html`. The label propagation algorithm (LP, Section 3.3.5) has been implemented within the igraph package [Csardi and Nepusz (2006)] and finally the infomap of Rosvall & Bergstrom (Section 3.3.7) can be downloaded from `http://www.tp.umu.se/~rosvall/code.html`.

The results for modularity based algorithms (FCE, LM, CNM & SC) and for LP are presented in Fig. 4.2 for undirected LFR graphs and in Fig. 4.3 for directed LFR graphs. Both figures are composed of 5 heat maps, corresponding to the different algorithms, and a summary plot which presents the cumulative area covered in each heat map, i.e. the proportion of the area of the experimental plane covered by NMI values larger than q, for $q \in [1, 0.4]$. The values of NMI were limited to 0.4 in this summary plot because this corresponds to the average value of NMI achieved for communities created at random. Obviously, all the curves closely collapse to 100% after this threshold.

Each heat map displays a smooth interpolation of the average value

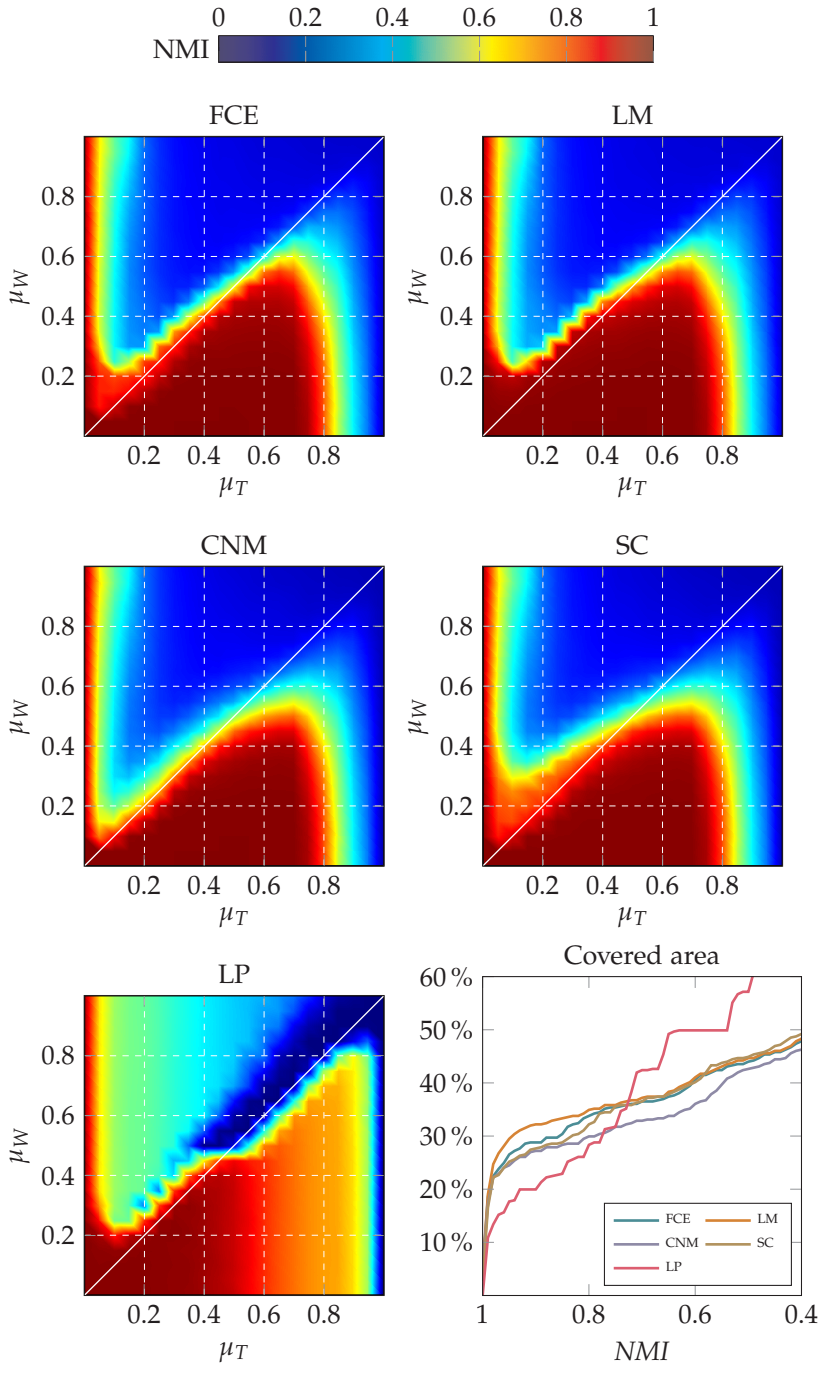


Fig. 4.2 Performance (NMI) of algorithms on the undirected LFR benchmark with $n = 1000$, $\langle k \rangle = 10$, $k_{max} = 20$ and community sizes in $[10, 100]$.

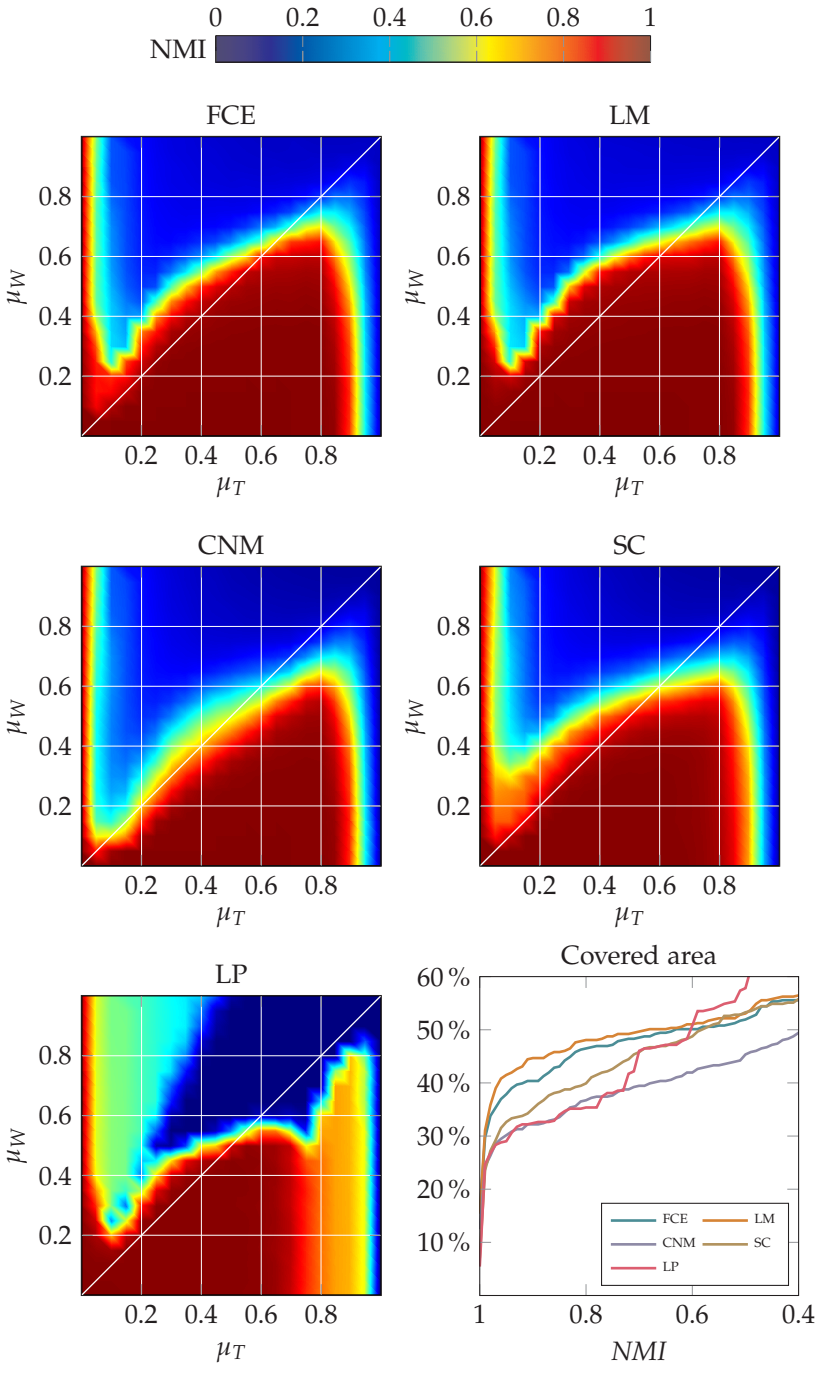


Fig. 4.3 Results for directed networks (See caption of Fig. 4.2).

of NMI computed on 50 graphs for each couple of mixing parameters $(\mu_T, \mu_W) \in [0,1] \times [0,1]$ discretized with a step size of 0.05. The x-axis captures the value of μ_T while the y-axis captures the value of μ_W. In this experiment, the parameters for the benchmark were set to $n = 1000$, $\langle k \rangle = 10$ and $k_{max} = 20$ with communities size distribution in the range $[10, 100]$. The other parameters were kept as default and do not influence much the results.

First, one can observe that all algorithms perform better under the diagonal which corresponds to the set of mixing parameters for which the expected weight of internal and external edges are equal (see Eq. (4.7) and Eq. (4.8)), i.e. communities start to be less well defined and harder to extract above the diagonal. However, it also becomes questionable if the a priori defined communities are still relevant in this range and if it remains desirable for the algorithms to extract them. LP is the only method with a clearly different behavior under the diagonal. This algorithm achieves lower value of NMI for $\mu_T > 0.5$ but has also a slower decay for larger values of μ_T. It is also interesting to observe that the performance of all the algorithms is superior on directed networks which can be seen from the larger covered areas in Fig. 4.3. The threshold for μ_T such that the communities can be extracted is around 0.8 for undirected networks and around 0.9 for directed networks.

In terms of quality, the main competitor of our FCE algorithm is LM which slightly outperforms our method for small values of μ_T and μ_W and just above the diagonal. The difference between the results of the 2 methods is rather small everywhere else and none of the algorithms is strictly dominating the other. The quality of SC is close to the performance of our algorithm for undirected networks but much less for directed networks. Finally, CNM achieves worse performances than all the other methods.

Since LM is slightly better than FCE for small values of the mixing parameters (with $\mu_W \geq \mu_T$), we analyzed more closely the partitions extracted by both algorithms. We observed that those partitions are in fact always very close to each other and that the average NMI of the partitions extracted by our algorithm is smaller due to more inaccurate merges of the true communities. For instance, a graph corresponding to $\mu_T = 0.1$ and $\mu_W = 0.15$ is illustrated in Fig. 4.4, with the true partition represented by the color of each node and the partitions extracted by FCE and LM represented as filled enclosing contours. One can see that most of the communities are correctly extracted and that both algorithms incorrectly merge two communities located in the top left of the figure. However,

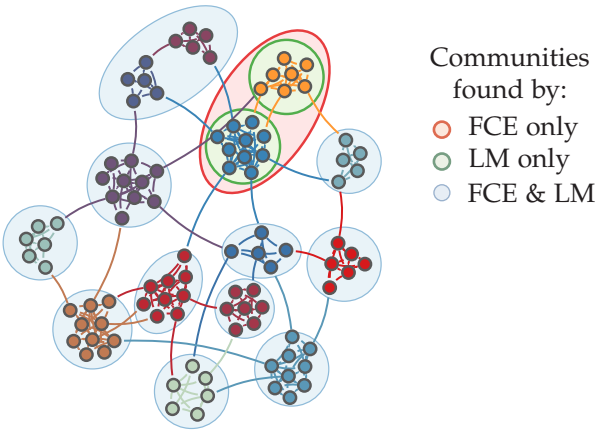


Fig. 4.4 Community partitions extracted by LM and FCE on the LFR benchmark with $n = 100$ and $\langle k \rangle = 5$. The color of each node indicates their true community assignment from the LFR benchmark.

FCE also merges two communities located on the top right of the figure while LM is able to correctly extract them. This highlights that, although FCE extracts a less accurate partition than LM, the true communities are not entirely split apart using FCE, hence the results are not as inaccurate as they may seem from Fig. 4.2 and Fig. 4.3. This also indicates an interesting direction for future development of the algorithm where additional correction steps might be defined to avoid this behavior.

We have also computed the average value of modularity achieved by each algorithm which is represented in Fig. 4.5 for the undirected LFR graphs and in Fig. 4.6 for the directed LFR graphs. Although it is well known that, as mentioned in Section 3.2.2 (see page 44), the modularity is not an accurate criterion to assess the quality of the different algorithms due to its large number of local optima [Good, de Montjoye, and Clauset (2010)], it is still interesting to observe how well each algorithm optimizes the measure as an objective function. Surprisingly, the results for all the algorithms are extremely close for undirected networks (except for LP but this algorithm does not optimize modularity) and only CNM seems to slightly underperform the other methods. For directed networks, FCE and LM outperform CNM and SC which is the same observation than for the

actual quality of the extracted partitions. Those observations confirm that the modularity value is not an accurate criterion to quantify the quality of a method but also that our FCE algorithm does a good job in optimizing this objective function.

Finally, we applied FCE and LM to the CPM cost function (see Section 3.2.3) and the results are represented in Fig. 4.7 for undirected LFR networks and in Fig. 4.8 for directed LFR networks. Both figures also contain the results of the classical and hierarchical Infomap (Section 3.3.7). As for modularity, FCE & LM produce very close results both in terms of NMI and as optimizer of the CPM cost function. The performances of both algorithm are even closer than for modularity. The infomap behaves extremely well in its classical version and outperforms all the other algorithms, but its hierarchical version is somehow less stable which is not surprising since we did not use a hierarchical benchmark model. This was somehow expected since the infomap mixes different optimization algorithms and uses a peculiar cost function. Moreover, it has been shown recently [Schaub, Lambiotte, and Barahona (2012)] that infomap was especially well suited to cluster graphs created using the LFR benchmark model. We will show in the following that the impressive quality of the partitions extracted using infomap comes at the cost of the large computational time required by the method.

We observe very similar behaviors of the algorithms for various parameters settings of the LFR benchmark and also for various graph sizes. We present in Fig. 4.9 and Fig. 4.10 the results for undirected and directed networks with $n = 5000$, $\langle k \rangle = 15$ and $k_{max} = 40$ and, in Fig. 4.11 and Fig. 4.12, the results for undirected and directed networks with $n = 10000$, $\langle k \rangle = 30$ and $k_{max} = 60$. One can observe that, as the size of the networks increases, the range of mixing parameters for which we can correctly extract community structures also increases as explained in the description of the benchmark model. The performances of CNM seems awkward in the last figure, however the poor quality of the partitions is due to implementation and memory issues rather than to the actual algorithm.

Our analyses show that our algorithm achieves similar or even higher qualitative performances than the other popular and efficient methods for community detection. However, the main achievement of our method is its very small computational time along with its highly parallelizable behavior. We analyzed the computational time required by each method to extract community partitions for benchmark graphs growing from $n = 10^3$ to $n = 10^6$. The average degree was set to one hundredth of the

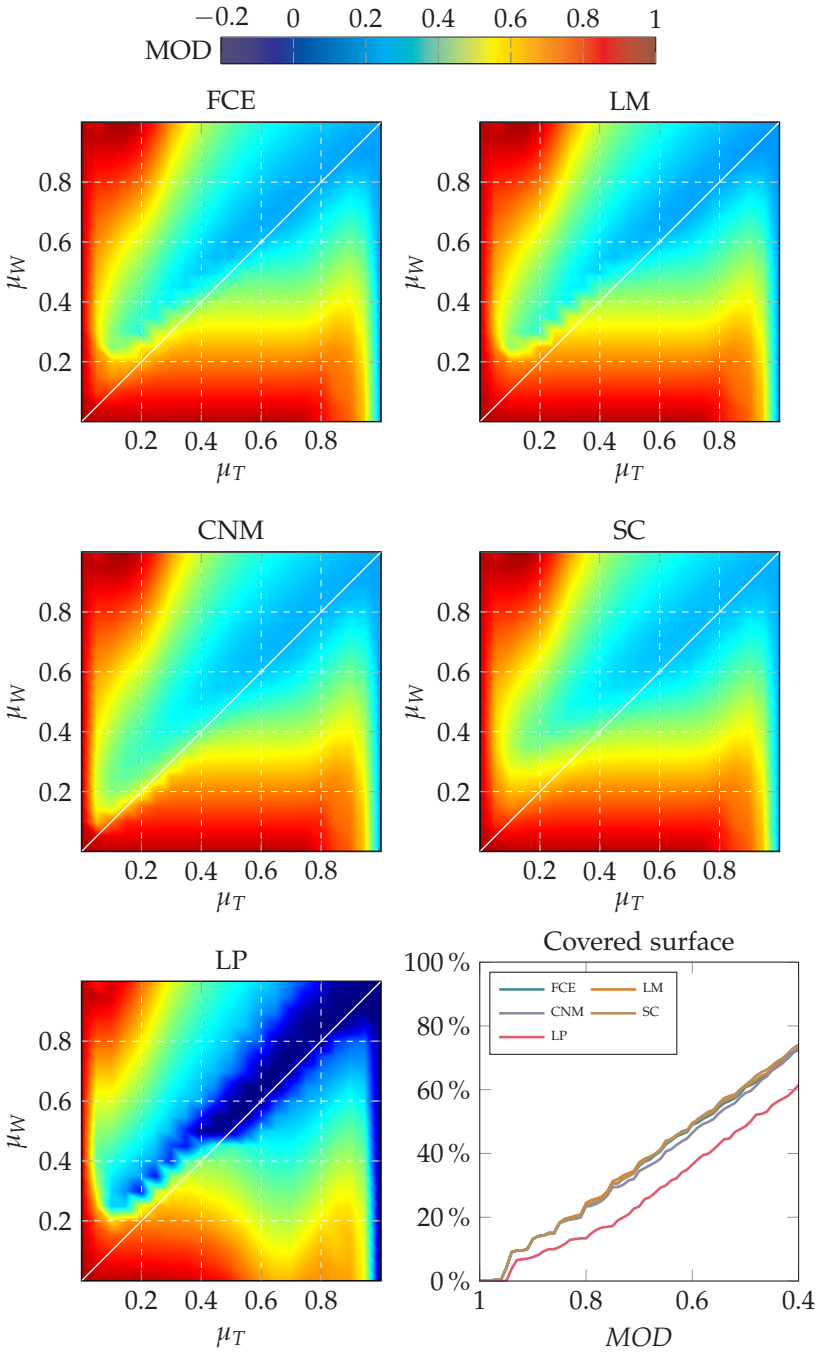


Fig. 4.5 Modularity value (See caption of Fig. 4.2).

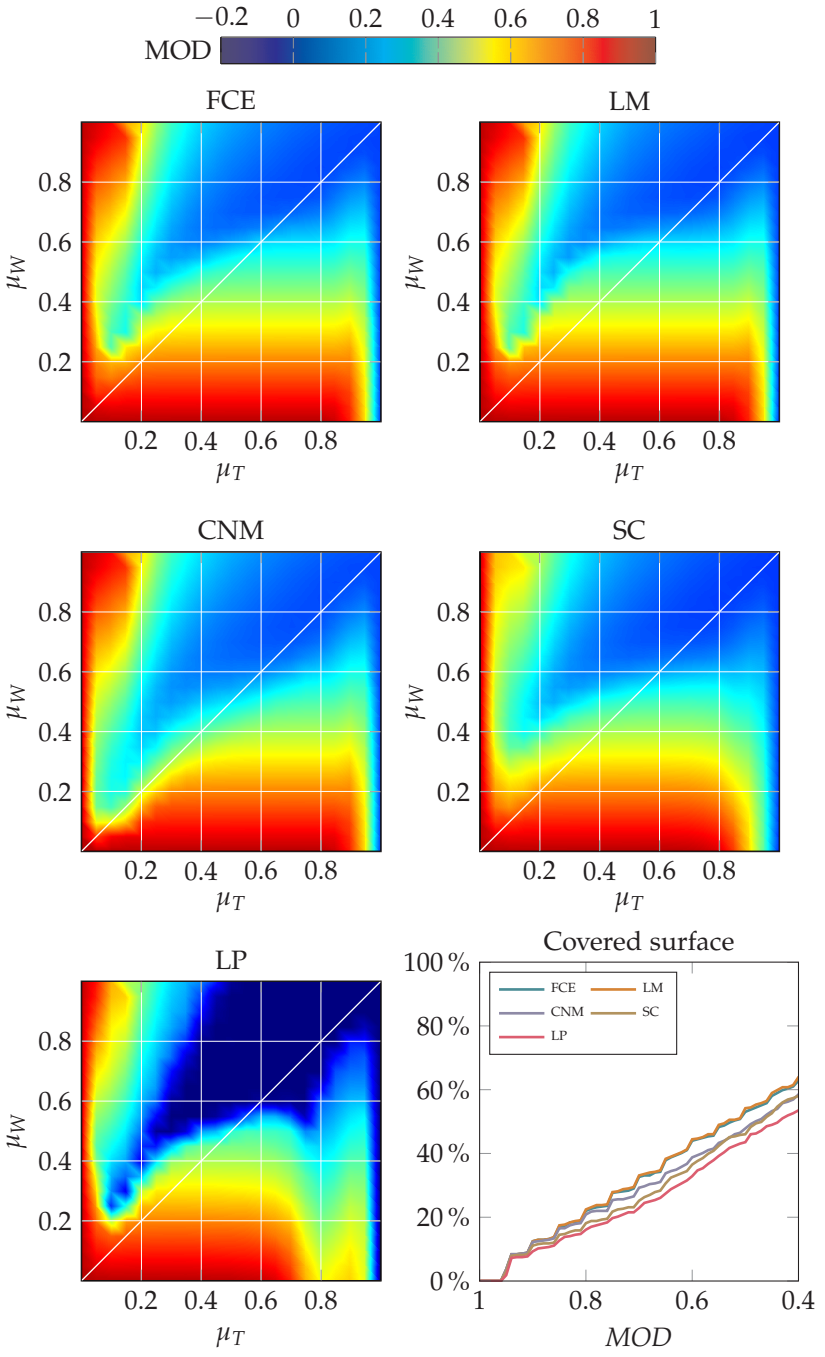


Fig. 4.6 Modularity value for directed networks (See caption of Fig. 4.2).

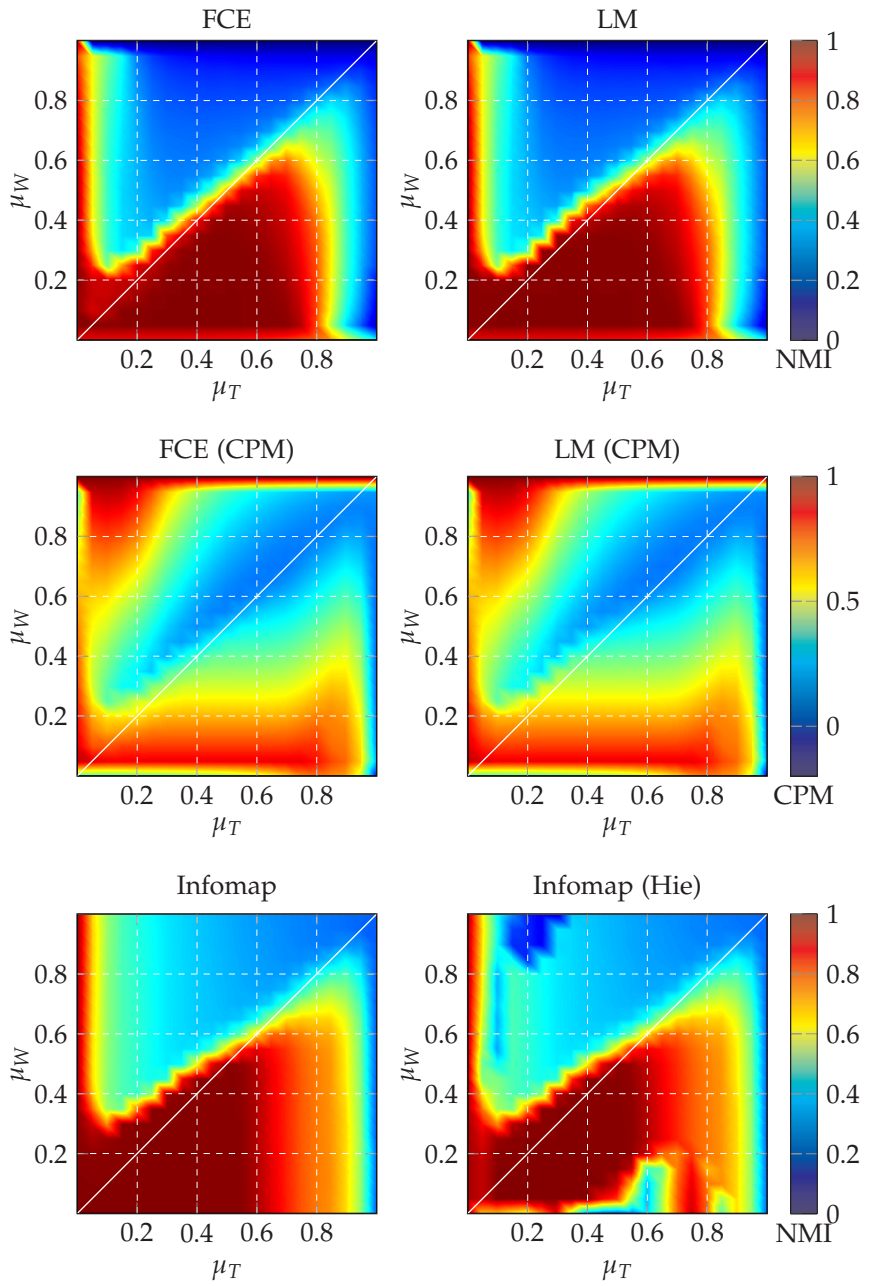


Fig. 4.7 Performance of FCE & LM on CPM and of Infomap (See caption of Fig. 4.2).

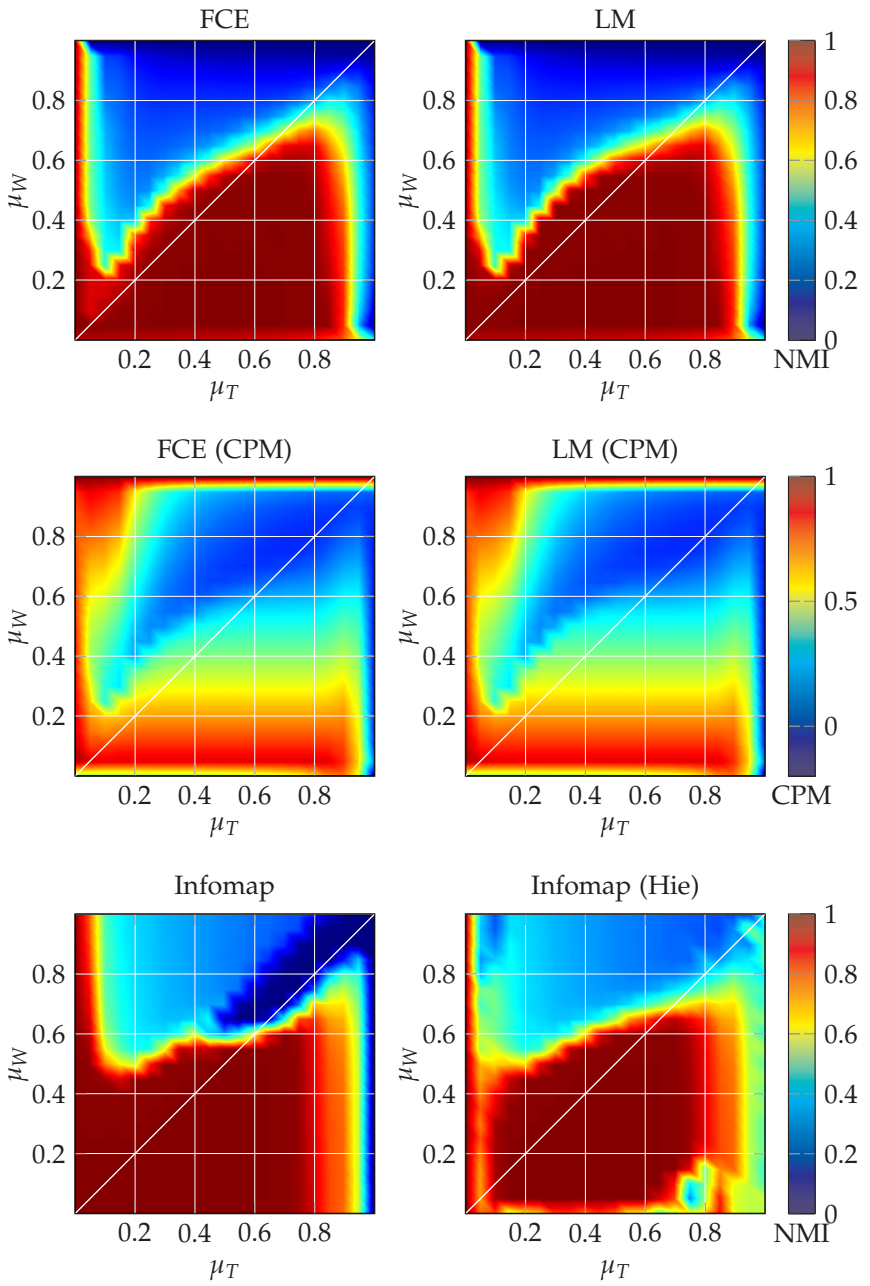


Fig. 4.8 Performance of FCE & LM on CPM and of Infomap on directed networks (See caption of Fig. 4.2).

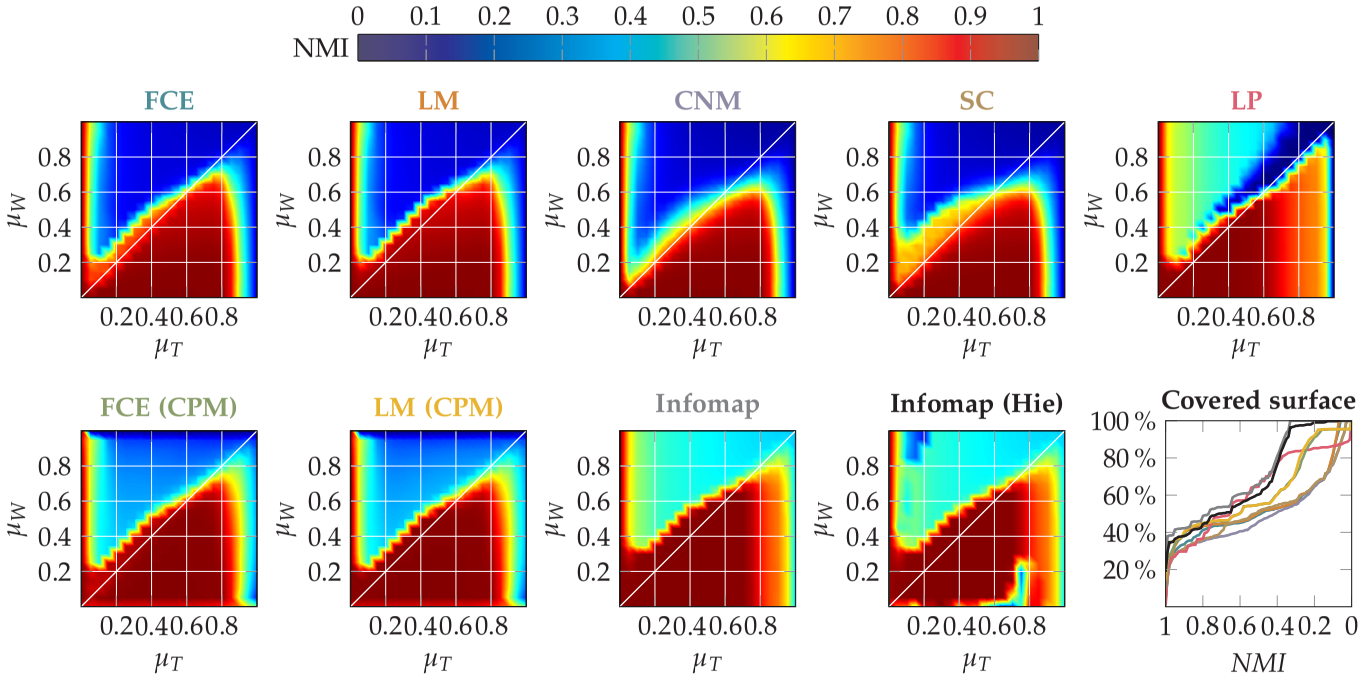


Fig. 4.9 Performance (NMI) of algorithms on the undirected LFR benchmark with $n = 5000$, $\langle k \rangle = 15$, $k_{max} = 40$ and community sizes in $[15, 150]$. The color codes of the titles of each algorithm give the legend of the last figure (Covered Surface).

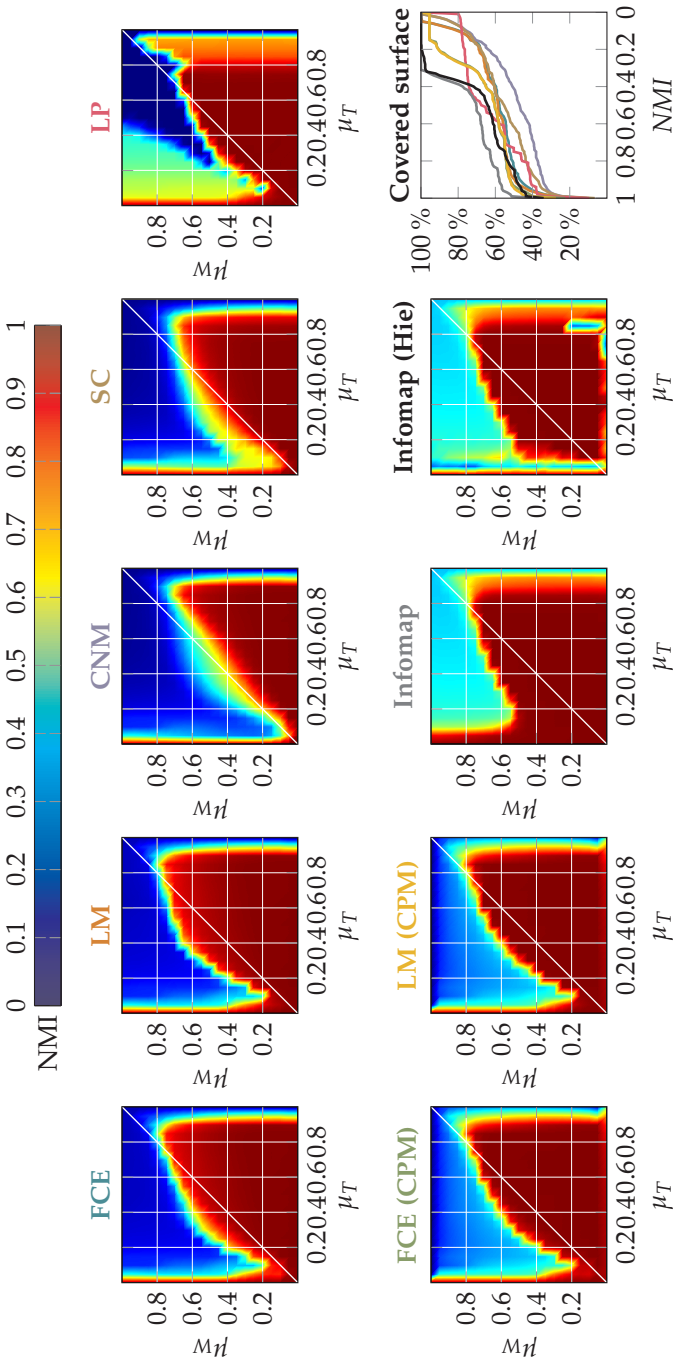


Fig. 4.10 Results for directed networks (See caption of Fig. 4.9).

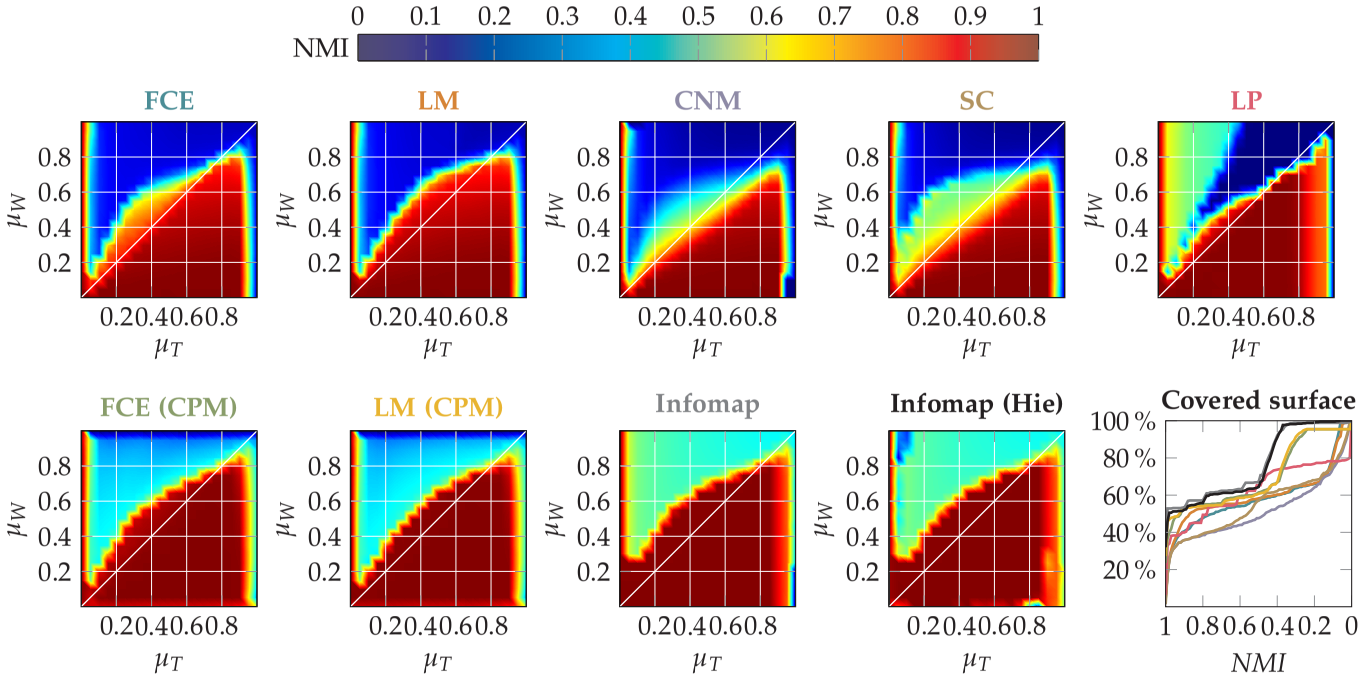


Fig. 4.11 Performance (NMI) of algorithms on the undirected LFR benchmark with $n = 10000$, $\langle k \rangle = 30$, $k_{max} = 60$ and community sizes in $[15, 150]$. The color codes of the titles of each algorithm give the legend of the last figure (Covered Surface).

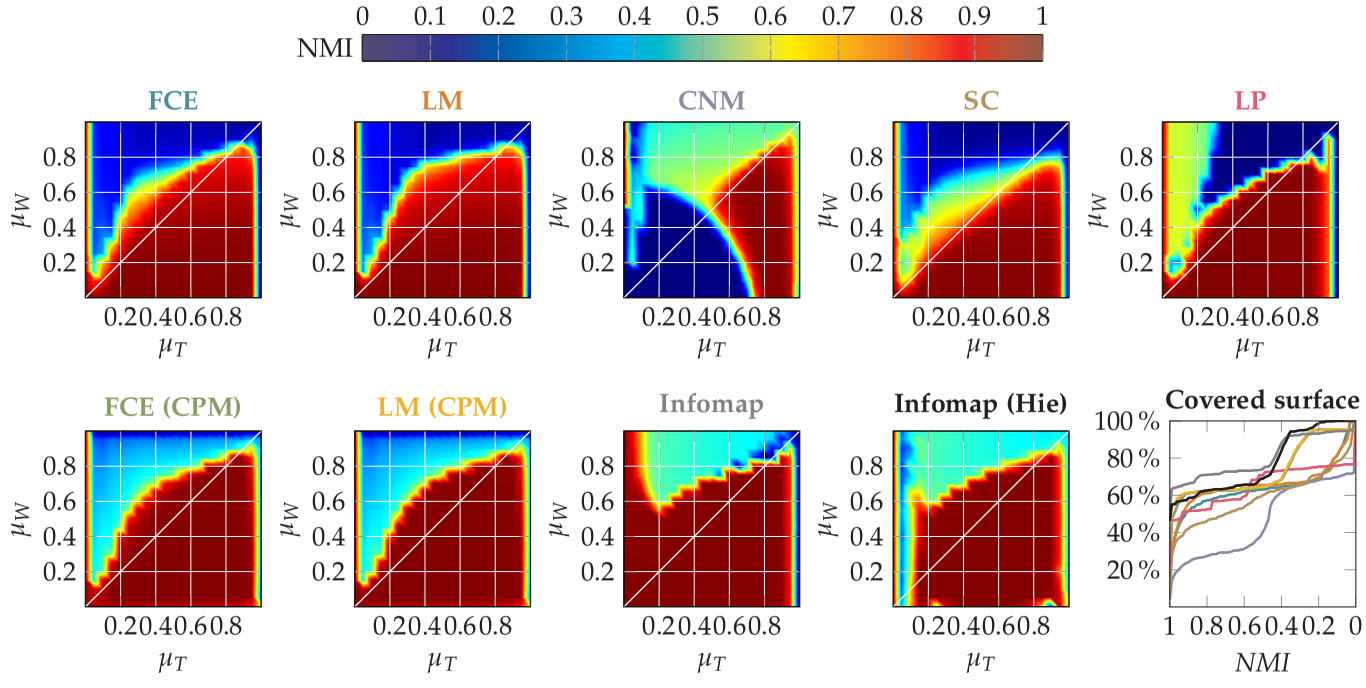


Fig. 4.12 Results for directed networks (See caption of Fig. 4.11).

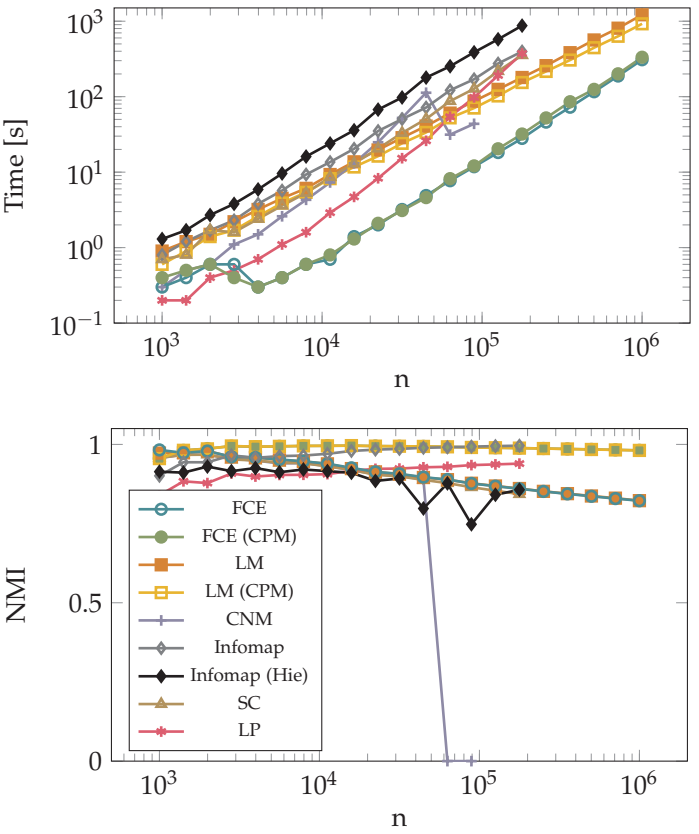


Fig. 4.13 Average computational time and NMI for each algorithm with respect to n.

number of nodes in the graph, $\langle k \rangle = n/100$, and the community sizes were chosen between one hundredth and one tenth of the number of nodes, $[n/100, n/10]$. The results are presented in Fig. 4.13 where the top panel shows the average computational time for each method and the bottom panel shows their average NMI value with respect to n. Each algorithm was applied to benchmark graphs with mixing parameters couples $\mu_T, \mu_W \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, $\mu_W \leq \mu_T$, i.e. for 15 different pairs. Each marker in the plot corresponds to the average computational time or NMI on 25 graph realizations per pair of mixing parameters.

As stated earlier, to have a fair comparison, all the different algorithms were tested using a C++ implementation and executed on dedicated single

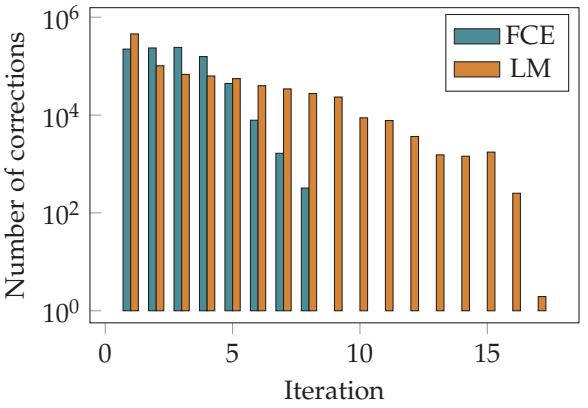


Fig. 4.14 Average number of corrections observed per iteration for FCE and LM.

processors. This implies that no external task could have slowed down the processor and perturbed the results. Some algorithms were not tested for the largest graph sizes, as soon as the computational time required to extract a partition was larger than 1500 seconds, and CNM was also discarded earlier because of memory issues leading to extremely poor quality of the extracted communities. It is clear from Fig. 4.13 that FCE outperforms all the other methods. We observed that FCE extracts partitions between 5 and 20 times faster than LM using either modularity or CPM, so the computational time performance seems to not be tied to the cost function. Moreover, even smaller computational time could have been reached using dedicated multiple processors architectures due to the highly parallelizable nature of our method.

One can observe that even if FCE is much faster than the other methods, the average quality of the extracted partitions is similar to the average quality of the partitions extracted by the other algorithms optimizing the same cost function. Hence, our method does not trade too much quality for time performance. The decay of quality for large sizes is due to the well known resolution limit problem of modularity previously mentioned in Section 3.2.2. This observation is confirmed by observing the NMI curves for FCE and LM applied to the CPM cost function which leads to partitions of impressive quality even for graphs with 1 million of nodes.

One can wonder why FCE is so much faster than LM using only a single processor since both methods are based on the same idea of moving

nodes from one community to another. The explanation behind the large speed-up of our algorithm lies in two observations. First, the two types of corrections we introduced in FCE allow to change the community assignment of a larger number of nodes in one iteration due to the assignment graph. This leads to a clear computational gain compared to sequential modifications of LM. Another observation is that even if the total number of corrections is about the same for both algorithms, our method allows to find a steady partition of the nodes in fewer iterations, as displayed in Fig. 4.14. Each iteration of LM requires to loop over all the nodes and edges of the graph to check for possible increases in the cost function. By reducing the required number of iterations to reach convergence, our algorithm extracts much faster the first hierarchical level, i.e. before the aggregation of the communities, and that is the dominant factor in the total time complexity.

4.1.2 Unweighted networks

We also applied the different algorithms on the unweighted LFR benchmark for which the results are displayed in Fig. 4.15. The NMI of the extracted partitions are represented in the left column for undirected networks and in the right column for directed networks. The ranking of the methods is not fundamentally different than for weighted networks but we present the results for the sake of completeness.

CNM clearly does not achieve good quality for any parameter settings. SC is very close to FCE for small ($n = 1000$) undirected networks but is less accurate for any other kind of networks. Our FCE algorithm is close to LM, and even more so for directed networks. The gap between the 2 NMI curves is mainly explained by the same observation as in Fig. 4.4, i.e. our method makes additional merges of small clusters compared to LM, without breaking the true communities. This was expected since on the diagonal of the mixing parameters set for weighted networks, the expected weight of internal and external edges is the same, so the networks are theoretically close to unweighted networks. Again, the gap between FCE and LM is much smaller for directed networks and almost absent for larger networks ($n = 10000$). LP produces good results but drops suddenly for specific thresholds of μ_T. Apart from a weird behavior for small μ_T, the hierarchical infomap outperforms all the other methods but its classical version which achieves equivalent NMI for undirected networks.

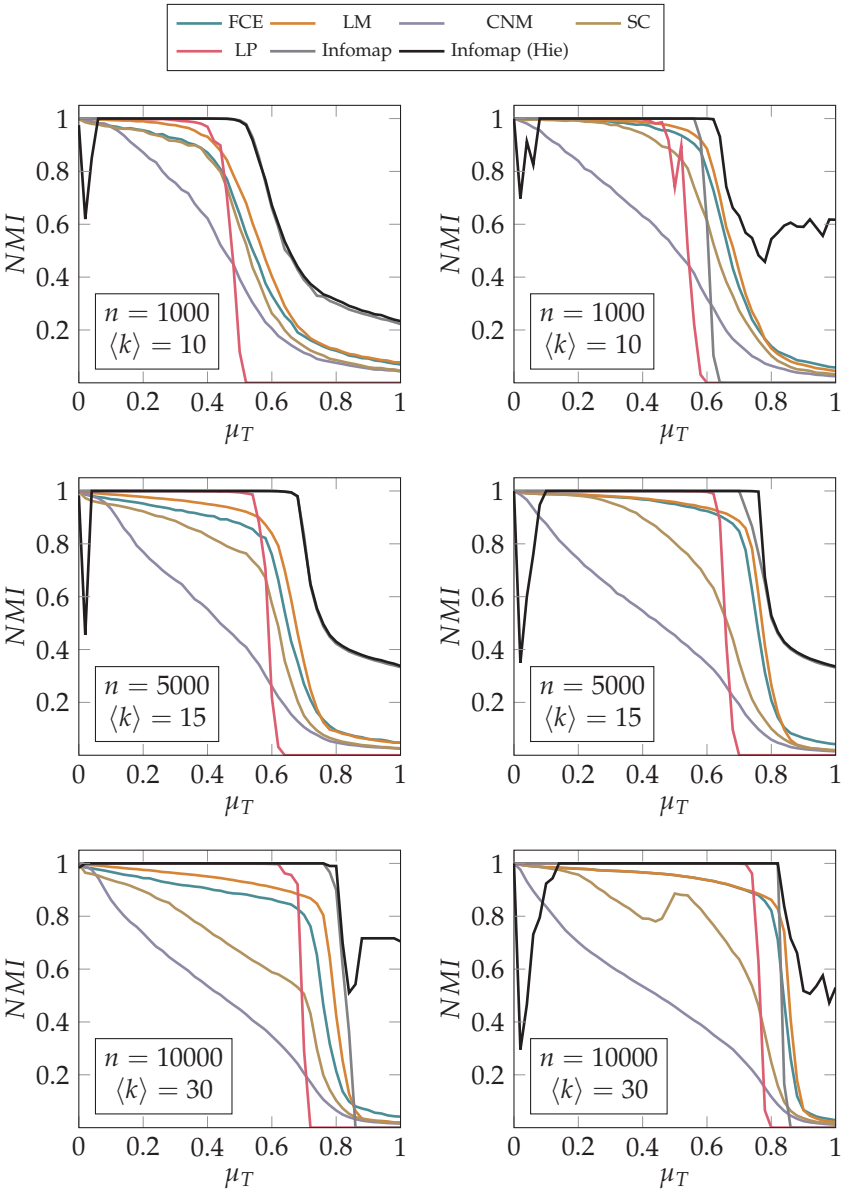


Fig. 4.15 Performance of algorithms on the unweighted LFR benchmark (undirected on the left side and directed on the right side).

4.1.3 Analysis of the probability parameter

Finally, we analyzed the effect of the probability parameter $p < 1$ introduced in Section 3.4.5 and required to guarantee the convergence of FCE. When p is small, only a few corrections are accepted per iteration and the method is close to LM (up to the assignment graph that defines the communities and forces some additional node switches) because we can safely assume that the accepted corrections will often be independent. On the other hand, when p is large, many corrections can be applied synchronously.

To this end, we applied FCE to networks with various mixing parameters values and with $n = 10\,000$ and $n = 100\,000$ and compute the average computational time required by the algorithm to extract communities (the computational time for smaller networks is too small to distinguish between the effect of p or some noisy measurements). The results are presented in Fig. 4.16. The first observation is that, as expected, when the value of p increases, the computational time decreases significantly. This is observed for both modularity and CPM, so this seems independent of the cost function. Moreover, when the communities are correctly extracted (NMI $\simeq 1$), the value of p has almost no effect on the outcome of the algorithm and the performance curves are very stable. Finally, for networks containing communities harder to extract, i.e. on the diagonal of the mixing parameters set, the quality of the extracted communities starts to decrease slightly after $p = 0.8$. This is why, in all our experiment, we set $p = 0.8$ which gives an appropriate balance between the quality of the extracted partitions and the computational time of FCE.

4.2 Image Processing

How many objects are present in a picture? This simple question from a human point of view (depending on what one considers to be an object) has raised a lot of research for the last decades and remains a very challenging problem for a computer. There are many practical applications where this problem occurs: discovering abnormal shadows on a CT or PET scan for tumor detection, detection of people and objects from images of surveillance cameras or from a camera at the front of a car in the context of collision detection, etc. This task is commonly known as image segmentation.

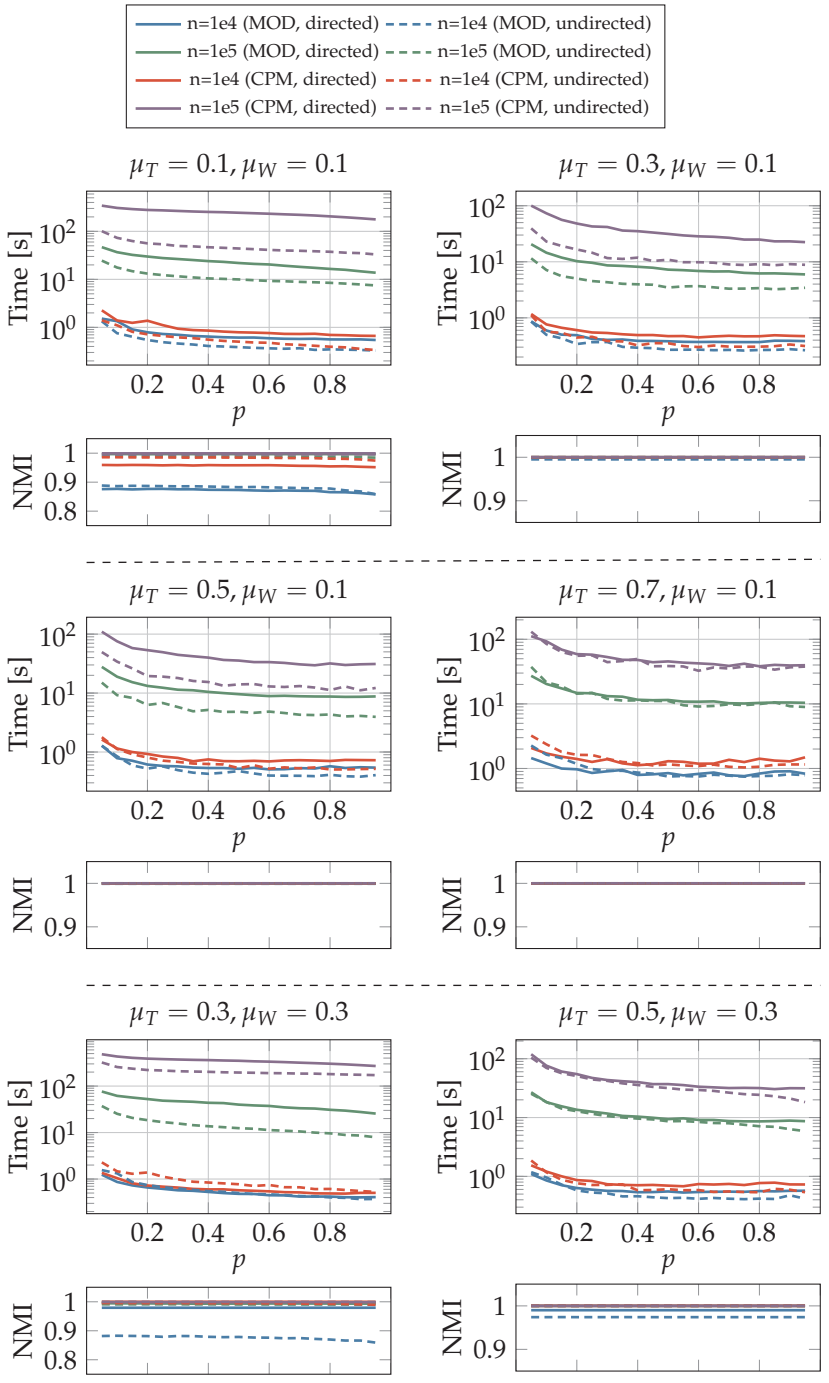


Fig. 4.16 Analysis of probability parameter p on the LFR benchmark.

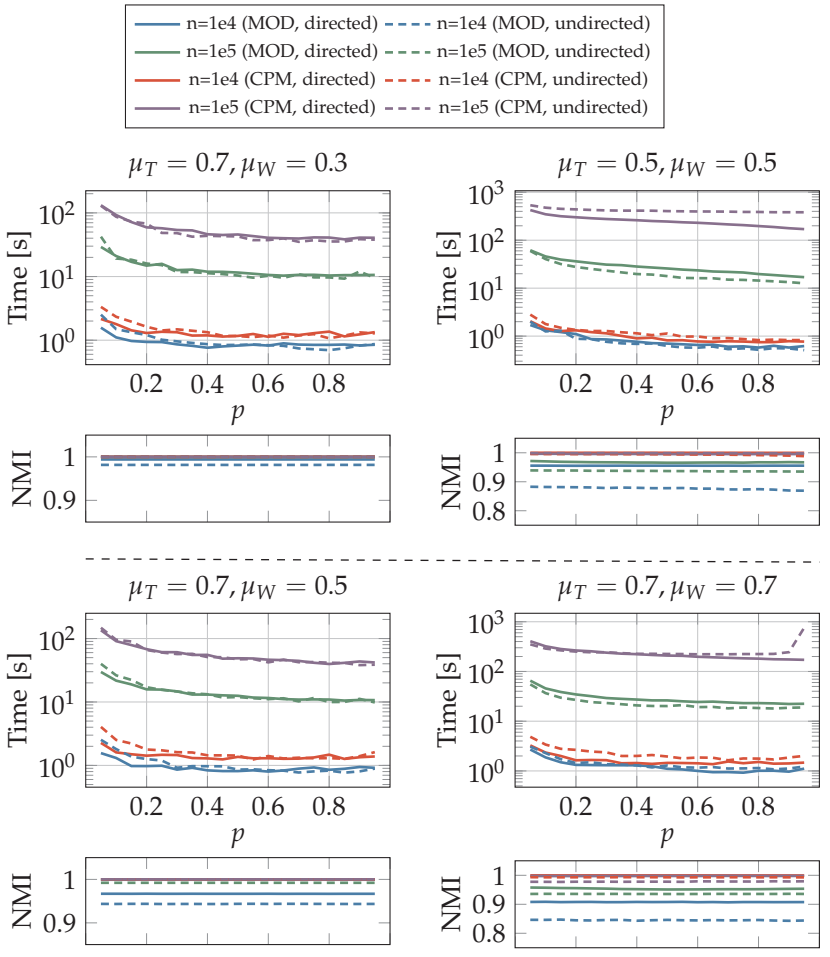


Fig. 4.16(cont'd) Analysis of probability parameter p on the LFR benchmark.

4.2.1 Picture graph

The number of techniques in computer vision is quite large, and the theory behind them comes from various fields such as histogram metrics [Werman, Peleg, and Rosenfeld (1985)], watershed techniques [Beucher

et al. (1991)], active contours [Olszewska (2009); Kass, Witkin, and Terzopoulos (1988)], manifold learning [Ho, Lee, Yang et al. (2004); Sundaramoorthi, Mennucci, Soatto et al. (2009)], etc. Some methods are based on graph theory [Cour, Benezit, and Shi (2005); Arbelaez, Maire, Fowlkes et al. (2011)]. Those procedures generally build a graph, called a picture graph, where each node represents a pixel of the input picture. Since the number of pixels may be very large, even for low resolution pictures, ideas have been proposed to reduce the number of nodes in the graph to a representative subsample of pixels or regions, [Alzoubi and Pan (2008)] and see other references therein.

The weighted edges of the undirected picture graph encode the similarity between pairs of pixels in the picture. Without a priori knowledge about the components of the input image, a classical way to define the weight of the edges is to use a Gaussian kernel function

$$w(i,j) = \begin{cases} e^{\frac{d(i,j)^2}{\sigma_x^2}} e^{\frac{|F(i)-F(j)|^2}{\sigma_i^2}} & \text{if } d(i,j) < d_{max}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.12}$$

where $d(i,j)$ is the distance between pixels i and j (e.g. the Euclidean or the Chebyshev distance) and $F(i)$ is a feature vector evaluated at pixel i. This feature vector can be for example the scalar intensity value for gray scaled images or the HSV transform for color images. The edge weight can be controlled by the user defined parameters σ_x, σ_i and d_{max}.

Based on this undirected weighted graph, classical methods [Wu and Leahy (1993)] try to find a set of edges, known as a cut, that optimizes a criterion, as for the classical minimum cut problem, Section 2.4. The cost function is in general defined as the ratio between the total weight of the cut edges and a scaling factor that penalizes the creation of very small components, i.e. to avoid that the optimal cut corresponds to the segmentation of the boundary or the corner of the picture which are naturally the less connected pixels. The ratio cut [Cox, Rao, and Zhong (1996)] uses a scaling factor based on the dimension of the components and the minimum mean cut [Wang and Siskind (2001)] uses the number of cut edges. The normalized cut [Shi and Malik (2000)] is based on the internal similarity of the components. Optimizing the normalized cut criterion is known to be NP-hard but a continuous relaxation can be solved in polynomial time. Another effective method proposed by [Felzenszwalb and Huttenlocher (2004)] extracts regions in the graph minimizing the inter-

(a) Input picture

(b) Segmentation using modularity

Fig. 4.17 Segmentation of a baby face (130×132 pixels) using modularity.

nal difference of regions while also maximizing the external difference between regions. Some recent works [Alzate and Suykens (2008); Frederix and Barel (2013)] proposed to avoid the high computational complexity of such methods by extracting an approximation of the optimal cut using an incomplete Cholesky decomposition. Unfortunately, in general these methods need to know the number of objects of interest in the input picture to define a stopping criterion or they have to set an arbitrary threshold on the minimum value of the cut criterion that may lead to inappropriate segmentation.

All those methods try to split the picture graph into salient regions by defining boundaries between the regions, hence acting as divisive algorithms. On the other hand, one can agglomerate adjacent connected nodes, which in turn defines regions within the picture. Hopefully, pixels that belong to the same object should be more similar with each other than with the other surrounding pixels. Therefore, pixels within an object should be linked with highly weighted edges, and loosely connected with neighboring pixels, hence revealing a community structure. This implies that we can apply our community detection algorithm on a picture graph to segment the underlying image. Moreover, our method does not require any a priori knowledge about the shape, the position or the number of objects displayed in the input picture, which is a main advantage in this field.

Fig. 4.17 shows the results of the segmentation of an image with our FCE community detection algorithm using modularity which extracts 34

Fig. 4.18 Segmentation of a baby face using windowed modularity.

regions. The parameters of the picture graph were set to $\sigma_x = 1.2$, $\sigma_i = 10$, $d_{max} = 3$. One can see that all the regions segmented by the algorithm are coherent but also that the objects of interest are clearly oversegmented (6 regions for the shirt and 7 for the face). This oversegmentation is due to the field of view limit [Schaub, Delvenne, Yaliraki et al. (2012)] already mentioned in Section 3.2.2. Each object (including the background) is spanned by long chains of locally interconnected nodes. The maximal distance between connected pixels d_{max} is small compared to the size of the picture which leads to a huge number of disconnected pairs of nodes in the picture graph. Therefore, modularity is not able to accurately extract the different objects.

There are at least two ways to tackle this resolution problem. First, one can increase the distance d_{max} which in turn reduces the number of disconnected pairs of nodes within communities. While this is a good theoretical solution that may work on very small images, it is not actually suitable. One can check that the number of neighbors for each pixel grows as $O(d_{max}^2)$, making this solution inappropriate for practical applications. Another possible solution is to modify the null model in such a way that it somehow takes into account the distance inside communities. We pursue this idea in the next section.

4.2.2 Windowed configuration null model

To avoid oversegmentation, we proposed to change the null model such that it takes into account the fact that an object can be spread over a large area of the picture and that the associated pixels can not be connected. To this end, we introduced the windowed configuration null model that

extends the resolution parameter introduced by Reichardt & Bornholdt, Section 3.2.1. The window configuration null model is given by

$$p_{ij} = \frac{1}{2m} s_i \Lambda_{ij} s_j, \tag{4.13}$$

where Λ is a mask matrix defined as

$$\Lambda_{ij} = \begin{cases} 1 \text{ if } d(i,j) \leq d_\Lambda \ , \\ 0 \text{ otherwise } , \end{cases} \tag{4.14}$$

and d_Λ is an additional parameter depending on the size of the picture and the expected size of the objects. The cost function to optimize over the community assignment of the nodes in the picture graph can be written as

$$Q_\Lambda(\sigma) = \frac{1}{m} \sum_{i,j \in V} \left[W - \frac{S \Lambda S}{m} \right]_{(i,j)} \delta(\sigma_i, \sigma_j) \tag{4.15}$$

where S is the diagonal matrix of the strength of the nodes, $S = diag(s)$. As for the CPM, we need to adjust the null model after the aggregation step such that merging nodes in the aggregated graph is equivalent to merging the associated communities in the original graph (see page 73). This can be done by using

$$\Lambda' = (S')^{-1} \left(C \, S \, \Lambda \, S \, C^T \right) (S')^{-1}, \tag{4.16}$$

where C is the community assignment matrix, $C \in \{0,1\}^{p \times n}, C(i,j) = 1$ if $\sigma(i) = j$, and S' is the diagonal matrix of the aggregated strength of the nodes, $S' = diag(C\, s)$. One can observe that this definition is consistent with the classical modularity, i.e. if we define $\Lambda = \mathbf{1}_n \mathbf{1}_n^T$ then $\Lambda' = \mathbf{1}_p \mathbf{1}_p^T$.

The communities extracted by FCE using the windowed modularity for the previous test picture is presented in Fig. 4.18 for $d_\Lambda = 15$. There are 8 regions defined by the algorithm which have very good visual relevance. The left eyelash of the baby is kept separated from the face as in Fig. 4.17. This shows that our windowed null model effectively produces larger communities by allowing additional merging but without occulting the small details extracted by the original cost function. Although, this may also introduce small errors like the right ear that is clustered with the shirt due to a tiny darker region on the right side of the baby face.

Another interesting feature of our method is that the extracted com-

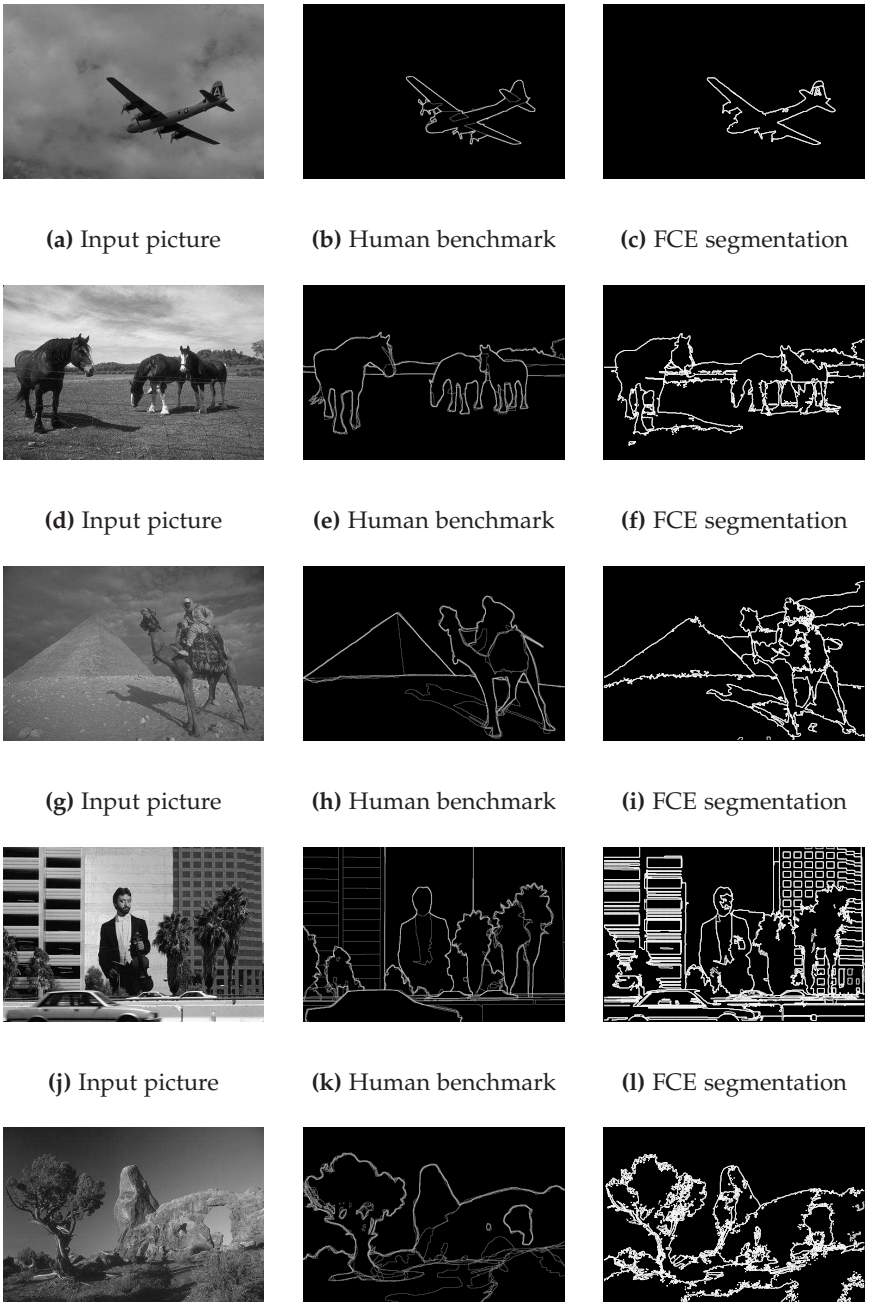
(a) FCE Segmentation.

(b) Penultimate step of FCE Segmentation.

Fig. 4.19 Segmentation of 2 elephants (320×480 pixels) using window modularity

munities are hierarchical. This might allow to extract smaller parts of objects contained in larger areas of the picture. For example, Fig. 4.19 shows the communities extracted at the last and the penultimate aggregation steps for a picture graph defined using $\sigma_x = 5$, $\sigma_i = 9$, $d_{max} = 3$ and $d_\Lambda = 25$. In Fig. 4.19a, representing the final community partition, there are 7 main regions segmented (and some small communities, with less than 10 pixels, due to image artifact), that are the sky, the elephants, the grass, one tusk and the ear shadow due to the large difference in pixel intensities. On the other hand, the communities extracted at the penultimate aggregation step, represented in Fig. 4.19b, allow to discover finer boundaries between the relevant parts of the picture showing that there are in fact 2 distinct elephants.

We ran our algorithm on a set of pictures taken from the Berkeley image segmentation database [Martin, Fowlkes, Tal et al. (2001)] and found relevant contours of objects when the parameters of the picture graph were correctly defined. Some results are displayed in Fig. 4.20 along with the initial pictures and human segmentation benchmarks. The different pictures are presented in increasing level of complexity according to the benchmark, and the parameters used for each picture graph are presented in Table 4.1. One can see that our algorithm is able to correctly identify the general contour of the objects in each picture but tends to have some difficulties to define boundaries in low gradient regions like the neck of the camel in the third row. There are also additional aggregations of communities that might be considered, for example the background behind the violinist in the fourth row or in the large rock at the right side of the figure in the fifth row.



(a) Input picture **(b)** Human benchmark **(c)** FCE segmentation

(d) Input picture **(e)** Human benchmark **(f)** FCE segmentation

(g) Input picture **(h)** Human benchmark **(i)** FCE segmentation

(j) Input picture **(k)** Human benchmark **(l)** FCE segmentation

(m) Input picture **(n)** Human benchmark **(o)** FCE segmentation

Fig. 4.20 Segmentation results on BSDS pictures.

plane	$\sigma_x = \sqrt{2}$, $\sigma_i = 5$, $d_{max} = 3$, $d_\Lambda = 20$
horses	$\sigma_x = \sqrt{2}$, $\sigma_i = 3$, $d_{max} = 2$, $d_\Lambda = 20$
camel	$\sigma_x = \sqrt{2}$, $\sigma_i = 2$, $d_{max} = 2$, $d_\Lambda = 25$
violinist	$\sigma_x = \sqrt{2}$, $\sigma_i = 4$, $d_{max} = 3$, $d_\Lambda = 20$
desert	$\sigma_x = 2$, $\sigma_i = 2$, $d_{max} = 4$, $d_\Lambda = 30$

Table 4.1 Parameters used for the segmentation presented in Fig. 4.20

While we observe good segmentation results, we also notice that the communities found can change dramatically with a minor modification of the parameters (σ_x, σ_i, d_{max} or d_Λ). This sensitivity to the parameters has been observed in most segmentation methods but it seems to affect strongly our results. Hence, we do not claim that our algorithm may outperform other dedicated segmentation methods, but due to its fast computational speed and to the overall quality of the extracted regions, it can be used as an accurate preprocessor for any input picture to largely simplify the task of more precisely tuned segmentation algorithm. Moreover, the cost function we used is not particularly designed for the image segmentation problem and other kind of criterion may be used. As soon as the cost function is somehow defined over the node community assignments, our algorithm is able to greedily optimize it. For example, the energy function used for active contours, also called snakes, is defined by

$$E(B) = \int_0^1 \Big[E_{int}\left(B(s)\right) + E_{con}\left(B(s)\right) + E_{ext}\left(B(s)\right) \Big]\, ds, \tag{4.17}$$

where $B(s)$ is the boundary of a region, $E_{int}\left(B(s)\right)$ is the internal energy which describes the flexibility and the resistance of the boundary, $E_{con}\left(B(s)\right)$ is the constraint energy to control the size of the region and $E_{ext}\left(B(s)\right)$ is the external energy described from the input picture to segment coherent regions. This functional is not directly computed from the node assignments but one could still compute the variation in the energy by switching some nodes from one community to another, hence modifying the boundary of the region. Therefore, this kind of cost function could be an excellent candidate, however the computational cost could also increase because one would need to compute the variation of the boundary

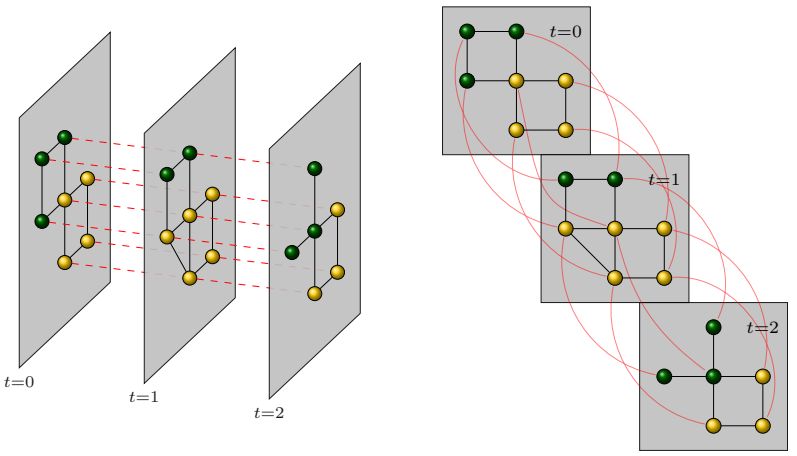


Fig. 4.21 Sliced network. Reprint with authorization from [Traag (2013)].

curve, etc.

4.2.3 Video tracking and 3D Segmentation

Solving the image segmentation problem allows one to automatically extract the position of different objects within a picture. Though, in many applications, one is also interested in following the evolution of those positions which is known as video tracking. While the image segmentation problem is already difficult, the video tracking problem raises even more challenges like changes in the pose (rotation, deformation,...) or in the ambient illumination, occlusion of the target, camera movements, and so on [Maggio and Cavallaro (2011)].

One way to handle video tracking is to consider each frame as an individual picture, to segment this picture and then try to match each object from one frame to the next one. This approach has the main advantage that it does not largely increase the computational cost of the algorithm since one only needs to segment individual pictures. Although, matching targets from one frame to the other might be very difficult when the appearance of other objects is similar to the target, which is known as image clutter.

On the other hand, one can use simultaneously the information of

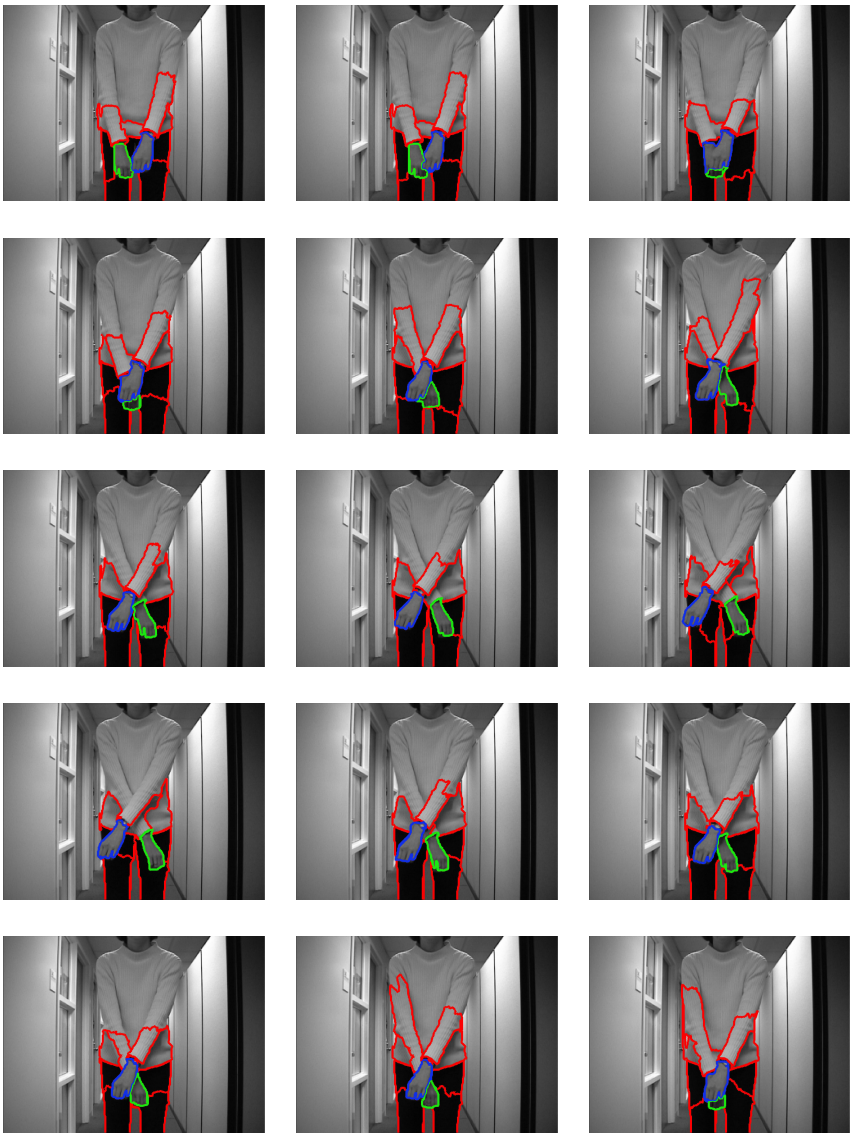


Fig. 4.22 Hands tracking.

multiple frames to segment the objects and acquire their trajectories at the same time. As a proof of concept, we applied our algorithm to this problem by considering a video as a sliced network as depicted in Fig.

4.21. Sliced or multilayer networks have become increasingly popular recently to study networks with multiple types of edges [De Domenico, Solé-Ribalta, Cozzo et al. (2013); Kivelä, Arenas, Barthelemy et al. (2014)]. Each frame of the video is represented by a picture graph in a slice. Then, multiple slices are connected using a similar measure than in the picture graph. Two nodes i and j in different slices are connected if the time interval between the slices is smaller than d_T and the distance between the associated pixels is again smaller than d_{max}. For example, in Fig. 4.21, $d_T = 1$ and $d_{max} = 1$. Finding the communities in a sliced network then hopefully reveals both the objects and the evolution of their positions in consecutive frames. Although, the more slices are connected, the larger is the network to cluster which may dramatically increase the computational time of the algorithm.

We applied this 3 dimensional segmentation to a video of someone moving their arms and try to track the position of the hands. The results are presented in Fig. 4.22. We initiated the tracking of the hands by defining one seed pixels per hand. We then computed the communities enclosing those seeds and tracked the evolution of those communities (and the surroundings).

We used $d_T = 2$ and initiated the network with 3 frames. After each segmentation, the sliced network is updated. We simply discarded the oldest frame and added a new slice to the network. The communities are then initialized as found in the previous step for the 2 remaining slices (according to their assignment graphs) and with 1 community per node in the new slice. Finally, our algorithm is applied to this updated sliced network using the initial partition. Each node in the new frame can either choose to join a community in the 2 previous frames (hopefully if the pixels belong to the same object) or to define a new community in the new frame (if a new object appears).

First, it is clear that this method will not be resilient to occlusions. If the tracked community disappears behind an obstacle, then the seed vanishes and our tracking technique will not allow to recover the object when it reappears. However, if the occlusion is only partial, then the community can still be tracked. This can be seen in the third and fourth frames of the video displayed in Fig. 4.22, where the right hand starts being occluded by the left hand. However, because there is still a small portion of the right hand that remains visible at the bottom, the community is recovered after the occlusion. One can see that the arms are also tracked even though their communities are only displayed when they are in the neighborhood of the

hands (to maintain the readability of the figure). Although the arms seem oversegmented, similarly as the baby face in Fig. 4.17 which may indicate that either d_Λ should have been chosen smaller or d_{max} larger.

Again, we do not claim that this method can outperform other existing algorithms specifically designed for video tracking purposes. However, the results are convincing and the method can potentially be tuned to reach high overall quality (mainly using ad hoc cost functions). This 3 dimensional segmentation could also be used for $3D$ reconstruction. For example, in medical imaging, one usually takes multiple consecutive scans of the region of interest (brain, hearth, etc.) to discover abnormalities. Hence, this kind of "video" could be segmented using our method to recover the external surface of medically interesting regions.

4.2.4 Boundary of inclusion in microstructure

Finally, let us conclude this chapter with a practical application in which our algorithm has been applied successfully. With the actual development of numerical methods and computer resources, engineers and researchers present a growing interest in being able to model materials behavior based on their experimentally observed microstructures, rather than under simplifying assumptions about their constituent. Materials characterization techniques have reached a point today where it is easy to obtain an image of a material's microstructure. These may include metallography or electron microscopy that are widely available in research laboratories and in the industry. Fig. 4.23 presents an experimental procedure to study the deformation of a Duplex stainless steel consisting of 2 phases. An image of the region of interest is taken after preparation of the surface. Then, the sample is inserted inside an adjusted opening in a plane strain compression specimen made of the same material and sealed by welding. The specimen is quickly heated at 850°C, annealed for 30s to ensure temperature homogeneity, deformed by plane strain compression and finally quenched to limit microstructural evolution. After deformation the sample is carefully extracted from the compressed block and an image of the region of interest after deformation is taken.

In this framework, image-based modeling has been the topic of intense research in the last decade. One of the most popular practices is to build a finite element mesh of the microstructure, as represented in Fig. 4.24, that will serve as input in an appropriate solver. One could build the mesh based on the pixel grid. However this approach may largely increase the

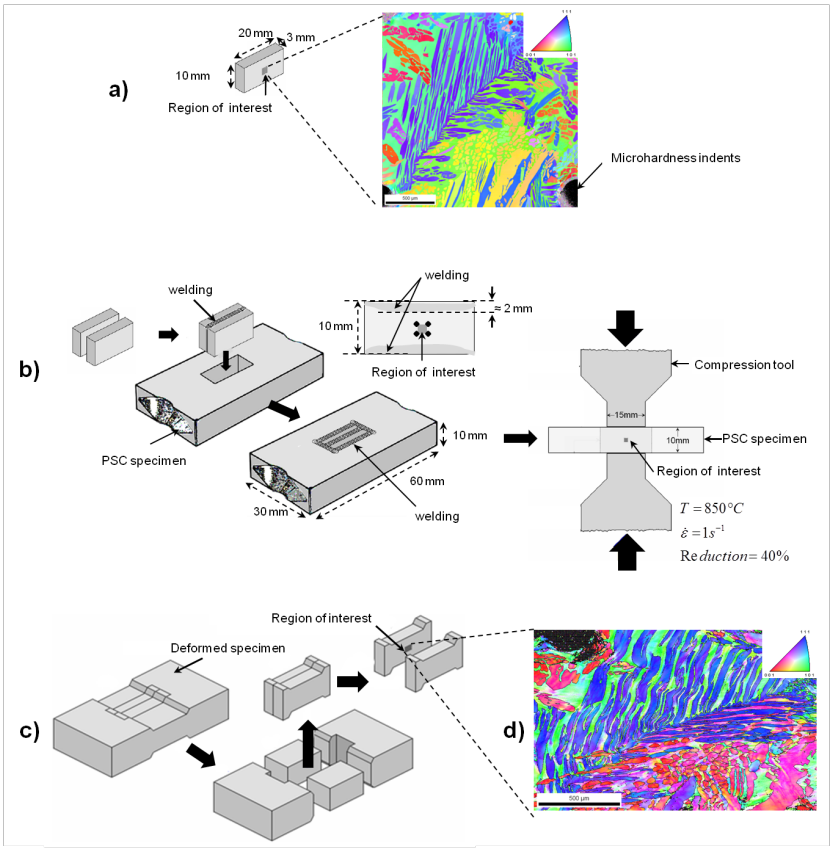


Fig. 4.23 Overview of the experimental procedure to study deformation.

number of elements in the mesh and the boundaries in the microstructure become staircase-shapped which increases the surface of the elements and may affect the computation of the studied fields.

This is where our algorithm has become convenient. Since the communities extracted are necessarily defining connected components of the graph, our method segments closed domains in an input picture, without intersection nor holes between them if the distance d_{max} is kept small. This allows to subsequently apply automated mesh generation algorithms that can usually produce good quality meshes even in the case of complex domain boundaries. Among many other meshing softwares, the free Gmsh project [Geuzaine and Remacle (2009)] provides a powerful environment

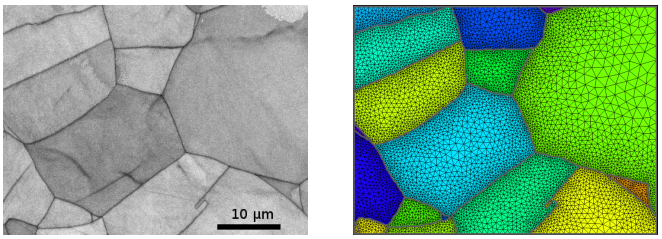


Fig. 4.24 Non uniform mesh generation on picture.

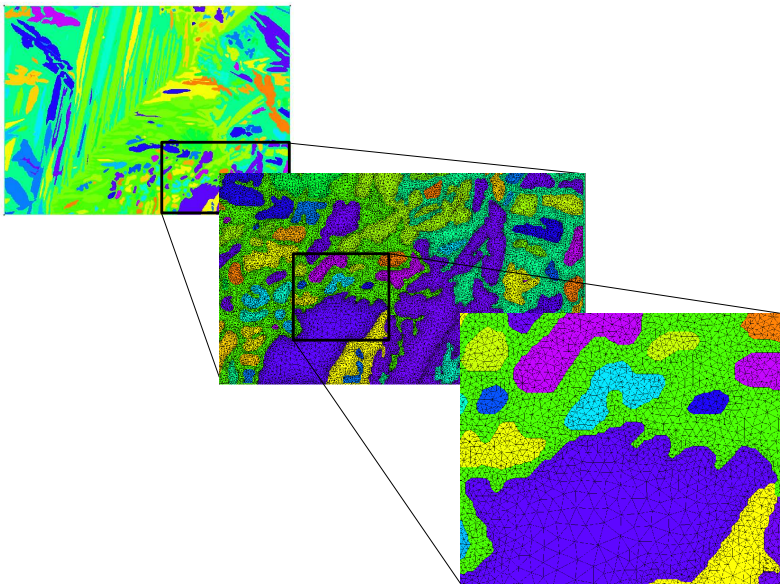
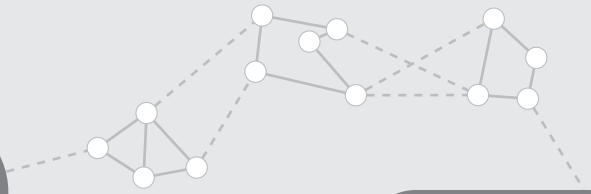


Fig. 4.25 Optimized mesh of the Duplex microstructure.

for this purpose.

The mesh obtained from Gmsh after segmentation of the microstructural image is shown in Fig. 4.25. Using this mesh, qualitative similarity between the experimental deformed microstructure and the simulation has been observed, but the analysis of the simulation goes far beyond the scope of the thesis.



5 Extraction of role structure

Until now, we have extensively presented the problem of community detection and shown that clustering is essential to comprehend large networks and reveal their relevant properties. However, this structural distribution of the nodes in a graph is not always representative. For instance, bipartite networks or cycle graphs do not contain communities in the sense of coherent clusters of vertices. Those kinds of graphs may enclose some densely connected groups of nodes, which could therefore be identified as communities, however any of those communities will always be composed of nodes from different classes and will therefore have a poor interpretation in terms of representative clusters. Though, those graphs may be heavily organized and less attention has been paid to uncover more general structures. This task is known as role extraction or block modeling [Wasserman and Faust (1994); Cason (2012)] and generalizes the concept of community.

In this chapter, we will first state precisely the role extraction problem. Then, we will present in Section 5.2 a model proposed by Reichardt & White which extends the energy function described in Section 3.2.1 for community detection. Yet, this model requires some prior knowledge on the role structure which might be problematic in practice. Therefore, in Section 5.3, we will show that one can handle the role extraction problem by defining a pairwise node similarity measure and thereafter present some of the similarity measures proposed in the literature. In Section 5.4, we will analyze a new efficient similarity measure briefly introduced in [Denayer (2012)] that is based on the number of common neighbors at any distance between each pair of nodes in the graph. Finally, computing the exact pairwise node similarity using this measure is computationally expensive, therefore we will introduce in Section 5.5 a low rank iterative scheme that approximates the similarity score and allows to analyze large networks.

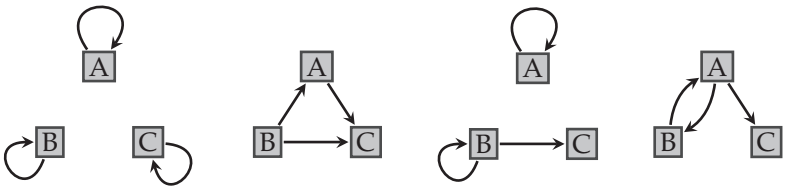
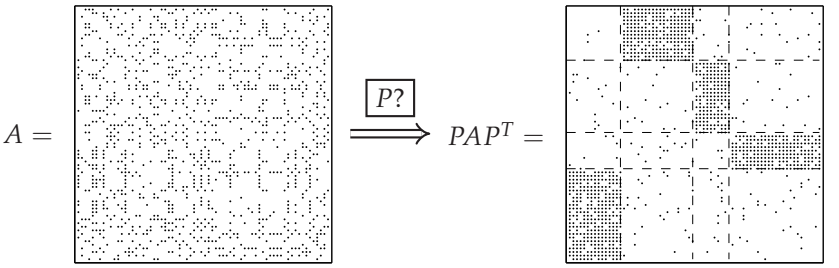


Fig. 5.1 Some examples of 3 roles structures.

5.1 Role model in network

The role extraction problem consists in finding a simple description of a graph that would be accurately represented by this smaller comprehensible structure called the reduced graph or the image graph. This problem is a generalization of the community detection problem. Roles can be broadly defined as groups of nodes that share roughly similar connectivity patterns across the graph. Hence, communities form a particular class of roles where nodes in a role mainly interact with nodes in the same role. However, many other kinds of role interactions may be defined like in a leader-follower model to represent social network interactions or in a block cycle model for food webs. For example, some possible role interactions, represented by their reduced graph, over a 3 roles structure are illustrated in Fig. 5.1. Note that, only the first role structure defines communities. In the following, we will always use ◯ to represent nodes of the original graph and □ to represent roles, i.e. nodes in the reduced graph.

The role extraction problem is sometimes referred as block modeling. This comes from the fact that, based on the adjacency matrix A of the graph, finding a representative role structure is equivalent to finding a permutation matrix P such that the edges of the relabeled graph, associated to the adjacency matrix PAP^T, are mainly concentrated within blocks:



For an in-depth review of block modeling, we refer the interested reader to [Doreian, Batagelj, and Ferligoj (2005)].

The role extraction problem is based on the assumption that nodes can be clustered according to a suitable measure of equivalence. The first relation of equivalence between graph vertices was introduced by [Lorrain and White (1971)] and is called the structural equivalence. Two nodes are structurally equivalent if they have exactly the same neighbors. This implies that all the blocks of size larger than 2, i.e. the equivalence classes, in the permuted adjacency matrix must be either null or complete, with potentially some variations allowed on the diagonal of the diagonal blocks. This measure of equivalence is rather restrictive and tends to extract many small roles when applied to real graphs. Therefore, another relation of equivalence has been introduced called the regular equivalence [White and Reitz (1983); Everett and Borgatti (1994, 1996)]. Two nodes are regularly equivalent if they share similar connections with other equivalence classes, where the notion of similar connection must be specified. In general, two nodes are considered regularly equivalent if they are connected to the same equivalence classes, while the number of such connections does not matter. This implies that the off diagonal blocks in the permuted adjacency matrix must be either null or contain at least one element per line and per column. Regular equivalence is a relaxation of structural equivalence. Clearly, structural equivalence implies regular equivalence but the opposite is not true. Originally, the structural and regular equivalence were strictly combinatorial and therefore not robustly applicable to real networks. However, some works have been done to define algorithms which approximate the optimal equivalence classes [Borgatti and Everett (1993); Luczkovich, Borgatti, Johnson et al. (2003)].

Based on a chosen equivalence criterion, one has to build a quality function to optimize over both the role structure and the role assignment of the nodes in the graph. More precisely, if we denote by B the adjacency

matrix of the reduced graph and by σ the assignment matrix of each node to a role, the problem can be stated as

$$(B^*, \sigma^*) = \arg\max_{B,\sigma} Q_A(B, \sigma)$$

where Q_A depends on the graph topology and the chosen equivalence criterion. One then has to solve a combinatorial optimization problem with respect to 2 variables, and therefore this problem is in general harder than the community detection problem.

The quality function $Q_A(B, \sigma)$ can be constructed either indirectly, based on a (dis)similarity measure between pairs of nodes, or directly, by measuring the fit of the clustering compared to an ideal clustering with perfect relations within and between clusters. For the latter, one has to choose a cost function that is sensitive to a measure of distance between the actual blocks $A(\sigma_i, \sigma_j)$ connecting clusters σ_i and σ_j and ideal blocks between those clusters. The structure of those ideal blocks depends on the relation of equivalence one is interested in. If we denote by $\mathcal{K}(\sigma_i, \sigma_j)$ the set of ideal blocks between the clusters σ_i and σ_j, the problem can then be stated as

$$\sigma^* = \arg\min_{\sigma} \sum_{\sigma_i, \sigma_j} \min_{K \in \mathcal{K}(\sigma_i, \sigma_j)} d\Big(A(\sigma_i, \sigma_j), K\Big),$$

where d is an appropriate measure of distance, for example

$$d\Big(A(\sigma_i, \sigma_j), K\Big) = \sum_{x \in \sigma_i, y \in \sigma_j} |A(x, y) - K(x, y)|\,.$$

Another approach of direct quality function based on the reduced graph has been introduced by [Reichardt and White (2007)] and will be detailed in the next section.

On the other hand, one can build the quality function based on a dissimilarity measure and define clusters as groups of nodes closed to each others. The general idea is to embed the graph vertices in a p-dimensional space based on some structural properties of the nodes. One can use local measures like the degree, the number of neighbors at distance d, the number of triads (i.e. the number of triangles as called by sociologists), etc. but also more global centrality measures like the closeness or the betweenness centrality (see Section 2.4 or [Newman and Girvan (2004); Klein (2010)]). Once the p measures have been computed for each node i and aggregated into an indicator vector t_i, one can compute the pairwise dissimilarity as

the distance between the indicator vectors

$$D(i,j) = d(t_i, t_j)$$

where d can be any type of norm like the Euclidian distance

$$d(t_i, t_j) = \sqrt{\sum_{s=1}^{p} \big(t_i(s) - t_j(s)\big)^2},$$

or the Manhattan distance

$$d(t_i, t_j) = \sum_{s=1}^{p} \big|t_i(s) - t_j(s)\big|.$$

Finally, one can also use a similarity or dissimilarity measure based on the adjacency matrix. This approach is generally more suited when one wants to establish clusters based on a specific equivalence criterion. For example, one can use

$$D(i,j) = \sqrt{\sum_{k} \big(A(i,k) - A(j,k)\big)^2}$$

which is a dissimilarity measure based on structural equivalence since it computes the number of non-common neighbors between nodes i and j. Another measure based on structural equivalence is given by the Pearson correlation between the rows or the columns of the adjacency matrix

$$D(i,j) = \frac{\sum_k (A(i,k) - \mu_i)(A(j,k) - \mu_j)}{n\sigma_i\sigma_j},$$

where $\mu_i = \sum_j A(i,j)/n$ and $\sigma_i = \sqrt{\sum_j (A(i,j) - \mu_i)^2/n}$. In Section 5.3, we will go over a number of types of similarity measures specifically designed for the extraction of role structure based on regular equivalence.

5.2 Quality function: Reichardt & White

The quality function proposed by [Reichardt and White (2007)] is a straightforward extension of Eq. (3.2) introduced for community detection in an unweighted graph. Let us first assume that we know a role structure, represented by its unweighted adjacency matrix B, such that $B(\sigma_i, \sigma_j) = 1$

if edges are allowed from block σ_i to block σ_j. The quality function $Q_{RW}(\sigma, B)$ measures the fit between a partition σ for the graph, represented by its adjacency matrix A, and the reduced graph as

$$Q_{RW}(\sigma, B) = \frac{1}{m} \sum_{i \neq j} \left[a_{ij} A(i,j) B(\sigma_i, \sigma_j) + b_{ij} (1 - A(i,j))(1 - B(\sigma_i, \sigma_j)) \right]. \tag{5.1}$$

where a_{ij} (resp. b_{ij}) rewards the presence (resp. absence) of an edge between nodes i and j if an edge is allowed (resp. forbidden) in the role model between σ_i and σ_j. Therefore, this cost function is based on regular equivalence between the nodes inside each cluster. If one is interested in structural equivalence, penalties for the presence of forbidden edges and for the absence of allowed edges can be added. The terms of Eq. (5.1) can be rearranged such that

$$Q_{RW}(\sigma, B) = Q_0 + \frac{1}{m} \sum_{i \neq j} \Big(\left(a_{ij} + b_{ij} \right) A(i,j) - b_{ij} \Big) B(\sigma_i, \sigma_j). \tag{5.2}$$

where Q_0 does not depend neither on the partition nor on the role structure. Reichardt & White proposed to balance the weight a_{ij} and b_{ij} because often there are not as much existing edges as missing ones. Hence, they imposed that $a_{ij} + b_{ij} = 1, \forall i, j$, and that

$$\sum_{i \neq j} a_{ij} A(i,j) = \sum_{i \neq j} b_{ij} \left(1 - A(i,j)\right).$$

To achieve this, they used $a_{ij} = 1 - p_{ij}$ and $b_{ij} = p_{ij}$ with $\sum_{i \neq j} p_{ij} = \sum_{i \neq j} A(i,j)$. Therefore, p_{ij} can be interpreted as the probability that there exists an edge from node i to node j, and they chose to use the configuration null model to define this probability. In this case, one can note that if the adjacency matrix of the reduced graph B is diagonal (i.e. the roles are communities), then we recover the unweighted modularity given by Eq. (3.13).

As previously, we can write the summation over the cluster indices rather than over the node indices which gives

$$Q_{RW}(B, \sigma) = \frac{1}{m} \sum_{r,s} (e_{rs} - \langle e_{rs} \rangle) B(r,s) \tag{5.3}$$

where e_{rs} is the actual number of edges between clusters r and s and $\langle e_{rs} \rangle$

is the expected number of such edges,

$$e_{rs} = \sum_{i \neq j} (a_{ij} + b_{ij}) A(i,j) \delta(\sigma_i, r) \delta(\sigma_j, s),$$
$$\langle e_{rs} \rangle = \sum_{i \neq j} b_{ij} \delta(\sigma_1, r) \delta(\sigma_j, s).$$

Using Eq. (5.3), one can see that from a given partition σ, we can easily compute the optimal adjacency matrix of the reduced graph. Indeed, maximizing $Q_{RW}(B, \sigma)$ is equivalent to maximizing every single term $(e_{rs} - \langle e_{rs} \rangle) B(r,s)$. Therefore, one simply needs to set

$$B(r,s) = \begin{cases} 1 & \text{if } e_{rs} - \langle e_{rs} \rangle > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

Based on this observation, Reichardt & White suggested to extract the role structure of an input graph by first maximizing

$$Q^*_{RW}(\sigma) = \frac{1}{m} \sum_{r,s} |e_{rs} - \langle e_{rs} \rangle| \tag{5.5}$$

and then defining the reduced graph using Eq. (5.4). The optimization procedure can be handled using any of the algorithms presented in Chapter 3, even though simulated annealing was originally chosen.

This quality function has been successfully applied to uncover the role structure of a commodity trade network [Reichardt and White (2007)]. However, it suffers from some important drawbacks. First, there is absolutely no guarantee that the optimization of Eq. (5.5) effectively extracts a relevant role structure since the reduced graph is removed from the optimizing scheme. Moreover, the method should also work for weighted networks, but an unweighted reduced graph seems inappropriate to represent clusters in a weighted network. Finally, this method is not able to extract a relevant role structure for some simple graphs like a complete bipartite graph. Clearly, the roles are strongly defined in this kind of network since each node within one of the disjoint sets is structurally equivalent to all the nodes within this set. However, one can compute that, for such network,

$$A(i,j) - p_{ij} = A(i,j) - \frac{k_i^{out} k_j^{in}}{m} = 0 \ , \ \forall i,j$$

and therefore, $Q_{RW}(\sigma, B) = 0$ for any partition σ and any reduced graph B. Therefore, the method can not extract any role in this type of graph.

5.3 Pairwise node similarity measures

As we have seen, the definition and the optimization of a direct quality criterion to extract roles in a network are really challenging and might yield unexpected results since one needs to optimize the cost function with respect to both the structure of the reduced graph and the cluster assignment of each node. Hence, indirect quality criteria based on the pairwise node similarity have been developed. The idea of such methods is to somehow compare each node of the input graph with the nodes of the reduced graph by measuring how similar they are in terms of flows or connectivity patterns. More precisely, given an input graph and a reduced graph, one is interested in defining a suitable pairwise node similarity $S(A, B)$ which sets a positive real value for every pair of nodes (i, j) with $i \in V_A$, the node set of A, and $j \in V_B$, the node set of B. Each node of the input graph can then be associated to the role, i.e. a node of the reduced graph, to which it is the most similar.

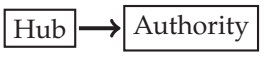
Although this approach seems to suffer from the same inconvenience than a direct quality criterion (that is we need to assume a particular structure for the reduced graph beforehand) one can in general use the criterion function to define a pairwise node self-similarity measure. This means that each node of the input graph is compared to all the nodes of the same graph using $S(A, A)$. Therefore, one can compute the similarity between each pair of nodes and then cluster highly similar nodes together. For example, one can apply a community detection algorithm on the similarity graph, whose weighted adjacency matrix is the pairwise node self-similarity measure $S(A, A)$, since nodes in the same cluster should be highly similar with each other and highly dissimilar with nodes in other clusters, hence revealing a community structure.

In the following sections, we will present some interesting pairwise node similarity measures previously introduced in the literature.

5.3.1 Blondel et al.

The similarity measure proposed by [Blondel, Gajardo, Heymans et al. (2004)] is a generalization of the hub and authority scores proposed by [Kleinberg (1999)] to rank web pages for search engines. The purpose of

the latter is to classify each web page as either a hub, i.e. a web page containing a lot of links pointing to relevant authorities, or an authority, i.e. a web page containing interesting information and hence pointed by many hubs. Therefore, the role structure for this ranking and the associated adjacency matrix are given by



$$B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

A node is an important hub if its outgoing edges are directed to important authorities, and is an important authority if its incoming edges are coming from important hubs. This suggests a reinforcement iteration to compute the scores as

$$h_{k+1} = \frac{A a_k}{\|A a_k\|} \quad , \quad a_{k+1} = \frac{A^T h_k}{\|A^T h_k\|}, \tag{5.6}$$

where h is the vector of hub scores and a the vector of authority scores. The normalization comes from the fact that we are only interested in the relative scores of each node rather than in their absolute values, i.e. a node would be classified as hub only if its hub score is larger than the hub score of the other nodes rather than if it achieves a particular value of hub score. Moreover, we want to ensure the convergence of the sequences h_k and a_k which are initialized with $h_0 = a_0 = \mathbf{1}$. If we denote by $[x \mid y]$ the horizontal concatenation of 2 vectors or matrices x and y, one can rewrite Eq. (5.6), up to the normalization factors, as

$$\begin{aligned} [h_{k+1} \mid a_{k+1}] &= A\,[a_k \mid 0] + A^T\,[0 \mid h_k]\,, \\ &= A\,[h_k \mid a_k] \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + A^T\,[h_k \mid a_k] \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \\ &= A\,[h_k \mid a_k]\,B^T + A^T\,[h_k \mid a_k]\,B. \end{aligned}$$

This illustrates the basic principle of the similarity introduced by Blondel et al. : two nodes $i \in V_A$ and $j \in V_B$ should be highly similar if the children of i are similar to the children of j or if the parents of i are similar to the parents of j. More precisely, the pairwise node similarity measure of Blondel et al. $S^B_{A,B}$ is given by the fixed point solution of

$$S_{k+1} = \frac{A\,S_k\,B^T + A^T\,S_k\,B}{\|A\,S_k\,B^T + A^T\,S_k\,B\|_F}. \tag{5.7}$$

with $S_0 = \mathbf{1}\mathbf{1}^T$. Using the property of the Kronecker product (denoted by $\otimes$) that $A^T X B = (B \otimes A)\, vec(X)$, where $vec(X)$ is the vectorization of the matrix X formed by stacking vertically the columns of X, Eq. (5.7) can be written as

$$s_{k+1} = \frac{\left(B \otimes A + B^T \otimes A^T\right) s_k}{\left\|\left(B \otimes A + B^T \otimes A^T\right) s_k\right\|_2} = \frac{M s_k}{\|M s_k\|_2}, \tag{5.8}$$

where $s_k = vec(S_k)$. One can prove that this sequence converges to $s(s_0) = \frac{\Pi s_0}{\|\Pi s_0\|_2}$ where Π is the orthogonal projector on the invariant subspace of M associated to the dominant eigenvalue $\rho(M)$. However, when $-\rho(M)$ is also an eigenvalue of M, then two converging and alternating sequences are observed $\{s_{even}(s_0), s_{odd}(s_0)\}$. In this case, the authors propose that the sequence $s_{even}(s_0)$ should be used as the pairwise node similarity matrix.

When one does not have a hypothetical role structure available for the graph, the measure can still be used to compute a pairwise node self-similarity $S^B_{A,A} = S^B_A$ as the fixed point solution of

$$S_{k+1} = \frac{A\, S_k\, A^T + A^T\, S_k\, A}{\left\|A\, S_k\, A^T + A^T\, S_k\, A\right\|_F}. \tag{5.9}$$

This measure tends to give higher similarity scores for pairs of nodes involving one high degree node. This can lead to a situation where some nodes are more similar to other nodes than to themselves. Therefore, one needs to apply a scaling of the similarity matrix such that each node is exactly similar to itself, $S(i,i) = 1$, and all other similarity scores $S(i,j) \leq 1$. This can be done using for example a diagonal scaling

$$\overline{S^B_A} = D_S^{-0.5}\, S^B_A\, D_S^{-0.5},$$

where D_S is the diagonal matrix of the unscaled self-similarity scores, $D_S = diag\left(S^B_A(i,i)\right)$.

This pairwise node similarity has been used for example to automatically extract synonyms from a dictionary by assuming that synonyms should have many words in common in their definition and also appear together in the definition of many words. However, the measure is not able to extract relevant similarity scores when the graph associated to A is regular or if A is normal. In those cases, one can prove that the similarity matrix is of rank 1 which implies that, after scaling, it is given by $\overline{S^B_A} = \mathbf{1}\mathbf{1}^T$ which makes the extraction of roles impossible.

Low rank approximation

Computing the exact pairwise similarity matrix of Blondel et al. can be computationally too expensive for large networks because after each iteration of Eq. (5.7) or Eq. (5.9), the number of non null elements in the matrix S_{k+1} tends to increase until this matrix eventually becomes full. Therefore, a low rank iterative scheme that approximates the pairwise similarity matrix has been proposed by [Cason, Absil, and Van Dooren (2013)]. Let $\Gamma_{A,B}[.]$ be the linear operator used in the iterative sequence of the similarity measure,

$$\Gamma_{A,B} : \mathbb{R}^{n\times c} \to \mathbb{R}^{n\times c} : \Gamma_{A,B}[X] = AXB^T + A^TXB, \tag{5.10}$$

where c is the number of roles, $B \in \mathbb{R}^{c\times c}$, and n is the number of nodes in the input graph, $A \in \mathbb{R}^{n\times n}$. One can observe that the iterative solutions of Eq. (5.7) or Eq. (5.9) are solution of the optimization problem

$$S_{k+1} = \underset{\|S\|_F=1}{\arg\max} \langle S, \Gamma\left[S_k\right]\rangle \tag{5.11}$$

where $\langle .\rangle$ denotes here the standard matrix inner product.

Using this observation, a low rank iterative scheme may then be defined as

$$S_{k+1}^{(r)} = f\left(S_k^{(r)}\right) = \underset{S\in\mathbb{R}_{\leq r}^{n\times c}}{\arg\max} \left\langle S, \Gamma^2\left[S_k^{(r)}\right]\right\rangle \tag{5.12}$$

where $S_k^{(r)}$ is the pairwise node similarity of rank at most r at iteration k and $\mathbb{R}_{\leq r}^{n\times c}$ is the set of matrices in $\mathbb{R}^{n\times c}$ of rank at most r,

$$\mathbb{R}_{\leq r}^{n\times c} = \left\{ U\Sigma V^T \;\middle|\; \begin{array}{c} U \in \mathbb{R}^{n\times r}, V \in \mathbb{R}^{c\times r}, \Sigma \in \mathbb{R}^{r\times r} \\ U^TU = I_n, V^TV = I_c, \Sigma \text{ diagonal} \end{array} \right\}.$$

Note that the operator $\Gamma[.]$ (which could be either $\Gamma_{A,B}[.]$ or $\Gamma_{A,A}[.]$) is applied twice in Eq. (5.12) because we are only interested in the fixed point solution of the even sequence when it differs from the odd sequence. In practice, each iterative solution of Eq. (5.12) can be computed using a truncated singular value decomposition (SVD). See [Cason (2012)] for a proof of convergence of the low rank iterative scheme.

5.3.2 Cooper & Barahona

Another pairwise node similarity measure has been introduced by Cooper & Barahona based on the number of paths originating from or leading to each node [Cooper and Barahona (2011); Beguerisse-Diaz, Vangelov, and Barahona (2013)]. First, let us remind that the number of paths of length l from a node i to a node j is given by $[A^l](i,j)$ while the number of such paths from j to i is given by $[A^{T^l}](i,j)$. One can compute the total number outgoing and incoming paths of length l from and to node i as $[A^l\mathbf{1}](i)$ and $[A^{T^l}\mathbf{1}](i)$.

The similarity score of Cooper & Barahona S_A^{CB} is a self-similarity measure based on this total number of paths. Intuitively, if nodes have similar connectivity patterns, they should roughly have the same number of neighbors at various distances, and therefore they should be highly similar. Based on this idea, one can compute an indicator matrix X containing the total number of paths of length $l \leq l_{max}$ as

$$X = \left[\beta A\mathbf{1} \mid \ldots \mid (\beta A)^{l_{max}}\,\mathbf{1} \mid \beta A^T\mathbf{1} \mid \ldots \mid \left(\beta A^T\right)^{l_{max}}\mathbf{1}\right] \tag{5.13}$$

where the first l_{max} columns correspond to the outgoing paths and the last l_{max} columns correspond to the incoming paths. The parameter $\beta = \frac{\alpha}{\rho(A)}$ ensures the convergence of the sequence with $\alpha \in [0,1]$ and $\rho(A)$ the dominant eigenvalue of A, i.e. its spectral radius. The value of α allows to tune the importance of long paths (global connectivity) with respect to short path (local connectivity).

Each row of X serves as an indicator vector of the total flow profile of the associated node. Since the similarity score should reward similar flow profiles, it is natural to define S_A^{CB} as

$$S_A^{CB}(i,j) = \frac{x_i x_j^T}{\|x_i\|\,\|x_j\|}, \tag{5.14}$$

where the scaling ensures $S_A^{CB}(i,i) = 1$ and $S_A^{CB}(i,j) \leq 1$. Note that this similarity matrix can also be computed for $l_{max} \to \infty$ as the normalized

sum of the two iterative sequences

$$S_{k+1}^{out} = A\left(\mathbf{1}\mathbf{1}^T + \left(\frac{\alpha}{\rho(A)}\right)^2 S_k^{out}\right)A^T \tag{5.15}$$

$$S_{k+1}^{in} = A^T\left(\mathbf{1}\mathbf{1}^T + \left(\frac{\alpha}{\rho(A)}\right)^2 S_k^{in}\right)A \tag{5.16}$$

which converge only if $\alpha < 1$.

The main advantage of the similarity score of Cooper & Barahona is that it is in general cheaper to compute than S_A^B when l_{max} is finite and this similarity measure has therefore been successfully applied on different networks [Beguerisse-Díaz, Vangelov, and Barahona (2013); Beguerisse-Díaz, Garduño-Hernández, Vangelov et al. (2013)]. However, much information, like the origins or the destinations of the paths, is lost since the measure only considers the total number of paths by computing the product of the powers of A and $\mathbf{1}$.

5.3.3 Leicht, Holme & Newman

Finally, let us discuss the self-similarity measure of [Leicht, Holme, and Newman (2006)] based on the predicate that a node i should be similar to a node j if i has a neighbor v that is itself similar to j. As a first candidate, the similarity measure of Leicht et al. , S_A^L, could be written as

$$S_A^L(i,j) = \phi \sum_v A(i,v)S(v,j) + \psi\delta(i,j) \tag{5.17}$$

$$= \left[\phi A\, S + \psi I\right](i,j) \tag{5.18}$$

where the first term corresponds to the similarity between j and the neighbors v of i, the second term enforces that each node is similar to itself and ϕ and ψ are parameters to be chosen. Working on Eq. (5.18), one can write it as

$$S_A^L = \psi\,(I - \phi A)^{-1}, \tag{5.19}$$

hence the parameter ψ is a simple multiplicative scaling which will not change the relative similarity scores and can therefore be discarded from the measure. If $\phi < \rho(A)$, then the solution of Eq. (5.19) always exists and can be computed as an infinite power series

$$S_A^L = I + \phi A + \phi^2 A^2 + \ldots \tag{5.20}$$

which shows that the parameter ϕ serves as a scaling factor to weight the number of long paths relatively to the number of short paths. However, applying a constant scaling ϕ^l to A^l seems inappropriate to build a pair-wise node similarity measure based on the number of paths since high degree nodes will necessarily be involved in more paths than low degree nodes. Therefore Leicht et al. relaxed this constrain by allowing a different scaling for each pair of nodes,

$$S_A^L = \sum_{l=0}^{\infty} C_l(i,j) \left[A^l\right](i,j). \tag{5.21}$$

The elements of the scaling matrix $C_l(i,j)$ should be inversely proportional to the expected number of paths from i to j in a random network with the same incoming and outgoing degree sequences. This way, a pair of nodes would be more similar if there are actually more paths between them than what would be observed by chance. To build the scaling matrix C_l, first observe that the number of paths of length l from i to j is given by the i^{th} entry of the vector p_l given by

$$p_l = A^l e_j,$$

where the vector $e_j(t) = \delta(j,t)$. Hence for large l, this vector converges to the dominant eigenvector of A associated to $\lambda := \rho(A)$, and

$$p_{l+1} = \lambda p_l$$

which shows that for large l, the number of paths increase approximately by a factor λ when the length of the paths is increased by 1. Obviously, this holds only in the limit of large l but will be used as a first approximation for small l too. One can however easily correct the number of paths of length 1 which is k_i^{out} and the total number of paths of length l from node i will be approximated by $k_i^{out}\lambda^{l-1}$. Out of all those paths, only a fraction k_j^{in}/m are expected to end up in node j in a random network so the scaling factor can be computed as

$$C_l(i,j) = \frac{m\lambda^{1-l}}{k_i^{out}k_j^{in}}.$$

Finally the pairwise node similarity reads

$$S_A^L(i,j) = \delta(i,j) + \frac{m\lambda}{k_i^{out}k_j^{in}} \sum_{l=1}^{\infty} \left(\frac{\alpha}{\lambda}\right)^l \left[A^l\right](i,j) \tag{5.22}$$

$$= \left[\left(1 - \frac{m\lambda}{k_i^{out}k_j^{in}}\right) I + \frac{m\lambda}{k_i^{out}k_j^{in}} \left(I - \frac{\alpha}{\lambda}A\right)^{-1}\right](i,j) \tag{5.23}$$

where the scaling factor $\alpha < 1$ is introduced to ensure the convergence of the sequence. Leicht et al. claim that an appropriate value is $\alpha = 0.97$. The first term of Eq. (5.23) only affects how each node is similar to itself and can be dropped from the final solution which is then given by

$$S_A^L = m\lambda \left(K^{out}\right)^{-1} \left(I - \frac{\alpha}{\lambda}A\right)^{-1} \left(K^{in}\right)^{-1} \tag{5.24}$$

where K is the diagonal matrix of in- or out-degrees $K = diag(k_i)$. In practice, one can compute this similarity matrix as the fixed point solution of the iteration

$$K^{out}\, S_{k+1}\, K^{in} = \frac{\alpha}{\lambda} A\, K^{out} S_k\, K^{in} + I \tag{5.25}$$

where one can note that the scaling $m\lambda$ has been removed since it does not change the relative similarity score.

This similarity measure has been applied to various networks containing some kind of hierarchical structures. However, it seems not suited to analyze the type of networks that we are interested in here. We expect that the edges of a graph will mostly connect nodes that are not similar, and instead that the similarity between nodes lies more in their connectivity patterns (which will be made clear in what follows), like for example in N-partite networks where all the edges connect nodes in different classes. In fact, the similarity measure of Leicht et al. is more suited to analyze networks with a community structure rather than a role structure.

5.4 Neighborhood patterns based similarity

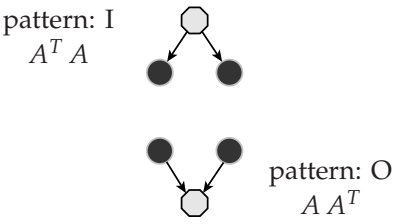
We consider that nodes sharing the same kind of behavior, i.e. having the same role, should have similar flow patterns across the network which can be somehow measured by comparing the neighborhood connectivity of the nodes, as for the structural equivalence criterion. We have just shown that this can be done using the similarity measure proposed by Blondel et al. and that, for this measure, the similarity matrix obtained can be easily

and well approximated using a low rank iterative scheme. However, it seems uneasy to interpret the measure in terms of neighborhood connectivity and we mentioned that it can not be used to extract role structure when the adjacency matrix of the graph is regular or normal, hence for undirected graphs. Another approach introduced by Cooper & Barahona computes an indicator vector for each node based on the number of its neighbors but much information gets lost in the process, i.e. the origin, the destination and the intermediate nodes involved in the transmission of the flow. Finally, we discussed the measure of Leicht et al. which has a clear interpretation and formalism but is much more appropriate to detect communities than role structures. In what follows, we will present a new pairwise node similarity measure introduced by [Denayer (2012)] which encompasses the different assets of the previously described measures. Our first contribution has been to precisely analyze the measure and its convergence properties, and to compare its behavior with the other methods on specific graphs which will be the topic of this section. Our next major contribution has been to define a low-rank approximation of the similarity matrix to allow the extraction of role structure in very large graphs, which will be presented in a consecutive section. Lastly, we will apply in the next chapter our low-rank similarity measure on benchmark and real graphs to illustrate its effectiveness.

Let us first define a neighborhood pattern of length ℓ for a node as a sequence of length ℓ of incoming (I) and outgoing (O) edges starting from the node, which we will call the *source* node. For example, the neighborhood patterns of length 1 consist in exactly one edge and end up either in a parent (pattern I) or in a child (pattern O) of the source. If we consider neighborhood patterns of length 2, then 4 different types of nodes can be reached: the parent of a parent (pattern I-I), the child of a parent (pattern I-O), the parent of a child (pattern O-I) or the child of a child (pattern O-O). One can easily see that when the length of the neighborhood patterns is increased by 1, the total number of types of reachable nodes, which we will call the *target* nodes, is doubled.

Our similarity measure should reflect that a pair of nodes is highly similar if they have many neighborhood patterns in common. In other words, as sources of the sequences, a pair of similar nodes should reach many common targets with the same neighborhood patterns and do so for patterns of various lengths. Let us illustrate this concept. Using the patterns of length 1, two source nodes will be more similar if they have many common parents (pattern I) or many common children (pattern O). If we

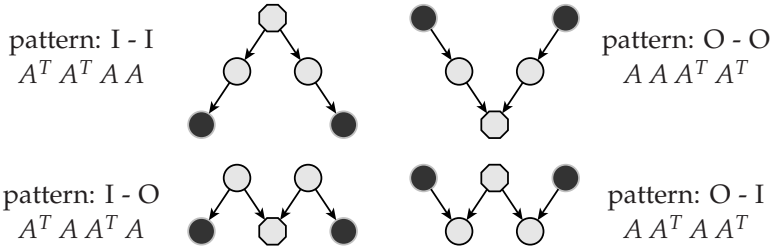
represent the source nodes as dark circles and the common target node as an octagon, this means that the graph topology around two similar source nodes should be one of the following



One can compute the number of common parents between a pair of nodes (i, j) as $\left[A^T A\right](i, j)$ while the the number common children is given by $\left[A A^T\right](i, j)$. Therefore, we can compute the number of common target nodes between every pair of source nodes using neighborhood patterns of length 1 as

$$T_1 = AA^T + A^T A.$$

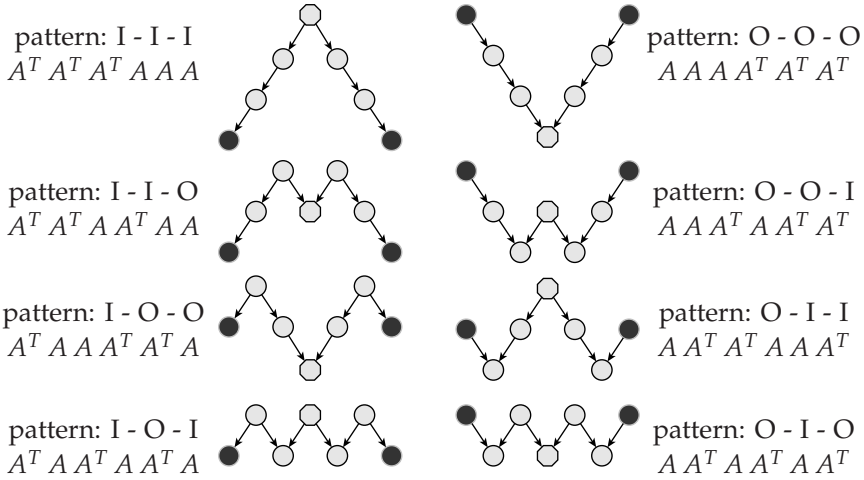
Similarly, using neighborhood patterns of length 2, the different graph topologies around 2 similar source nodes are represented by



and the number of common target nodes for neighborhood patterns of length 2 is given by

$$T_2 = AAA^T A^T + AA^T AA^T + A^T AA^T A + A^T A^T AA.$$

Let us represent, as a final example, the available neighborhood patterns of length 3



from which the number of common targets T_3 can be trivially computed.

We define our pairwise node similarity measure $S \in \mathbb{R}^{n\times n}$ as the weighted sum of the number of common target nodes using neighborhood patterns of any length

$$S = \sum_{\ell=1}^{\infty} \beta^{2(\ell-1)} T_\ell, \tag{5.26}$$

where $\beta \in \mathbb{R}$ is a scaling parameter. As in the previous similarity measures, the value of β balances the relative importance of long neighborhood patterns with respect to short neighborhood patterns since the number of common targets tends to naturally grow when using longer patterns. This similarity measure was briefly introduced in [Denayer (2012)] and resembles the similarity measure of Leicht et al. given by Eq. (5.21), although considering totally different types of node connectivity patterns that are more appropriate to study networks containing a role structure.

Using the linear operator $\Gamma_A[.] = \Gamma_{A,A}[.]$ defined by Eq. (5.10), one can see that

$$\begin{aligned} T_1 &= \Gamma_A\left[I\right], \\ T_2 &= \Gamma_A\left[T_1\right] = \Gamma_A^2\left[I\right], \\ T_3 &= \Gamma_A\left[T_2\right] = \Gamma_A^3\left[I\right], \end{aligned}$$

where $\Gamma_A^k[.]$ corresponds to applying k times the operator $\Gamma_A[.]$. Therefore

$$S = \sum_{\ell=1}^{\infty} \beta^{2(\ell-1)} \Gamma_A^{\ell}[I], \tag{5.27}$$

which can be computed as the limit when $k \to \infty$ of the iterative sequence

$$S_{k+1} = \Gamma_A \left[I + \beta^2 S_k \right], \tag{5.28}$$

since for an initial matrix S_0, we have

$$S_{k+1} = \Gamma_A [I] + \cdots + \left(\beta^2\right)^k \Gamma_A^{k+1} [I] + \left(\beta^2\right)^{k+1} \Gamma_A^{k+1} [S_0] \tag{5.29}$$

Hence, our similarity measure is given by the fixed point solution of the iterative sequence Eq. (5.28)

$$S = \Gamma_A [I] + \beta^2 \Gamma_A [S]. \tag{5.30}$$

Note that our similarity matrix S is necessarily symmetric and positive semidefinite since it is computed as the weighted sum of infinitely many symmetric and positive semidefinite matrices.

The parameter β can be tuned to vary the relative weight of long neighborhood patterns but must be chosen wisely to ensure the convergence of the sequence S_k. If we initialize $S_0 = 0$, the iterative sequence of Eq. (5.28) can be written for $k \geq 1$ as

$$S_{k+1} = S_1 + \beta^2 \Gamma_A [S_k], \tag{5.31}$$

where

$$S_1 = \Gamma_A [I] = T_1 = AA^T + A^T A, \tag{5.32}$$

and the fixed point solution of Eq. (5.31) is then given by

$$S = S_1 + \beta^2 \left(ASA^T + A^T SA \right),$$

if the sequence converges. Using the same property of the Kronecker product than for S_A^B, this fixed point solution can be written as

$$vec(S) = \left[I - \beta^2 \left(A \otimes A + (A \otimes A)^T \right) \right]^{-1} vec\,(S_1).$$

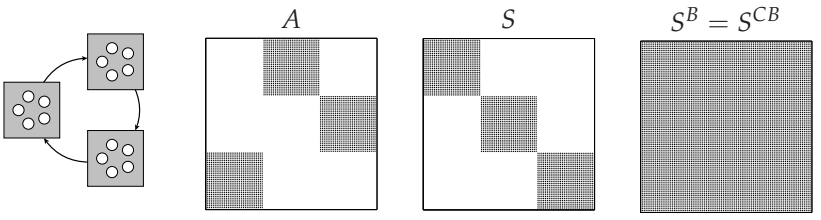


Fig. 5.2 Block cycle role structure and the associated node similarities.

It follows that, to ensure convergence, one can choose β such that

$$\beta^2 < \frac{1}{\rho\left(A \otimes A + (A \otimes A)^T\right)} \tag{5.33}$$

Computing exactly this upper bound for the parameter β to ensure convergence might be computationally expensive due to the Kronecker products $A \otimes A \in \mathbb{R}^{n^2 \times n^2}$ if A is not symmetric. However, one can easily compute a more restrictive bound

$$\beta^2 \leq \frac{1}{\rho\left((A + A^T) \otimes (A + A^T)\right)} = \frac{1}{\rho\left((A + A^T)\right)^2} \tag{5.34}$$

which ensures that the constraint of Eq. (5.33) is satisfied. This bound is clearly not tight to the actual upper bound, however we did not find any closer bound which can be computed easily.

Note that our similarity measure S is somehow linked to the similarity of Blondel et al. S^B (Section 5.3.1). Indeed, the actual upper bound for the parameter β is exactly the inverse of the dominant eigenvalue associated the invariant subspace which defines the orthogonal projector to compute the solution $S^B(s_0)$. Moreover, one can see that the iterative sequences are not independent, and while one can not compute S^B from S (nor vice versa), if we choose a matrix Φ such that the initial value of $S_0^B = I + \beta^2\Phi$, then

$$S_1^B = \frac{S_1 + \beta^2\Gamma\left[\Phi\right]}{\left\|S_1 + \beta^2\Gamma\left[\Phi\right]\right\|_F}$$
$$S_{k+1}^B = \frac{S_{k+1} - S_k + \beta^{2(k+1)}\Gamma^{k+1}\left[\Phi\right]}{\alpha_k} \rightarrow \frac{S_{k+1} - S_k}{\alpha_k}$$

where $\alpha_k = \left\| S_{k+1} - S_k + \beta^{2(k+1)} \Gamma^{k+1} [\Phi] \right\|_F$. However, the similarity measure of Blondel et al. has some drawbacks that are avoided using our iterative scheme defined by Eq. (5.31). First, one can see that the sequence S_k converges for any initial matrix S_0 to a unique fixed point solution. From Eq. (5.29), it is clear that when $k \to \infty$, $\left(\beta^2\right)^{k+1} \Gamma_A^{k+1} [S_0] \to 0$ since the spectral radius of the linear operator $\rho\left(\beta^2 \Gamma [.]\right) < 1$. This also supports our choice $S_0 = 0$. On the other hand, S^B strongly depends on the initialization of its iterative sequence and, as we have seen, the solution may alternate between 2 converging sequences. The choices $S_0^B = \mathbf{1}\mathbf{1}^T$ and $S = S_{even}(S_0)$ seem very arbitrary and not based on relevant properties of the measure. The argument brought by the authors, that is $S_{even}(\mathbf{1}\mathbf{1}^T)$ is the matrix of largest 1-norm, is unconvincing. Moreover, it is known that the similarity score of Blondel et al. S^B is of rank 1 when the adjacency matrix A is normal. For example, let us consider a regular block cycle graph, as represented in Fig. 5.2 along with its adjacency matrix, where each role contains the same number of nodes and each node is connected to all the nodes in the following role in the cycle. The similarity matrix S^B is of rank 1 for this role structure and therefore, after scaling, $\overline{S^B}$ is the matrix of all ones which makes the extraction of the role structure impossible. On the contrary, our measure computes in this case a similarity matrix of rank equal to the number of roles in the network, with an obvious clustering that reveals the different roles.

Our similarity measure can also be seen as a generalization of the measure proposed by Cooper & Barahona for which the pairwise similarity S^{CB} only compares the total number of paths originating or leading to a node, without comparing the targets nor the sources of those paths. Furthermore, this similarity score S^{CB} does not consider all the types of neighborhood patterns that we introduced, but only restricts the measure to direct paths (which are defined by the patterns I-I-...-I and O-O-...-O). Therefore, the pairwise node similarity for the regular block cycle graph, illustrated in Fig. 5.2, is of rank 1 because all the nodes have a constant number of in/out neighbors at any distances. This makes the extraction of the role structure of this network also impossible using S^{CB}. One can see that any 2 nodes of the same role in the input graph are isomorphic, while any 2 nodes of different roles are not. This is accurately represented by our measure S but not by S^{CB}.

Unfortunately, even if β is small enough to guarantee the convergence of the sequence given by Eq. (5.31), it might be impossible to compute

the fixed point solution up to a small tolerance because of the increasing computational cost and memory requirements. Indeed, even if A is sparse, the matrix S_k tends to fill in as k increases and the complexity of each single iteration of Eq. (5.31) is $O(mn^2)$. This leads us to define a low-rank projected iteration to approximate the pairwise node similarity matrix.

5.5 Low-rank similarity approximation

Because the full rank fixed point solution of Eq. (5.31) is often computationally too expensive to extract, let us introduce a low-rank iterative scheme that approximates the pairwise node similarity matrix S with a matrix $S^{(r)}$ of rank at most r. We will thereafter prove the convergence of this low rank iteration.

Let us first assume that we have a low rank approximation of the solution at iteration k, $S_k^{(r)}$. Using the same formulation as Eq. (5.31), we define the low rank iterative scheme as

$$S_{k+1}^{(r)} = \Pi^{(r)}\left[S_1^{(r)} + \beta^2 \Gamma_A\left[S_k^{(r)}\right]\right] = X_{k+1}\, X_{k+1}^T \tag{5.35}$$

where $X_{k+1} \in \mathbb{R}^{n\times r}$ and $\Pi^{(r)}[.]$ is the best low-rank projector on the dominant subspace of dimension at most r which can be computed using a truncated singular value decomposition (SVD). $S_1^{(r)}$ is the best low-rank approximation of S_1 which can be written as

$$S_1 = AA^T + A^TA = \left[A \mid A^T\right]\left[A \mid A^T\right]^T, \tag{5.36}$$

for which we remind that $\left[A \mid A^T\right]$ is the horizontal concatenation of A and A^T. This allows us to efficiently compute $S_1^{(r)}$ which is naturally defined as

$$S_1^{(r)} = \Pi^{(r)}\left[\left[A \mid A^T\right]\left[A \mid A^T\right]^T\right]$$

For this, we first compute a singular value decomposition of $\left[A \mid A^T\right]$, which can be done efficiently using sparse matrix algorithms, e.g. ARPACK,

$$\left[A \mid A^T\right] = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T \tag{5.37}$$

where the columns of the unitary matrix $U_1 \in \mathbb{R}^{n\times r}$ span the dominant subspace of dimension at most r of $\left[A \mid A^T\right]$ and $\Sigma_1 \in \mathbb{R}^{r\times r}$ is the diag-

onal matrix of the dominant singular values, $\Sigma_1(i,i) > \Sigma_2(j,j), \forall i,j$. This leads to

$$\left[A \mid A^T\right]\left[A \mid A^T\right]^T = U_1\Sigma_1^2U_1^T + U_2\Sigma_2^2U_2^T \tag{5.38}$$

which implies that the best low rank projection of S_1 is given by

$$S_1^{(r)} = U_1\Sigma_1^2U_1^T = X_1X_1^T \tag{5.39}$$

To compute each iterative solution of Eq. (5.35), one can see that

$$\begin{aligned} S_1^{(r)} + \beta^2\Gamma_A\left[S_k^{(r)}\right] &= X_1X_1^T + \beta^2AX_kX_k^TA^T + \beta^2A^TX_kX_k^TA \\ &= Y_k\,Y_k^T \end{aligned}$$

where

$$Y_k = \left[X_1 \mid \beta AX_k \mid \beta A^TX_k\right]. \tag{5.40}$$

Therefore the matrix $Y_kY_k^T$ is of rank at most $3r$ and

$$X_{k+1}X_{k+1}^T = \Pi^{(r)}\left[Y_kY_k^T\right]. \tag{5.41}$$

Finally, to efficiently compute X_{k+1}, we first apply a QR factorization of Y_k

$$Y_k = Q_kR_k \tag{5.42}$$

for which the computational cost can be reduced by keeping the first r columns of Q_{k-1} and R_{k-1} since those correspond to the factorization of X_1 which is constant for all k. Then, we compute a truncated SVD of rank at most r of R_k such that

$$R_k \approx \mathcal{U}_k\Omega_k\mathcal{V}_k \tag{5.43}$$

and finally compute X_{k+1} as

$$X_{k+1} = Q_k\mathcal{U}_k\Omega_k. \tag{5.44}$$

One can prove using perturbation theory [Stewart (1973)] that the iterative scheme defined by Eq. (5.35) converges locally to a fixed point solu-

tion

$$S^{(r)} = XX^T = \Pi^{(r)} \left[YY^T \right],$$
$$YY^T = \left[X_1 \mid \beta AX \mid \beta A^T X \right].$$

if the spectral gap of YY^T at the r^{th} eigenvalue is sufficiently large. Let us recall the perturbation theorem which is stated as follows:

Theorem (4.11 Stewart (1973)). *Let $A, E \in \mathbb{C}^{n\times n}$. Let $X = [X_1 \mid X_2]$ be unitary with $X_1 \in \mathbb{C}^{n\times r}$ and suppose that $span(X_1)$ is an invariant subspace of A. Let $X^H AX$ and $X^H EX$ be partitioned conformally with X in the forms*

$$X^H AX = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

and

$$X^H EX = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix}.$$

Let

$$\delta = sep(A_{11}, A_{22}) - \|E_{11}\| - \|E_{22}\|,$$

where $sep(A, B)$ is the eigenvalue gap between A and B,

$$sep(A, B) = min\,|\lambda(A) - \lambda(B)|$$

(when A and B are hermitian matrices).
If

$$\frac{\|E_{21}\|\,(\|A_{12}\| - \|E_{12}\|)}{\delta^2} \leq \frac{1}{4}, \tag{5.45}$$

then there is a matrix P satisfying

$$\|P\| \leq 2\frac{\|E_{21}\|}{\delta}, \tag{5.46}$$

such that $X_1^{'} = (X_1 + X_2 P)\left(I + P^H P\right)^{-1/2}$ span an invariant subspace of $A + E$.

We will show that we are indeed satisfying the assumption of the the-

orem and use it to prove the local convergence of our low rank iterative scheme.

First, we consider the function

$$f(S) = S_1^{(r)} + \beta^2 \Gamma_A [S],$$

which defines the low rank iteration before the projection on the dominant subspace. Then, let us define the function

$$g(\beta, S) = S - \Pi^{(r)} [f(S)] \tag{5.47}$$

with a slight abuse of notation, since $f(S)$ depends on β, that hopefully will not confuse the reader. Clearly, there is at least one solution of $g(\beta, S) = 0$ given by

$$g\left(0, S_1^{(r)}\right) = S_1^{(r)} - \Pi^{(r)} \left[S_1^{(r)}\right], \tag{5.48}$$

$$= S_1^{(r)} - S_1^{(r)} = 0. \tag{5.49}$$

Using the implicit function theorem [Weir, Hass, and Thomas (2013)], this implies that there exists a low rank matrix $S^{(r)}$ such that, for β sufficiently close to 0,

$$g\left(\beta, S^{(r)}\right) = S^{(r)} - \Pi^{(r)} \left[f\left(S^{(r)}\right)\right] = 0. \tag{5.50}$$

This requires that the low projector $\Pi^{(r)}$ is differentiable which is the case if there is a non-negative singular value gap at the r^{th} singular value of $S_1^{(r)}$. Therefore, we know that there exists a fixed point solution of our low rank iterative scheme

$$S^{(r)} = \Pi^{(r)} \left[f\left(S^{(r)}\right)\right]. \tag{5.51}$$

Since $S^{(r)}$ is a symmetric positive semidefinite matrix of rank at most r, we can express it as

$$S^{(r)} = XX^T = U\Sigma^2 U^T \tag{5.52}$$

with $X, U \in \mathbb{R}^{n \times r}$, $U^T U = I$, and $\Sigma \in \mathbb{R}^{r \times r}$ diagonal. Moreover, since $S^{(r)}$ is a fixed point solution of the low rank iteration, it implies that U is also

the dominant subspace of $f\left(S^{(r)}\right)$ which can then be expressed as

$$f\left(S^{(r)}\right) = U\Sigma^2 U^T + V\sigma^2 V^T \tag{5.53}$$

with $V \in \mathbb{R}^{n\times 2r}$, $\sigma \in \mathbb{R}^{2r\times 2r}$ diagonal, $\Sigma(i,i) > \sigma(j,j)\ \forall i,j$ because we assumed that the fixed point solution has a positive spectral gap at the r^{th} singular value, and $U^T V = 0$. This can also be written in the form of the theorem as

$$[U\ V]^T\ f(S^{(r)})\ [U\ V] = \begin{bmatrix} \Sigma^2 & \\ & \sigma^2 \end{bmatrix} \tag{5.54}$$

Then, let us consider a small symmetric perturbation Δ around the low rank fixed point solution. Using the linearity of the operator $\Gamma_A[.]$, one can write that

$$\begin{aligned} f(S^{(r)} + \Delta) &= S_1^{(r)} + \beta^2 \Gamma_A \left[S^{(r)} + \Delta\right], && (5.55)\\ &= S_1^{(r)} + \beta^2 \Gamma_A \left[S^{(r)}\right] + \beta^2 \Gamma_A \left[\Delta\right], && (5.56)\\ &= f(S^{(r)}) + \beta^2 \Gamma_A \left[\Delta\right]. && (5.57) \end{aligned}$$

Clearly, U is in general not an invariant subspace of $f(S^{(r)} + \Delta)$ and

$$[U\ V]^T\ \left(f(S^{(r)}) + \beta^2 \Gamma[\Delta]\right)\ [U\ V] = \begin{bmatrix} F_{11} & F_{21}^T \\ F_{21} & F_{22} \end{bmatrix}. \tag{5.58}$$

with $F_{21} \neq 0$. However, if the condition of the perturbation theorem holds, then we know that there exists a unitary transformation $Q \in \mathbb{R}^{n\times n}$ such that the first r columns of the matrix

$$[U'\ V'] = [U\ V]Q$$

span an invariant subspace of $f(S^{(r)} + \Delta)$. The matrix Q is given by

$$Q = \begin{pmatrix} I & -P^T \\ P & I \end{pmatrix} \begin{pmatrix} (I + P^T P)^{-1/2} & 0 \\ 0 & (I + PP^T)^{-1/2} \end{pmatrix} \tag{5.59}$$

where P is solution of the non-linear equation

$$P\,F_{11} - F_{22}\,P = F_{21} - P\,F_{21}^T P. \tag{5.60}$$

Equivalently, the first r columns of $[U\ V]\,Q$ are given, as stated by the perturbation theorem, by

$$U^{'} = (U + VP)\left(I + P^T P\right)^{-1/2}, \tag{5.61}$$

and

$$([U\ V]\,Q)^T\ \left(f(S^{(r)}) + \beta^2\Gamma[\Delta]\right)\ ([U\ V]\,Q) = \begin{bmatrix} \widetilde{F}_{11} & 0 \\ 0 & \widetilde{F}_{22} \end{bmatrix}. \tag{5.62}$$

Let us now prove that, indeed, the hypothesis of the perturbation theorem holds here. Since $\beta^2\Gamma[\Delta]$ is a small symmetric perturbation, we can write, in the same form as the theorem,

$$[U\ V]^T\ \beta^2\Gamma[\Delta]\ [U\ V] = \begin{bmatrix} E_{11} & E_{21}^T \\ E_{21} & E_{22} \end{bmatrix}, \tag{5.63}$$

and we need to prove that

$$\frac{\|E_{21}\|^2}{\delta^2} \leq \frac{1}{4} \tag{5.64}$$

since $E_{12} = E_{21}^T$ and $V^T f(S^{(r)})U = 0$.

As previously stated, we need the matrix $S^{(r)}$ to have a non null spectral gap at the r^{th} singular value. Therefore, if the matrices Σ and σ are sorted in decreasing order,

$$sep(\Sigma^2, \sigma^2) = \Sigma_k^2 - \sigma_1^2 > 0,$$

and Eq. (5.64) can be written as

$$2\,\|E_{21}\| \leq \delta \tag{5.65}$$

$$2\,\|E_{21}\| + \|E_{11}\| + \|E_{22}\| \leq \Sigma_k^2 - \sigma_1^2 \tag{5.66}$$

An appropriate norm is the Frobenius norm for which we know that

[Horn and Johnson (1990)]

$$\|ABC\|_F \leq \|A\|_2 \|B\|_F \|C\|_2$$

and therefore

$$\|E_{21}\|_F = \beta^2 \left\| V^T \Gamma[\Delta] U \right\|_F \tag{5.67}$$

$$\leq \beta^2 \|V\|_2 \|\Gamma[\Delta]\|_F \|U\|_2 = \beta^2 \|\Gamma[\Delta]\|_F \tag{5.68}$$

since the matrices U and V are isometries, so $\|V\|_2 = \|U\|_2 = 1$. The exact same upper bound can be obtained for the other matrices E_{ij} which leads to an upper bound for the left hand side of equation Eq. (5.66) as

$$2\|E_{21}\|_F + \|E_{11}\|_F + \|E_{22}\|_F \leq 4\beta^2 \|\Gamma[\Delta]\|_F \tag{5.69}$$

$$\leq 4\beta^2 \left\| A \otimes A + A^T \otimes A^T \right\|_2 \|\Delta\|_F \tag{5.70}$$

Therefore, if Δ is not too large, in the sense that

$$4\beta^2 \left\| A \otimes A + A^T \otimes A^T \right\|_2 \|\Delta\|_F \leq \Sigma_k^2 - \sigma_1^2, \tag{5.71}$$

then Eq. (5.66) holds and the upper bound on the norm of the matrix P is given by

$$\|P\| \leq \frac{2\beta^2 \|\Gamma[\Delta]\|_F}{s_k^2 - \sigma_1^2 - 2\beta^2 \|\Gamma[\Delta]\|_F}. \tag{5.72}$$

This proves that one can compute an invariant subspace of the perturbed matrix $f(S^{(r)} + \Delta)$ from the dominant and invariant subspace U of $f(S^{(r)})$ (which is unknown in practice of course). But, we can also use this result to prove the local convergence of our low rank iterative sequence.

We know that U' is an invariant subspace of $f(S^{(r)} + \Delta)$. However, if the perturbation Δ is not too large, the eigenvalues of $f(S^{(r)} + \Delta)$ will be close to the eigenvalues of $f(S^{(r)})$ and the rotation matrix Q will not perturb too much the dominant subspace of $f(S^{(r)})$. Therefore, we can safely assume that the set of eigenvalues of the perturbed low-rank fixed point solution $\lambda\left(f(S^{(r)} + \Delta)\right) = \lambda\left(\widetilde{F}_{11}\right) \cup \lambda\left(\widetilde{F}_{22}\right)$ will be such that

$$\lambda_i\left(\widetilde{F}_{11}\right) > \lambda_j\left(\widetilde{F}_{22}\right), \quad \forall i, j, \tag{5.73}$$

i.e. $\widetilde{F}_{11}$ is somehow a perturbation of Σ and $\widetilde{F}_{22}$ a perturbation of σ. Therefore, $U^{'}$ will not only be an invariant, but also the dominant subspace of $f(S^{(r)}+\Delta)$.

From this, we can write the solution of the low rank projection of $f(S^{(r)}+\Delta)$ as

$$\Pi^{(r)}\left[f(S^{(r)}+\Delta)\right] = U'\,U'^{T}\,f(S^{(r)}+\Delta)\,U'\,U'^{T} \tag{5.74}$$

where the matrix $U'\,U'^{T}$ is given by

$$U'\,U'^{T} = (U+VP)(I+P^{T}P)^{-1}(U^{T}+P^{T}V^{T}). \tag{5.75}$$

Let us analyze separately the projections of the 2 terms of Eq. (5.57). First, note that

$$\begin{aligned}(U^{T}+P^{T}V^{T})\,f(S^{(r)})\,(U+VP) =& \Sigma^{2}+P^{T}\sigma^{2}\,P & (5.76)\\ =& \Sigma^{2}+O(\Delta)^{2} & (5.77)\end{aligned}$$

since $\|P\| \leq O(\|\Delta\|)$ from Eq. (5.72). Then, one can write, as a first order approximation by dropping all the quadratic terms in P that are $O(\Delta)^{2}$, that

$$U'U'^{T}f(S^{(r)})U'U'^{T} = U\Sigma^{2}U^{T}+VP\Sigma^{2}U^{T}+U\Sigma^{2}P^{T}V^{T}+O(\Delta)^{2} \tag{5.78}$$

Similarly, for the term $\beta^{2}\Gamma[\Delta]$ of Eq. (5.57),

$$(U^{T}+P^{T}V^{T})\,\beta^{2}\Gamma[\Delta]\,(U+VP) \quad = \quad E_{11}+O(\Delta)^{2} \tag{5.79}$$

since all the terms involving E_{ij} and P at the same time are in $O(\Delta)^{2}$. This leads to

$$U'U'^{T}\beta^{2}\Gamma[\Delta]U'U'^{T} \quad = \quad UE_{11}U^{T}+O(\Delta)^{2} \tag{5.80}$$

Finally, this shows that

$$\begin{aligned}\left\|S^{(r)} - \Pi^{(r)}\left[f(S^{(r)}+\Delta)\right]\right\| &= \left\|U\Sigma^2U^T - U'U'^Tf(S^{(r)}+\Delta)U'U'^T\right\| \\ &= \left\|VP\Sigma^2U^T + U\Sigma^2P^TV^T + UE_{11}U^T + O(\Delta)^2\right\| \\ &\leq \|P\|\left\|\Sigma^2\right\| + \|P\|\left\|\Sigma^2\right\| + \|E_{11}\| + \left\|O(\Delta)^2\right\| \\ &\leq 2\left\|\Sigma^2\right\|\frac{2\beta^2\|\Gamma[\Delta]\|}{s_k^2 - \sigma_1^2 - 2\beta^2\|\Gamma[\Delta]\|} + \beta^2\|\Gamma[\Delta]\|\end{aligned}$$

However, we need to impose

$$4\beta^2\|\Gamma[\Delta]\| \leq \Sigma_k^2 - \sigma_1^2 \tag{5.81}$$

to satisfy the hypothesis of the perturbation theorem and guarantee the existence of a bounded matrix P. Working on this equation shows

$$\frac{1}{\Sigma_k^2 - \sigma_1^2 - 2\beta^2\|\Gamma[\Delta]\|} \leq \frac{2}{\Sigma_k^2 - \sigma_1^2}. \tag{5.82}$$

Finally, one obtains

$$\left\|S^{(r)} - \Pi^{(r)}\left[f(S^{(r)}+\Delta)\right]\right\| \leq \beta^2\left\|A\otimes A + A^T\otimes A^T\right\|_2\left(\frac{8\left\|\Sigma^2\right\|_F}{\Sigma_k^2 - \sigma_1^2} + 1\right)\|\Delta\|_F.$$

This guarantees the convergence of the low rank iteration if β is chosen sufficiently small, i.e.

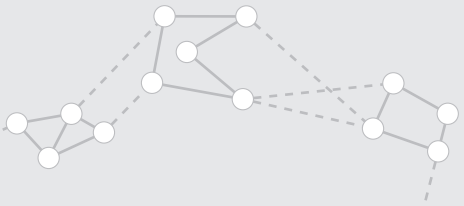
$$\beta^2 < \frac{1}{\|A\otimes A + A^T\otimes A^T\|_F\left(\frac{8\left\|\Sigma^2\right\|}{\Sigma_k^2 - \sigma_1^2} + 1\right)}.$$

Indeed, the distance between the fixed point solution $S^{(r)}$ and the perturbed matrix $S^{(r)} + \Delta$ is naturally $\|\Delta\|$ while the distance between the fixed point solution and the new iterate computed from the perturbed matrix becomes strictly smaller than $\|\Delta\|$.

While we had to assume that β was small enough to ensure the existence of a low-rank fixed point solution in Eq. (5.51), this final inequality ensures the existence of a value for β such that our low-rank iterative scheme converges. Even if theoretically the convergence is not guaranteed

for any perturbation Δ, we never observed divergence of the low-rank sequence using $S_0^{(r)} = 0$ and $\beta > 0$ but small enough. In practice, the maximal value of β will be unknown, so one has to observe the behavior of the low-rank iterates for an initial value of β and adjust it accordingly.

We believe that our pairwise node similarity measure is informative to analyze networks containing a role structure and will be useful in many practical applications. In the next chapter, we will apply and compare our full-rank and our low-rank similarity measures to benchmark and to real graphs and show that our similarity measure allows to successfully extract the different roles contained in the graph. We will also show that the extracted partition for the low and the full rank similarity measure are very much alike, hence justifying the use of our low rank similarity measure in practical context when the full rank similarity can not be computed.



6 Applications to role extraction problems

In this last chapter, we will first apply our similarity measure, introduced in Section 5.4, and its low-rank approximation, detailed in Section 5.5, to random graphs containing a structural block distribution of their nodes. We will show that both measures successfully extract the different roles within those graphs even when the graph topology is fairly noisy. We will also provide some observations on the fact that the evolution of the low-rank similarity matrix for increasing value of the rank can reveal the number of roles in the network. Lastly, we will show that the accuracy of both measures are quantitatively equivalent hence justifying the application of our low-rank iterative scheme in practical contexts.

Then, we will apply our similarity measures to real graphs that are supposed to contain a role structure. In Section 6.2, we will analyze a food web of the Florida bay ecosystem and show that the clustering of the similarity matrix reveals relevant trophic levels according to the diet of the species living in the bay. Then, in Section 6.3, we will present early results about the clustering of a network of reefs in a specific area of the Great Barrier Reef in Australia. This network has already been studied to extract communities of reefs exchanging coral larvae [Thomas, Lambrechts, Wolanski et al. (2014)] and we will show that our low-rank similarity measure corroborates this existence of communities of reefs. Finally, in Section 6.4, we will investigate a graph of words built from co-occurrence in a book. The purpose was to determine if our low-rank similarity measure is able to correctly identify different types of words (nouns, adjectives, adverbs, etc.) without any prior knowledge on the structure of the language. This task is commonly known as part-of-speech tagging and is a challenging problem of natural language processing. We will show that, although the clustering of words is not perfect, some of the clusters computed using our pairwise node similarity indeed extract similar types of words sometimes with great accuracy.

6.1 Benchmark graphs

To assess the quality of our role extraction methodology, we have analyzed its accuracy on synthetic networks with built-in role structure. Our method, extensively applied throughout this chapter, can be described in 2 steps

1. From the adjacency matrix of the input graph, compute the fixed point solution of the pairwise node similarity S^* using Eq. (5.31) or its low-rank approximation $S^{(r)}$ using Eq. (5.35), with suitable values for β and r.
2. Extract communities, using our algorithm described in Section 3.4 and either the modularity or the CPM objective function, on the similarity graph, defined as the graph whose adjacency matrix is the similarity matrix, to identify the roles in the input graph.

As we have pointed out, the extraction of role structure has received less attention than the community detection problem and therefore, there exists much less dedicated models to build a random network containing a role structure rather than a community structure. Hence, we will propose an Erdős-Rényi model, with 2 probability parameters, that allows one to build a random graph with any kind of prescribed role structure. This model will be thoroughly analyzed in Section 6.1.1. Then, in Section 6.1.2, we will apply our methodology to the model proposed by [Guimerà, Sales-Pardo, and Amaral (2007)] which creates bipartite networks containing blocks of similar nodes in each independent set. Finally, we will examine the results of the extraction of roles on LFR benchmark graphs in Section 6.1.3. As we have shown, this kind of random networks contains community structure but, since communities form a particular class of role interactions, one may expect to extract them using a role extraction procedure.

6.1.1 Erdős-Rényi graphs

We applied our pairwise node similarity measures to extract roles in Erdős-Rényi random graphs containing a prescribed block structure. To build such graphs, we first choose a directed reduced graph $G_B(V_B, E_B)$, i.e. each node in G_B defines a role that we would like to identify. Some of the reduced graphs that we considered are represented on the left hand side of Fig. 6.1 to 6.4. Following the notations introduced in the previous

chapter, the large gray filled rectangles represent the roles, i.e. the nodes of the reduced graph, while the small white circles represent the nodes of the input graph.

The reduced graph in Fig. 6.1 corresponds to a community structure where nodes in a role interact mainly with other nodes in the same role. Fig. 6.2 represents a block cycle reduced graph, as already presented in Fig. 5.2, where each node in a role interacts mainly with nodes in the following role in the cycle. This reduced graph might be used for example to condense the interactions between animals in a food web. Lastly, in Fig. 6.3 and 6.4, the reduced graphs were simply chosen as representative examples for more complex role interactions without precise real life examples in mind.

Once a reduced graph G_B has been chosen, we build a random graph $G_A(V_A, E_A)$ where each node of G_A is assigned to a role of G_B. That is, for each node $i \in V_A$, we select a role $R(i) \in V_B$. Then, we add the edges in E_A using 2 probability parameters. For every pair of nodes $i, j \in V_A$, we add the edge (i, j) to E_A with probability p_{in} if there is an edge between the corresponding roles in G_B, i.e. $(R(i), R(j)) \in E_B$. If there is no edge between the corresponding roles in G_B, the edge (i, j) might still be added to E_A but with a probability p_{out}. Therefore, if p_{in} is much larger than p_{out}, the reduced graph G_B is an appropriate representation of the role structure of the graph G_A and we expect that our pairwise similarities between the vertices V_A should allow the extraction of those roles. On the other hand, if p_{out} is much larger than p_{in}, the different roles in G_A are more accurately represented by the complement of the reduced graph G_B, whose adjacency matrix is $\mathbf{1}\mathbf{1}^T - B$. However, the role structure is still strongly existing in the graph G_A and we expect that our similarity measures should still be able to extract it. It is when the 2 probabilities p_{in} and p_{out} are close to each other that extracting the different roles becomes challenging but, at the same time, the graph G_A becomes close to a classical Erdős-Rényi random graph which is known to be free of any structure.

Each of the figures 6.1 to 6.4 is divided into 4 panels, according to dashed lines, corresponding to different values of p_{in} and p_{out} for the same reduced graph as follows

$p_{in} = 0.9 / p_{out} = 0.1$	$p_{in} = 0.8 / p_{out} = 0.2$
$p_{in} = 0.7 / p_{out} = 0.3$	$p_{in} = 0.6 / p_{out} = 0.4$

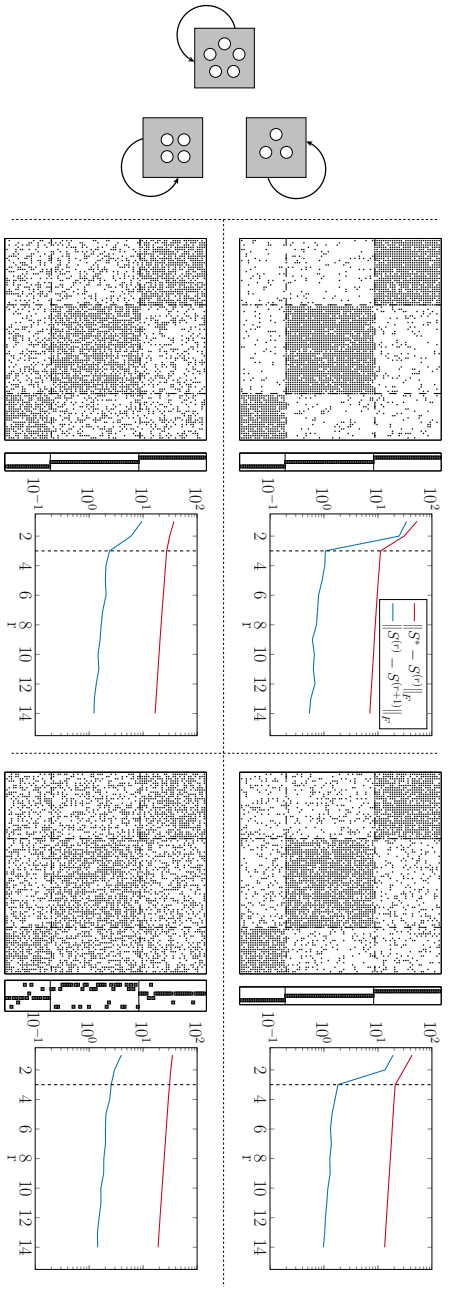


Fig. 6.1 Extraction of communities as roles.

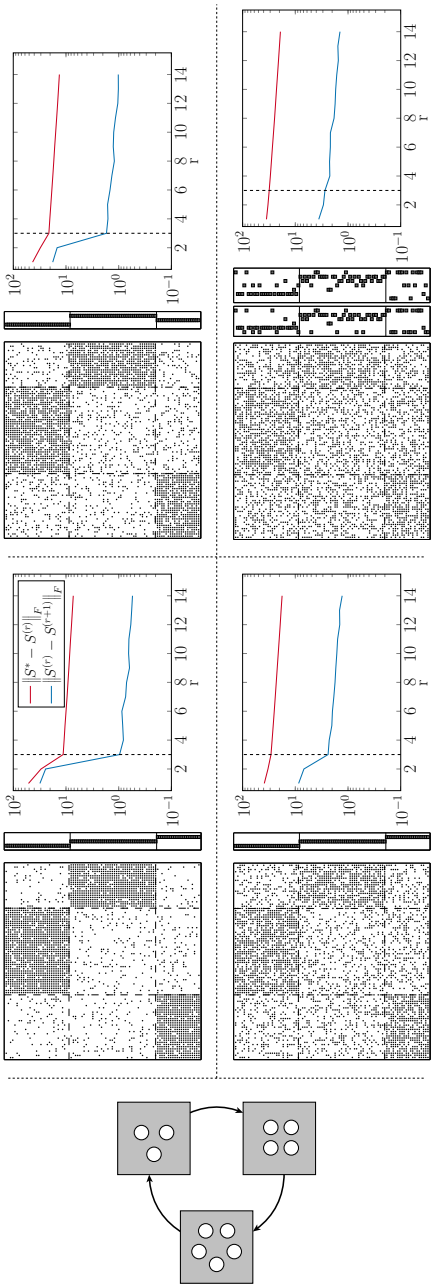


Fig. 6.2 Extraction of cycle role structure.

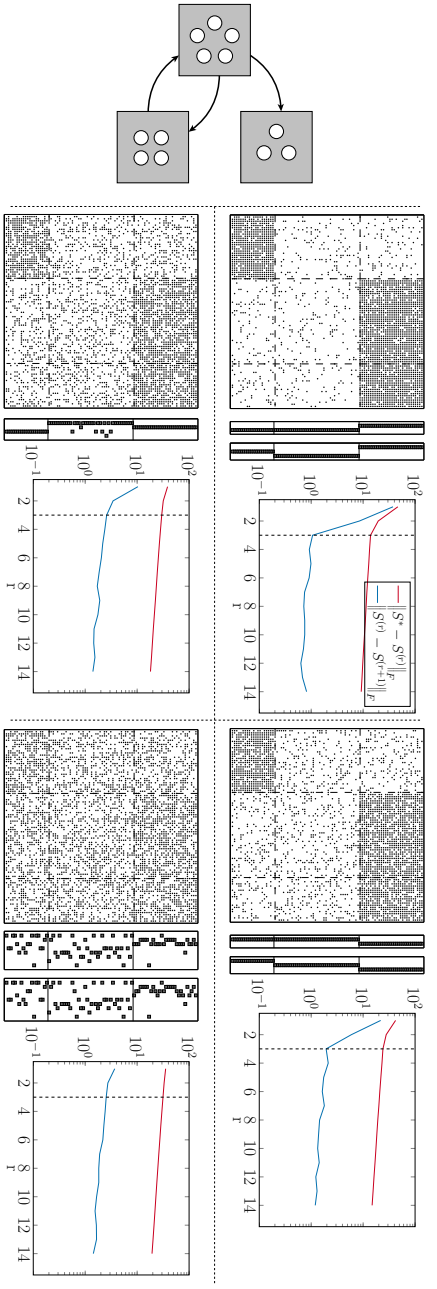


Fig. 6.3 Extraction of role structure with almost isomorphic roles.

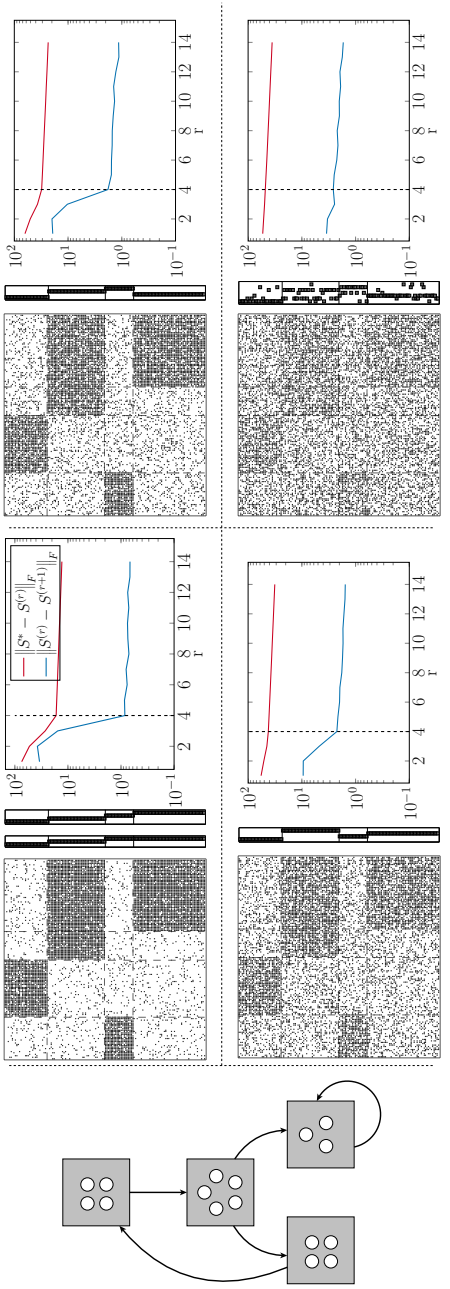


Fig. 6.4 Extraction of 4 roles structure.

In each panel, we first show the adjacency matrix of one typical realization of the random graphs G_A generated. For visual clarity, the adjacency matrices have been permuted such that nodes in the same role are next to each other. Then, we represent the role assignment of each node as extracted using our community detection algorithm applied to our low-rank similarity matrix $S^{(r)}$ for $r = 10$. Since our community detection algorithm may produce different hierarchical levels of clustering, we present each level of roles when different levels were extracted from the similarity graph. The role assignments are represented by the clustering matrix $\sigma(i, j) = 1$ if node i belongs to cluster j and 0 otherwise. The last plot in each panel represents the evolution of $S^{(r)}$ for increasing values of r. That is, we compute the norm of the difference between the full-rank fixed point solution and each low-rank fixed point solutions, $\left\| S^* - S^{(r)} \right\|_F$, and between "consecutive" low-rank fixed point solutions, $\left\| S^{(r)} - S^{(r+1)} \right\|_F$ for increasing values of r. This should reveal the minimal rank required for $S^{(r)}$ to be a qualitatively good approximation of S^*.

The main result is that the different roles within each network can be well extracted by clustering our low-rank similarity graph up to some high level of noise. For the first role structure, Fig. 6.1, each community is perfectly extracted up to $p_{in} = 0.7$ and $p_{out} = 0.3$. The network starts to be really noisy for probability parameters closer to each other, i.e. $p_{in} = 0.6$ and $p_{out} = 0.4$, as represented by the adjacency matrix, however the first and the third communities are still pretty well clustered and the second community is mostly split in 2. The same observation applies to the block cycle reduced graph, presented in Fig. 6.2, for which all the roles are perfectly extracted for the first three pairs of probability parameters. Note that, for the last pair of probability parameters, two hierarchical levels of clustering are extracted, i.e. the final level is represented on the left hand side and the first level on the right hand side. In the final level of clustering, the second and third roles are each essentially split in 2 different clusters and there are only a few nodes, compared to the size of the network, with inappropriate role assignment.

Those first 2 reduced graphs define strong role structures. If two nodes of the reduced graph are isomorphic, i.e. if they have exactly the same incoming and outgoing neighborhoods, we say that, by extension, the associated roles are isomorphic. Nodes belonging to different isomorphic roles have therefore the exact same probability distribution of their edges across the graph and it is impossible to distinguish them and deter-

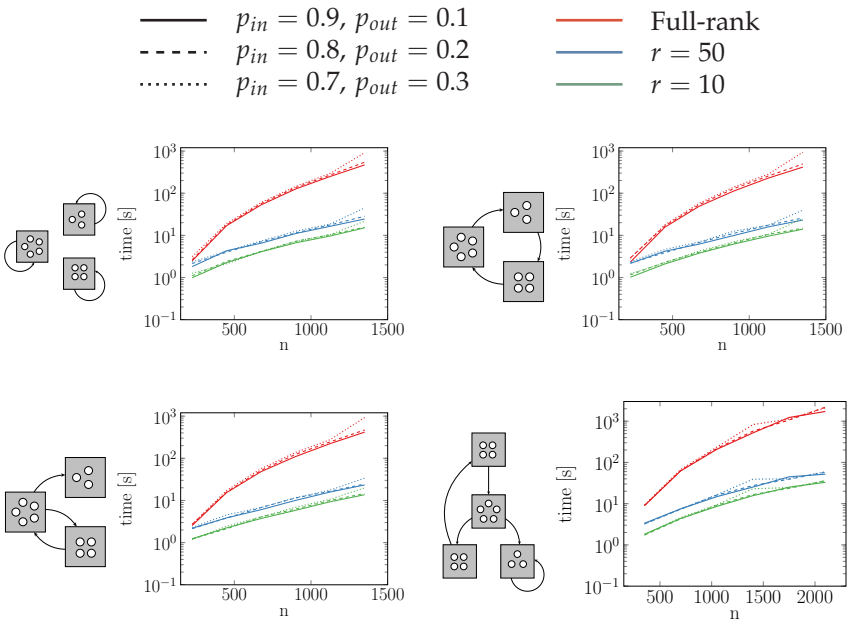


Fig. 6.5 Computational time required to compute S^* and $S^{(r)}$ (for $r = 50$ and $r = 10$) with respect to the number of nodes in the graph.

mine their exact role. One can observe that, in the first 2 reduced graphs, one would need to add relatively many edges to create isomorphic roles. Therefore, our method correctly extracts the role structures even for high level of noise. On the other hand, the third reduced graph, in Fig. 6.3, is less strongly defined because if a single edge is added in the reduced graph from the second block to the first block, the second and the third roles would become isomorphic. Indeed, we observe that a small set of nodes is incorrectly clustered from the second role to the third role for $p_{in} = 0.7$ and $p_{out} = 0.3$. This might also explain why the second and third roles are merged together in the final level of clustering for the first two pairs of probability parameters but this might also be due to some resolution limit of the community detection cost function. For higher level of noise, clustering the pairwise similarity matrix does not provide an accurate partition, however, the adjacency matrix clearly indicates that the role structure is very weak, if even existing.

Lastly, for the reduced graph in Fig. 6.4 composed of 4 distinct roles,

the results are again reasonably good. Except for an additional merge in the last level of clustering for $p_{in} = 0.9$ and $p_{out} = 0.1$, all the roles are perfectly extracted, and even for the last pair of probability parameters, there is only a small number of nodes incorrectly classified for the first and last roles and the second and third roles are mostly bisected as previously observed.

One can also observe that the evolution of the low-rank similarity matrix $S^{(r)}$, for increasing value of r, might be used to actually reveal the number of roles in the network. Indeed, when the different roles in a network are strongly defined, we observe an abrupt variation in the decay of the norm of the differences $\left\|S^* - S^{(r)}\right\|_F$ and $\left\|S^{(r)} - S^{(r+1)}\right\|_F$. This abrupt variation indicates that we might not need to consider larger values of the rank to extract qualitatively good roles in this network, since the gain in precision for the similarity matrix starts to increase very slowly afterwards. The most interesting point is that this abrupt variation always occurs when the rank hits the exact number of roles in the network. When the network is highly noisy, we do not observe such an abrupt variation which could indicate that the clustering of the nodes according to the similarity matrix will not produce a relevant partition. Observing the evolution of the low-rank similarity matrix could become a strong indicator of the quality of the extracted roles for real networks when the exact block structure is unknown. Indeed, as depicted in Fig. 6.5, computing the full-rank similarity matrix would be computationally too expensive for real networks and, as expected, computing the low-rank similarity matrix exhibits a much smaller time complexity which is mostly independent of the parameters of the model.

Finally, we compare quantitatively the extracted clusters using our full-rank similarity S^* and our low-rank similarity $S^{(r)}$ measures. For each of the different reduced graphs previously described, we compute the normalized mutual information (Section 4.1) between the exact role structure and the extracted role partition using S^* or $S^{(r)}$ with $r = 10$. For each reduced graph, we generated 20 random realizations for each couple of probability parameters p_{in} and p_{out} in $[0, 1]$ with a discretization step size of 0.05, and we computed the average NMI on those 20 realizations. The results are presented in Fig. 6.6. As expected, we observe that the extracted roles are accurate when either $p_{in} \gg p_{out}$ or the opposite. As mentioned previously, the third reduced graph seems harder to recover due to either a resolution limit phenomenon or to the almost isomorphic

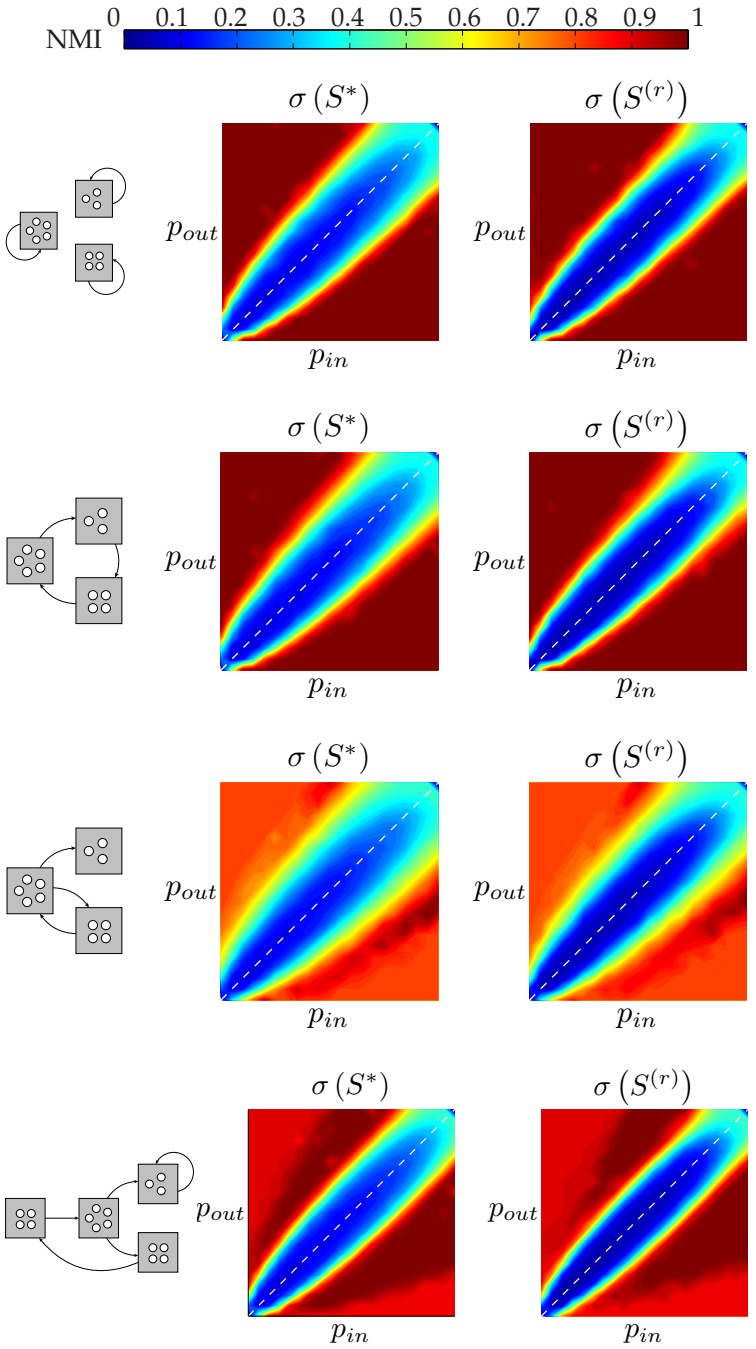


Fig. 6.6 Comparison of the quality of the extracted partitions using S^* or $S^{(r)}$.

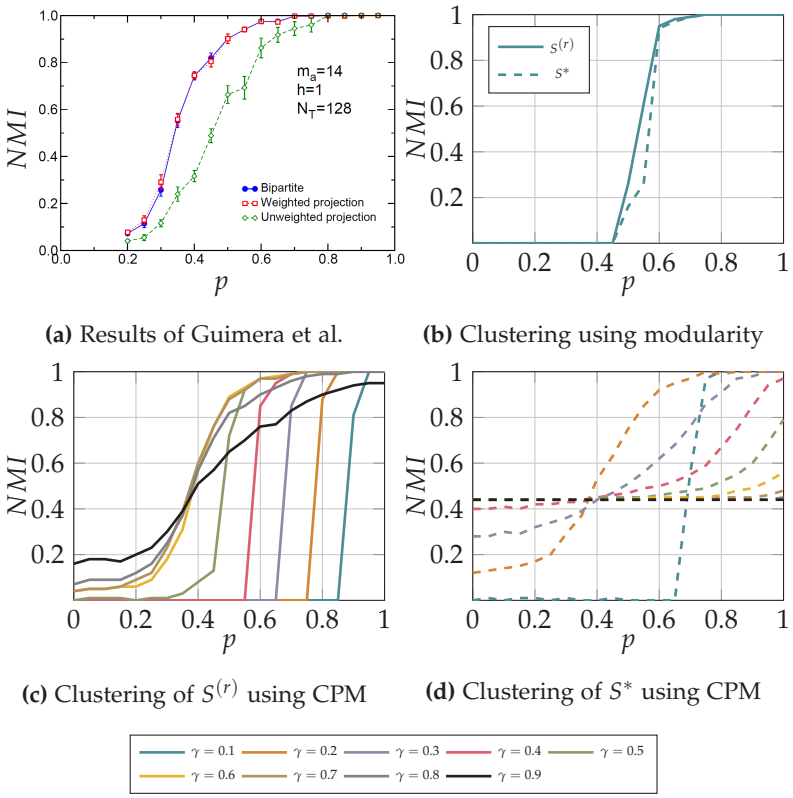
behavior of two of the roles. Nevertheless, we observe that the low-rank similarity matrix $S^{(r)}$ produces almost identical results than the full-rank similarity matrix S^*. This leads us to conclude that, if the rank is sufficiently large, one can always use our low-rank pairwise node similarity measure to extract the role structure in a network. The low-rank similarity matrix will always be cheaper to compute and will produce analogous partitions.

6.1.2 Guimera et al. model

We also applied our role extraction procedure on the model proposed by [Guimerà, Sales-Pardo, and Amaral (2007)]. The networks created using this model are bipartite and composed of a set of actors and a set of teams. The underlying assumption behind the creation of a network is that there exist groups of similar actors which should therefore be involved in many teams together.

More precisely, a random network is built as follows. First, we choose the number of groups of actors N_g and the number of actors n_i in each group i. Then, we create the teams and connect them to some actors following the hypothesis that actors in the same group should have a high probability to appear in the same teams. That is, we choose the number of teams N_t and, for each team, we randomly select one preferential group of actors. Then, we choose the number of actors that will be involved in each team m_a and for each position in a team, we select an actor of its preferential group with a probability p or any of the actors with a probability $1 - p$. Hence, the probability parameter p controls the team homogeneity in term of groups of actors. When $p = 1$, each team is composed solely of actors from its preferential group and therefore the role structure, defined by the groups of actors, is strongly present in the network. On the other hand, when $p = 0$, all the teams are composed of actors randomly selected from all the existing groups and the prescribed role structure is irrelevant.

In their work, [Guimerà, Sales-Pardo, and Amaral (2007)] considered three different approaches to uncover the groups of actors. First, they considered modularity optimization on the unweighted projection of the graph over the set of actors. That is, they created a graph of actors where two actors are connected if they are involved in at least one common team. They also analyzed the weighted projection over the set of actors where each connection between two actors is weighted by the number of teams



(a) Results of Guimera et al.

(b) Clustering using modularity

(c) Clustering of $S^{(r)}$ using CPM

(d) Clustering of S^* using CPM

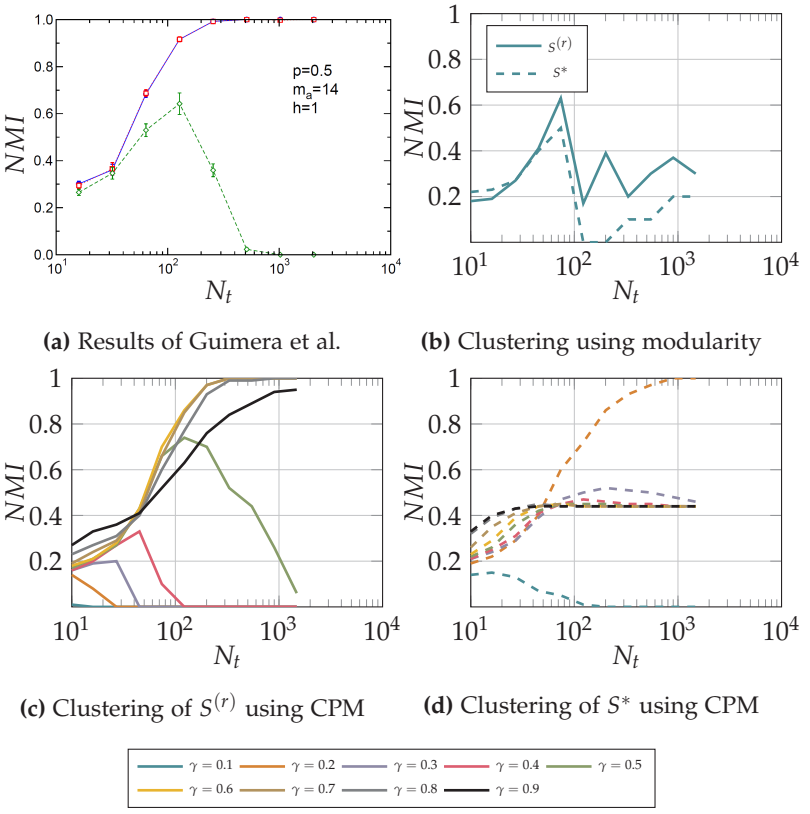
Fig. 6.7 Performance of role extraction procedures with respect to team homogeneity. (a) reprinted from [Guimerà, Sales-Pardo, and Amaral (2007)].

in which there are commonly involved. Finally, they proposed a cost function inspired from modularity but specifically designed to cluster nodes in a bipartite network and which does not require any projection. The optimization of the cost functions (either modularity on the projected networks or the bipartite modularity as the authors named it) was performed using simulated annealing (see Section 3.3.2) which is known to be accurate but slow.

The results of the extraction of the role structure, measured according to the NMI between the true partition in groups of actors and the extracted partitions, for $p \in [0,1]$ is presented in Fig. 6.7. The parameters of the model were taken as in the paper of Guimera et al. to have a fair com-

parison between their results and ours, i.e. $N_g = 4$ groups of actors with $n_i = 32$ actors each and $N_T = 128$ teams with $m_a = 14$ actors each. First, Fig. 6.7a presents the results obtained by Guimera et al. for each of their three approaches. Then, Fig. 6.7b presents the results of the optimization of modularity on our similarity matrices S^* and $S^{(r)}$ with $r = 10$. Finally, we present in Fig. 6.7c and 6.7d the results obtained for the optimization of CPM, using different values of γ, again on our similarity matrices. The main results of Guimera et al. are that the unweighted projection performs pretty badly and should not be used, and that the weighted projection and their bipartite cost function perform much better and produce indistinguishable results. We observe that the clustering of our similarity graphs using modularity produces accurate results for $p \geq 0.6$ but starts to behave rather poorly for smaller values of p. In fact, for small value of p, our algorithm clusters all the actors together which yields necessarily a NMI of 0. We observe that, in parallel, all the teams are also clustered together but within another cluster. Therefore, this method extracts the bipartite nature of network but nothing more. Clearly, using either our full-rank similarity measure or its low-rank approximation yields mostly the same results. The results for the optimization of CPM are much better. Surprisingly, the clustering of our low-rank similarity matrix produces more accurate results than using our full-rank similarity which is unexpected. The best results are obtained for $\gamma = 0.6$ for the low-rank measure and for $\gamma = 0.2$ for the full-rank measure. Moreover, one can note that the full-rank measure is much more sensitive to the value of γ and quickly produces partitions with either all the actors together or all the actors alone. Finally, let us note that even the best results we obtained using our low-rank similarity measure and appropriate value of γ are slightly under the accuracy of the method of Guimera et al.

We also analyzed the impact of the number of teams on the clustering of the similarity matrices. To this end, the probability parameter was set to a constant value of $p = 0.5$ and we let the number of teams grows from 10 to 1500. The results are presented in Fig. 6.8 following the same layout as in the previous figure. First, note that the number of teams affects very differently each of the approaches considered by Guimera et al. Again, the optimization of modularity on the weighted projection and the optimization of the bipartite modularity yield extremely close results. As the number of teams grows, the network somehow contains more information about the structure of the groups of actors and therefore both approaches are better able to extract them. On the other hand, the un-



(a) Results of Guimera et al.

(b) Clustering using modularity

(c) Clustering of $S^{(r)}$ using CPM

(d) Clustering of S^* using CPM

Fig. 6.8 Performance of role extraction procedures with respect to number of team. (a) reprinted from [Guimerà, Sales-Pardo, and Amaral (2007)].

weighted projection quickly becomes unable to extract any relevant partition because when there are more teams, the projected network becomes close to a complete graph since any pair of actors has a high probability to appear together in at least one of the teams. In this experiment, we observe that modularity optimization on our similarity graphs produces extremely poor results, in fact even not better than the unweighted projection approach. Clearly, the problem comes from the cost function rather than from our similarity graph since the optimization of CPM produces accurate partitions. As previously, the clustering obtained using the full-rank similarity matrix seems much more sensitive to the value of γ than

using the low-rank similarity matrix. The results show that one needs to use $\gamma = 0.2$ for the full-rank similarity and any other value of γ produces poor partitions. On the contrary, the clustering of the low-rank similarity produces good partitions for $\gamma \in [0.6, 0.8]$. The best results are obtained for the optimization of CPM with $\gamma = 0.6$ on the low-rank similarity graph and it seems that the accuracy of our method in this case is extremely close to the accuracy of the two best approaches of Guimera et al.

In the end, our method does not outperform the approach of Guimera et al. but produces results of similar accuracy when the parameter γ is correctly chosen. This clearly indicates that the outcome of the clustering of the similarity matrix is strongly dependent on the choice of the cost function. Moreover, let us note that the bipartite modularity is specifically designed to extract partitions in bipartite networks and therefore, it makes sense that this method is more suited here. However, it lacks the generality of our similarity measures which allow to study any other kind of network and role structure. Moreover, Guimera et al. used simulated annealing to optimize their cost functions which is known to produce better partitions than our community detection algorithm, but is completely inappropriate to study larger networks. It is possible that, using simulated annealing, the clustering of our similarity graph would produce even better results. Finally, when using CPM, we always choose a constant value of γ for all the networks created using a fixed set of parameters, while it might be more appropriate to use slightly different values of γ for each network. Although, finding an appropriate value of γ for a network under study is a research topic in itself.

This leads us to conclude that our similarity measure seems an appropriate surrogate to discover the role structure within this kind of bipartite graphs, and interestingly, the low-rank similarity measures seems even more appropriate than its full-rank counterpart. However, a dedicated cost function to cluster the similarity graphs should be defined. Clearly, the similarity graphs have some specific properties that should be considered by the cost function to achieve the best possible results. That gives an interesting direction for future research.

6.1.3 LFR model

To conclude this section about benchmark graphs, let us discuss the results obtained for the extraction of role structure in random graphs created using the LFR benchmark presented in Section 4.1. As mentioned, commu-

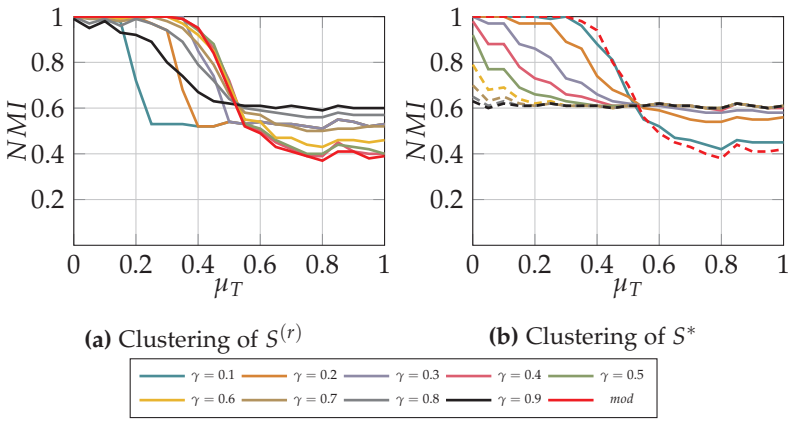


Fig. 6.9 Performance of role extraction on the LFR benchmark.

nities form a particular class of roles and therefore, one can wonder how our similarity measures behave on this kind of graphs. The results are presented in Fig. 6.9 for unweighted undirected LFR networks with 1000 nodes and $\langle k \rangle = 10$ and should be compared to the results obtained in Section 4.1.2. The quality of the partitions obtained using both CPM and modularity are presented in Fig. 6.9a for the low-rank similarity matrix with $r = 10$ and in Fig. 6.9b for the full-rank similarity matrix.

Surprisingly, the results are completely different than what we observed in the previous section. The optimization of the modularity on the low-rank similarity graph yields very accurate partitions, in fact almost identical to the optimization of CPM with the most appropriate value of γ. Furthermore, the optimization of modularity achieves the best performance for the full-rank similarity graph. As for the Erdős-Rényi random graphs presented in Section 6.1.1, the low-rank and the full-rank similarity graphs produce partitions of equivalent accuracy.

When compared to the results presented in Fig. 4.15 for small undirected networks, it is clear that our similarity measures enforce the detection of the community structure existing in the graph since our algorithm is able to correctly partition the network for much larger values of the topological mixing parameter μ_T. However, we are still not able to outperform what we observed to be the best community detection algorithm, i.e. infomap. But again, it is not clear if the differences are due to the measures of similarity or to the clustering algorithm applied to the simi-

larity graphs. It might very well be that applying infomap either on the similarity graphs or directly on the input graph would produce results of comparable accuracy. Though, for large networks, infomap would not be suitable since there are always much more edges in the similarity graphs than in the original graph and we have shown how badly this algorithm scales with the size of the network. Finally, we can conclude that our similarity measures, at least, do not weaken the community structure and is therefore an appropriate way to cluster networks where the topological node distribution is completely unknown.

6.2 Florida bay food web

We now turn to the analysis of some real graphs. The first network under study is a food web of the Florida bay ecosystem[1] [Ulanowicz and DeAngelis (1999)]. The 122 different biological species living in the bay can be classified in 7 subgroups as given by Table 6.1. The network is built on a who-eats-whom relationship such that there is a weighted edge from i to j if species j eats species i and the weight of the edge represents the average exchange of biomass from i to j. The purpose here is to try to find relevant groups of animals that have the same kind of diet. Hence, we are not interested in the absolute value of the biomass exchanges but rather in the proportional composition of the diet of each species. Therefore, we rescaled the weight of the edges such that each node has an incoming strength of 1 (or 0 in the case of primary producers which do not eat any other species). The weight of an edge between i and j can then be interpreted as the proportion of species i in the diet of species j.

The reduced graph for the natural biological classification of species is represented in Fig. 6.10a. The reduced graph is computed here as the aggregation of the original graph according to the chosen clustering, with the incoming edge weight rescaled by the number of species per cluster. Therefore, the weight of an edge between 2 clusters in the reduced graph gives the average proportion of the diet of species in one cluster that is coming from species in another cluster. For example, one can observe that species of the first group do not eat anything since they have no incoming edge, and that the species of the second group have on average a diet composed of 40% of species from the first group and of 60% of species from their own group. For visual clarity, we do not add to the reduced graph the edges with a weight smaller than 0.1 which explains why the

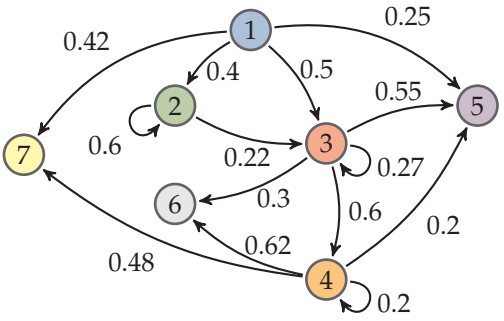
[1] http://www.cbl.umces.edu/~atlss/FBay001.html

1. Primary producers	2um Spherical Phytoplankton ; Synnedococcus ; Oscillatoria ; Small Diatoms ; Big Diatoms ; Dinoflagellates ; Other Phytoplankton ; Benthic Microalgae; Thalassia testudinum ; Halodule wrightii ; Syringodium filiforme ; Roots ; Drift Algae ; Epiphytes
2. Microfauna	Free Bacteria ; Water-Column Flagellates ; Water-Column Cilliates ; Acartia tonsa ; Oithona nana ; Paracalanus sp. ; Other Copepods ; Meroplankton ; Other Zooplankton ; Benthic Flagellates ; Benthic Cilliates ; Meiofauna
3. Macroinvertebrates	Sponges ; Coral ; Other Cnidaria ; Echinodermata ; Bivalves ; Detritivorous Gastropods ; Epiphyte Grazing Gastropods ; Predatory Gastropods ; Detritivorous Polychaetes ; Predatory Polychaetes ; Pelagic Feeding Polychaetes ; Macrobenthos ; Benthic Crustaceans ; Detrivorous Amphipods ; Herbivorous Amphipods ; Isopods ; Herbivorous Shrimp ; Predaceous Shrimp ; Pink Shrimp ; Thor floridanus ; Spiny Lobster ; Detritivorous Crabs ; Omnivorous Crabs ; Predatory Crabs ; Callinectes spp. ; Stone Crab
4. Fishes	Sharks ; Rays ; Tarpon and Ladyfish ; Bonefish ; Sardines ; Anchovies ; Bay Anchovy ; Lizardfish ; Catfishes ; Eels ; Toadfish ; Brotulas and Batfishes ; Halfbeaks and Flyingfish ; Needlefishes ; Killifishes ; Floridychthys carpio ; Lucania parva ; Snooks ; Poecilids ; Silversides ; Seahorses and Pipefishes ; Sygnathus scovelli ; Hippocampus zosterae ; Groupers ; Jacks and Runners ; Pompano and Permits ; Snappers ; Gray Snapper ; Mojarras and Jennies ; Grunts ; Porgies ; Pinfish ; Sciaenid fishes ; Spotted Seatrout ; Red Drum ; Spadefish ; Parrotfishes ; Mackerels ; Mullets ; Barracudas ; Blennies ; Code Goby ; Clown Goby ; Flatfishes ; Filefishes and Trigger fishes ; Puffers and Burrfishes ; Other Pelagics ; Other Demersals
5. Herpetofauna	American Crocodile ; Loggerhead Turtle ; Green Turtle ; Hawksbill Turtle
6.Avifauna	Loons ; Grebes ; Pelicans ; Cormorants ; Big Herons and Egrets ; Small Herons and Egrets ; Ibis ; Roseate Spoonbill ; Herbivorous Ducks ; Omnivorous Ducks ; Predaceous Ducks ; Raptors ; Gruiformes ; Small Shorebirdss ; Gulls and Terns ; Kingfishers
7. Mammals	Dolphins ; Manatee

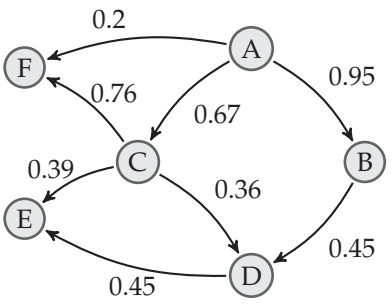
Table 6.1 Composition of biological compartments in the Florida Bay network

incoming strength of the node representing fishes is only 0.8.

The reduced graph of the natural classification exhibits correctly the differences between the different groups of animals. For example mammals are different from reptiles and amphibians (herpetofauna): while they both eat primary producers and fishes, mammals do not eat macroinvertebrates. Similarly, birds (avifauna) are classified differently from reptiles because they do not eat primary producers in the bay, even though they eat fishes and macroinvertebrates. However, one may expect to find an even more stratified classification in the sense that animals in a group should not eat species of their own group, i.e. one may expect to find a reduced graph without self loops.



(a) Natural classification.



(b) Role partition.

Fig. 6.10 Reduced graphs for the natural partition and the extracted role structure.

We have applied our pairwise node similarity measure to this ecological network of the Florida bay and clustered the similarity graph using our community detection algorithm and to the modularity cost function. We also applied the CPM cost function but the results look extremely sensitive to the value of γ and a stability analysis should be done to select a correct value. We used both our full-rank and our low-rank similarity with $r = 20$ and observed very similar results. The reduced graph that we obtained for the extraction of the role structure, using our low-rank similarity measure with $r = 20$, is represented in Fig. 6.10b and it seems to be accurately capturing most of the biomass exchanges in the bay. Note that, using our methodology, we did not find any self loop in the reduced graph, i.e. each group eats animals coming mainly from other

A	2um Spherical Phytoplankton ; Synnedococcus ; Oscillatoria ; Small Diatoms ; Big Diatoms ; Dinoflagellates ; Other Phytoplankton ; Benthic Microalgae; Thalassia testudinum ; Halodule wrightii ; Syringodium filiforme ; Roots ; Drift Algae ; Epiphytes ; Free Bacteria ; Water-Column Flagellates ;
B	Water-Column Cilliates; Acartia tonsa ; Oithona nana; Paracalanus sp.; Other Copepods; Other Zooplankton; Sponges; Bivalves; Detritivorous Gastropods; Detritivorous Polychaetes; Pelagic Feeding Polychaetes;
C	Benthic Flagellates; Echinodermata; Epiphyte Grazing Gastropods; Predatory Polychaetes ; Macrobenthos ; Herbivorous Amphipods ; Isopods; Herbivorous Shrimp ; Predaceous Shrimp ; Pink Shrimp; Thor floridanus; Spiny Lobster; Detritivorous Crabs; Omnivorous Crabs; Halfbeaks and Flyingfish ; Poecilids ; Parrotfishes ; Mullets ; Other Demersals ; Green Turtle ; Herbivorous Ducks ; Omnivorous Ducks ; Manatee;
D	Meroplankton; Coral; Other Cnidaria; Predatory Gastropods; Stone Crab; Rays ; Bonefish ; Sardines ; Anchovies ; Bay Anchovy ; Catfishes ; Killifishes ; Floridychthys carpio ; Lucania parva ; Silversides ; Seahorses and Pipefishes ; Sygnathus scovelli ; Hippocampus zosterae ; Pompano and Permits ; Mojarras and Jennies ; Grunts ; Porgies ; Pinfish ; Blennies ; Code Goby ; Clown Goby ; Filefishes and Trigger fishes ; Puffers and Burrfishes ; Hawksbill Turtle ;
E	Predatory Crabs; Callinectes spp.; Sharks ; Tarpon and Ladyfish ; Lizardfish ; Eels ; Toadfish ; Brotulas and Batfishes ; Needlefishes ; Snooks ; Groupers ; Jacks and Runners ; Snappers ; Gray Snapper ; Sciaenid fishes ; Spotted Seatrout ; Red Drum ; Spadefish ; Mackerels ; Barracudas; Flatfishes ; Other Pelagics ; American Crocodile ; Loggerhead Turtle ; Loons ; Grebes ; Pelicans ; Cormorants ; Big Herons and Egrets ; Small Herons and Egrets ; Ibis ; Roseate Spoonbill ; Predaceous Ducks; Raptors ; Gruiformes ; Small Shorebirdss ; Gulls and Terns ; Kingfishers ; Dolphins;
F	Benthic Cilliates; Meiofauna; Benthic Crustaceans; Detrivorous Amphipods;

Table 6.2 Composition of roles in the Florida Bay network

groups (at least, more than 90% of their diet). This is a clear improvement over the reduced graph of the natural classification. However, note that we may capture less information for some particular species. For example, we observe that 67% of the diet of species in group B is coming from species of group A and the rest is spread over the reduced graph and we do not capture it. On the other hand, we also observe that some groups are particularly well clustered like group B from which animals have a diet composed of 95% of species in group A. None of the natural groups achieves such a high weight on any of its edges.

Let us comment the composition of each cluster according to our role extraction procedure which is detailed in Table 6.2. Cluster A is composed of all the primary producers (perfectly clustered together) and 2 species of the microfauna, the free bacteria, which do not eat anything in the network and therefore really look like primary producers, and the water-column flagellates, which eat mostly one primary producer and free bacteria but, at the same time, feed only species of cluster B and none of the other species, so are therefore better classified in A. Cluster B is composed of 6 species of microfauna and 5 species of macroinvertebrates that

have the particularity to feed almost exclusively species of the cluster D and to eat only primary producers. Cluster C contains a large portion of the remaining macroinvertabrates. However, this cluster also contains some fishes, the green turtle, two types of ducks, and the manatee which makes sense since those animals are mainly herbivorous and eat mostly primary producers as the macroinvertebrates of this cluster (the green turtle and the manatee eat in fact only primary producers). Let us note that this cluster is different than cluster B because it feeds also cluster E and F. Cluster D is composed mainly of small fishes, like Sardines, that eat macroinvertebrates and species of cluster B. Cluster E contains the rest of the large fishes, like Sharks or Barracudas, and the birds that are eating macroinvertebrates but also the small fishes of cluster D. Note that due to their diet, the Dolphins are classified as large fishes. Finally, cluster F contains only 2 species of microfauna and 2 species of macroinvertebrates which seems to have a different diet than the others.

In conclusion, our role structure is accurately partitioning the network in groups of animals having the same diet, although they may be very different like large birds and sharks. This is mostly due to our rescaling of the incoming edge weight such that animals are more similar if the have the same kind of diet. If we had rescaled the edges according to the outgoing strength, then we might have observed that groups of similar species have instead the same kind of predators.

6.3 Great Barrier Reef network

Our low-rank similarity measure has also been applied to a simulated network of reefs in the Great Barrier Reef of Australia, previously studied in [Thomas, Lambrechts, Wolanski et al. (2014)]. By analyzing this network, one is interested in understanding how the coral larvae spread between reefs during the early phases of their existence. Once a larvae has fixated onto a reef, it will spend its entire life on this reef so analyzing the reefs connectivity is essential to provide better conservation policies of the Marine Protected Areas. However, it remains extremely difficult to directly observe or measure the transport and exchange of larvae between reefs because of the small size of the larvae, the distance between the reefs and their dispersal or the variability in the time before fixation for each larva. Therefore, an oceanic model has been proposed to simulate the exchange of larvae between roughly 1000 reefs (see [Thomas, Lambrechts, Wolanski et al. (2014)] for details about the model or its numerical reso-

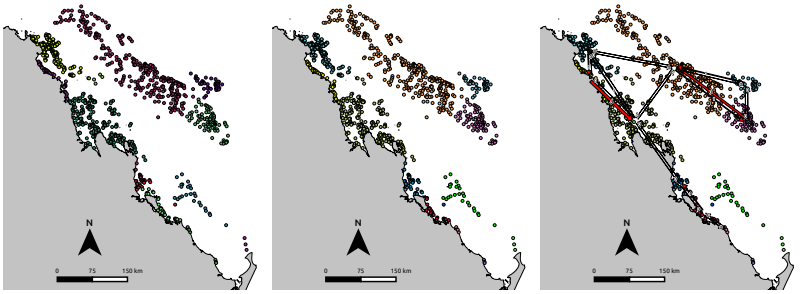
lution). Based on the simulated network topology, communities of reefs exchanging larvae mostly with each others were detected using the CPM cost function.

The question was then to know whether a community structure is really the most representative distribution of reefs, and therefore, we applied our role structure identification methodology to this network. The results are presented in Fig. 6.11 and, what we observed is that, for each resolution level defining a stable partition of the network in communities, we found a corresponding resolution parameter γ that clusters the similarity graph such that the extracted roles are indeed communities.

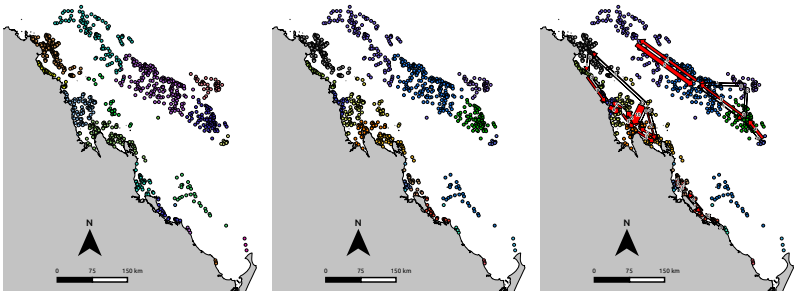
More precisely, in each row of Fig. 6.11, we first represent a stable partition obtained by clustering the graph in communities, then we represent the partition obtained by clustering our similarity graph, and finally, we represent the inter-reefs total connectivity in our role partition, i.e. the edges of the reduced graph that are not self loops. The first row corresponds to the most coarse-grained resolution of communities and the last row to the finest stable resolution obtained. Each reef in the map is placed at its exact position in the Great Barrier Reef and the color indicates the partition. Unexpectedly, the results obtained for the extraction of roles are extremely similar to the results obtained using community detection. In the first and second rows, one can observe that both partitions are very much alike and when looking at the inter-reefs connectivity in the role partition, one can note that only a very small fraction of larvae is exchanged between different clusters of reefs. Therefore, those role structures are indeed community structures. We observed some differences in the partitions only for the finest resolution level, i.e. some of the communities in the top left corner of the figure are more or less bisected in the role partition which produces exchanges of around 50% of larvae between each component of the community. While this indicates that the partitioning results may be slightly improved using our role structure identification methodology, the community structure seems indeed very representative of the actual distribution of reefs.

6.4 Part-of-speech tagging

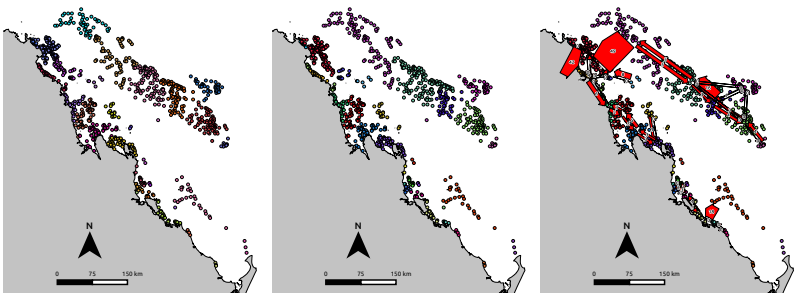
We conclude this chapter with an experiment on the automatic classification of words in a text. Our basic assumption is that most of the grammatically correct sentences should have roughly the same kind of structure and therefore one can expect to extract the type of a word as its role



(a) Communities, $\gamma = 5.9e^{-4}$ **(b)** Roles, $\gamma = 4.2e^{-3}$ **(c)** Roles, $\gamma = 4.2e^{-3}$



(d) Communities, $\gamma = 1.2e^{-3}$ **(e)** Roles, $\gamma = 2e^{-2}$ **(f)** Roles, $\gamma = 2e^{-2}$



(g) Communities, $\gamma = 5.4e^{-3}$ **(h)** Roles, $\gamma = 3.5e^{-2}$ **(i)** Roles, $\gamma = 3.5e^{-2}$

Fig. 6.11 Communities and roles extracted in the Great Barrier Reef network.

in the sentence. For example one would expect that most sentences are in the form "*subject*"⇒"*verb*"⇒"*agreement*" or that adjectives are mostly followed by nouns, etc.

This kind of automatic text analysis is known as natural language processing and in particular, finding the type of a word (noun, verb, adjective, adverb, etc.) is known as part-of-speech (POS) tagging. The most clever algorithms for POS tagging use the structure of the sentence but also the definition of the words, some descriptive tags and constraints between words and often prior knowledge on the possible tags for each word coming from a corpus. For example, the Brown corpus is a collection of more than 1.000.000 words that were manually tagged and improved over many years. Obviously, some words can have different meanings and therefore different tags depending on the semantics which largely hardens the task.

We were interested in knowing how our similarity measure would behave on this particular problem because it has the nice property to not require any prior knowledge and only use the topology of the graph to identify groups of nodes having the same role. Our method could potentially be used as a preprocessing of any text to cluster together similar words which in turn could largely help linguistics studies. Naturally, the actual type of words in a cluster could not be determined by our method but we let that responsibility to linguists.

We considered a graph of words built by automatically reading the book "Language, an introduction to the study of speech" by [Sapir (1921)]. This book is freely available on the *Project Gutenberg* website (`http://www.gutenberg.org/`) and was chosen simply because we believed that its title fits appropriately our subject. The graph contains one node per word found in the book and there is a weighted edge from a word i to a word j if the word j followed the word i in a sentence. The weight of each edge encodes the number of times the two words co-occurred. To have some sort of ground truth to evaluate the quality of our partitioning, we used the Stanford POS tagger [Toutanova, Klein, Manning et al. (2003)] and the natural language processing toolbox available in Python (NLTK, `http://www.nltk.org/`). Fig. 6.12 shows the distribution of words per type according to the classification of the Stanford POS tagger. One can see that around 90% of the words are tagged as either nouns (singular and plural being differentiated), adjectives, verbs (with different type of conjugation) or adverbs.

We applied our low-rank similarity measure with $r = 100$ on the graph of words which contains around 7000 words and the results are repre-

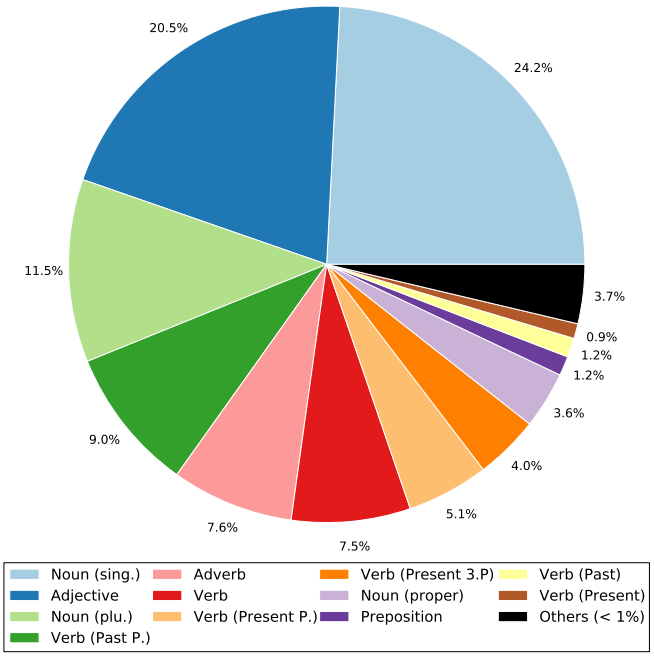


Fig. 6.12 Distribution of types of words in [Sapir (1921)].

sented in Fig. 6.13. In the figure, each pie chart corresponds to a cluster of similar words as extracted by our method and the distribution within a pie chart corresponds to the distribution of types of words in that cluster according to the Stanford POS tagger. We also give a list of the 15 most frequent words per cluster in Table 6.3.

First, note that the areas in the pie charts are not scaled by the number of times each word appears in the text. This means that the word "of" that appears 3636 times takes exactly the same area in a pie chart as any other word. If rescaled by the number of occurrences, the pie charts could have looked very differently. Let us now discuss the content of each cluster. Cluster 1 contains mainly adjectives but does not seem to be as well separated as it could be. Although containing some misplaced pronouns (“him” and “me”), cluster 2 is strikingly good and contains mostly verbs. Cluster 3 is also very good and contains adjectives and proper nouns that often serve the same purpose. Note that, surprisingly, this cluster contains all the adjectives related to countries (“greek”, “cambodgian”, “siamese”,etc.). An explanation behind the fact that this cluster is

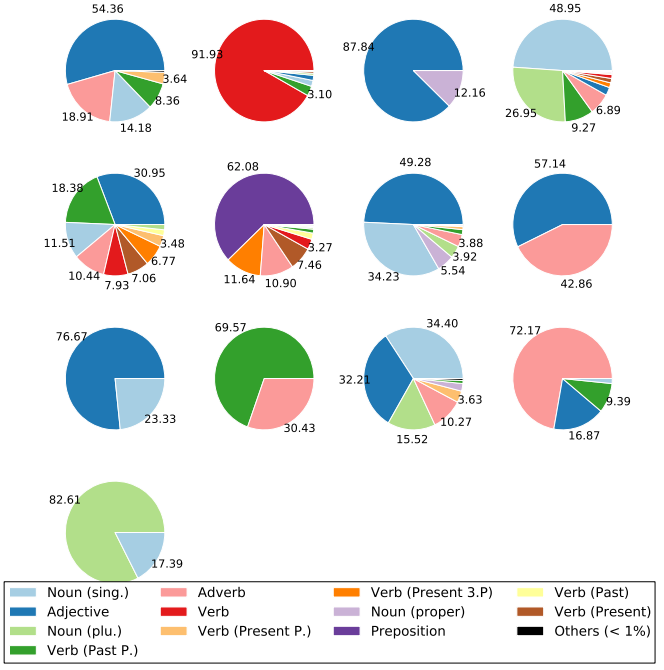


Fig. 6.13 Distribution of types of words per role extracted.

composed of adjectives and proper nouns is that some of those country related adjectives are in fact categorized as proper nouns by the Stanford POS tagger. Cluster 4 contains mainly prepositions and nouns that appear just before verbs (“it”, “one”, “there”) and its composition is actually better than it looks from the pie chart. Cluster 5 is clearly not a good cluster and contains many different types of words. Cluster 6 is again very satisfying and contains mostly short preposition. However, it also contains some verbs (“is”, “are”, “have”). Our guess is that this is due to some interrogative sentences present in the book that reverse the order of the words (“Is it…?”, “Are you…?”). Clusters 7 and 9 are composed of nouns and adjectives and we do not have a clear explanation why those words are cluster together, although they are better partitioned than cluster 5 since their content is much more uniform. Cluster 8 contains the adjectives and adverbs that often have the same position in a sentence. Cluster 10 also contains adverbs but that are more resembling past participles. By looking at the pie chart, cluster 11 seems mostly spread but, in fact, its structure is much more uniform and it contains mostly small

articles (“the”, “a”, “this”). The pie chart could again be biased by the Stanford tagger. Finally, cluster 12 contains mostly adverbs that seem correctly grouped together and cluster 13 contains only nouns which is excellent.

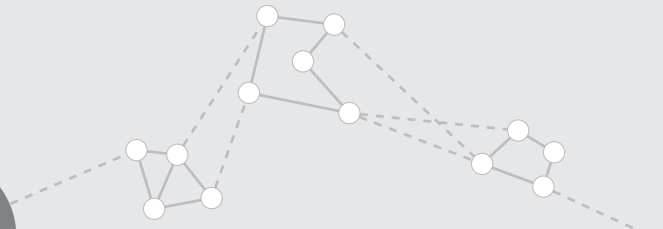
All those results indicate that, even if the clustering of words is far from perfect, most of the clusters extracted with our low-rank similarity are relevant and contain uniform types of words. Potentially, this could be of great interests for people working in the natural language processing field.

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
little (57)	be (582)	greek (48)	it (802)	well (47)	of (3636)	language (447)
simple (41)	say (128)	elusive (10)	languages (300)	related (47)	to (1820)	more (262)
further (39)	go (37)	cambodgian (9)	one (252)	possible (43)	in (1647)	english (254)
given (34)	become (33)	archaic (7)	there (222)	likely (36)	is (1557)	other (244)
greater (30)	him (28)	satisfying (5)	form (184)	difficult (33)	and (1514)	word (243)
definite (24)	show (24)	grouped (4)	elements (179)	known (32)	that (1220)	phonetic (141)
thing (24)	believe (23)	siamese (4)	element (176)	need (31)	as (1100)	radical (135)
dialect (24)	me (17)	nowhere (4)	forms (135)	seem (27)	or (795)	two (118)
good (22)	animate (17)	inclusive (4)	type (107)	nothing (27)	are (766)	sounds (107)
few (21)	indicate (16)	explicit (4)	expression (104)	according (26)	not (607)	sentence (105)
perfectly (16)	look (15)	religious (4)	number (102)	necessary (26)	by (411)	verb (104)
considerable (15)	fall (14)	infixes (4)	group (95)	contrast (25)	with (391)	most (100)
slight (13)	observe (14)	treated (4)	expressed (88)	come (23)	but (387)	drift (97)
polysynthetic (13)	serve (12)	formless (4)	far (80)	belong (21)	for (366)	same (95)
readily (13)	run (12)	syllabic (3)	types (79)	said (19)	have (364)	long (92)

Table 6.3 15 most frequent words per cluster (with number of occurrences).

Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12	Cluster 13
otherwise (9)	indefinite (9)	looked (16)	the (4934)	only (137)	nouns (32)
feminine (6)	infixed (8)	conveniently (7)	a (1673)	still (64)	imagery (8)
indirect (6)	abstraction (7)	enormously (4)	we (604)	too (60)	classifications (6)
six (4)	integral (6)	rapidly (4)	an (363)	entirely (34)	sing (5)
instrumental (4)	adequate (5)	eliminated (4)	this (304)	never (33)	agglutination (4)
intuitively (3)	exclusive (5)	termed (4)	all (299)	hardly (31)	automatic (4)
techniques (3)	immense (5)	barely (4)	its (292)	identical (25)	plan (4)
partial (3)	apple (4)	obsolete (3)	such (281)	really (22)	radically (4)
label (3)	all-important (3)	sleeping (3)	which (272)	clear (16)	release (4)
tsimshian (3)	entering (3)	swept (2)	words (229)	concerned (15)	guinea (3)
doer (3)	uncomfortable (3)	subdivided (2)	they (227)	red (13)	pre-verbal (3)
continuously (3)	absolutive (3)	multiplied (2)	these (218)	obvious (12)	sweeping (3)
lax (2)	unlimited (3)	diffused (2)	speech (216)	indicated (11)	vanished (3)
sister (2)	abbreviated (3)	tired (2)	i (212)	constantly (10)	similarities (3)
impeded (2)	imitative (2)	abundantly (1)	no (191)	done (10)	behavior (3)

Table 6.3(cont'd) 15 most frequent words per cluster (with number of occurrences).



7 Conclusion

Graph theory can be used to represent basically any complex system and we hope to have successfully convinced the reader that clustering is a topic of main interest in the study of modern networks. When one wants to extract information out of a mobile phone database, when one wants to analyze friendships or business relationships over online social networks with their millions of users, when one wants to study the topology of the internet or peer-to-peer networks containing billions of routers,... All these examples put the emphasis on the need for fast, efficient and dedicated algorithms and methods to cluster massive networks.

We have seen that one way to partition a network is to extract communities, i.e. groups of densely connected nodes that are supposed to share the same characteristics and features. Community detection is a challenging task but it provides powerful insights about the structure of networks and allows to analyze complex phenomena at different scales. After introducing the problem and presenting different existing methods, we have proposed a fast and highly parallelizable algorithm which produces partitions that are quantitatively similar to some of its competitors but clearly outperforms them in terms of speed and, for the first time, offers the possibility to study massive networks. We have shown that, in our framework, the only persistent elements in a network are its nodes and therefore developed our algorithm based on the assignment of each node to another node rather than to a partitioning set as done in most other methods. We have also shown that our algorithm is independent on the choice of the cost function that describes what one wants to identify as a community. We have proposed two types of local corrections of the node assignments based on global measures; however our framework is not constrained by those rules and other types of corrections could be introduced. This gives interesting opportunities for future research where corrections of the node assignments based on prior knowledge of the structure of the network could be developed. We have extensively tested our algorithm on benchmark graphs and also applied it to some real graphs. This has highlighted that the potential applications of our

method are extremely wide but also that dedicated cost functions should be developed in order to obtain the best possible clustering results in practical contexts. We hope that our community detection algorithm can help future scholars in their investigation on network properties.

But, communities are not always representative of the actual structure of a graph and we have presented different examples where the node similarity is somehow hidden in the graph topology. In this case, we have shown that the node distribution can be accurately represented by a role structures which form a generalization of the concept of communities. We have then introduced a pairwise node similarity measure with the hypothesis that nodes having the same role, i.e. being similar, should have the same kind of connectivity patterns across the graph. Our similarity measure computes the weighted sum of the number of common neighbors at any distance between every pair of nodes. However, we have shown that computing the exact similarity matrix can be computationally expensive for large graphs. Therefore, we proposed a low-rank approximation computed as the fixed point solution of an iterative scheme and proved its convergence when the weighting parameter is not too large. In our works, we have always applied a constant weighting parameter and previous studies have shown that sometimes a more appropriate scaling can be obtained by taking into account the length of the neighborhood patterns. Additional work in this direction could be done to improve the quality of the similarity measure; however, it seems much harder to compute the expected number of common neighbors rather than the expected number of paths in a graph.

Finally, we described how our pairwise node similarity measure, or its low-rank approximation, allows to correctly extract a partition using our community detection algorithm on the similarity graph. We applied our methodology to Erdős-Rényi graphs and to other models of random graphs containing a block structure and showed that, if the noise level is not too large and the block structure correctly represents the different roles of the nodes in the network, our similarity measures correctly extract these roles. We also observed that analyzing the evolution of the low-rank similarity measure might reveal the number of roles in a network and might also indicate if the extracted clusters are relevant. We demonstrated that both measures produce very similar results, hence justifying the use of our low-rank approximation in practical examples when computing the full rank measure is computationally too expensive. Lastly, we applied our low-rank similarity measure to three different real world examples.

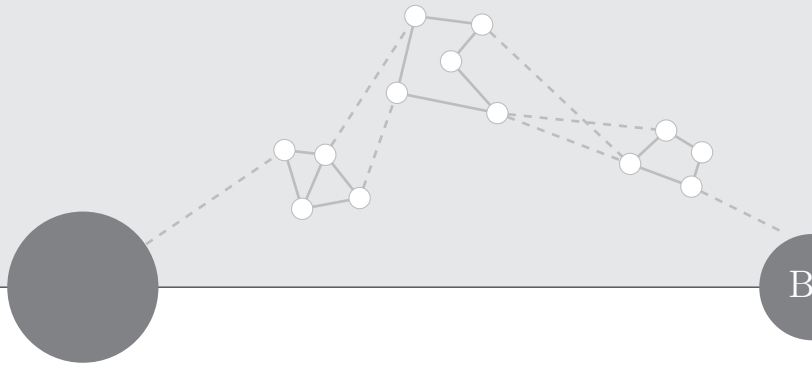
First, we have seen that one can correctly extract different trophic levels in an ecological food web and that the associated partition can even make more sense than the natural biological classification. Then, we explained how our similarity measure has been used to validate the existence of a community structure in a simulated network of reefs in the Great Barrier Reef of Australia. Lastly, we studied the partition obtained for a network of co-occurrence of words in a book which reveals similar types of words without prior knowledge on the structure of language. All those analyses have shown that additional research towards an appropriate cost function to partition the similarity graph are required. Yet, our results give a strong framework to tackle the network partitioning problem.

Perspective and future works

Our works open different interesting directions for future research and some open questions remain. We already stated some possible modifications of our fast community extraction algorithm in Section 3.4.6 but there are still a few properties of the method that are worth being investigated. First, a parallel implementation of the algorithm is still lacking and might reveal some sort of limitations because, as most community detection algorithms, our method requires mainly memory access rather than processing power. Therefore, at a certain point, increasing the number of processors might not speed up the computation of the communities since a bottleneck in the memory reading could emerge. This might constrain the size of the networks that can be analyzed and the actual behavior of a parallel implementation of the algorithm is worth studying. Regarding the partitioning problem in itself, a clear mathematical formalism to describe what are good communities is still to be discovered. We believe that the actual definition of the optimal community partition in a given network will always be dependent on the particular application one is interested in. This highlights another unstudied aspect of our algorithm which is its application on real graphs (and possibly using specific cost functions). However, different collaborations have been initiated with the development of our community extraction procedure and it starts to attract the attention of researchers and could lead to interesting results.

The role extraction problem suffers from the same drawbacks because it is somehow the same problem as for community detection with the additional constraint that the reduced graph is unknown. This makes the extraction of roles even harder and naturally leads to even more open

questions. First, it would be very interesting to describe a network model which will spontaneously lead the graphs created to have a role structure. Indeed, we used some random graph models to analyze the efficiency of our pairwise node similarity measure but these were fairly artificial. A model of growth, like the preferential attachment which creates small world networks with a power-law degree distribution, would be extremely interesting to study although rather hard to define. Regarding our similarity measure and our role extraction procedure, there also remains a few questions to consider. A thorough comparison of our results with other existing similarity measures should be conducted to confirm the efficiency of our role extraction method. Although, the appropriate choice of the similarity measure may depend on the type of network to study. Another issue is that the role extraction problem is tackled using two different procedure, i.e. first the computation of the pairwise node similarity measure and then the detection of communities within the similarity graph. Therefore, it could be interesting to study how the performance are affected by a modification of one or another of those procedures. Finally, we have studied some spectral properties of the low-rank similarity matrix to prove the convergence of our low-rank iterative scheme. One may have observed that we did not make any use of the singular vectors in the decomposition of Eq. (5.44). However, it is easy to see that if the similarity matrix is of sufficiently small rank then those singular vectors may indicate an appropriate partition of the network in roles and one would not require to apply a community detection method. The spectral properties of the similarity matrix may give further insights on the existing role structure.



Bibliography

Ahn, YY, Bagrow, JP, and Lehmann, S (2010). Link communities reveal multiscale complexity in networks. *Nature*, 466. doi: 10.1038/nature09182.

Albert, R and Barabási, AL (2002). Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97. doi: 10.1103/RevModPhys.74.47.

Albert, R, Jeong, H, and Barabási, AL (1999). Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131. doi: 10.1038/43601.

Aldecoa, R and Marín, I (2013). Surprise maximization reveals the community structure of complex networks. *Scientific Reports*, 3:1060. doi: 10.1038/srep01060.

Aldecoa, R and Marín, I (2011). Deciphering network community structure by surprise. *PLoS ONE*, 6(9):e24195. doi: 10.1371/journal.pone.0024195.

Alzate, C and Suykens, JAK (2008). Sparse kernel models for spectral clustering using the incomplete cholesky decomposition. In *IEEE International Joint Conference on Neural Networks (IJCNN), 2008*, pages 3556–3563. doi: 10.1109/IJCNN.2008.4634306.

Alzoubi, H and Pan, WD (2008). Fast and accurate global motion estimation algorithm using pixel subsampling. *Information Sciences*, 178(17):3415 – 3425. doi: 10.1016/j.ins.2008.05.004.

Anteneodo, C, Malmgren, RD, and Chialvo, DR (2010). Poissonian bursts in e-mail correspondence. *The European Physical Journal B*, 75(3):389–394. doi: 10.1140/epjb/e2010-00139-9.

Arbelaez, P, Maire, M, Fowlkes, C, and Malik, J (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916. doi: 10.1109/TPAMI.2010.161.

Arenas, A, Fernández, A, Fortunato, S, and Gómez, S (2008). Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224001. doi: 10.1088/1751-8113/41/22/224001.

Arenas, A, Fernández, A, and Gómez, S (2008). Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039. doi: 10.1088/1367-2630/10/5/053039.

Arnau, V, Mars, S, and Marín, I (2005). Iterative cluster analysis of protein interaction data. *Bioinformatics*, 21(3):364–378. doi: 10.1093/bioinformatics/bti021.

Bagrow, JP (2012). Communities and bottlenecks: Trees and tree-like networks have high modularity. *Phys. Rev. E*, 85:066118. doi: 10.1103/PhysRevE.85.066118.

Balcan, D, Colizza, V, Gonçalves, B, Hu, H, Ramasco, JJ, et al. (2009). Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the National Academy of Sciences*. doi: 10.1073/pnas.0906910106.

Barabási, AL (2005). The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–11. doi: 10.1038/nature03459.

Barabási, AL and Albert, R (1999). Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512. doi: 10.1126/science.286.5439.509.

Barrat, A, Barthélemy, M, Pastor-Satorras, R, and Vespignani, A (2004). The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752. doi: 10.1073/pnas.0400087101.

Beguerisse-Díaz, M, Garduño-Hernández, G, Vangelov, B, Yaliraki, SN, and Barahona, M (2013). Communities, roles, and informational organigrams in directed networks: the Twitter network of the UK riots. *ArXiv e-prints*. arXiv:physics.soc-ph/1311.6785.

Beguerisse-Diaz, M, Vangelov, B, and Barahona, M (2013). Finding role communities in directed networks using role-based similarity, markov stability and the relaxed minimum spanning tree. *ArXiv e-prints*. arXiv:1309.1795.

Beguerisse-Díaz, M, Vangelov, B, and Barahona, M (2013). Finding role communities in directed networks using role-based similarity, markov stability and the relaxed minimum spanning tree. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 937–940. doi: 10.1109/GlobalSIP.2013.6737046.

Berry, JW, Hendrickson, B, LaViolette, RA, and Phillips, CA (2011). Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E*, 83:056119. doi: 10.1103/PhysRevE.83.056119.

Beucher, S et al. (1991). The watershed transformation applied to image segmentation. In *Scanning Microscopy International*, pages 299–314.

Bhowmick, S and Srinivasan, S (2013). A template for parallelizing the louvain method for modularity maximization. In Mukherjee, A, Choudhury, M, Peruani, F, Ganguly, N, and Mitra, B, editors, *Dynamics On and Of Complex Networks, Volume 2*, Modeling and Simulation in Science, Engineering and Technology, pages 111–124. Springer New York. doi: 10.1007/978-1-4614-6729-8_6.

Blondel, V, Deville, P, Morlot, F, Smoreda, Z, Van Dooren, P, et al. (2011). Voice on the border: Do cellphones redraw the maps? http://www.paristechreview.com/2011/11/15/voice-border-cellphones-redraw-maps/. [online].

Blondel, V, Gajardo, A, Heymans, M, Senellart, P, and Van Dooren, P (2004). A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Review*, 46(4):647–666. doi: 10.1137/S0036144502415960.

Blondel, V, Krings, G, and Thomas, I (2010). Regions and borders of mobile telephony in belgium and in the brussels metropolitan zone. *Brussels Studies*, 42(4). http://www.brusselsstudies.be/publications/index/index/id/129/lang/en.

Blondel, VD, Guillaume, JL, Lambiotte, R, and Lefebvre, E (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008. doi: 10.1088/1742-5468/2008/10/P10008.

Bollobás, B (2001). *Random Graphs*. Number 4 in Wiley Series in Disc. Math. and Opt. Cambridge University Press. ISBN 0521797225. doi: 10.1214/aoms/1177706098.

Borgatti, SP and Everett, MG (1993). Two algorithms for computing regular equivalence. *Social Networks*, 15(4):361 – 376. doi: http://dx.doi.org/10.1016/0378-8733(93)90012-A.

Brandes, U, Delling, D, Gaertler, M, Gorke, R, Hoefer, M, et al. (2008). On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188. doi: 10.1109/TKDE.2007.190689.

Brandes, U, Delling, D, Gaertler, M, Görke, R, Hoefer, M, et al. (2006). On modularity – np-completeness and beyond.

Brandes, U, Gaertler, M, and Wagner, D (2003). Experiments on graph clustering algorithms. In Battista, G and Zwick, U, editors, *Algorithms - ESA 2003*, volume 2832 of *Lecture Notes in Computer Science*, pages 568–579. Springer Berlin Heidelberg. doi: 10.1007/978-3-540-39658-1_52.

Browet, A (2011). Community detection for hierarchical video segmentation. *10th IMACS International Symposium on Iterative Methods in Scientific Computing*, page 22.

Browet, A, Absil, PA, and Van Dooren, P (2011). Community detection for hierarchical image segmentation. In Aggarwal, J, Barneva, R, Brimkov, V, Koroutchev, K, and Korutcheva, E, editors, *Combinatorial Image Analysis*, volume 6636 of *Lecture Notes in Computer Science*, pages 358–371. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-21073-0_32.

Browet, A, Absil, PA, and Van Dooren, P (2013). Fast community detection using local neighbourhood search. *ArXiv e-prints*. arXiv:1308.6276.

Browet, A and Van Dooren, P (2013). Low-rank Similarity Measure for Role Model Extraction. *ArXiv e-prints*. arXiv:1312.4860.

Burt, RS (1976). Positions in networks. *Social forces*, 55(1):93–122. doi: 10.2307/2577097.

Carli, R, Fagnani, F, Frasca, P, and Zampieri, S (2010). Gossip consensus algorithms via quantized communication. *Automatica*, 46(1):70 – 80. doi: http://dx.doi.org/10.1016/j.automatica.2009.10.032.

Cason, TP (2012). *Role Extraction in Networks*. Ph.D. thesis, Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université catholique de Louvain. http://hdl.handle.net/2078.1/114511.

Cason, TP, Absil, PA, and Van Dooren, P (2013). Iterative methods for low rank approximation of graph similarity matrices. *Linear Algebra and its Applications*, 438(4):1863 – 1882. doi: 10.1016/j.laa.2011.12.004.

Clauset, A and Eagle, N (2007). Persistence and periodicity in a dynamic proximity network. *October*.

Clauset, A, Newman, MEJ, and Moore, C (2004). Finding community structure in very large networks. *Phys. Rev. E*, 70:066111. doi: 10.1103/PhysRevE.70.066111.

Clauset, A, Shalizi, C, and Newman, M (2009). Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703. doi: 10.1137/070710111.

Condon, A and Karp, RM (2001). Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2):116–140. doi: 10.1002/1098-2418(200103)18:2<116::AID-RSA1001>3.0.CO;2-2.

Conover, MD, Davis, C, Ferrara, E, McKelvey, K, Menczer, F, et al. (2013). The Geospatial Characteristics of a Social Movement Communication Network. *PLoS ONE*, 8(3):e55957. doi: 10.1371/journal.pone.0055957.

Cooper, K and Barahona, M (2011). Role-similarity based comparison of directed networks. *ArXiv e-prints*. arXiv:1103.5582.

Cour, T, Benezit, F, and Shi, J (2005). Spectral segmentation with multiscale graph decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2005.*, volume 2, pages 1124–1131 vol. 2. doi: 10.1109/CVPR.2005.332.

Cover, TM and Thomas, JA (2012). *Elements of information theory*. John Wiley & Sons. ISBN 0471241954.

Cox, IJ, Rao, S, and Zhong, Y (1996). "Ratio regions": a technique for image segmentation. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 2, pages 557–564. doi: 10.1109/ICPR.1996.546886.

Csardi, G and Nepusz, T (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems(1695):1695.

Csáji, BC, Browet, A, Traag, V, Delvenne, JC, Huens, E, et al. (2013). Exploring the mobility of mobile phone users. *Physica*

A: Statistical Mechanics and its Applications, 392(6):1459 – 1473. doi: 10.1016/j.physa.2012.11.040.

Danon, L, Díaz-Guilera, A, and Arenas, A (2006). The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(11):P11010.

Danon, L, Díaz-Guilera, A, Duch, J, and Arenas, A (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008. doi: 10.1088/1742-5468/2005/09/P09008.

De Domenico, M, Solé-Ribalta, A, Cozzo, E, Kivelä, M, Moreno, Y, et al. (2013). Mathematical formulation of multilayer networks. *Phys. Rev. X*, 3:041022. doi: 10.1103/PhysRevX.3.041022.

Delvenne, JC, Schaub, M, Yaliraki, SN, and Barahona, M (2013). The stability of a graph partition: A dynamics-based framework for community detection. In Mukherjee, A, Choudhury, M, Peruani, F, Ganguly, N, and Mitra, B, editors, *Dynamics On and Of Complex Networks, Volume 2*, Modeling and Simulation in Science, Engineering and Technology, pages 221–242. Springer New York. ISBN 978-1-4614-6728-1. doi: 10.1007/978-1-4614-6729-8_11.

Delvenne, JC, Yaliraki, SN, and Barahona, M (2010). Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences*, 107(29):12755–12760. doi: 10.1073/pnas.0903215107.

Denayer, D (2012). *Modélisation par rôles de grands graphes*. Master's thesis, Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université catholique de Louvain.

Diestel, R (2005). *Graph theory*. Springer, 4 edition. ISBN 978-3-642-14278-9.

Doreian, P, Batagelj, V, and Ferligoj, A (2005). *Generalized blockmodeling*. 25. Cambridge University Press. ISBN 9780521840859.

Erdős, P and Rényi, A (1960). On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61.

Euler, L (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140.

Evans, T, Lambiotte, R, and Panzarasa, P (2011). Community structure and patterns of scientific collaboration in business and management. *Scientometrics*, 89(1):381–396. doi: 10.1007/s11192-011-0439-1.

Everett, MG and Borgatti, SP (1994). Regular equivalence: General theory. *The Journal of Mathematical Sociology*, 19(1):29–52. doi: 10.1080/0022250X.1994.9990134.

Everett, MG and Borgatti, SP (1996). Exact colorations of graphs and digraphs. *Social Networks*, 18(4):319 – 331. doi: http://dx.doi.org/10.1016/0378-8733(95)00286-3.

Fan, Y, Li, M, Zhang, P, Wu, J, and Di, Z (2007). Accuracy and precision of methods for community identification in weighted networks. *Physica A: Statistical Mechanics and its Applications*, 377(1):363 – 372. doi: 10.1016/j.physa.2006.11.036.

Farkas, I, Ábel, D, Palla, G, and Vicsek, T (2007). Weighted network modules. *New Journal of Physics*, 9(6):180. doi: 10.1088/1367-2630/9/6/180.

Felzenszwalb, P and Huttenlocher, D (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181. doi: 10.1023/B:VISI.0000022288.19776.77.

Fiedler, M (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305. permalink: http://dml.cz/dmlcz/101168.

Fortunato, S (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75–174. doi: 10.1016/j.physrep.2009.11.002.

Fortunato, S and Barthélemy, M (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41. doi: 10.1073/pnas.0605965104.

Frederix, K and Barel, MV (2013). Sparse spectral clustering method based on the incomplete cholesky decomposition. *Journal of Computational and Applied Mathematics*, 237(1):145 – 161. doi: 10.1016/j.cam.2012.07.019.

Garlaschelli, D (2009). The weighted random graph model. *New Journal of Physics*, 11(7):073005. doi: 10.1088/1367-2630/11/7/073005.

Geuzaine, C and Remacle, J (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331. doi: 10.1002/nme.2579.

Girvan, M and Newman, MEJ (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826. doi: 10.1073/pnas.122653799.

González, MC and Barabási, AL (2007). Complex networks: From data to models. *Nature Physics*, 3(4):224–225. doi: 10.1038/nphys581.

Gonzalez, MC, Hidalgo, CA, and Barabasi, AL (2008). Understanding individual human mobility patterns. *Nature*, 453(7196):479–482. doi: 10.1038/nature06958.

Good, BH, de Montjoye, YA, and Clauset, A (2010). Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106. doi: 10.1103/PhysRevE.81.046106.

Granovetter, M (1973). The Strength of Weak Ties. *American Journal of Sociology*, 78(6):1360–1380. doi: 10.1086/225469.

Granovetter, M (1983). The strength of weak ties: A network theory revisited. *Sociological Theory*, 1:201–233. doi: 10.2307/202051.

Guimerà, R, Mossa, S, Turtschi, A, and Amaral, LAN (2005). The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, 102(22):7794–7799. doi: 10.1073/pnas.0407994102.

Guimera, R and Amaral, LAN (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900. doi: 10.1038/nature03288.

Guimerà, R, Sales-Pardo, M, and Amaral, LAN (2004). Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:025101. doi: 10.1103/PhysRevE.70.025101.

Guimerà, R, Sales-Pardo, M, and Amaral, LAN (2007). Module identification in bipartite and directed networks. *Phys. Rev. E*, 76:036102. doi: 10.1103/PhysRevE.76.036102.

Guimerà, R, Stouffer, DB, Sales-Pardo, M, Leicht, EA, Newman, MEJ, et al. (2010). Origin of compartmentalization in food webs. *Ecology*, 91(10):2941–2951. doi: 10.1890/09-1175.1.

Hagen, L and Kahng, A (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085. doi: 10.1109/43.159993.

Halevy, A, Norvig, P, and Pereira, F (2009). The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12. doi: 10.1109/MIS.2009.36.

Haynes, J and Perisic, I (2009). Mapping search relevance to social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, pages 1–7. ACM, New York, NY, USA. doi: 10.1145/1731011.1731013.

Ho, J, Lee, KC, Yang, MH, and Kriegman, D (2004). Visual tracking using learned linear subspaces. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–782–I–789 Vol.1. doi: 10.1109/CVPR.2004.1315111.

Holme, P, Kim, BJ, Yoon, CN, and Han, SK (2002). Attack vulnerability of complex networks. *Phys. Rev. E*, 65:056109. doi: 10.1103/PhysRevE.65.056109.

Horn, RA and Johnson, CR (1990). *Matrix Analysis*. Cambridge University Press. ISBN 0521386322.

Hu, D, Ronhovde, P, and Nussinov, Z (2012). Replica inference approach to unsupervised multiscale image segmentation. *Phys. Rev. E*, 85:016101. doi: 10.1103/PhysRevE.85.016101.

Hu, Y, Chen, H, Zhang, P, Li, M, Di, Z, et al. (2008). Comparative definition of community and corresponding identifying algorithm. *Phys. Rev. E*, 78:026121. doi: 10.1103/PhysRevE.78.026121.

Huffman, D (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101. doi: 10.1109/JRPROC.1952.273898.

Janson, S, Luczak, T, and Kolchin, V (2000). *Random graphs*. Cambridge Univ Press. ISBN 978-0-471-17541-4.

Jeong, H, Tombor, B, Albert, R, Oltvai, ZN, and Barabasi, AL (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654. doi: 10.1038/35036627.

Kang, U, Tsourakakis, C, Appel, AP, Faloutsos, C, and Leskovec, J (2008). *HADI: Fast diameter estimation and mining in massive graphs with Hadoop*. Carnegie Mellon University, School of Computer Science, Machine Learning Department.

Kashtan, N and Alon, U (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–8. doi: 10.1073/pnas.0503610102.

Kass, M, Witkin, A, and Terzopoulos, D (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331. doi: 10.1007/BF00133570.

Kim, J, Krapivsky, PL, Kahng, B, and Redner, S (2002). Infinite-order percolation and giant fluctuations in a protein interaction network. *Physical Review E*, 66:055101+. doi: 10.1103/physreve.66.055101.

Kirkpatrick, S, Gelatt, CD, and Vecchi, MP (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. doi: 10.1126/science.220.4598.671.

Kivelä, M, Arenas, A, Barthelemy, M, Gleeson, JP, Moreno, Y, et al. (2014). Multilayer networks. *Journal of Complex Networks*. doi: 10.1093/comnet/cnu016.

Klein, D (2010). Centrality measure in graphs. *Journal of Mathematical Chemistry*, 47(4):1209–1223. doi: 10.1007/s10910-009-9635-0.

Kleinberg, JM (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632. doi: 10.1145/324133.324140.

Kolaczyk, ED (2009). Statistical Analysis of Network Data. *Network*, 66(2):123–152. doi: 10.1007/978-0-387-88146-1.

Krings, G, Dabin, D, and Blondel, V (2011). Communities in a crime network. In *NetSci2011*.

Kumpula, JM, Kivelä, M, Kaski, K, and Saramäki, J (2008). Sequential algorithm for fast clique percolation. *Phys. Rev. E*, 78:026109. doi: 10.1103/PhysRevE.78.026109.

Lambiotte, R, Delvenne, JC, and Barahona, M (2008). Laplacian Dynamics and Multiscale Modular Structure in Networks. *ArXiv e-prints*. arXiv:0812.1770.

Lancichinetti, A and Fortunato, S (2009a). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 80:016118. doi: 10.1103/PhysRevE.80.016118.

Lancichinetti, A and Fortunato, S (2009b). Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117. doi: 10.1103/PhysRevE.80.056117.

Lancichinetti, A, Fortunato, S, and Radicchi, F (2008). Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110. doi: 10.1103/PhysRevE.78.046110.

Lancichinetti, A, Radicchi, F, Ramasco, JJ, and Fortunato, S (2011). Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961. doi: 10.1371/journal.pone.0018961.

Lazer, D, Pentland, A, Adamic, L, Aral, S, Barabasi, AL, et al. (2009). Computational Social Science. *Science*, 323(5915):721–723. doi: 10.1126/science.1167742.

Leicht, EA, Holme, P, and Newman, MEJ (2006). Vertex similarity in networks. *Phys. Rev. E*, 73:026120. doi: 10.1103/PhysRevE.73.026120.

Leicht, EA and Newman, MEJ (2008). Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703. doi: 10.1103/PhysRevLett.100.118703.

Leskovec, J, Adamic, LA, and Huberman, BA (2006). The dynamics of viral marketing. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 228–237. ACM Press. doi: 10.1145/1134707.1134732.

Liu, Z and Hu, B (2005). Epidemic spreading in community networks. *Europhysics Letters*, 72(2):315–321. doi: 10.1209/epl/i2004-10550-5.

Lorrain, F and White, HC (1971). Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80. doi: 10.1080/0022250X.1971.9989788.

Luczak, T (1989). Sparse random graphs with a given degree sequence. In *Proceedings of the Symposium on Random Graphs, Poznan*, pages 165–182.

Luczkovich, JJ, Borgatti, SP, Johnson, JC, and Everett, MG (2003). Defining and measuring trophic role similarity in food webs using regular equivalence. *Journal of Theoretical Biology*, 220(3):303 – 321. doi: http://dx.doi.org/10.1006/jtbi.2003.3147.

Lupu, Y and Traag, VA (2012). Trading communities, the networked structure of international relations, and the kantian peace. *Journal of Conflict Resolution*. doi: 10.1177/0022002712453708.

MacKay, DJ (2003). *Information theory, inference and learning algorithms*. Cambridge university press. ISBN 978-0521642989.

Maggio, DE and Cavallaro, DA (2011). *Video Tracking: Theory and Practice*. Wiley Publishing, 1st edition. ISBN 978-0-470-74964-7.

Martin, D, Fowlkes, C, Tal, D, and Malik, J (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423. doi: 10.1109/ICCV.2001.937655.

Meilă, M (2007). Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873 – 895. doi: 10.1016/j.jmva.2006.11.013.

Milenković, T, Filippis, I, Lappe, M, and Pržulj, N (2009). Optimized null model for protein structure networks. *PLoS ONE*, 4(6):e5967. doi: 10.1371/journal.pone.0005967.

Milo, R, Shen-Orr, S, Itzkovitz, S, Kashtan, N, Chklovskii, D, et al. (2002). Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827. doi: 10.1126/science.298.5594.824.

Molloy, M and Reed, B (1995). A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2-3):161–180. doi: 10.1002/rsa.3240060204.

Mondragón, RJ (2014). Network null-model based on maximal entropy and the rich-club. *Journal of Complex Networks*, 2(3):288–298. doi: 10.1093/comnet/cnu006.

de Montgolfier, F, Soto, M, and Viennot, L (2011). Asymptotic modularity of some graph classes. In *Proceedings of the 22Nd International Conference on Algorithms and Computation*, pages 435–444. Springer-Verlag. doi: 10.1007/978-3-642-25591-5_45.

Mucha, PJ, Richardson, T, Macon, K, Porter, MA, and Onnela, JP (2010). Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878. doi: 10.1126/science.1184819.

Newman, M (2003). The structure and function of complex networks. *SIAM Review*, 45(2):167–256. doi: 10.1137/S003614450342480.

Newman, M (2010). *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA. ISBN 0199206651.

Newman, MEJ (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409. doi: 10.1073/pnas.98.2.404.

Newman, MEJ (2004a). Analysis of weighted networks. *Phys. Rev. E*, 70:056131. doi: 10.1103/PhysRevE.70.056131.

Newman, MEJ (2004b). Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133. doi: 10.1103/PhysRevE.69.066133.

Newman, MEJ (2006). Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104. doi: 10.1103/PhysRevE.74.036104.

Newman, MEJ and Girvan, M (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113. doi: 10.1103/PhysRevE.69.026113.

Newman, MEJ, Strogatz, SH, and Watts, DJ (2001). Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64:026118. doi: 10.1103/PhysRevE.64.026118.

Nowak, MA (2006). *Evolutionary dynamics: exploring the equations of life*. Harvard University Press. ISBN 0674023382.

Olszewska, JI (2009). *Unified framework for multi-feature active contour*. Ph.D. thesis, Universite catholique de Louvain, Ecole polytechnique.

Page, L, Brin, S, Motwani, R, and Winograd, T (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab.

Palla, G, Derényi, I, Farkas, I, and Vicsek, T (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818. doi: 10.1038/nature03607.

Perry, PO and Wolfe, PJ (2012). Null models for network data. *ArXiv e-prints*. arXiv:math.ST/1201.5871.

Porter, MA, Onnela, JP, and Mucha, PJ (2009). Communities in networks. *Notices of the AMS*, 56(9):1082–1097. http://www.ams.org/notices/200909/rtx090901082p.pdf.

Radicchi, F, Castellano, C, Cecconi, F, Loreto, V, and Parisi, D (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663. doi: 10.1073/pnas.0400054101.

Raghavan, UN, Albert, R, and Kumara, S (2007). Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106. doi: 10.1103/PhysRevE.76.036106.

Rath, B (2010). *Asymptotic behavior of random graphs evolving in time*. Ph.D. thesis, Institute of Mathematics, Budapest University of Technology and Economics.

Ratti, C, Sobolevsky, S, Calabrese, F, Andris, C, Reades, J, et al. (2010). Redrawing the Map of Great Britain from a Network of Human Interactions. *PLoS ONE*, 5(12). doi: 10.1371/journal.pone.0014248.

Reichardt, J and Bornholdt, S (2006). Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110. doi: 10.1103/PhysRevE.74.016110.

Reichardt, J and White, DR (2007). Role models for complex networks. *The European Physical Journal B*, 60(2):217–224. doi: 10.1140/epjb/e2007-00340-y.

Richardson, T, Mucha, PJ, and Porter, MA (2009). Spectral tripartitioning of networks. *Phys. Rev. E*, 80:036111. doi: 10.1103/PhysRevE.80.036111.

Ronhovde, P and Nussinov, Z (2009). Multiresolution community detection for megascale networks by information-based replica correlations. *Phys. Rev. E*, 80:016109. doi: 10.1103/PhysRevE.80.016109.

Ronhovde, P and Nussinov, Z (2010). Local resolution-limit-free potts model for community detection. *Phys. Rev. E*, 81:046114. doi: 10.1103/PhysRevE.81.046114.

Rosvall, M and Bergstrom, CT (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123. doi: 10.1073/pnas.0706851105.

Rosvall, M and Bergstrom, CT (2010). Mapping change in large networks. *PLoS ONE*, 5(1):e8694. doi: 10.1371/journal.pone.0008694.

Rosvall, M and Bergstrom, CT (2011). Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE*, 6(4):e18209. doi: 10.1371/journal.pone.0018209.

Sapir, E (1921). *Language: an introduction to the study of speech*. New York: Harcourt, Brace and company. ISBN 1108063780.

Sarlette, A, Tuna, SE, Blondel, V, and Sepulchre, R (2008). Global synchronization on the circle. In *Proceedings of the 17th IFAC World Congress*.

Schaub, MT, Delvenne, JC, Yaliraki, SN, and Barahona, M (2012). Markov dynamics as a zooming lens for multiscale community detection: Non clique-like communities and the field-of-view limit. *PLoS ONE*, 7(2):e32210. doi: 10.1371/journal.pone.0032210.

Schaub, MT, Lambiotte, R, and Barahona, M (2012). Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation. *Phys. Rev. E*, 86:026112. doi: 10.1103/PhysRevE.86.026112.

Schuetz, P and Caflisch, A (2008). Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Phys. Rev. E*, 77:046112. doi: 10.1103/PhysRevE.77.046112.

Scott, J (2000). *Social Network Analysis: A Handbook*. Sage Publications. ISBN 0761963391.

Shannon, CE (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Shaw, L and Tunc, I (2012). Epidemic spread in adaptive social networks with community structure. In *Bio-Inspired Models of Network, Information, and Computing Systems*, volume 87, pages 519–520. Springer. doi: 10.1007/978-3-642-32615-8_49.

Shi, J and Malik, J (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905. doi: 10.1109/34.868688.

Simon, HA and Ando, A (1961). Aggregation of variables in dynamic systems. *Econometrica*, 29(2):111–138.

Skillicorn, D and Talia, D (2012). Mining large data sets on grids: Issues and prospects. *Computing and Informatics*, 21:347–362. http://www.cai.sk/ojs/index.php/cai/article/viewArticle/488.

Staudt, C and Meyerhenke, H (2013). Engineering high-performance community detection heuristics for massive graphs. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 180–189. doi: 10.1109/ICPP.2013.27.

Stewart, G (1973). Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15(4):727–764. doi: 10.1137/1015095.

Sundaramoorthi, G, Mennucci, A, Soatto, S, and Yezzi, A (2009). Tracking deforming objects by filtering and prediction in the space of curves. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, pages 2395–2401. doi: 10.1109/CDC.2009.5400786.

Thomas, CJ, Lambrechts, J, Wolanski, E, Traag, VA, Blondel, VD, et al. (2014). Numerical modelling and graph theory tools to study ecological connectivity in the great barrier reef. *Ecological Modelling*, 272(0):160 – 174. doi: 10.1016/j.ecolmodel.2013.10.002.

Tibély, G and Kertész, J (2008). On the equivalence of the label propagation method of community detection and a potts model approach. *Physica A: Statistical Mechanics and its Applications*, 387(19–20):4982 – 4984. doi: 10.1016/j.physa.2008.04.024.

Tizzoni, M, Bajardi, P, Decuyper, A, King, GKK, Schneider, CM, et al. (2013). On the use of human mobility proxy for the modeling of epidemics. *ArXiv e-prints*. arXiv:1309.7272.

Torralba, A, Fergus, R, and Freeman, W (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(11):1958–1970. doi: 10.1109/TPAMI.2008.128.

Toutanova, K, Klein, D, Manning, CD, and Singer, Y (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180. Association for Computational Linguistics, Stroudsburg, PA, USA. doi: 10.3115/1073445.1073478.

Traag, V (2013). *Groups and Reputation in Social Networks Graph Theoretic Algorithms and Dynamical Models*. Ph.D. thesis, Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université catholique de Louvain. http://dial.academielouvain.be/handle/boreal:134615.

Traag, V, Krings, G, and Van Dooren, P (2013). Significant scales in community structure. *Scientific Reports*, 3:2930. doi: 10.1038/srep02930.

Traag, VA, Browet, A, Calabrese, F, and Morlot, F (2011). Social event detection in massive mobile phone data using probabilistic location inference. In *IEEE 3rd International Conference on Social Computing (socialcom)*, pages 625–628. doi: 10.1109/PASSAT/SocialCom.2011.133.

Traag, VA, Van Dooren, P, and De Leenheer, P (2013). Dynamical models explaining social balance and evolution of cooperation. *PLoS ONE*, 8(4):e60063. doi: 10.1371/journal.pone.0060063.

Traag, VA, Van Dooren, P, and Nesterov, Y (2011). Narrow scope for resolution-limit-free community detection. *Phys. Rev. E*, 84:016114. doi: 10.1103/PhysRevE.84.016114.

Ulanowicz, RE and DeAngelis, DL (1999). Network analysis of trophic dynamics in south florida ecosystems. In *Proceedings of the South Florida Restoration Science Forum*, pages 114–115.

Wakita, K and Tsurumi, T (2007). Finding community structure in mega-scale social networks: [extended abstract]. In *Proceedings of the 16th International Conference on World Wide Web*, pages 1275–1276. ACM, New York, NY, USA. doi: 10.1145/1242572.1242805.

Wang, G, Shen, Y, and Ouyang, M (2008). A vector partitioning approach to detecting community structure in complex networks. *Computers & Mathematics with Applications*, 55(12):2746 – 2752. doi: 10.1016/j.camwa.2007.10.028.

Wang, S and Siskind, J (2001). Image segmentation with minimum mean cut. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 517–524 vol.1. doi: 10.1109/ICCV.2001.937560.

Wasserman, S and Faust, K (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press. ISBN 0521387078.

Watts, DJ and Strogatz, SH (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442. doi: 10.1038/30918.

Weir, MD, Hass, J, and Thomas, GB (2013). *Thomas' calculus*. Pearson Education. ISBN 9780321878960.

Werman, M, Peleg, S, and Rosenfeld, A (1985). A distance metric for multi-dimensional histograms. *Computer Vision, Graphics, and Image Processing*, 32(3):328 – 336. doi: 10.1016/0734-189X(85)90055-6.

West, DB (2001). *Introduction to Graph Theory*. Dover books on advanced mathematics. Prentice Hall. ISBN 0130144002.

White, DR and Reitz, KP (1983). Graph and semigroup homomorphisms on networks of relations. *Social Networks*, 5(2):193 – 234. doi: 10.1016/0378-8733(83)90025-4.

Wu, X and Liu, Z (2008). How community structure influences epidemic spread in social networks. *Physica A: Statistical Mechanics and its Applications*, 387(2-3):623–630. doi: 10.1016/j.physa.2007.09.039.

Wu, Z and Leahy, R (1993). An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(11):1101–1113. doi: 10.1109/34.244673.

Zaslavsky, T (1982). Signed graphs. *Discrete Applied Mathematics*, 4(1):47–74. doi: 10.1016/0166-218x(82)90033-6.