Physics Reports 486 (2010) 75–174

Contents lists available at ScienceDirect

Physics Reports

journal homepage: www.elsevier.com/locate/physrep

# Community detection in graphs

Santo Fortunato *

*Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino, I, Italy*

ARTICLE INFO

*Article history:*
Accepted 5 November 2009
Available online 4 December 2009
editor: I. Procaccia

*Keywords:*
Graphs
Clusters
Statistical physics

ABSTRACT

The modern science of networks has brought significant advances to our understanding of complex systems. One of the most relevant features of graphs representing real systems is community structure, or clustering, i.e. the organization of vertices in clusters, with many edges joining vertices of the same cluster and comparatively few edges joining vertices of different clusters. Such clusters, or communities, can be considered as fairly independent compartments of a graph, playing a similar role like, e.g., the tissues or the organs in the human body. Detecting communities is of great importance in sociology, biology and computer science, disciplines where systems are often represented as graphs. This problem is very hard and not yet satisfactorily solved, despite the huge effort of a large interdisciplinary community of scientists working on it over the past few years. We will attempt a thorough exposition of the topic, from the definition of the main elements of the problem, to the presentation of most methods developed, with a special focus on techniques designed by statistical physicists, from the discussion of crucial issues like the significance of clustering and how methods should be tested and compared against each other, to the description of applications to real networks.



**Contents**



* Tel.: +39 011 6603090; fax: +39 011 6600049.
*E-mail address:* fortunato@isi.it.

0370-1573/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.physrep.2009.11.002

## 1. Introduction

The origin of graph theory dates back to Euler's solution of the puzzle of Königsberg's bridges in 1736 [1]. Since then a lot has been learned about graphs and their mathematical properties [2]. In the 20th century they have also become extremely useful as the representation of a wide variety of systems in different areas. Biological, social, technological, and information networks can be studied as graphs, and graph analysis has become crucial to understand the features of these systems. For instance, social network analysis started in the 1930's and has become one of the most important topics in sociology [3,4]. In recent times, the computer revolution has provided scholars with a huge amount of data and computational resources to process and analyze these data. The size of real networks one can potentially handle has also grown considerably, reaching millions or even billions of vertices. The need to deal with such a large number of units has produced a deep change in the way graphs are approached [5–10].

Graphs representing real systems are not regular like, e.g., lattices. They are objects where order coexists with disorder. The paradigm of disordered graph is the random graph, introduced by Erdös and Rényi [11]. In it, the probability of having an edge between a pair of vertices is equal for all possible pairs (see Appendix). In a random graph, the distribution of edges among the vertices is highly homogeneous. For instance, the distribution of the number of neighbors of a vertex, or *degree*, is binomial, so most vertices have equal or similar degree. Real networks are not random graphs, as they display big inhomogeneities, revealing a high level of order and organization. The degree distribution is broad, with a tail that often follows a power law: therefore, many vertices with low degree coexist with some vertices with large degree. Furthermore, the distribution of edges is not only globally, but also locally inhomogeneous, with high concentrations of edges within special groups of vertices, and low concentrations between these groups. This feature of real networks is called *community*

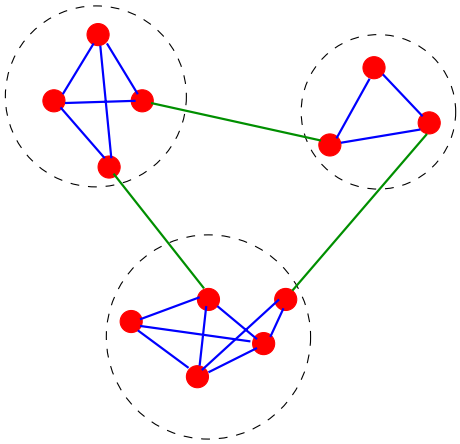

**Fig. 1.** A simple graph with three communities, enclosed by the dashed circles. Reprinted figure with permission from Ref. [16]. © 2009, by Springer.

*structure* [12], or *clustering*, and is the topic of this review (for earlier reviews see Refs. [13–17]). Communities, also called *clusters* or *modules*, are groups of vertices which probably share common properties and/or play similar roles within the graph. In Fig. 1 a schematic example of a graph with communities is shown.

Society offers a wide variety of possible group organizations: families, working and friendship circles, villages, towns, nations. The diffusion of Internet has also led to the creation of virtual groups, that live on the Web, like online communities. Indeed, social communities have been studied for a long time [18–21]. Communities also occur in many networked systems from biology, computer science, engineering, economics, politics, etc. In protein–protein interaction networks, communities are likely to group proteins having the same specific function within the cell [22–24], in the graph of the World Wide Web they may correspond to groups of pages dealing with the same or related topics [25,26], in metabolic networks they may be related to functional modules such as cycles and pathways [27,28], in food webs they may identify compartments [29,30], and so on.

Communities can have concrete applications. Clustering Web clients who have similar interests and are geographically near to each other may improve the performance of services provided on the World Wide Web, in that each cluster of clients could be served by a dedicated mirror server [31]. Identifying clusters of customers with similar interests in the network of purchase relationships between customers and products of online retailers (like, e.g., www.amazon.com) enables one to set up efficient recommendation systems [32], that better guide customers through the list of items of the retailer and enhance the business opportunities. Clusters of large graphs can be used to create data structures in order to efficiently store the graph data and to handle navigational queries, like path searches [33,34]. *Ad hoc networks* [35], i.e. self-configuring networks formed by communication nodes acting in the same region and rapidly changing (because the devices move, for instance), usually have no centrally maintained routing tables that specify how nodes have to communicate to other nodes. Grouping the nodes into clusters enables one to generate compact routing tables while the choice of the communication paths is still efficient [36].

Community detection is important for other reasons, too. Identifying modules and their boundaries allows for a classification of vertices, according to their structural position in the modules. So, vertices with a central position in their clusters, i.e. sharing a large number of edges with the other group partners, may have an important function of control and stability within the group; vertices lying at the boundaries between modules play an important role of mediation and lead the relationships and exchanges between different communities (alike to Csermely's "creative elements" [37]). Such a classification seems to be meaningful in social [38–40] and metabolic networks [27]. Finally, one can study the graph where vertices are the communities and edges are set between clusters if there are connections between some of their vertices in the original graph and/or if the modules overlap. In this way one attains a coarse-grained description of the original graph, which unveils the relationships between modules.[1] Recent studies indicate that networks of communities have a different degree distribution with respect to the full graphs [28]; however, the origin of their structures can be explained by the same mechanism [43].

Another important aspect related to community structure is the hierarchical organization displayed by most networked systems in the real world. Real networks are usually composed by communities including smaller communities, which in turn include smaller communities, etc. The human body offers a paradigmatic example of hierarchical organization: it is composed by organs, organs are composed by tissues, tissues by cells, etc. Another example is represented by business firms,

[1] Coarse-graining a graph generally means mapping it onto a smaller graph having similar properties, which is easier to handle. For this purpose, the vertices of the original graph are not necessarily grouped in communities. Gfeller and De Los Rios have proposed coarse-graining schemes that keep the properties of dynamic processes acting on the graph, like random walks [41] and synchronization [42].

which are characterized by a pyramidal organization, going from the workers to the president, with intermediate levels corresponding to work groups, departments and management. Herbert A. Simon has emphasized the crucial role played by hierarchy in the structure and evolution of complex systems [44]. The generation and evolution of a system organized in interrelated stable subsystems are much quicker than if the system were unstructured, because it is much easier to assemble the smallest subparts first and use them as building blocks to get larger structures, until the whole system is assembled. In this way it is also far more difficult that errors (mutations) occur along the process.

The aim of community detection in graphs is to identify the modules and, possibly, their hierarchical organization, by only using the information encoded in the graph topology. The problem has a long tradition and it has appeared in various forms in several disciplines. The first analysis of community structure was carried out by Weiss and Jacobson [45], who searched for work groups within a government agency. The authors studied the matrix of working relationships between members of the agency, which were identified by means of private interviews. Work groups were separated by removing the members working with people of different groups, which act as connectors between them. This idea of cutting the bridges between groups is at the basis of several modern algorithms of community detection (Section 5). Research on communities actually started even earlier than the paper by Weiss and Jacobson. Already in 1927, Stuart Rice looked for clusters of people in small political bodies, based on the similarity of their voting patterns [46]. Two decades later, George Homans showed that social groups could be revealed by suitably rearranging the rows and the columns of matrices describing social ties, until they take an approximate block-diagonal form [47]. This procedure is now standard. Meanwhile, traditional techniques to find communities in social networks are hierarchical clustering and partitional clustering (Sections 4.2 and 4.3), where vertices are joined into groups according to their mutual similarity.

Identifying graph communities is a popular topic in computer science, too. In parallel computing, for instance, it is crucial to know what is the best way to allocate tasks to processors so as to minimize the communications between them and enable a rapid performance of the calculation. This can be accomplished by splitting the computer cluster into groups with roughly the same number of processors, such that the number of physical connections between processors of different groups is minimal. The mathematical formalization of this problem is called *graph partitioning* (Section 4.1). The first algorithms for graph partitioning were proposed in the early 1970's.

In a seminal paper appeared in 2002, Girvan and Newman proposed a new algorithm, aiming at the identification of edges lying between communities and their successive removal, a procedure that after some iterations leads to the isolation of the communities [12]. The intercommunity edges are detected according to the values of a centrality measure, the edge betweenness, that expresses the importance of the role of the edges in processes where signals are transmitted across the graph following paths of minimal length. The paper triggered a big activity in the field, and many new methods have been proposed in the last years. In particular, physicists entered the game, bringing in their tools and techniques: spin models, optimization, percolation, random walks, synchronization, etc., became ingredients of new original algorithms. The field has also taken advantage of concepts and methods from computer science, nonlinear dynamics, sociology, discrete mathematics.

In this manuscript we try to cover in some detail the work done in this area. We shall pay special attention to the contributions made by physicists, but we shall also give proper credit to important results obtained by scholars of other disciplines. Section 2 introduces communities in real networks, and is supposed to make the reader acquainted with the problem and its relevance. In Section 3 we define the basic elements of community detection, i.e. the concepts of community and partition. Traditional clustering methods in computer and social sciences, i.e. graph partitioning, hierarchical, partitional and spectral clustering are reviewed in Section 4. Modern methods, divided into categories based on the type of approach, are presented in Sections 5 to 10. Algorithms to find overlapping communities, multiresolution and hierarchical techniques, are separately described in Sections 11 and 12, respectively, whereas Section 13 is devoted to the detection of communities evolving in time. We stress that our categorization of the algorithms is not sharp, because many algorithms may enter more categories: we tried to classify them based on what we believe is their main feature/purpose, even if other aspects may be present. Sections 14 and 15 are devoted to the issues of defining when community structure is significant, and deciding about the quality of algorithms' performances. In Sections 16 and 17 we describe general properties of clusters found in real networks, and specific applications of clustering algorithms. Section 18 contains the summary of the review, along with a discussion about future research directions in this area. The review makes use of several concepts of graph theory, that are defined and explained in the Appendix. Readers not acquainted with these concepts are urged to read the Appendix first.

## 2. Communities in real-world networks

In this section we shall present some striking examples of real networks with community structure. In this way we shall see what communities look like and why they are important.

Social networks are paradigmatic examples of graphs with communities. The word community itself refers to a social context. People naturally tend to form groups, within their work environment, family, friends.

In Fig. 2 we show some examples of social networks. The first example (Fig. 2a) is Zachary's network of karate club members [50], a well-known graph regularly used as a benchmark to test community detection algorithms (Section 15.1). It consists of 34 vertices, the members of a karate club in the United States, who were observed during a period of three years. Edges connect individuals who were observed to interact outside the activities of the club. At some point, a conflict between the club president and the instructor led to the fission of the club into two separate groups, supporting the instructor and

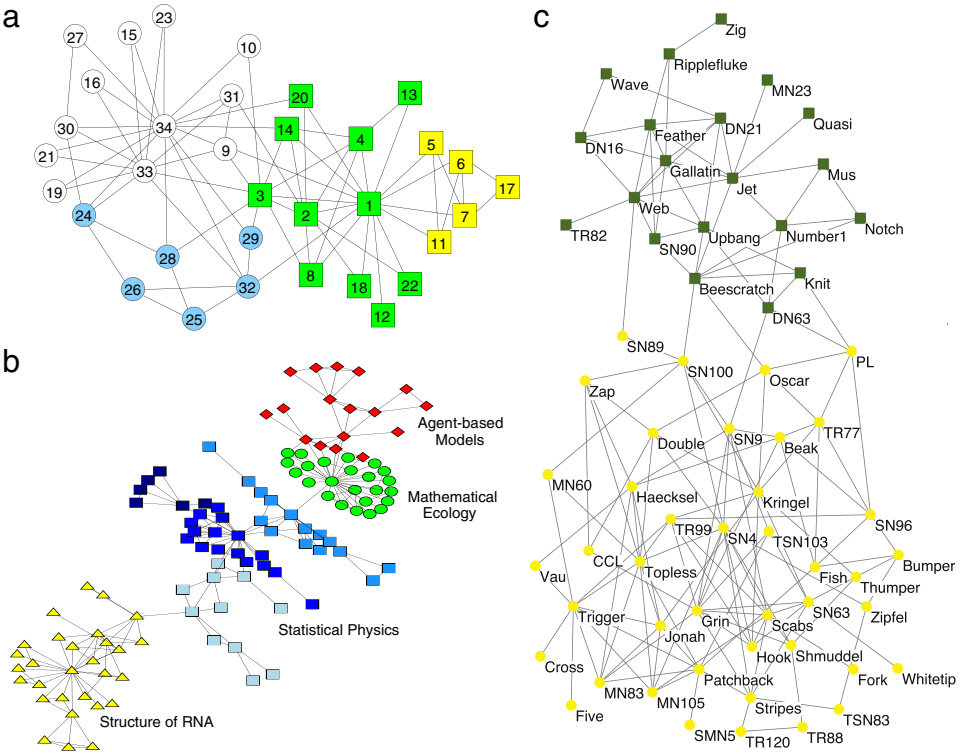


**Fig. 2.** Community structure in social networks. (a) Zachary's karate club, a standard benchmark in community detection. The colors correspond to the best partition found by optimizing the modularity of Newman and Girvan (Section 6.1). Reprinted figure with permission from Ref. [48].

(b) Collaboration network between scientists working at the Santa Fe Institute. The colors indicate high level communities obtained by the algorithm of Girvan and Newman (Section 5.1) and correspond quite closely to research divisions of the institute. Further subdivisions correspond to smaller research groups, revolving around project leaders. Reprinted figure with permission from Ref. [12].

(c) Lusseau's network of bottlenose dolphins. The colors label the communities identified through the optimization of a modified version of the modularity of Newman and Girvan, proposed by Arenas et al. [49] (Section 12.1). The partition matches the biological classification of the dolphins proposed by Lusseau. Reprinted figure with permission from Ref. [49].


the president, respectively (indicated by squares and circles). The question is whether from the original network structure it is possible to infer the composition of the two groups. Indeed, by looking at Fig. 2a one can distinguish two aggregations, one around vertices 33 and 34 (34 is the president), the other around vertex 1 (the instructor). One can also identify several vertices lying between the two main structures, like 3, 9, 10; such vertices are often misclassified by community detection methods.

Fig. 2b displays the largest connected component of a network of collaborations of scientists working at the Santa Fe Institute (SFI). There are 118 vertices, representing resident scientists at SFI and their collaborators. Edges are placed between scientists that have published at least one paper together. The visualization layout allows to distinguish disciplinary groups. In this network one observes many cliques, as authors of the same paper are all linked to each other. There are but a few connections between most groups.

In Fig. 2c we show the network of bottlenose dolphins living in Doubtful Sound (New Zealand) analyzed by Lusseau [51]. There are 62 dolphins and edges were set between animals that were seen together more often than expected by chance. The dolphins separated in two groups after a dolphin left the place for some time (squares and circles in the figure). Such groups are quite cohesive, with several internal cliques, and easily identifiable: only six edges join vertices of different groups. Due to this natural classification Lusseau's dolphins' network, like Zachary's karate club, is often used to test algorithms for community detection (Section 15.1).

Protein–protein interaction (PPI) networks are subject of intense investigations in biology and bioinformatics, as the interactions between proteins are fundamental for each process in the cell [52]. Fig. 3 illustrates a PPI network of the rat proteome [53]. Each interaction is derived by homology from experimentally observed interactions in other organisms. In our example, the proteins interact very frequently with each other, as they belong to metastatic cells, which have a high motility and invasiveness with respect to normal cells. Communities correspond to functional groups, i.e. to proteins having the same or similar functions, which are expected to be involved in the same processes. The modules are labeled by the overall function or the dominating protein class. Most communities are associated to cancer and metastasis, which indirectly shows how important detecting modules in PPI networks is.

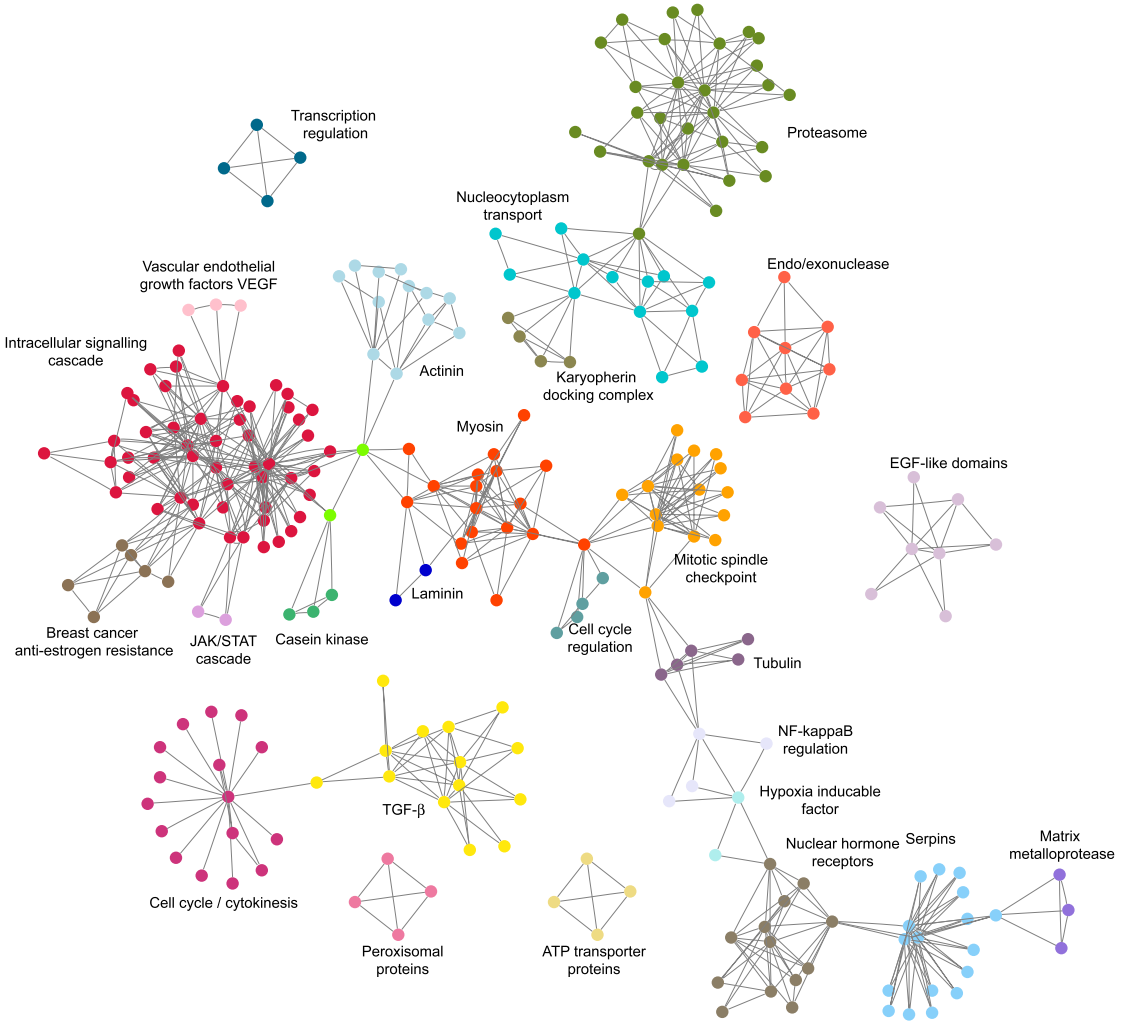


**Fig. 3.** Community structure in protein–protein interaction networks. The graph pictures the interactions between proteins in cancerous cells of a rat. Communities, labeled by colors, were detected with the Clique Percolation Method by Palla et al. (Section 11.1). Reprinted figure with permission from Ref. [53].


Relationships/interactions between elements of a system need not be reciprocal. In many cases they have a precise direction, that needs to be taken into account to understand the system as a whole. As an example we can cite predator-prey relationships in food webs. In Fig. 4 we see another example, taken from technology. The system is the World Wide Web, which can be seen as a graph by representing web pages as vertices and the hyperlinks that make users move from one page to another as edges [55]. Hyperlinks are directed: if one can move from page A to page B by clicking on a hyperlink of A, one usually does not find on B a hyperlink taking back to A. In fact, very few hyperlinks (less than 10%) are reciprocal. Communities of the web graph are groups of pages having topical similarities. Detecting communities in the web graph may help to identify the artificial clusters created by link farms in order to enhance the PageRank [56] value of web sites and grant them a higher Google ranking. In this way one could discourage this unfair practice. One usually assumes that the existence of a hyperlink between two pages implies that they are content-related, and that this similarity is independent of the hyperlink direction. Therefore it is customary to neglect the directedness of the hyperlinks and to consider the graph as undirected, for the purpose of community detection. On the other hand, taking properly into account the directedness of the edges can considerably improve the quality of the partition(s), as one can handle a lot of precious information about the system. Moreover, in some instances neglecting edge directedness may lead to strange results [57,58]. Developing methods of community detection for directed graphs is a hard task. For instance, a directed graph is characterized by asymmetrical matrices (adjacency matrix, Laplacian, etc.), so spectral analysis is much more complex. Only a few techniques can be easily extended from the undirected to the directed case. Otherwise, the problem must be formulated from scratch.

Edge directedness is not the only complication to deal with when facing the problem of graph clustering. In many real networks vertices may belong to more than one group. In this case one speaks of *overlapping communities* and uses the

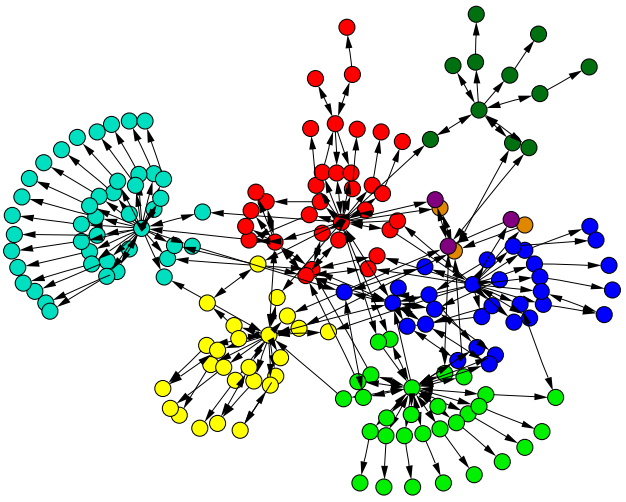

**Fig. 4.** Community structure in technological networks. Sample of the web graph consisting of the pages of a web site and their mutual hyperlinks, which are directed. Communities, indicated by the colors, were detected with the algorithm of Girvan and Newman (Section 5.1), by neglecting the directedness of the edges. Reprinted figure with permission from Ref. [54].


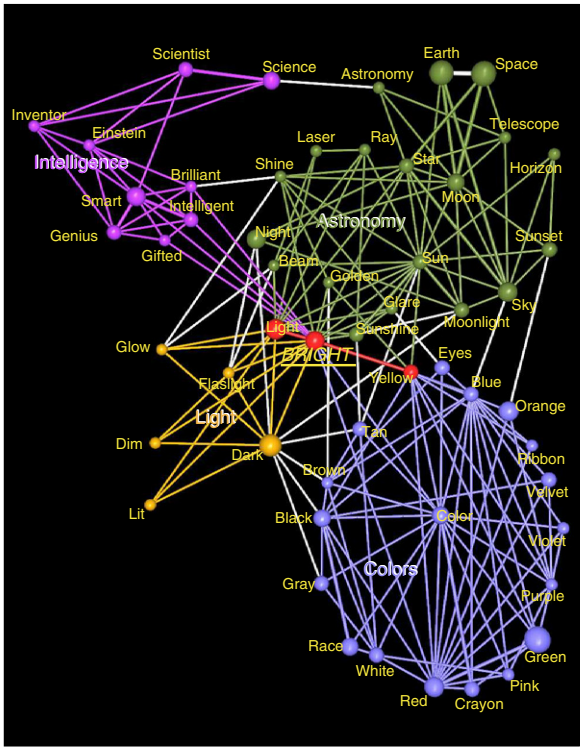


**Fig. 5.** Overlapping communities in a network of word association. The groups, labeled by the colors, were detected with the Clique Percolation Method by Palla et al. (Section 11.1). Reprinted figure with permission from Ref. [28].


term *cover*, rather than partition, whose standard definition forbids multiple memberships of vertices. Classical examples are social networks, where an individual usually belongs to different circles at the same time, from that of work colleagues to family, sport associations, etc. Traditional algorithms of community detection assign each vertex to a single module. In so doing, they neglect potentially relevant information. Vertices belonging to more communities are likely to play an important role of intermediation between different compartments of the graph. In Fig. 5 we show a network of word association derived starting from the word "bright". The network builds on the University of South Florida Free Association Norms [59]. An edge

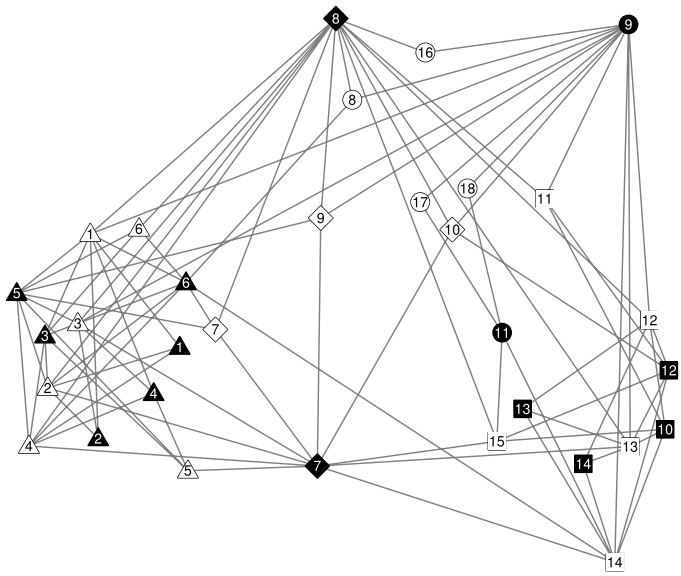

**Fig. 6.** Community structure in multipartite networks. This bipartite graph refers to the Southern Women Event Participation data set. Women are represented as open symbols with black labels, events as filled symbols with white labels. The illustrated vertex partition has been obtained by maximizing a modified version of the modularity by Newman and Girvan, tailored on bipartite graphs [60] (Section 6.2). Reprinted figure with permission from Ref. [60]. © 2007, by the American Physical Society.

between words *A* and *B* indicates that some people associate *B* to the word *A*. The graph clearly displays four communities, corresponding to the categories *Intelligence*, *Astronomy*, *Light* and *Colors*. The word "bright" is related to all of them by construction. Other words belong to more categories, e.g. "dark" (Colors and Light). Accounting for overlapping communities introduces a further variable, the membership of vertices in different communities, which enormously increases the number of possible covers with respect to standard partitions. Therefore, searching for overlapping communities is usually much more computationally demanding than detecting standard partitions.

So far we have discussed examples of unipartite graphs. However, it is not uncommon to find real networks with different classes of vertices, and edges joining only vertices of different classes. An example is a network of scientists and papers, where edges join scientists and the papers they have authored. Here there is no edge between any pair of scientists or papers, so the graph is bipartite. For a multipartite network the concept of community does not change much with respect to the case of unipartite graphs, as it remains related to a large density of edges between members of the same group, with the only difference that the elements of each group belong to different vertex classes. Multipartite graphs are usually reduced to unipartite projections of each vertex class. For instance, from the bipartite network of scientists and papers one can extract a network of scientists only, who are related by co-authorship. In this way one can adopt standard techniques of network analysis, in particular standard clustering methods, but a lot of information gets lost. Detecting communities in multipartite networks can have interesting applications in, e.g., marketing. Large shopping networks, in which customers are linked to the products they have bought, allow one to classify customers based on the types of product they purchase more often: this could be used both to organize targeted advertising, as well as to give recommendations about future purchases [61]. The problem of community detection in multipartite networks is not trivial, and usually requires *ad hoc* methodologies. Fig. 6 illustrates the famous bipartite network of Southern Women studied by Davis et al. [62]. There are 32 vertices, representing 18 women from the area of Natchez, Mississippi, and 14 social events. Edges represent the participation of the women in the events. From the figure one can see that the network has a clear community structure.

In some of the previous examples, edges have (or can have) weights. For instance, the edges of the collaboration network of Fig. 2b could be weighted by the number of papers coauthored by pairs of scientists. Similarly, the edges of the word association network of Fig. 5 are weighted by the number of times pairs of words have been associated by people. Weights are precious additional information on a graph, and should be considered in the analysis. In many cases methods working on unweighted graphs can be simply extended to the weighted case.

## 3. Elements of community detection

The problem of graph clustering, intuitive at first sight, is actually not well defined. The main elements of the problem themselves, i.e. the concepts of community and partition, are not rigorously defined, and require some degree of arbitrariness and/or common sense. Indeed, some ambiguities are hidden and there are often many equally legitimate ways of resolving them. Therefore, it is not surprising that there are plenty of recipes in the literature and that people do not even try to ground the problem on shared definitions. It is important to stress that the identification of structural clusters is possible

only if graphs are *sparse*, i.e. if the number of edges $m$ is of the order of the number of nodes $n$ of the graph. If $m \gg n$, the distribution of edges among the nodes is too homogeneous for communities to make sense.[2] In this case the problem turns into something rather different, close to data clustering [63], which requires concepts and methods of a different nature. The main difference is that, while communities in graphs are related, explicitly or implicitly, to the concept of edge density (inside versus outside the community), in data clustering communities are sets of points which are "close" to each other, with respect to a measure of distance or similarity, defined for each pair of points. Some classical techniques for data clustering, like *hierarchical*, *partitional* and *spectral clustering* will be discussed later in the review (Sections 4.2–4.4), as they are sometimes adopted for graph clustering too. Other standard procedures for data clustering include *neural network clustering* techniques like, e. g., *self-organizing maps* and *multi-dimensional scaling* techniques like, e.g., *singular value decomposition* and *principal component analysis* [63].

In this section we shall attempt an ordered exposition of the fundamental concepts of community detection. After a brief discussion of the issue of computational complexity for the algorithms, we shall review the notions of community and partition.

### 3.1. Computational complexity

The massive amount of data on real networks currently available makes the issue of the efficiency of clustering algorithms essential. The *computational complexity* of an algorithm is the estimate of the amount of resources required by the algorithm to perform a task. This involves both the number of computation steps needed and the number of memory units that need to be simultaneously allocated to run the computation. Such demands are usually expressed by their scalability with the size of the system at study. In the case of a graph, the size is typically indicated by the number of vertices $n$ and/or the number of edges $m$. The computational complexity of an algorithm cannot always be calculated. In fact, sometimes this is a very hard task, or even impossible. In these cases, it is however important to have at least an estimate of the *worst-case* complexity of the algorithm, which is the amount of computational resources needed to run the algorithm in the most unfavorable case for a given system size.

The notation $O(n^\alpha m^\beta)$ indicates that the computer time grows as a power of both the number of vertices and edges, with exponents $\alpha$ and $\beta$, respectively. Ideally, one would like to have the lowest possible values for the exponents, which would correspond to the lowest possible computational demands. Samples of the Web graph, with millions of vertices and billions of edges, cannot be tackled by algorithms whose running time grows faster than $O(n)$ or $O(m)$.

Algorithms with polynomial complexity form the class **P**. For some important decision and optimization problems, there are no known polynomial algorithms. Finding solutions of such problems in the worst-case scenario may demand an exhaustive search, which takes a time growing faster than any polynomial function of the system size, e.g. exponentially. Problems whose solutions can be verified in a polynomial time span the class **NP** of *non-deterministic polynomial time* problems, which includes **P**. A problem is **NP**-hard if a solution for it can be translated into a solution for any **NP**-problem. However, a **NP**-hard problem needs not be in the class **NP**. If it does belong to **NP** it is called **NP**-complete. The class of **NP**-complete problems has drawn a special attention in computer science, as it includes many famous problems like the Traveling Salesman, Boolean Satisfiability (SAT), Linear Programming, etc. [64,65]. The fact that **NP** problems have a solution which is verifiable in polynomial time does not mean that **NP** problems have polynomial complexity, i.e., that they are in **P**. In fact, the question of whether $\mathbf{NP} = \mathbf{P}$ is the most important open problem in theoretical computer science. **NP**-hard problems need not be in **NP** (in which case they would be **NP**-complete), but they are at least as hard as **NP**-complete problems, so they are unlikely to have polynomial complexity, although a proof of that is still missing.

Many clustering algorithms or problems related to clustering are **NP**-hard. In this case, it is pointless to use exact algorithms, which could be applied only to very small systems. Moreover, even if an algorithm has a polynomial complexity, it may still be too slow to tackle large systems of interest. In all such cases it is common to use *approximation algorithms*, i.e. methods that do not deliver an exact solution to the problem at hand, but only an approximate solution, with the advantage of a lower complexity. Approximation algorithms are often non-deterministic, as they deliver different solutions for the same problem, for different initial conditions and/or parameters of the algorithm. The goal of such algorithms is to deliver a solution which differs by a constant factor from the optimal solution. In any case, one should give provable bounds on the goodness of the approximate solution delivered by the algorithm with respect to the optimal solution. In many cases it is not possible to approximate the solution within any constant, as the goodness of the approximation strongly depends on the specific problem at study. Approximation algorithms are commonly used for optimization problems, in which one wants to find the maximum or minimum value of a given cost function over a large set of possible system configurations.

### 3.2. Communities

#### 3.2.1. Basics

The first problem in graph clustering is to look for a quantitative definition of community. No definition is universally accepted. As a matter of fact, the definition often depends on the specific system at hand and/or application one has in mind.

[2] This is not necessarily true if graphs are weighted with a heterogeneous distribution of weights. In such cases communities may still be identified as subgraphs with a high internal density of weight.

From intuition and the examples of Section 2 we get the notion that there must be more edges "inside" the community than edges linking vertices of the community with the rest of the graph. This is the reference guideline at the basis of most community definitions. But many alternative recipes are compatible with it. Moreover, in most cases, communities are algorithmically defined, i.e. they are just the final product of the algorithm, without a precise *a priori* definition.

Let us start with a subgraph $\mathcal{C}$ of a graph $\mathcal{G}$, with $|\mathcal{C}| = n_c$ and $|\mathcal{G}| = n$ vertices, respectively. We define the *internal* and *external* degree of vertex $v \in \mathcal{C}$, $k_v^{int}$ and $k_v^{ext}$, as the number of edges connecting $v$ to other vertices of $\mathcal{C}$ or to the rest of the graph, respectively. If $k_v^{ext} = 0$, the vertex has neighbors only within $\mathcal{C}$, which is likely to be a good cluster for $v$; if $k_v^{int} = 0$, instead, the vertex is disjoint from $\mathcal{C}$ and it should be better assigned to a different cluster. The *internal degree* $k_{int}^{\mathcal{C}}$ of $\mathcal{C}$ is the sum of the internal degrees of its vertices. Likewise, the *external degree* $k_{ext}^{\mathcal{C}}$ of $\mathcal{C}$ is the sum of the external degrees of its vertices. The *total degree* $k^{\mathcal{C}}$ is the sum of the degrees of the vertices of $\mathcal{C}$. By definition, $k^{\mathcal{C}} = k_{int}^{\mathcal{C}} + k_{ext}^{\mathcal{C}}$.

We define the *intra-cluster density* $\delta_{int}(\mathcal{C})$ of the subgraph $\mathcal{C}$ as the ratio between the number of internal edges of $\mathcal{C}$ and the number of all possible internal edges, i.e.

$$\delta_{int}(\mathcal{C}) = \frac{\#\text{ internal edges of } \mathcal{C}}{n_c(n_c - 1)/2}. \tag{1}$$

Similarly, the *inter-cluster density* $\delta_{ext}(\mathcal{C})$ is the ratio between the number of edges running from the vertices of $\mathcal{C}$ to the rest of the graph and the maximum number of inter-cluster edges possible, i.e.

$$\delta_{ext}(\mathcal{C}) = \frac{\#\text{ inter-cluster edges of } \mathcal{C}}{n_c(n - n_c)}. \tag{2}$$

For $\mathcal{C}$ to be a community, we expect $\delta_{int}(\mathcal{C})$ to be appreciably larger than the average link density $\delta(\mathcal{G})$ of $\mathcal{G}$, which is given by the ratio between the number of edges of $\mathcal{G}$ and the maximum number of possible edges $n(n-1)/2$. On the other hand, $\delta_{ext}(\mathcal{C})$ has to be much smaller than $\delta(\mathcal{G})$. Searching for the best tradeoff between a large $\delta_{int}(\mathcal{C})$ and a small $\delta_{ext}(\mathcal{C})$ is implicitly or explicitly the goal of most clustering algorithms. A simple way to do that is, e.g., maximizing the sum of the differences $\delta_{int}(\mathcal{C}) - \delta_{ext}(\mathcal{C})$ over all clusters of the partition[3] [66].

A required property of a community is *connectedness*. We expect that for $\mathcal{C}$ to be a community there must be a path between each pair of its vertices, running only through vertices of $\mathcal{C}$. This feature simplifies the task of community detection on disconnected graphs, as in this case one just analyzes each connected component separately, unless special constraints are imposed on the resulting clusters.

With these basic requirements in mind, we can now introduce the main definitions of community. Social network analysts have devised many definitions of subgroups with various degrees of internal cohesion among vertices [3,4,21]. Many other definitions have been introduced by computer scientists and physicists. We distinguish three classes of definitions: local, global and based on vertex similarity. Other definitions will be given in the context of the algorithms for which they were introduced.

### 3.2.2. Local definitions

Communities are parts of the graph with a few ties with the rest of the system. To some extent, they can be considered as separate entities with their own autonomy. So, it makes sense to evaluate them independently of the graph as a whole. Local definitions focus on the subgraph under study, including possibly its immediate neighborhood, but neglecting the rest of the graph. We start with a listing of the main definitions adopted in social network analysis, for which we shall closely follow the exposition of Ref. [3]. There, four types of criterion were identified: *complete mutuality*, *reachability*, *vertex degree* and the *comparison of internal versus external cohesion*. The corresponding communities are mostly *maximal subgraphs*, which cannot be enlarged with the addition of new vertices and edges without losing the property which defines them.

Social communities can be defined in a very strict sense as subgroups whose members are all "friends" to each other [67] (complete mutuality). In graph terms, this corresponds to a *clique*, i.e. a subset whose vertices are all adjacent to each other. In social network analysis, a clique is a maximal subgraph, whereas in graph theory it is common to also call cliques non-maximal subgraphs. Triangles are the simplest cliques, and are frequent in real networks. But larger cliques are less frequent. Moreover, the condition is really too strict: a subgraph with all possible internal edges except one would be an extremely cohesive subgroup, but it would not be considered a community under this recipe. Another problem is that all vertices of a clique are absolutely symmetric, with no differentiation between them. In many practical examples, instead, we expect that within a community there is a whole hierarchy of roles for the vertices, with core vertices coexisting with peripheral ones. We remark that vertices may belong to more cliques simultaneously, a property which is at the basis of the Clique Percolation Method of Palla et al. [28] (see Section 11.1). From a practical point of view, finding cliques in a graph is an **NP**-complete problem [68]. The Bron–Kerbosch method [69] runs in a time growing exponentially with the size of the graph.

It is however possible to relax the notion of clique, defining subgroups which are still clique-like objects. A possibility is to use properties related to reachability, i.e. to the existence (and length) of paths between vertices. An *n-clique* is a maximal

[3] In Ref. [66] one actually computes the inter-cluster density by summing the densities for each pair of clusters. Therefore the function to minimize is not exactly $\sum_{\mathcal{C}}[\delta_{int}(\mathcal{C}) - \delta_{ext}(\mathcal{C})]$, but essentially equivalent.

subgraph such that the distance of each pair of its vertices is not larger than $n$ [70,71]. For $n = 1$ one recovers the definition of clique, as all vertices are adjacent, so each geodesic path between any pair of vertices has length 1. This definition, more flexible than that of clique, still has some limitations, deriving from the fact that the geodesic paths need not run on the vertices of the subgraph at study, but may run on vertices outside the subgraph. In this way, there may be two disturbing consequences. First, the diameter of the subgraph may exceed $n$, even if in principle each vertex of the subgraph is less than $n$ steps away from any of the others. Second, the subgraph may be disconnected, which is not consistent with the notion of cohesion one tries to enforce. To avoid these problems, Mokken [72] has suggested two possible alternatives, the *n-clan* and the *n-club*. An $n$-clan is an $n$-clique whose diameter is not larger than $n$, i.e. a subgraph such that the distance between any two of its vertices, computed over shortest paths within the subgraph, does not exceed $n$. An $n$-club, instead, is a maximal subgraph of diameter $n$. The two definitions are quite close: the difference is that an $n$-clan is maximal under the constraint of being an $n$-clique, whereas an $n$-club is maximal under the constraint imposed by the length of the diameter.

Another criterion for subgraph cohesion relies on the adjacency of its vertices. The idea is that a vertex must be adjacent to some minimum number of other vertices in the subgraph. In the literature on social network analysis there are two complementary ways of expressing this. A *k-plex* is a maximal subgraph in which each vertex is adjacent to all other vertices of the subgraph except at most $k$ of them [73]. Similarly, a *k-core* is a maximal subgraph in which each vertex is adjacent to at least $k$ other vertices of the subgraph [74]. So, the two definitions impose conditions on the minimal number of absent or present edges. The corresponding clusters are more cohesive than $n$-cliques, just because of the existence of many internal edges. In any graph there is a whole hierarchy of cores of different order, which can be identified by means of a recent efficient algorithm [75]. A $k$-core is essentially the same as a *p-quasi complete* subgraph, which is a subgraph such that the degree of each vertex is larger than $p(k-1)$, where $p$ is a real number in $[0, 1]$ and $k$ the order of the subgraph [76]. Determining whether a graph includes a $1/2$-quasi complete subgraph of order at least $k$ is **NP**-complete.

As cohesive as a subgraph can be, it would hardly be a community if there is a strong cohesion as well between the subgraph and the rest of the graph. Therefore, it is important to compare the internal and external cohesion of a subgraph. In fact, this is what is usually done in the most recent definitions of community. The first recipe, however, is not recent and stems from social network analysis. An *LS-set* [77], or *strong community* [78], is a subgraph such that the internal degree of each vertex is greater than its external degree. This condition is quite strict and can be relaxed into the so-called *weak* definition of community [78], for which it suffices that the internal degree of the subgraph exceeds its external degree. An *LS*-set is also a weak community, while the converse is not generally true. Hu et al. [79] have introduced alternative definitions of strong and weak communities: a community is strong if the internal degree of any vertex of the community exceeds the number of edges that the vertex shares with any other community; a community is weak if its total internal degree exceeds the number of edges shared by the community with the other communities. These definitions are in the same spirit of the *planted partition model* (Section 15). An *LS*-set is also a strong community in the sense of Hu et al.. Likewise, a weak community according to Radicchi et al. is also a weak community for Hu et al.. In both cases the converse is not true, however. Another definition focuses on the robustness of clusters to edge removal and uses the concept of *edge connectivity*. The edge connectivity of a pair of vertices in a graph is the minimal number of edges that need to be removed in order to disconnect the two vertices, i.e. such that there is no path between them. A *lambda set* is a subgraph such that any pair of vertices of the subgraph has a larger edge connectivity than any pair formed by one vertex of the subgraph and one outside the subgraph [80]. However, vertices of a lambda-set need not be adjacent and may be quite distant from each other.

Communities can also be identified by a *fitness measure*, expressing to which extent a subgraph satisfies a given property related to its cohesion. The larger the fitness, the more definite is the community. This is the same principle behind *quality functions*, which give an estimate of the goodness of a graph partition (see Section 3.3.2). The simplest fitness measure for a cluster is its intra-cluster density $\delta_{int}(\mathcal{C})$. One could assume that a subgraph $\mathcal{C}$ with $k$ vertices is a cluster if $\delta_{int}(\mathcal{C})$ is larger than a threshold, say $\xi$. Finding such subgraphs is an **NP**-complete problem, as it coincides with the **NP**-complete Clique Problem when the threshold $\xi = 1$ [64]. It is better to fix the size of the subgraph because, without this conditions, any clique would be one of the best possible communities, including trivial two-cliques (simple edges). Variants of this problem focus on the number of internal edges of the subgraph [81–83]. Another measure of interest is the *relative density* $\rho(\mathcal{C})$ of a subgraph $\mathcal{C}$, defined as the ratio between the internal and the total degree of $\mathcal{C}$. Finding subgraphs of a given size with $\rho(\mathcal{C})$ larger than a threshold is **NP**-complete [84]. Fitness measures can also be associated to the connectivity of the subgraph at study to the other vertices of the graph. A good community is expected to have a small cut size (see Appendix A.1), i.e. a small number of edges joining it to the rest of the graph. This sets a bridge between community detection and graph partitioning, which we shall discuss in Section 4.1.

### 3.2.3. Global definitions

Communities can also be defined with respect to the graph as a whole. This is reasonable in those cases in which clusters are essential parts of the graph, which cannot be taken apart without seriously affecting the functioning of the system. The literature offers many global criteria to identify communities. In most cases they are indirect definitions, in which some global property of the graph is used in an algorithm that delivers communities at the end. However, there is a class of proper definitions, based on the idea that a graph has community structure if it is different from a random graph. A random graph à la Erdös–Rényi (Appendix A.3), for instance, is not expected to have community structure, as any two vertices have the same probability to be adjacent, so there should be no preferential linking involving special groups of vertices. Therefore, one can define a *null model*, i.e. a graph which matches the original in some of its structural features, but which is otherwise a random

graph. The null model is used as a term of comparison, to verify whether the graph at study displays community structure or not. The most popular null model is that proposed by Newman and Girvan and consists of a randomized version of the original graph, where edges are rewired at random, under the constraint that the expected degree of each vertex matches the degree of the vertex in the original graph [54]. This null model is the basic concept behind the definition of *modularity*, a function which evaluates the goodness of partitions of a graph into clusters. Modularity will be discussed at length in this review, because it has the unique privilege of being at the same time a global criterion to define a community, a quality function and the key ingredient of the most popular method of graph clustering. In the standard formulation of modularity, a subgraph is a community if the number of edges inside the subgraph exceeds the expected number of internal edges that the same subgraph would have in the null model. This expected number is an average over all possible realizations of the null model. Several modifications of modularity have been proposed (Section 6.2). A general class of null models, including modularity as a special case, has been designed by Reichardt and Bornholdt [85] (Section 6.2).

*3.2.4. Definitions based on vertex similarity*

It is natural to assume that communities are groups of vertices similar to each other. One can compute the similarity between each pair of vertices with respect to some reference property, local or global, no matter whether they are connected by an edge or not. Each vertex ends up in the cluster whose vertices are most similar to it. Similarity measures are at the basis of traditional methods, like hierarchical, partitional and spectral clustering, to be discussed in Sections 4.2–4.4. Here we discuss some popular measures used in the literature.

If it were possible to embed the graph vertices in an $n$-dimensional Euclidean space, by assigning a position to them, one could use the *distance* between a pair of vertices as a measure of their similarity (it is actually a measure of dissimilarity because similar vertices are expected to be close to each other). Given the two data points $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$, one could use any norm $L_m$, like the *Euclidean distance* ($L_2$-norm),

$$d_{AB}^E = \sum_{k=1}^{n} \sqrt{(a_k - b_k)^2}, \tag{3}$$

the *Manhattan distance* ($L_1$-norm)

$$d_{AB}^M = \sum_{k=1}^{n} |a_k - b_k|, \tag{4}$$

and the $L_\infty$-norm

$$d_{AB}^\infty = \max_{k\in[1,n]} |a_k - b_k|. \tag{5}$$

Another popular spatial measure is the *cosine similarity*, defined as

$$\rho_{AB} = \arccos \frac{\mathbf{a}\cdot\mathbf{b}}{\sqrt{\sum_{k=1}^{n} a_k^2}\sqrt{\sum_{k=1}^{n} b_k^2}}, \tag{6}$$

where $\mathbf{a}\cdot\mathbf{b}$ is the dot product of the vectors $\mathbf{a}$ and $\mathbf{b}$. The variable $\rho_{AB}$ is defined in the range $[0, \pi)$.

If the graph cannot be embedded in space, the similarity must be necessarily inferred from the adjacency relationships between vertices. A possibility is to define a distance [39,3] between vertices like

$$d_{ij} = \sqrt{\sum_{k\neq i,j} (A_{ik} - A_{jk})^2}, \tag{7}$$

where $\mathbf{A}$ is the adjacency matrix. This is a dissimilarity measure, based on the concept of structural equivalence [86]. Two vertices are structurally equivalent if they have the same neighbors, even if they are not adjacent themselves. If $i$ and $j$ are structurally equivalent, $d_{ij} = 0$. Vertices with large degree and different neighbors are considered very "far" from each other. Alternatively, one could measure the *overlap* between the neighborhoods $\Gamma(i)$ and $\Gamma(j)$ of vertices $i$ and $j$, given by the ratio between the intersection and the union of the neighborhoods, i.e.

$$\omega_{ij} = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}. \tag{8}$$

Another measure related to structural equivalence is the Pearson correlation between columns or rows of the adjacency matrix,

$$C_{ij} = \frac{\sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{n\sigma_i\sigma_j}, \tag{9}$$

where the averages $\mu_i = (\sum_j A_{ij})/n$ and the variances $\sigma_i = \sqrt{\sum_j (A_{ij} - \mu_i)^2/n}$.

An alternative measure is the number of edge- (or vertex-) independent paths between two vertices. Independent paths do not share any edge (vertex), and their number is related to the maximum flow that can be conveyed between the two vertices under the constraint that each edge can carry only one unit of flow (max-flow/min-cut theorem [87]). The maximum flow can be calculated in a time $O(m)$, for a graph with $m$ edges, using techniques like the augmenting path algorithm [88]. Similarly, one could consider all paths running between two vertices. In this case, there is the problem that the total number of paths is infinite, but this can be avoided if one performs a weighted sum of the number of paths. For instance, paths of length $l$ can be weighted by the factor $\alpha^l$, with $\alpha < 1$. Another possibility, suggested by Estrada and Hatano [89,90], is to weigh paths of length $l$ with the inverse factorial $1/l!$. In both cases, the contribution of long paths is strongly suppressed and the sum converges.

Another important class of measures of vertex similarity is based on properties of random walks on graphs. One of this properties is the *commute-time* between a pair of vertices, which is the average number of steps needed for a random walker, starting at either vertex, to reach the other vertex for the first time and to come back to the starting vertex. Saerens and coworkers [91–94] have extensively studied and used the commute-time (and variants thereof) as (dis)similarity measure: the larger the time, the farther (less similar) the vertices. The commute-time is closely related [95] to the *resistance distance* introduced by Klein and Randic [96], expressing the effective electrical resistance between two vertices if the graph is turned into a resistor network. White and Smyth [97] and Zhou [98] used instead the average first passage time, i.e. the average number of steps needed to reach for the first time the target vertex from the source. Harel and Koren [99] proposed to build measures out of quantities like the probability to visit a target vertex in no more than a given number of steps after it leaves a source vertex[4] and the probability that a random walker starting at a source visits the target exactly once before hitting the source again. Another quantity used to define similarity measures is the *escape probability*, defined as the probability that the walker reaches the target vertex before coming back to the source vertex [102,103]. The escape probability is related to the effective conductance between the two vertices in the equivalent resistor network. Other authors have exploited properties of modified random walks. For instance, the algorithm by Gori and Pucci [104] and that by Tong et al. [103] used similarity measures derived from Google's PageRank process [56].

### 3.3. Partitions

#### 3.3.1. Basics

A *partition* is a division of a graph in clusters, such that each vertex belongs to one cluster. As we have seen in Section 2, in real systems vertices may be shared among different communities. A division of a graph into overlapping (or *fuzzy*) communities is called *cover*.

The number of possible partitions in $k$ clusters of a graph with $n$ vertices is the *Stirling number of the second kind* $S(n, k)$ [105]. The total number of possible partitions is the $n$-th *Bell number* $B_n = \sum_{k=0}^{n} S(n, k)$ [105]. In the limit of large $n$, $B_n$ has the asymptotic form [106]

$$B_n \sim \frac{1}{\sqrt{n}}[\lambda(n)]^{n+1/2} e^{\lambda(n)-n-1}, \tag{10}$$

where $\lambda(n) = e^{W(n)} = n/W(n)$, $W(n)$ being the *Lambert W function* [107]. Therefore, $B_n$ grows faster than exponentially with the graph size $n$, which means that an enumeration and/or evaluation of all partitions of a graph is impossible, unless the graph consists of very few vertices.

Partitions can be *hierarchically ordered*, when the graph has different levels of organization/structure at different scales. In this case, clusters display in turn community structure, with smaller communities inside, which may again contain smaller communities, and so on (Fig. 7). As an example, in a social network of children living in the same town, one could group the children according to the schools they attend, but within each school one can make a subdivision into classes. Hierarchical organization is a common feature of many real networks, where it is revealed by a peculiar scaling of the clustering coefficient for vertices having the same degree $k$, when plotted as a function of $k$ [108,109].

A natural way to represent the hierarchical structure of a graph is to draw a *dendrogram*, like the one illustrated in Fig. 8. Here, partitions of a graph with twelve vertices are shown. At the bottom, each vertex is its own module (the "leaves" of the tree). By moving upwards, groups of vertices are successively aggregated. Mergers of communities are represented by horizontal lines. The uppermost level represents the whole graph as a single community. Cutting the diagram horizontally at some height, as shown in the figure (dashed line), displays one partition of the graph. The diagram is hierarchical by construction: each community belonging to a level is fully included in a community at a higher level. Dendrograms are regularly used in sociology and biology. The technique of hierarchical clustering, described in Section 4.2, lends itself naturally to this kind of representation.

[4] In the clustering method by Latapy and Pons [100] (discussed in Section 8.2) and in a recent analysis by Nadler et al. [101], one defined a dissimilarity measure called "diffusion distance", which is derived from the probability that the walker visits the target after a fixed number of steps.

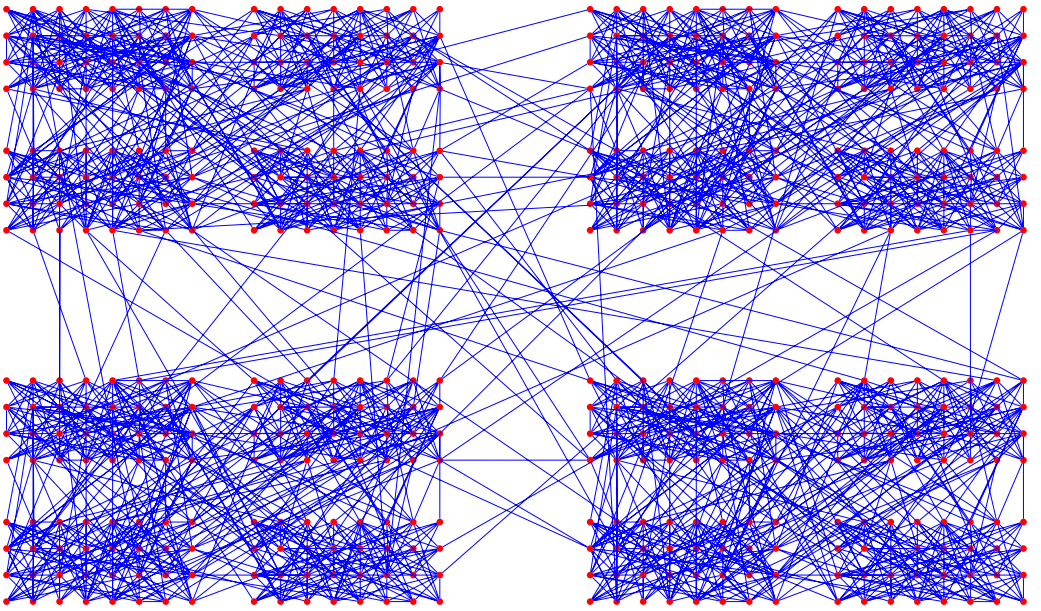

**Fig. 7.** Schematic example of a hierarchical graph. Sixteen modules with 32 vertices each clearly form four larger clusters. All vertices have degree 64. Reprinted figure with permission from Ref. [110].


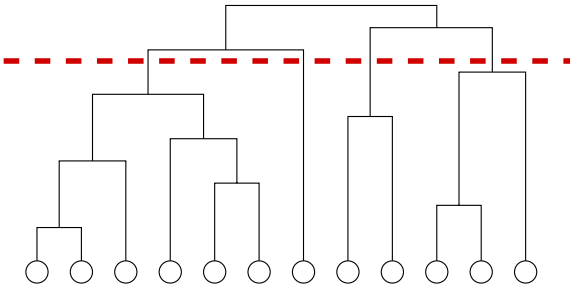

**Fig. 8.** A dendrogram, or hierarchical tree. Horizontal cuts correspond to partitions of the graph in communities. Reprinted figure with permission from Ref. [54].


### *3.3.2. Quality functions: Modularity*

Reliable algorithms are supposed to identify *good* partitions. But what is a good clustering? In order to distinguish between "good" and "bad" clusterings, it would be useful to require that partitions satisfy a set of basic properties, intuitive and easy to agree upon. In the wider context of data clustering, this issue has been studied by Jon Kleinberg [111], who has proved an important *impossibility theorem*. Given a set $S$ of points, a *distance function* $d$ is defined, which is positive definite and symmetric (the triangular inequality is not explicitly required). One wishes to find a clustering $f$ based on the distances between the points. Kleinberg showed that no clustering satisfies at the same time the three following properties:

1. *Scale-invariance*: given a constant $\alpha$, multiplying *any* distance function $d$ by $\alpha$ yields the same clustering.
2. *Richness*: any possible partition of the given point set can be recovered if one chooses a suitable distance function $d$.
3. *Consistency*: given a partition, any modification of the distance function that does not decrease the distance between points of different clusters and that does not increase the distance between points of the same cluster, yields the same clustering.

The theorem cannot be extended to graph clustering because the distance function cannot be in general defined for a graph which is not complete. For weighted complete graphs, like correlation matrices [112], it is often possible to define a distance function. On a generic graph, except for the first property, which does not make sense without a distance function,[5] the other two are quite well defined. The property of richness implies that, given a partition, one can set edges between the vertices in such a way that the partition is a natural outcome of the resulting graph (e.g., it could be achieved by setting edges only between vertices of the same cluster). Consistency here implies that deleting inter-cluster edges and adding intra-cluster edges yields the same partition.

[5] The traditional shortest-path distance between vertices is not suitable here, as it is integer by definition.

Many algorithms are able to identify a subset of meaningful partitions, ideally one or just a few, whereas some others, like techniques based on hierarchical clustering (Section 4.2), deliver a large number of partitions. That does not mean that the partitions found are equally good. Therefore it is helpful (sometimes even necessary) to have a quantitative *criterion* to assess the goodness of a graph partition. A *quality function* is a function that assigns a number to each partition of a graph. In this way one can rank partitions based on their score given by the quality function. Partitions with high scores are "good", so the one with the largest score is by definition the best. Nevertheless, one should keep in mind that the question of when a partition is better than another one is ill-posed, and the answer depends on the specific concept of community and/or quality function adopted.

A quality function $Q$ is *additive* if there is an elementary function $q$ such that, for any partition $\mathcal{P}$ of a graph

$$Q(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} q(\mathcal{C}), \tag{11}$$

where $\mathcal{C}$ is a generic cluster of partition $\mathcal{P}$. Eq. (11) states that the quality of a partition is given by the sum of the qualities of the individual clusters. The function $q(\mathcal{C})$ could be any of the cluster fitness functions discussed in Section 3.2.2, for instance. Most quality functions used in the literature are additive, although it is not a necessary requirement.

An example of quality function is the *performance* $P$, which counts the number of correctly "interpreted" pairs of vertices, i.e. two vertices belonging to the same community and connected by an edge, or two vertices belonging to different communities and not connected by an edge. The definition of performance, for a partition $\mathcal{P}$, is

$$P(\mathcal{P}) = \frac{|\{(i,j)\in E, C_i = C_j\}| + |\{(i,j)\notin E, C_i \neq C_j\}|}{n(n-1)/2}. \tag{12}$$

By definition, $0 \leq P(\mathcal{P}) \leq 1$. Another example is *coverage*, i.e. the ratio of the number of intra-community edges by the total number of edges: by definition, an ideal cluster structure, where the clusters are disconnected from each other, yields a coverage of 1, as all edges of the graph fall within clusters.

The most popular quality function is the modularity of Newman and Girvan [54]. It is based on the idea that a random graph is not expected to have a cluster structure, so the possible existence of clusters is revealed by the comparison between the actual density of edges in a subgraph and the density one would expect to have in the subgraph if the vertices of the graph were attached regardless of community structure. This expected edge density depends on the chosen *null model*, i.e. a copy of the original graph keeping some of its structural properties but without community structure. Modularity can then be written as follows

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - P_{ij}\right)\delta(C_i, C_j), \tag{13}$$

where the sum runs over all pairs of vertices, $A$ is the adjacency matrix, $m$ the total number of edges of the graph, and $P_{ij}$ represents the expected number of edges between vertices $i$ and $j$ in the null model. The $\delta$-function yields one if vertices $i$ and $j$ are in the same community ($C_i = C_j$), zero otherwise. The choice of the null model graph is in principle arbitrary, and several possibilities exist. For instance, one could simply demand that the graph keeps the same number of edges as the original graph, and that edges are placed with the same probability between any pair of vertices. In this case (Bernoulli random graph), the null model term in Eq. (13) would be a constant (i.e. $P_{ij} = p = 2m/[n(n-1)]$, $\forall i, j$). However this null model is not a good descriptor of real networks, as it has a Poissonian degree distribution which is very different from the skewed distributions found in real networks. Due to the important implications that broad degree distributions have for the structure and function of real networks [5,9,7,113,8,10], it is preferable to go for a null model with the same degree distribution of the original graph. The standard null model of modularity imposes that the expected degree sequence (after averaging over all possible configurations of the model) matches the actual degree sequence of the graph. This is a stricter constraint than merely requiring the match of the degree distributions, and is essentially equivalent[6] to the *configuration model*, which has been subject of intense investigation in the recent literature on networks [114,115]. In this null model, a vertex could be attached to any other vertex of the graph and the probability that vertices $i$ and $j$, with degrees $k_i$ and $k_j$, are connected, can be calculated without problems. In fact, in order to form an edge between $i$ and $j$ one needs to join two *stubs* (i.e. half-edges), incident with $i$ and $j$. The probability $p_i$ to pick at random a stub incident with $i$ is $k_i/2m$, as there are $k_i$ stubs incident with $i$ out of a total of $2m$. The probability of a connection between $i$ and $j$ is then given by the product $p_i p_j$, since edges are placed independently of each other. The result is $k_i k_j/4m^2$, which yields an expected number $P_{ij} = 2m p_i p_j = k_i k_j/2m$ of edges between $i$ and $j$. So, the final expression of modularity reads

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(C_i, C_j). \tag{14}$$

[6] The difference is that the configuration model maintains the same degree sequence of the original graph for each realization, whereas in the null model of modularity the degree sequence of a realization is in general different, and only the average/expected degree sequence coincides with that of the graph at hand. The two models are equivalent in the limit of infinite graph size.

Since the only contributions to the sum come from vertex pairs belonging to the same cluster, we can group these contributions together and rewrite the sum over the vertex pairs as a sum over the clusters

$$Q = \sum_{c=1}^{n_c} \left[ \frac{l_c}{m} - \left( \frac{d_c}{2m} \right)^2 \right]. \tag{15}$$

Here, $n_c$ is the number of clusters, $l_c$ the total number of edges joining vertices of module $c$ and $d_c$ the sum of the degrees of the vertices of $c$. In Eq. (15), the first term of each summand is the fraction of edges of the graph inside the module, whereas the second term represents the expected fraction of edges that would be there if the graph were a random graph with the same expected degree for each vertex.

A nice feature of modularity is that it can be equivalently expressed both in terms of the intra-cluster edges, as in Eq. (15), and in terms of the inter-cluster edges [116]. In fact, the maximum of modularity can be expressed as

$$\begin{aligned} Q_{\max} &= \max_{\mathcal{P}} \left\{ \sum_{c=1}^{n_c} \left[ \frac{l_c}{m} - \left( \frac{d_c}{2m} \right)^2 \right] \right\} = \frac{1}{m} \max_{\mathcal{P}} \left\{ \sum_{c=1}^{n_c} \left[ l_c - \mathrm{Ex}(l_c) \right] \right\} \\ &= -\frac{1}{m} \min_{\mathcal{P}} \left\{ - \sum_{c=1}^{n_c} \left[ l_c - \mathrm{Ex}(l_c) \right] \right\}, \end{aligned} \tag{16}$$

where $\max_{\mathcal{P}}$ and $\min_{\mathcal{P}}$ indicates the maximum and the minimum over all possible graph partitions $\mathcal{P}$ and $\mathrm{Ex}(l_c) = d_c^2/4m$ indicates the expected number of links in cluster $c$ in the null model of modularity. By adding and subtracting the total number of edges $m$ of the graph one finally gets

$$\begin{aligned} Q_{\max} &= -\frac{1}{m} \min_{\mathcal{P}} \left\{ \left[ \left( m - \sum_{c=1}^{n_c} l_c \right) - \left( m - \sum_{c=1}^{n_c} \mathrm{Ex}(l_c) \right) \right] \right\} \\ &= -\frac{1}{m} \min_{\mathcal{P}} (|\mathrm{Cut}_{\mathcal{P}}| - \mathrm{ExCut}_{\mathcal{P}}). \end{aligned} \tag{17}$$

In the last expression $|\mathrm{Cut}_{\mathcal{P}}| = m - \sum_{c=1}^{n_c} l_c$ is the number of inter-cluster edges of partition $\mathcal{P}$, and $\mathrm{ExCut}_{\mathcal{P}} = m - \sum_{c=1}^{n_c} \mathrm{Ex}(l_c)$ is the expected number of inter-cluster edges of the partition in modularity's null model.

According to Eq. (15), a subgraph is a module if the corresponding contribution to modularity in the sum is positive. The more the number of internal edges of the cluster exceeds the expected number, the better defined the community. So, large positive values of the modularity indicate good partitions.[7] The maximum modularity of a graph generally grows if the size of the graph and/or the number of (well-separated) clusters increases [117]. Therefore, modularity should not be used to compare the quality of the community structure of graphs which are very different in size. The modularity of the whole graph, taken as a single community, is zero, as the two terms of the only summand in this case are equal and opposite. Modularity is always smaller than one, and can be negative as well. For instance, the partition in which each vertex is a community is always negative: in this case the sum runs over $n$ terms, which are all negative as the first term of each summand is zero. This is a nice feature of the measure, implying that, if there are no partitions with positive modularity, the graph has no community structure. On the contrary, the existence of partitions with large negative modularity values may hint to the existence of subgroups with very few internal edges and many edges lying between them (*multipartite structure*) [118]. Holmström et al. [119] have shown that the distribution of modularity values across the partitions of various graphs, real and artificial (including random graphs with no apparent community structure), has some stable features, and that the most likely modularity values correspond to partitions in clusters of approximately equal size.

Modularity has been employed as quality function in many algorithms, like some of the divisive algorithms of Section 5. In addition, modularity optimization is itself a popular method for community detection (see Section 6.1). Modularity also allows one to assess the stability of partitions [120] (Section 14), it can be used to design layouts for graph visualization [121] and to perform a sort of renormalization of a graph, by transforming a graph into a smaller one with the same community structure [122].

## 4. Traditional methods

### 4.1. Graph partitioning

The problem of graph partitioning consists of dividing the vertices in $g$ groups of predefined size, such that the number of edges lying between the groups is minimal. The number of edges running between clusters is called *cut size*. Fig. 9 presents the solution of the problem for a graph with fourteen vertices, for $g = 2$ and clusters of equal size.

[7] This is not necessarily true, as we will see in Section 6.3.

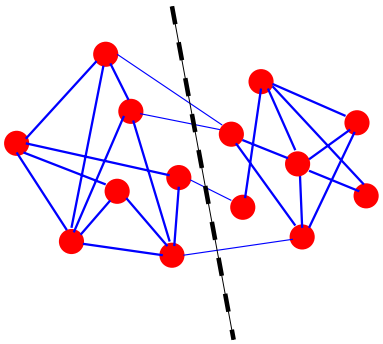

**Fig. 9.** Graph partitioning. The dashed line shows the solution of the minimum bisection problem for the graph illustrated, i.e. the partition in two groups of equal size with minimal number of edges running between the groups. Reprinted figure with permission from Ref. [16].


Specifying the number of clusters of the partition is necessary. If one simply imposed a partition with the minimal cut size, and left the number of clusters free, the solution would be trivial, corresponding to all vertices ending up in the same cluster, as this would yield a vanishing cut size. Specifying the size is also necessary, as otherwise the most likely solution of the problem would consist of separating the lowest degree vertex from the rest of the graph, which is quite uninteresting. This problem can be actually avoided by choosing a different measure to optimize for the partitioning, which accounts for the size of the clusters. Some of these measures will be briefly introduced at the end of this section.

Graph partitioning is a fundamental issue in parallel computing, circuit partitioning and layout, and in the design of many serial algorithms, including techniques to solve partial differential equations and sparse linear systems of equations. Most variants of the graph partitioning problem are NP-hard. There are however several algorithms that can do a good job, even if their solutions are not necessarily optimal [123]. Many algorithms perform a bisection of the graph. Partitions into more than two clusters are usually attained by iterative bisectioning. Moreover, in most cases one imposes the constraint that the clusters have equal size. This problem is called *minimum bisection* and is NP-hard.

The *Kernighan–Lin algorithm* [124] is one of the earliest methods proposed and is still frequently used, often in combination with other techniques. The authors were motivated by the problem of partitioning electronic circuits onto boards: the nodes contained in different boards need to be linked to each other with the least number of connections. The procedure is an optimization of a benefit function $Q$, which represents the difference between the number of edges inside the modules and the number of edges lying between them. The starting point is an initial partition of the graph in two clusters of the predefined size: such an initial partition can be random or suggested by some information on the graph structure. Then, subsets consisting of equal numbers of vertices are swapped between the two groups, so that $Q$ has the maximal increase. The subsets can consist of single vertices. To reduce the risk to be trapped in local maxima of $Q$, the procedure includes some swaps that decrease the function $Q$. After a series of swaps with positive and negative gains, the partition with the largest value of $Q$ is selected and used as starting point of a new series of iterations. The Kernighan–Lin algorithm is quite fast, scaling as $O(n^2 \log n)$ ($n$ being as usual the number of vertices), if only a constant number of swaps are performed at each iteration. The most expensive part is the identification of the subsets to swap, which requires the computation of the gains/losses for any pair of candidate subsets. On sparse graphs, a slightly different heuristic allows to lower the complexity to $O(n^2)$. The partitions found by the procedure are strongly dependent on the initial configuration and other algorithms can do better. It is preferable to start with a good guess about the sought partition, otherwise the results are quite poor. Therefore the method is typically used to improve on the partitions found through other techniques, by using them as starting configurations for the algorithm. The Kernighan–Lin algorithm has been extended to extract partitions in any number of parts [125], however the run-time and storage costs increase rapidly with the number of clusters.

Another popular technique is the *spectral bisection method* [126], which is based on the properties of the spectrum of the Laplacian matrix. Spectral clustering will be discussed more thoroughly in Section 4.4, here we focus on its application to graph partitioning.

Every partition of a graph with $n$ vertices in two groups can be represented by an index vector $\mathbf{s}$, whose component $\mathbf{s}_i$ is $+1$ if vertex $i$ is in one group and $-1$ if it is in the other group. The cut size $R$ of the partition of the graph in the two groups can be written as

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s}, \tag{18}$$

where $\mathbf{L}$ is the Laplacian matrix and $\mathbf{s}^T$ the transpose of vector $\mathbf{s}$. Vector $\mathbf{s}$ can be written as $\mathbf{s} = \sum_i a_i \mathbf{v}_i$, where $\mathbf{v}_i, i = 1, \ldots, n$ are the eigenvectors of the Laplacian. If $\mathbf{s}$ is properly normalized, then

$$R = \sum_i a_i^2 \lambda_i, \tag{19}$$

where $\lambda_i$ is the Laplacian eigenvalue corresponding to eigenvector $\mathbf{v}_i$. It is worth remarking that the sum contains at most $n-1$ terms, as the Laplacian has at least one zero eigenvalue. Minimizing $R$ equals to the minimization of the sum on the right-hand side of Eq. (19). This task is still very hard. However, if the second lowest eigenvector $\lambda_2$ is close enough to zero, a good approximation of the minimum can be attained by choosing $\mathbf{s}$ parallel with the corresponding eigenvector $\mathbf{v}_2$, which is called the *Fiedler vector* [127]: this would reduce the sum to $\lambda_2$, which is a small number. But the index vector cannot be perfectly parallel with $\mathbf{v}_2$ by construction, because all its components are equal in modulus, whereas the components of $\mathbf{v}_2$ are not. The best choice is to match the signs of the components. So, one can set $\mathbf{s}_i = +1\,(-1)$ if $\mathbf{v}_2^i > 0\,(<0)$. It may happen that the sizes of the two corresponding groups do not match the predefined sizes one wishes to have. In this case, if one aims at a split in $n_1$ and $n_2 = n - n_1$ vertices, the best strategy is to order the components of the Fiedler vector from the lowest to the largest values and to put in one group the vertices corresponding to the first $n_1$ components from the top or the bottom, and the remaining vertices in the second group. This procedure yields two partitions: the better solution is naturally the one that gives the smaller cut size.

The spectral bisection method is quite fast. The first eigenvectors of the Laplacian can be computed by using the Lanczos method [128]. The time required to compute the first $k$ eigenvectors of a matrix with the Lanczos method depends on the size of the eigengap $|\lambda_{k+1} - \lambda_k|$ [129]. If the eigenvalues $\lambda_{k+1}$ and $\lambda_k$ are well separated, the running time of the algorithm is much shorter than the time required to calculate the complete set of eigenvectors, which scales as $O(n^3)$. The method gives in general good partitions, that can be further improved by applying the Kernighan–Lin algorithm.

The well known max-flow min-cut theorem by Ford and Fulkerson [130] states that the minimum cut between any two vertices $s$ and $t$ of a graph, i.e. any minimal subset of edges whose deletion would topologically separate $s$ from $t$, carries the maximum flow that can be transported from $s$ to $t$ across the graph. In this context edges play the role of water pipes, with a given carrying capacity (e.g. their weights), and vertices the role of pipe junctions. This theorem has been used to determine minimal cuts from maximal flows in clustering algorithms. There are several efficient routines to compute maximum flows in graphs, like the algorithm of Goldberg and Tarjan [131]. Flake et al. [132,25] have recently used maximum flows to identify communities in the graph of the World Wide Web. The web graph is directed but for the purposes of the calculation Flake et al. treated the edges as undirected. Web communities are defined to be "strong" (LS-sets), i.e. the internal degree of each vertex must not be smaller than its external degree [78]. An artificial sink $t$ is added to the graph and one calculates the maximum flows from a source vertex $s$ to the sink $t$: the corresponding minimum cut identifies the community of vertex $s$, provided $s$ shares a sufficiently large number of edges with the other vertices of its community, otherwise one could get trivial separations and meaningless clusters.

Other popular methods for graph partitioning include level-structure partitioning, the geometric algorithm, multilevel algorithms, etc. A good description of these algorithms can be found in Ref. [123].

Graphs can be also partitioned by minimizing measures that are affine to the cut size, like *conductance* [2]. The conductance $\Phi(\mathcal{C})$ of the subgraph $\mathcal{C}$ of a graph $\mathcal{G}$ is defined as

$$\Phi(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{\min(k_{\mathcal{C}}, k_{\mathcal{G}\setminus\mathcal{C}})}, \tag{20}$$

where $c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})$ is the cut size of $\mathcal{C}$, and $k_{\mathcal{C}}$, $k_{\mathcal{G}\setminus\mathcal{C}}$ are the total degrees of $\mathcal{C}$ and of the rest of the graph $\mathcal{G} \setminus \mathcal{C}$, respectively. Cuts are defined only between non-empty sets, otherwise the measure would not be defined (as the denominator in Eq. (20) would vanish). The minimum of the conductance is obtained in correspondence of low values of the cut size and of large values for the denominator in Eq. (20), which peaks when the total degrees of the cluster and its complement are equal. In practical applications, especially on large graphs, close values of the total degrees correspond to clusters of approximately equal size. The problem of finding a cut with minimal conductance is **NP**-hard [84]. Similar measures are the *ratio cut* [133] and the *normalized cut* [134,135]. The ratio cut of a cluster $\mathcal{C}$ is defined as

$$\Phi_C(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{n_{\mathcal{C}} n_{\mathcal{G}\setminus\mathcal{C}}}, \tag{21}$$

where $n_{\mathcal{C}}$ and $n_{\mathcal{G}\setminus\mathcal{C}}$ are the numbers of vertices of the two subgraphs. The normalized cut of a cluster $\mathcal{C}$ is

$$\Phi_N(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{k_{\mathcal{C}}}, \tag{22}$$

where $k_{\mathcal{C}}$ is again the total degree of $\mathcal{C}$. As for the conductance, minimizing the ratio cut and the normalized cut favors partitions into clusters of approximately equal size, measured in terms of the number of vertices or edges, respectively. On the other hand, graph partitioning requires preliminary assumptions on the cluster sizes, whereas the minimization of conductance, ratio cut and normalized cut does not. The ratio cut was introduced for circuit partitioning [133] and its optimization is an **NP**-hard problem [136]. The normalized cut is frequently used in image segmentation [137] and its optimization is **NP**-complete [135]. The cut ratio and the normalized cut can be quite well minimized via spectral clustering [138,139] (Section 4.4).

Algorithms for graph partitioning are not good for community detection, because it is necessary to provide as input the number of groups and in some cases even their sizes, about which in principle one knows nothing. Instead, one would like an algorithm capable of producing this information in its output. Besides, from the methodological point of view, using

iterative bisectioning to split the graph in more pieces is not a reliable procedure. For instance, a split into three clusters is necessarily obtained by breaking either cluster of the original bipartition in two parts, whereas in many cases a minimum cut partition is obtained if the third cluster is a merger of parts of both initial clusters.

### 4.2. Hierarchical clustering

In general, very little is known about the community structure of a graph. It is uncommon to know the number of clusters in which the graph is split, or other indications about the membership of the vertices. In such cases clustering procedures like graph partitioning methods can hardly be of help, and one is forced to make some reasonable assumptions about the number and size of the clusters, which are often unjustified. On the other hand, the graph may have a hierarchical structure, i.e. may display several levels of grouping of the vertices, with small clusters included within large clusters, which are in turn included in larger clusters, and so on. Social networks, for instance, often have a hierarchical structure (Section 3.3.1). In such cases, one may use *hierarchical clustering algorithms* [140], i.e. clustering techniques that reveal the multilevel structure of the graph. Hierarchical clustering is very common in social network analysis, biology, engineering, marketing, etc.

The starting point of any hierarchical clustering method is the definition of a similarity measure between vertices. After a measure is chosen, one computes the similarity for each pair of vertices, no matter if they are connected or not. At the end of this process, one is left with a new $n \times n$ matrix $X$, the similarity matrix. In Section 3.2.4 we have listed several possible definitions of similarity. Hierarchical clustering techniques aim at identifying groups of vertices with high similarity, and can be classified in two categories:

1. *Agglomerative algorithms*, in which clusters are iteratively merged if their similarity is sufficiently high;
2. *Divisive algorithms*, in which clusters are iteratively split by removing edges connecting vertices with low similarity.

The two classes refer to opposite processes: agglomerative algorithms are bottom–up, as one starts from the vertices as separate clusters (singletons) and ends up with the graph as a unique cluster; divisive algorithms are top–down as they follow the opposite direction. Divisive techniques have been rarely used in the past (meanwhile they have become more popular, see Section 5), so we shall concentrate here on agglomerative algorithms.

Since clusters are merged based on their mutual similarity, it is essential to define a measure that estimates how similar clusters are, out of the matrix $X$. This involves some arbitrariness and several prescriptions exist. In *single linkage clustering*, the similarity between two groups is the minimum element $x_{ij}$, with $i$ in one group and $j$ in the other. On the contrary, the maximum element $x_{ij}$ for vertices of different groups is used in the procedure of *complete linkage clustering*. In *average linkage clustering* one has to compute the average of the $x_{ij}$.

The procedure can be better illustrated by means of dendrograms (Section 3.3.1), like the one in Fig. 8. Sometimes, stopping conditions are imposed to select a partition or a group of partitions that satisfy a special criterion, like a given number of clusters or the optimization of a quality function (e.g. modularity).

Hierarchical clustering has the advantage that it does not require a preliminary knowledge on the number and size of the clusters. However, it does not provide a way to discriminate between the many partitions obtained by the procedure, and to choose that or those that better represent the community structure of the graph. The results of the method depend on the specific similarity measure adopted. The procedure also yields a hierarchical structure by construction, which is rather artificial in most cases, since the graph at hand may not have a hierarchical structure at all. Moreover, vertices of a community may not be correctly classified, and in many cases some vertices are missed even if they have a central role in their clusters [13]. Another problem is that vertices with just one neighbor are often classified as separated clusters, which in most cases does not make sense. Finally, a major weakness of agglomerative hierarchical clustering is that it does not scale well. If points are embedded in space, so that one can use the distance as dissimilarity measure, the computational complexity is $O(n^2)$ for single linkage, $O(n^2 \log n)$ for the complete and average linkage schemes. For graph clustering, where distance is not trivially defined, the complexity can become much heavier if the calculation of the chosen similarity measure is costly.

### 4.3. Partitional clustering

*Partitional clustering* indicates another popular class of methods to find clusters in a set of data points. Here, the number of clusters is preassigned, say $k$. The points are embedded in a metric space, so that each vertex is a point and a distance measure is defined between pairs of points in the space. The distance is a measure of dissimilarity between vertices. The goal is to separate the points in $k$ clusters such to maximize/minimize a given cost function based on distances between points and/or from points to *centroids*, i.e. suitably defined positions in space. Some of the most used functions are listed below:

- *Minimum k-clustering*. The cost function here is the *diameter* of a cluster, which is the largest distance between two points of a cluster. The points are classified such that the largest of the $k$ cluster diameters is the smallest possible. The idea is to keep the clusters very "compact".
- *k-clustering sum*. Same as minimum $k$-clustering, but the diameter is replaced by the average distance between all pairs of points of a cluster.
- *k-center*. For each cluster $i$ one defines a reference point $x_i$, the centroid, and computes the maximum $d_i$ of the distances of each cluster point from the centroid. The clusters and centroids are self-consistently chosen such to minimize the largest value of $d_i$.
- *k-median*. Same as $k$-center, but the maximum distance from the centroid is replaced by the average distance.

The most popular partitional technique in the literature is *k-means clustering* [141]. Here the cost function is the total intra-cluster distance, or squared error function

$$\sum_{i=1}^{k} \sum_{\mathbf{x_j} \in S_i} \|\mathbf{x_j} - \mathbf{c_i}\|^2, \tag{23}$$

where $S_i$ indicates the subset of points of the $i$-th cluster and $\mathbf{c_i}$ its centroid. The $k$-means problem can be simply solved with the Lloyd's algorithm [142]. One starts from an initial distribution of centroids such that they are as far as possible from each other. In the first iteration, each vertex is assigned to the nearest centroid. Next, the centers of mass of the $k$ clusters are estimated and become a new set of centroids, which allows for a new classification of the vertices, and so on. After a small number of iterations, the positions of the centroids are stable, and the clusters do not change any more. The solution found is not optimal, and it strongly depends on the initial choice of the centroids. Nevertheless, Lloyd's heuristic has remained popular due to its quick convergence, which makes it suitable for the analysis of large data sets. The result can be improved by performing more runs starting from different initial conditions, and picking the solution which yields the minimum value of the total intra-cluster distance. Extensions of $k$-means clustering to graphs have been proposed by some authors [143–145].

Another popular technique, similar in spirit to $k$-means clustering, is *fuzzy k-means clustering* [146,147]. This method accounts for the fact that a point may belong to two or more clusters at the same time and is widely used in pattern recognition. The associated cost function is

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^m \|\mathbf{x_i} - \mathbf{c_j}\|^2, \tag{24}$$

where $u_{ij}$ is the *membership matrix*, which measures the degree of membership of point $i$ (with position $\mathbf{x_i}$) in cluster $j$, $m$ is a real number greater than 1 and $\mathbf{c_j}$ is the center of cluster $j$

$$\mathbf{c_j} = \frac{\sum_{i=1}^{n} u_{ij}^m \mathbf{x_i}}{\sum_{i=1}^{n} u_{ij}^m}. \tag{25}$$

The matrix $u_{ij}$ is normalized so that the sum of the memberships of every point in all clusters yields 1. The membership $u_{ij}$ is related to the distance of point $i$ from the center of cluster $j$, as it is reasonable to assume that the larger this distance, the lower $u_{ij}$. This can be expressed by the following relation

$$u_{ij} = \frac{1}{\sum_{l=1}^{k} \left( \frac{\|\mathbf{x_i} - \mathbf{c_j}\|}{\|\mathbf{x_i} - \mathbf{c_l}\|} \right)^{\frac{2}{m-1}}}. \tag{26}$$

The cost function $J_m$ can be minimized by iterating Eqs. (25) and (26). One starts from some initial guess for $u_{ij}$ and uses Eq. (25) to compute the centers, which are then plugged back into Eq. (26), and so on. The process stops when the corresponding elements of the membership matrix in consecutive iterations differ from each other by less than a predefined tolerance. It can be shown that this procedure indeed delivers a local minimum of the cost function $J_m$ of Eq. (24). This procedure has the same problems of Lloyd's algorithm for $k$-means clustering, i.e. the minimum is a local minimum, and depends on the initial choice of the matrix $u_{ij}$.

The limitation of partitional clustering is the same as that of the graph partitioning algorithms: the number of clusters must be specified at the beginning, the method is not able to derive it. In addition, the embedding in a metric space can be natural for some graphs, but rather artificial for others.

### 4.4. Spectral clustering

Let us suppose to have a set of $n$ objects $x_1, x_2, \ldots, x_n$ with a pairwise similarity function $S$ defined between them, which is symmetric and non-negative (i.e., $S(x_i, x_j) = S(x_j, x_i) \geq 0, \forall i, j = 1, \ldots, n$). *Spectral clustering* includes all methods and techniques that partition the set into clusters by using the eigenvectors of matrices, like $S$ itself or other matrices derived from it. In particular, the objects could be points in some metric space, or the vertices of a graph. Spectral clustering consists of a transformation of the initial set of objects into a set of points in space, whose coordinates are elements of eigenvectors: the set of points is then clustered via standard techniques, like $k$-means clustering (Section 4.3). One may wonder why it is necessary to cluster the points obtained through the eigenvectors, when one can directly cluster the initial set of objects, based on the similarity matrix. The reason is that the change of representation induced by the eigenvectors makes the cluster properties of the initial data set much more evident. In this way, spectral clustering is able to separate data points that could not be resolved by applying directly $k$-means clustering, for instance, as the latter tends to deliver convex sets of points.

The first contribution on spectral clustering was a paper by Donath and Hoffmann [148], who used the eigenvectors of the adjacency matrix for graph partitions. In the same year, Fiedler [127] realized that from the eigenvector of the second

smallest eigenvalue of the Laplacian matrix it was possible to obtain a bipartition of the graph with very low cut size, as we have explained in Section 4.1. For a historical survey see Ref. [149]. In this section we shall follow the nice tutorial by von Luxburg [150], with a focus on spectral graph clustering. The concepts and methods discussed below apply to both unweighted and weighted graphs.

The Laplacian is by far the most used matrix in spectral clustering. In Appendix A.2 we see that the unnormalized Laplacian of a graph with $k$ connected components has $k$ zero eigenvalues. In this case the Laplacian can be written in block-diagonal form, i.e. the vertices can be ordered in such a way that the Laplacian displays $k$ square blocks along the diagonal, with (some) entries different from zero, whereas all other elements vanish. Each block is the Laplacian of the corresponding subgraph, so it has the trivial eigenvector with components $(1, 1, 1, \ldots, 1, 1)$. Therefore, there are $k$ degenerate eigenvectors with equal non-vanishing components in correspondence with the vertices of a block, whereas all other components are zero. In this way, from the components of the eigenvectors one can identify the connected components of the graph. For instance, let us consider the $n \times k$ matrix, whose columns are the $k$ eigenvectors above mentioned. The $i$-th row of this matrix is a vector with $k$ components representing vertex $i$ of the graph. Vectors representing vertices in the same connected component of the graph coincide, and their tip lies on one of the axes of a $k$-dimensional system of coordinates (i.e. they are all vectors of the form $(0, 0, \ldots 0, 1, 0, \ldots, 0, 0)$). So, by drawing the vertex vectors one would see $k$ distinct points, each on a different axis, corresponding to the graph components.

If the graph is connected, but consists of $k$ subgraphs which are weakly linked to each other, the spectrum of the unnormalized Laplacian will have one zero eigenvalue, all others being positive. Now the Laplacian cannot be put in block-diagonal form: even if one enumerates the vertices in the order of their cluster memberships (by listing first the vertices of one cluster, then the vertices of another cluster, etc.) there will always be some non-vanishing entries outside of the blocks. However, the lowest $k-1$ non-vanishing eigenvalues are still close to zero, and the vertex vectors of the first $k$ eigenvectors should still enable one to clearly distinguish the clusters in a $k$-dimensional space. Vertex vectors corresponding to the same cluster are now not coincident, in general, but still rather close to each other. So, instead of $k$ points, one would observe $k$ groups of points, with the points of each group localized close to each other and far from the other groups. Techniques like $k$-means clustering (Section 4.3) can then easily recover the clusters.

The scenario we have described is expected from perturbation theory [151,152]. In principle all symmetric matrices that can be put in block-diagonal form have a set of eigenvectors (as many as the blocks), such that the elements of each eigenvector are different from zero on the vertices of a block and zero otherwise, just like the Laplacian. The adjacency matrix itself has the same property, for example. This is a necessary condition for the eigenvectors to be successfully used for graph clustering, but it is not sufficient. In the case of the Laplacian, for a graph with $k$ connected components, we know that the eigenvectors corresponding to the $k$ lowest eigenvalues come each from one of the components. In the case of the adjacency matrix $\mathbf{A}$ (or of its weighted counterpart $\mathbf{W}$), instead, it may happen that large eigenvalues refer to the same component. So, if one takes the eigenvectors corresponding to the $k$ largest eigenvalues,[8] some components will be over-represented, while others will be absent. Therefore, using the eigenvectors of $\mathbf{A}$ (or $\mathbf{W}$) in spectral graph clustering is in general not reliable. Moreover, the elements of the eigenvectors corresponding to the components should be sufficiently far from zero. To understand why, suppose that we take a (symmetric, block-diagonal) matrix, and that one or more elements of one of the eigenvectors corresponding to the connected components is very close to zero. If one perturbs the graph by adding edges between different components, all entries of the perturbed eigenvectors will become non-zero and some may have comparable values as the lowest elements of the eigenvectors on the blocks. Therefore distinguishing vertices of different components may become a problem, even when the perturbation is fairly small, and misclassifications are likely. On the other hand, the non-vanishing elements of the (normalized) eigenvectors of the unnormalized Laplacian, for instance, are all equal to $1/\sqrt{n_i}$, where $n_i$ is the number of vertices in the $i$-th component. In this way, there is a gap between the lowest element (here they are all equal for the same eigenvector) and zero. This holds as well for the normalized Laplacian $\mathbf{L_{rw}}$ (Appendix A.2). For the other normalized Laplacian $\mathbf{L_{sym}}$ (Appendix A.2), the non-zero elements of the eigenvectors corresponding to the connected components are proportional to the square root of the degree of the corresponding vertex. So, if degrees are very different from each other, and especially if there are vertices with very low degree, some eigenvector elements may be quite small. As we shall see below, in the context of the technique by Ng et al. [153], a suitable normalization procedure is introduced to alleviate this problem.

Now that we have explained why the Laplacian matrix is particularly suitable for spectral clustering, we proceed with the description of three popular methods: *unnormalized spectral clustering* and two *normalized spectral clustering* techniques, proposed by Shi and Malik [134,135] and by Ng et al. [153], respectively.

Unnormalized spectral clustering uses the unnormalized Laplacian $\mathbf{L}$. The inputs are the adjacency matrix $\mathbf{A}$ ($\mathbf{W}$ for weighted graphs) and the number $k$ of clusters to be recovered. The first step consists of computing the eigenvectors corresponding to the lowest $k$ eigenvalues of $\mathbf{L}$. Then, one builds the $n \times k$ matrix $\mathbf{V}$, whose columns are the $k$ eigenvectors. The $n$ rows of $\mathbf{V}$ are used to represent the graph vertices in a $k$-dimensional Euclidean space, through a Cartesian system of coordinates. The points are then grouped in $k$ clusters by using $k$-means clustering or similar techniques (Section 4.3). Normalized spectral clustering works in the same way. In the version by Shi and Malik [134,135], one uses the eigenvectors

[8] Large eigenvalues of the adjacency matrix are the counterpart of the low eigenvalues of the Laplacian, since $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix whose elements are the vertex degrees.

of the normalized Laplacian $\mathbf{L_{rw}}$ (Appendix A.2). In the algorithm by Ng et al. [153] one adopts the normalized Laplacian $\mathbf{L_{sym}}$ (Appendix A.2). Here, however, the matrix $\mathbf{V}$ is normalized by dividing the elements of each row by their sum, obtaining a new matrix $\mathbf{U}$, whose rows are then used to represent the vertices in space, as in the other methods. By doing so, it is much more unlikely that eigenvector components for a well-separated cluster are close to zero, a scenario which would make the classification of the corresponding vertices problematic, as we have said above. However, if the graph has some vertices with low degree, they may still be misclassified.

Spectral clustering is closely related to graph partitioning. Relaxed versions of the minimization of ratio cut and normalized cut (see Section 4.1) can be turned into spectral clustering problems, by following similar procedures as in spectral graph partitioning. The measure to minimize can be expressed in matrix form, obtaining similar expressions as for the cut size (see Eq. (18)), with index vectors defining the partition of the graph in groups through the values of their entries. For instance, for the minimum cut bipartition of Section 4.1, there is only one index vector $\mathbf{s}$, whose components equal $\pm 1$, where the signs indicate the two clusters. The relaxation consists in performing the minimization over all possible vectors $\mathbf{s}$, allowing for real-valued components as well. This version of the problem is exactly equivalent to spectral clustering. The relaxed minimization of ratio cut for a partition in $k$ clusters yields the $n$ $k$-dimensional vertex vectors of unnormalized spectral clustering [150]; for a normalized cut one obtains the $n$ $k$-dimensional vertex vectors of normalized spectral clustering, with the normalized Laplacian $\mathbf{L_{rw}}$ [134]. The problem is then to turn the resulting vectors into a partition of the graph, which can be done by using techniques like $k$-means clustering, as we have seen above. However, it is still unclear what is the relation between the original minimum cut problem over actual graph partitions and the relaxed version of it, in particular how close one can come to the real solution via spectral clustering.

Random walks on graphs are also related to spectral clustering. In fact, by minimizing the number of edges between clusters (properly normalized for measures like, e.g., ratio cut and normalized cut) one forces random walkers to spend more time within clusters and to move more rarely from one cluster to another. In particular, unnormalized spectral clustering with the Laplacian $\mathbf{L_{rw}}$ has a natural link with random walks, because $\mathbf{L_{rw}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ (Appendix A.2), where $\mathbf{D}^{-1}\mathbf{A}$ is the transfer matrix $\mathbf{T}$. This has interesting consequences. For instance, Meilă and Shi have proven that the normalized cut for a bipartition equals the total probability that a random walker moves from one of the clusters to the other in either sense [154]. In this way, minimizing the normalized cut means looking for a partition minimizing the probability of transitions between clusters.

Spectral clustering requires the computation of the first $k$ eigenvectors of a Laplacian matrix. If the graph is large, an exact computation of the eigenvectors is impossible, as it would require a time $O(n^3)$. Fortunately there are approximate techniques, like the power method or Krylov subspace techniques like the Lanczos method [129], whose speed depends on the size of the eigengap $|\lambda_{k+1} - \lambda_k|$, where $\lambda_k$ and $\lambda_{k+1}$ are the $k$-th and $(k+1)$-th smallest eigenvalue of the matrix. The larger the eigengap, the faster the convergence. In fact, the existence of large gaps between pairs of consecutive eigenvalues could suggest the number of clusters of the graph, an information which is not delivered by spectral clustering and which has to be given as input. We know that, for a disconnected graph with $k$ components, the first $k$ eigenvalues of the Laplacian matrix (normalized or not) are zero, whether the $(k+1)$-th is non-zero. If the clusters are weakly connected to each other, one expects that the first $k$ eigenvalues remain close to zero, and that the $(k+1)$-th is clearly different from zero. By reversing this argument, the number of clusters of a graph could be derived by checking whether there is an integer $k$ such that the first $k$ eigenvalues are small and the $(k+1)$-th is relatively large. However, when the clusters are very mixed with each other, it may be hard to identify significant gaps between the eigenvalues.

The last issue we want to point out concerns the choice of the Laplacian matrix to use in the applications. If the graph vertices have the same or similar degrees, there is no substantial difference between the unnormalized and the normalized Laplacians. If there are big inhomogeneities among the vertex degrees, instead, the choice of the Laplacian considerably affects the results. In general, normalized Laplacians are more promising because the corresponding spectral clustering techniques implicitly impose a double optimization on the set of partitions, such that the intracluster edge density is high and, at the same time, the intercluster edge density is low. On the contrary, the unnormalized Laplacian is related to the intercluster edge density only. Moreover, unnormalized spectral clustering does not always converge, and sometimes yields trivial partitions in which one or more clusters consist of a single vertex. Of the normalized Laplacians, $\mathbf{L_{rw}}$ is more reliable than $\mathbf{L_{sym}}$ because the eigenvectors of $\mathbf{L_{rw}}$ corresponding to the lowest eigenvalues are cluster indicator vectors, i.e., they have equal non-vanishing entries in correspondence of the vertices of each cluster, and zero elsewhere, if the clusters are disconnected. The eigenvectors of $\mathbf{L_{sym}}$, instead, are obtained by (left-) multiplying those of $\mathbf{L_{rw}}$ by the matrix $\mathbf{D}^{1/2}$: in this way, eigenvector components corresponding to vertices of the same cluster are no longer equal, in general, a complication that may induce artifacts in the spectral clustering procedure.

## 5. Divisive algorithms

A simple way to identify communities in a graph is to detect the edges that connect vertices of different communities and remove them, so that the clusters get disconnected from each other. This is the philosophy of divisive algorithms. The crucial point is to find a property of intercommunity edges that could allow for their identification. Divisive methods do not introduce substantial conceptual advances with respect to traditional techniques, as they just perform hierarchical clustering on the graph at study (Section 4.2). The main difference with divisive hierarchical clustering is that here one removes inter-cluster edges instead of edges between pairs of vertices with low similarity and there is no guarantee *a priori*

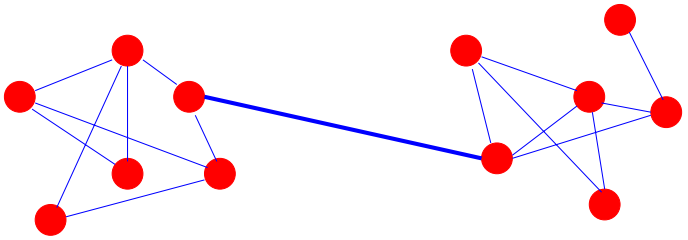

**Fig. 10.** Edge betweenness is highest for edges connecting communities. In the figure, the edge in the middle has a much higher betweenness than all other edges, because all shortest paths connecting vertices of the two communities run through it. Reprinted figure with permission from Ref. [16]. 

that inter-cluster edges connect vertices with low similarity. In some cases vertices (with all their adjacent edges) or whole subgraphs may be removed, instead of single edges. Being hierarchical clustering techniques, it is customary to represent the resulting partitions by means of dendrograms.

### 5.1. The algorithm of Girvan and Newman

The most popular algorithm is that proposed by Girvan and Newman [12,54]. The method is historically important, because it marked the beginning of a new era in the field of community detection and opened this topic to physicists. Here edges are selected according to the values of measures of *edge centrality*, estimating the importance of edges according to some property or process running on the graph. The steps of the algorithm are:

1. Computation of the centrality for all edges;
2. Removal of edge with largest centrality: in case of ties with other edges, one of them is picked at random;
3. Recalculation of centralities on the running graph;
4. Iteration of the cycle from step 2.

Girvan and Newman focused on the concept of *betweenness*, which is a variable expressing the frequency of the participation of edges to a process. They considered three alternative definitions: geodesic edge betweenness, random-walk edge betweenness and current-flow edge betweenness. In the following we shall refer to them as edge betweenness, random-walk betweenness and current-flow betweenness, respectively.

Edge betweenness is the number of shortest paths between all vertex pairs that run along the edge. It is an extension to edges of the popular concept of site betweenness, introduced by Freeman in 1977 [40] and expresses the importance of edges in processes like information spreading, where information usually flows through the shortest paths. Historically edge betweenness was introduced before site betweenness in a never published technical report by Anthonisse [155]. It is intuitive that intercommunity edges have a large value of the edge betweenness, because many shortest paths connecting vertices of different communities will pass through them (Fig. 10). As in the calculation of site betweenness, if there are two or more geodesic paths with the same endpoints that run through an edge, the contribution of each of them to the betweenness of the edge must be divided by the multiplicity of the paths, as one assumes that the signal/information propagates equally along each geodesic path. The betweenness of all edges of the graph can be calculated in a time that scales as $O(mn)$, or $O(n^2)$ on a sparse graph, with techniques based on breadth-first-search [54,156,157].

In the context of information spreading, one could imagine that signals flow across random rather than geodesic paths. In this case the betweenness of an edge is given by the frequency of the passages across the edge of a random walker running on the graph (random-walk betweenness). A random walker moving from a vertex follows each adjacent edge with equal probability. A pair of vertices is chosen at random, $s$ and $t$. The walker starts at $s$ and keeps moving until it hits $t$, where it stops. One computes the probability that each edge was crossed by the walker, and averages over all possible choices for the vertices $s$ and $t$. It is meaningful to compute the *net* crossing probability, which is proportional to the number of times the walk crossed the edge in one direction. In this way one neglects back and forth passages that are accidents of the random walk and tell nothing about the centrality of the edge. Calculation of random-walk betweenness requires the inversion of an $n \times n$ matrix (once), followed by obtaining and averaging the flows for all pairs of nodes. The first task requires a time $O(n^3)$, the second $O(mn^2)$, for a total complexity $O[(m+n)n^2]$, or $O(n^3)$ for a sparse matrix. The complete calculation requires a time $O(n^3)$ on a sparse graph.

Current-flow betweenness is defined by considering the graph a resistor network, with edges having unit resistance. If a voltage difference is applied between any two vertices, each edge carries some amount of current, that can be calculated by solving Kirchoff's equations. The procedure is repeated for all possible vertex pairs: the current-flow betweenness of an edge is the average value of the current carried by the edge. It is possible to show that this measure is equivalent to random-walk betweenness, as the voltage differences and the random walks net flows across the edges satisfy the same equations [158]. Therefore, the calculation of current-flow betweenness has the same complexity $O[(m+n)n^2]$, or $O(n^3)$ for a sparse graph.

Calculating edge betweenness is much faster than current-flow or random walk betweenness [$O(n^2)$ versus $O(n^3)$ on sparse graphs]. In addition, in practical applications the Girvan–Newman algorithm with edge betweenness gives better results than adopting the other centrality measures [54]. Numerical studies show that the recalculation step 3 of Girvan–Newman algorithm is essential to detect meaningful communities. This introduces an additional factor $m$ in the

running time of the algorithm: consequently, the edge betweenness version scales as $O(m^2n)$, or $O(n^3)$ on a sparse graph. On graphs with strong community structure, that quickly break into communities, the recalculation step needs to be performed only within the connected component including the last removed edge (or the two components bridged by it if the removal of the edge splits a subgraph), as the edge betweenness of all other edges remains the same. This can help saving some computer time, although it is impossible to give estimates of the gain since it depends on the specific graph at hand. Nevertheless, the algorithm is quite slow, and applicable to sparse graphs with up to $n \sim 10\,000$ vertices, with current computational resources. In the original version of Girvan–Newman's algorithm [12], the authors had to deal with the whole hierarchy of partitions, as they had no procedure to say which partition is the best. In a successive refinement [54], they selected the partition with the largest value of modularity (see Section 3.3.2), a criterion that has been frequently used ever since. The method can be simply extended to the case of weighted graphs, by suitably generalizing the edge betweenness. The betweenness of a weighted edge equals the betweenness of the edge in the corresponding unweighted graph, divided by the weight of the edge [159]. There have been countless applications of the Girvan–Newman method: the algorithm is now integrated in well known libraries of network analysis programs.

Tyler et al. proposed a modification of the Girvan–Newman algorithm, to improve the speed of the calculation [160,161]. The gain in speed was required by the analysis of graphs of gene co-occurrences, which are too large to be analyzed by the algorithm of Girvan and Newman. Algorithms computing site/edge betweenness start from any vertex, taken as center, and compute the contribution to betweenness from all paths originating at that vertex; the procedure is then repeated for all vertices [54,156,157]. Tyler et al. proposed to calculate the contribution to edge betweenness only from a limited number of centers, chosen at random, deriving a sort of Monte Carlo estimate. Numerical tests indicate that, for each connected subgraph, it suffices to pick a number of centers growing as the logarithm of the number of vertices of the component. For a given choice of the centers, the algorithm proceeds just like that of Girvan and Newman. The stopping criterion is different, though, as it does not require the calculation of modularity on the resulting partitions, but relies on a particular definition of community. According to such a definition, a connected subgraph with $n_0$ vertices is a community if the edge betweenness of any of its edges does not exceed $n_0 - 1$. Indeed, if the subgraph consists of two parts connected by a single edge, the betweenness value of that edge would be greater than or equal to $n_0 - 1$, with the equality holding only if one of the two parts consists of a single vertex. Therefore, the condition on the betweenness of the edges would exclude such situations, although other types of cluster structure might still be compatible with it. In this way, in the method of Tyler et al., edges are removed until all connected components of the partition are "communities" in the sense explained above. The Monte Carlo sampling of the edge betweenness necessarily induces statistical errors. As a consequence, the partitions are in general different for different choices of the set of center vertices. However, the authors showed that, by repeating the calculation many times, the method gives good results on a network of gene co-occurrences [161], with a substantial gain of computer time. The technique has been also applied to a network of people corresponding via email [160]. In practical examples, only vertices lying at the boundary between communities may not be clearly classified, and be assigned sometimes to a group, sometimes to another. This is actually a nice feature of the method, as it allows one to identify overlaps between communities, as well as the degree of membership of overlapping vertices in the clusters they belong to. The algorithm of Girvan and Newman, which is deterministic, is unable to accomplish this.[9] Another fast version of the Girvan–Newman algorithm has been proposed by Rattigan et al. [145]. Here, a quick approximation of the edge betweenness values is carried out by using a *network structure index*, which consists of a set of vertex annotations combined with a distance measure [162]. Basically one divides the graph into regions and computes the distances of every vertex from each region. In this way Rattigan et al. showed that it is possible to lower the complexity of the algorithm to $O(m)$, by keeping a fair accuracy in the estimate of the edge betweenness values. This version of the Girvan–Newman algorithm gives good results on the benchmark graphs proposed by Brandes et al. [163] (see also Section 15.1), as well as on a collaboration network of actors and on a citation network.

Chen and Yuan have pointed out that counting all possible shortest paths in the calculation of the edge betweenness may lead to unbalanced partitions, with communities of very different size, and proposed to count only *non-redundant* paths, i.e. paths whose endpoints are all different from each other: the resulting betweenness yields better results than standard edge betweenness for mixed clusters on the benchmark graphs of Girvan and Newman [24]. Holme et al. have used a modified version of the algorithm in which vertices, rather than edges, are removed [164]. A centrality measure for the vertices, proportional to their site betweenness, and inversely proportional to their indegree, is chosen to identify boundary vertices, which are then iteratively removed with all their edges. This modification, applied to study the hierarchical organization of biochemical networks, is motivated by the need to account for reaction kinetic information, that simple site betweenness does not include. The indegree of a vertex is solely used because it indicates the number of substrates to a metabolic reaction involving that vertex; for the purpose of clustering the graph is considered undirected, as usual.

The algorithm of Girvan and Newman is unable to find overlapping communities, as each vertex is assigned to a single cluster. Pinney and Westhead have proposed a modification of the algorithm in which vertices can be split between communities [165]. To do that, they also compute the betweenness of all vertices of the graph. Unfortunately the values of edge and site betweenness cannot be simply compared, due to their different normalization, but the authors remarked

[9] It may happen that, at a given iteration, two or more edges of the graph have the same value of maximal betweenness. In this case one can pick any of them at random, which may lead in general to (slightly) different partitions at the end of the computation.

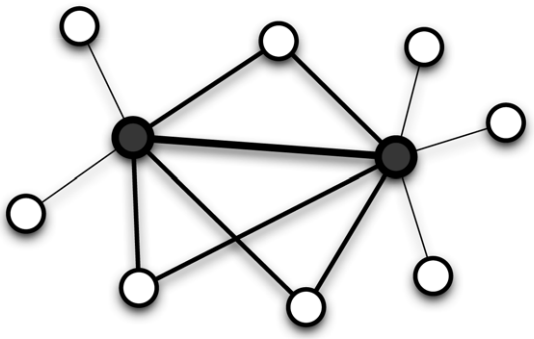

**Fig. 11.** Schematic illustration of the edge clustering coefficient introduced by Radicchi et al. [78]. The two gray vertices have five and six other neighbors, respectively. Of the five possible triangles based on the edge connecting the gray vertices, three are actually there, yielding an edge clustering coefficient $C^3 = 3/5$. Courtesy by F. Radicchi.

that the two endvertices of an inter-cluster edge should have similar betweenness values, as the shortest paths crossing one of them are likely to reach the other one as well through the edge. So they take the edge with largest betweenness and remove it only if the ratio of the betweenness values of its end-vertices is between $\alpha$ and $1/\alpha$, with $\alpha = 0.8$. Otherwise, the vertex with highest betweenness (with all its adjacent edges) is temporarily removed. When a subgraph is split by vertex or edge removal, all deleted vertices belonging to that subgraph are "copied" in each subcomponent, along with all their edges. Gregory [166] has proposed a similar approach, named CONGA (Cluster Overlap Newman–Girvan Algorithm), in which vertices are split among clusters if their site betweenness exceeds the maximum value of the betweenness of the edges. A vertex is split by assigning some of its edges to one of its duplicates, and the rest to the other. There are several possibilities to do that, Gregory proposed to go for the split that yields the maximum of a new centrality measure, called *split betweenness*, which is the number of shortest paths that would run between two parts of a vertex if the latter were split. The method has a worst-case complexity $O(m^3)$, or $O(n^3)$ on a sparse graph, like the algorithm of Girvan and Newman. The code can be found at http://www.cs.bris.ac.uk/~steve/networks/index.html.

### 5.2. Other methods

Another promising track to detect inter-cluster edges is related to the presence of cycles, i.e. closed non-intersecting paths, in the graph. Communities are characterized by a high density of edges, so it is reasonable to expect that such edges form cycles. On the contrary, edges lying between communities will hardly be part of cycles. Based on this intuitive idea, Radicchi et al. proposed a new measure, the edge clustering coefficient, such that low values of the measure are likely to correspond to intercommunity edges [78]. The edge clustering coefficient generalizes to edges the notion of clustering coefficient introduced by Watts and Strogatz for vertices [167] (Fig. 11). The clustering coefficient of a vertex is the number of triangles including the vertex divided by the number of possible triangles that can be formed (Appendix A.1). The edge clustering coefficient is defined as

$$\tilde{C}_{i,j}^{(g)} = \frac{z_{i,j}^{(g)} + 1}{s_{i,j}^{(g)}}, \tag{27}$$

where $i$ and $j$ are the extremes of the edge, $z_{i,j}^{(g)}$ the number of cycles of length $g$ built upon edge $ij$ and $s_{i,j}^{(g)}$ the possible number of cycles of length $g$ that one could build based on the existing edges of $i, j$ and their neighbors. The number of actual cycles in the numerator is augmented by 1 to enable a ranking among edges without cycles, which would all yield a coefficient $\tilde{C}_{i,j}^{(g)}$ equal to zero, independently of the degrees of the extremes $i$ and $j$ and their neighbors. Usually, cycles of length $g = 3$ (triangles) or 4 are considered. The measure is (anti)correlated with edge betweenness: edges with low edge clustering coefficient usually have high betweenness and vice versa, although the correlation is not perfect. The method works as the algorithm by Girvan and Newman. At each iteration, the edge with smallest clustering coefficient is removed, the measure is recalculated again, and so on. If the removal of an edge leads to a split of a subgraph in two parts, the split is accepted only if both clusters are LS-sets ("strong") or "weak" communities (see Section 3.2.2). The verification of the community condition on the clusters is performed on the full adjacency matrix of the initial graph. If the condition were satisfied only for one of the two clusters, the initial subgraph may be a random graph, as it can be easily seen that by cutting a random graph á la Erdös and Rényi in two parts, the larger of them is a strong (or weak) community with very high probability, whereas the smaller part is not. Enforcing the community condition on both clusters, it is more likely that the subgraph to be split indeed has a cluster structure. Therefore, the algorithm stops when all clusters produced by the edge removals are communities in the strong or weak sense, and further splits would violate this condition. The authors suggested to use the same stopping criterion for the algorithm of Girvan and Newman, to get structurally well-defined clusters. Since the edge clustering coefficient is

a local measure, involving at most an extended neighborhood of the edge, it can be calculated very quickly. The running time of the algorithm to completion is $O(m^4/n^2)$, or $O(n^2)$ on a sparse graph, if $g$ is small, so it is much shorter than the running time of the Girvan–Newman method. The recalculation step becomes slow if $g$ is not so small, as in this case the number of edges whose coefficient needs to be recalculated may reach a sizeable fraction of the edges of the graph; likewise, counting the number of cycles based on one edge becomes lengthier. If $g \sim 2d$, where $d$ is the diameter of the graph (which is usually a small number for real networks), the cycles span the whole graph and the measure becomes global and no longer local. The computational complexity in this case exceeds that of the algorithm of Girvan and Newman, but it can come close to it for practical purposes even at lower values of $g$. So, by tuning $g$ one can smoothly interpolate between a local and a global centrality measure. The software of the algorithm can be found in http://filrad.homelinux.org/Data/. In a successive paper [168] the authors extended the method to the case of weighted networks, by modifying the edge clustering coefficient of Eq. (27), in that the number of cycles $z_{i,j}^{(g)}$ is multiplied by the weight of the edge $ij$. The definitions of strong and weak communities can be trivially extended to weighted graphs by replacing the internal/external degrees of the vertices/clusters with the corresponding strengths. More recently, the method has been extended to bipartite networks [169], where only cycles of even length are possible ($g = 4, 6, 8$, etc.). The algorithm by Radicchi et al. may give poor results when the graph has few cycles, as it happens in some social and many non-social networks. In this case, in fact, the edge clustering coefficient is small and fairly similar for most edges, and the algorithm may fail to identify the bridges between communities.

An alternative measure of centrality for edges is information centrality. It is based on the concept of efficiency [170], which estimates how easily information travels on a graph according to the length of shortest paths between vertices. The efficiency of a network is defined as the average of the inverse distances between all pairs of vertices. If the vertices are "close" to each other, the efficiency is high. The information centrality of an edge is the relative variation of the efficiency of the graph if the edge is removed. In the algorithm by Fortunato et al. [171], edges are removed according to decreasing values of information centrality. The method is analogous to that of Girvan and Newman. Computing the information centrality of an edge requires the calculation of the distances between all pairs of vertices, which can be done with breadth-first-search in a time $O(mn)$. So, in order to compute the information centrality of all edges one requires a time $O(m^2n)$. At this point one removes the edge with the largest value of information centrality and recalculates the information centrality of all remaining edges with respect to the running graph. Since the procedure is iterated until there are no more edges in the network, the final complexity is $O(m^3n)$, or $O(n^4)$ on a sparse graph. The partition with the largest value of modularity is chosen as most representative of the community structure of the graph. The method is much slower than the algorithm of Girvan and Newman. Partitions obtained with both techniques are rather consistent, mainly because information centrality has a strong correlation with edge betweenness. The algorithm by Fortunato et al. gives better results when communities are mixed, i.e. with a high degree of interconnectedness, but it tends to isolate leaf vertices and small loosely bound subgraphs.

A measure of vertex centrality based on loops, similar to the clustering coefficient by Watts and Strogatz [167], has been introduced by Vragovi̇c and Louis [172]. The idea is that neighbors of a vertex well inside a community are "close" to each other, even in the absence of the vertex, due to the high density of intra-cluster edges. Suppose that $j$ and $k$ are neighbors of a vertex $i$: $d_{jk/i}$ is the length of a shortest path between $j$ and $k$, if $i$ is removed from the graph. Naturally, the existence of alternative paths to $j - i - k$ implies the existence of loops in the graph. Vragovi̇c and Louis defined the *loop coefficient* of $i$ as the average of $1/d_{jk/i}$ over all pairs of neighbors of $i$, somewhat reminding of the concept of information centrality used in the method by Fortunato et al. [171]. High values of the loop coefficient are likely to identify core vertices of communities, whereas low values correspond to vertices lying at the boundary between communities. Clusters are built around the vertices with highest values of the loop coefficient. The method has time complexity $O(nm)$; its results are not so accurate, as compared to popular clustering techniques.

## 6. Modularity-based methods

Newman–Girvan modularity $Q$ (Section 3.3.2), originally introduced to define a stopping criterion for the algorithm of Girvan and Newman, has rapidly become an essential element of many clustering methods. Modularity is by far the most used and best known quality function. It represented one of the first attempts to achieve a first principle understanding of the clustering problem, and it embeds in its compact form all essential ingredients and questions, from the definition of community, to the choice of a null model, to the expression of the "strength" of communities and partitions. In this section we shall focus on all clustering techniques that require modularity, directly and/or indirectly. We will examine fast techniques that can be used on large graphs, but which do not find good optima for the measure [173–184]; more accurate methods, which are computationally demanding [185–187]; algorithms giving a good tradeoff between high accuracy and low complexity [188–192]. We shall also point out other properties of modularity, discuss some extensions/modifications of it, as well as highlight its limits.

### 6.1. Modularity optimization

By assumption, high values of modularity indicate good partitions.[10] So, the partition corresponding to its maximum value on a given graph should be the best, or at least a very good one. This is the main motivation for modularity

[10] This is not true in general, as we shall discuss in Section 6.3.

maximization, by far the most popular class of methods to detect communities in graphs. An exhaustive optimization of $Q$ is impossible, due to the huge number of ways in which it is possible to partition a graph, even when the latter is small. Besides, the true maximum is out of reach, as it has been recently proved that modularity optimization is an NP-complete problem [193], so it is probably impossible to find the solution in a time growing polynomially with the size of the graph. However, there are currently several algorithms able to find fairly good approximations of the modularity maximum in a reasonable time.

*6.1.1. Greedy techniques*

The first algorithm devised to maximize modularity was a greedy method of Newman [173]. It is an agglomerative hierarchical clustering method, where groups of vertices are successively joined to form larger communities such that modularity increases after the merging. One starts from $n$ clusters, each containing a single vertex. Edges are not initially present, they are added one by one during the procedure. However, the modularity of partitions explored during the procedure is always calculated from the full topology of the graph, as we want to find the modularity maximum on the space of partitions of the full graph. Adding a first edge to the set of disconnected vertices reduces the number of groups from $n$ to $n - 1$, so it delivers a new partition of the graph. The edge is chosen such that this partition gives the maximum increase (minimum decrease) of modularity with respect to the previous configuration. All other edges are added based on the same principle. If the insertion of an edge does not change the partition, i.e. the edge is internal to one of the clusters previously formed, modularity stays the same. The number of partitions found during the procedure is $n$, each with a different number of clusters, from $n$ to 1. The largest value of modularity in this subset of partitions is the approximation of the modularity maximum given by the algorithm. At each iteration step, one needs to compute the variation $\Delta Q$ of modularity given by the merger of any two communities of the running partition, so that one can choose the best merger. However, merging communities between which there are no edges can never lead to an increase of $Q$, so one has to check only the pairs of communities which are connected by edges, of which there cannot be more than $m$. Since the calculation of each $\Delta Q$ can be done in constant time, this part of the calculation requires a time $O(m)$. After deciding which communities are to be merged, one needs to update the matrix $e_{ij}$ expressing the fraction of edges between clusters $i$ and $j$ of the running partition (necessary to compute $Q$), which can be done in a worst-case time $O(n)$. Since the algorithm requires $n - 1$ iterations (community mergers) to run to completion, its complexity is $O((m + n)n)$, or $O(n^2)$ on a sparse graph, so it enables one to perform a clustering analysis on much larger networks than the algorithm of Girvan and Newman (up to an order of 100 000 vertices with current computers). In a later paper [174], Clauset et al. pointed out that the update of the matrix $e_{ij}$ in Newman's algorithm involves a large number of useless operations, due to the sparsity of the adjacency matrix. This operation can be performed more efficiently by using data structures for sparse matrices, like *max-heaps*, which rearrange the data in the form of binary trees. Clauset et al. maintained the matrix of modularity variations $\Delta Q_{ij}$, which is also sparse, a max-heap containing the largest elements of each row of the matrix $\Delta Q_{ij}$ as well as the labels of the corresponding communities, and a simple array whose elements are the sums of the elements of each row of the old matrix $e_{ij}$. The optimization of modularity can be carried out using these three data structures, whose update is much quicker than in Newman's technique. The complexity of the algorithm is $O(md \log n)$, where $d$ is the depth of the dendrogram describing the successive partitions found during the execution of the algorithm, which grows as $\log n$ for graphs with a strong hierarchical structure. For those graphs, the running time of the method is then $O(n \log^2 n)$, which allows one to analyze the community structure of very large graphs, up to $10^6$ vertices. The greedy optimization of Clauset et al. is currently one of the few algorithms that can be used to estimate the modularity maximum on such large graphs. The code can be freely downloaded from http://cs.unm.edu/~aaron/research/fastmodularity.htm.

This greedy optimization of modularity tends to form quickly large communities at the expenses of small ones, which often yields poor values of the modularity maxima. Danon et al. suggested to normalize the modularity variation $\Delta Q$ produced by the merger of two communities by the fraction of edges incident to one of the two communities, in order to favor small clusters [175]. This trick leads to better modularity optima as compared to the original recipe of Newman, especially when communities are very different in size. Wakita and Tsurumi [178] have noticed that, due to the bias towards large communities, the fast algorithm by Clauset et al. is inefficient, because it yields very unbalanced dendrograms, for which the relation $d \sim \log n$ does not hold, and as a consequence the method often runs at its worst-case complexity. To improve the situation they proposed a modification in which, at each step, one seeks the community merger delivering the largest value of the product of the modularity variation $\Delta Q$ times a factor (*consolidation ratio*), that peaks for communities of equal size. In this way there is a tradeoff between the gain in modularity and the balance of the communities to merge, with a big gain in the speed of the procedure, that enables the analysis of systems with up to $10^7$ vertices. Interestingly, this modification often leads to better modularity maxima than those found with the version of Clauset et al., at least on large social networks. The code can be found at http://www.is.titech.ac.jp/~wakita/en/software/community-analysis-software/. Another trick to avoid the formation of large communities was proposed by Schuetz and Caflisch and consists in allowing for the merger of more community pairs, instead of one, at each iteration [180,181]. This generates several "centers" around which communities are formed, which grow simultaneously so that a condensation into a few large clusters is unlikely. This modified version of the greedy algorithm is combined with a simple refinement procedure in which single vertices are moved to the neighboring community that yields the maximum increase of modularity. The method has the same complexity of the fast optimization by Clauset et al., but comes closer to the modularity maximum. The software is available at http://www.biochem-caflisch.uzh.ch/public/5/network-clusterization-algorithm.html. The accuracy

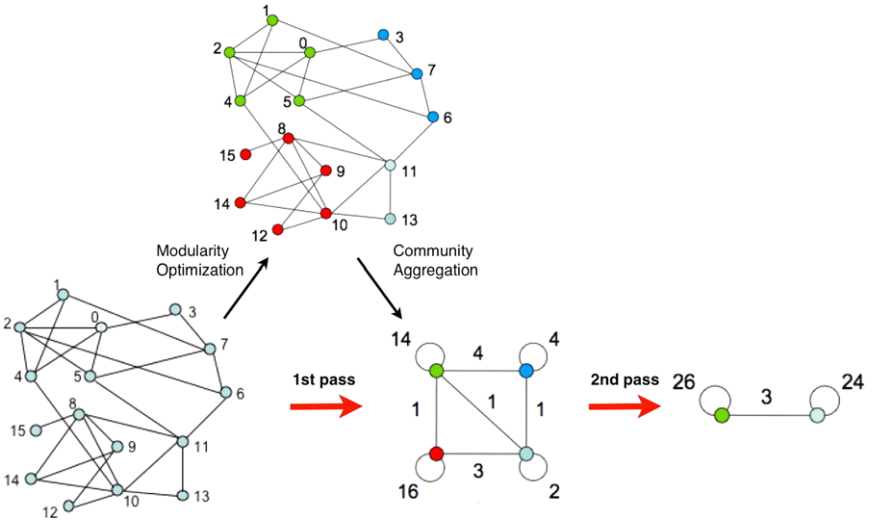


**Fig. 12.** Hierarchical optimization of modularity by Blondel et al. [179]. The diagram shows two iterations of the method, starting from the graph on the left. Each iteration consists of a step, in which every vertex is assigned to the (local) cluster that produces the largest modularity increase, followed by a successive transformation of the clusters into vertices of a smaller (weighted) graph, representing the next higher hierarchical level. Reprinted figure with permission from Ref. [179].


of the greedy optimization can be significantly improved if the hierarchical agglomeration is started from some reasonable intermediate configuration, rather than from the individual vertices [176,177]. Xiang et al. suggested to start from a configuration obtained by merging the original isolated vertices into larger subgraphs, according to the values of a measure of topological similarity between subgraphs [183]. A similar approach has been described by Ye et al. [194]: here the initial partition is such that no single vertex can be moved from its cluster to another without decreasing $Q$. Higher-quality modularities can be also achieved by applying refinement strategies based on local search at various steps of the greedy agglomeration [182]. Such refinement procedures are similar to the technique proposed by Newman to improve the results of his spectral optimization of modularity ([190] and Section 6.1.4). Another good strategy consists of alternating greedy optimization with stochastic perturbations of the partitions [184].

A different greedy approach has been introduced by Blondel et al. [179], for the general case of weighted graphs. Initially, all vertices of the graph are put in different communities. The first step consists of a sequential sweep over all vertices. Given a vertex $i$, one computes the gain in weighted modularity (Eq. (35)) coming from putting $i$ in the community of its neighbor $j$ and picks the community of the neighbor that yields the largest increase of $Q$, as long as it is positive. At the end of the sweep, one obtains the first level partition. In the second step communities are replaced by supervertices, and two supervertices are connected if there is at least an edge between vertices of the corresponding communities. In this case, the weight of the edge between the supervertices is the sum of the weights of the edges between the represented communities at the lower level. The two steps of the algorithm are then repeated, yielding new hierarchical levels and supergraphs (Fig. 12). We remark that modularity is always computed from the initial graph topology: operating on supergraphs enables one to consider the variations of modularity for partitions of the original graph after merging and/or splitting of groups of vertices. Therefore, at some iteration, modularity cannot increase any more, and the algorithm stops. The technique is more limited by storage demands than by computational time. The latter grows like $O(m)$, so the algorithm is extremely fast and graphs with up to $10^9$ edges can be analyzed in a reasonable time on current computational resources. The software can be found at http://findcommunities.googlepages.com/. The modularity maxima found by the method are better than those found with the greedy techniques by Clauset et al. [174] and Wakita and Tsurumi [178]. However, closing communities within the immediate neighborhood of vertices may be inaccurate and yield spurious partitions in practical cases. So, it is not clear whether some of the intermediate partitions could correspond to meaningful hierarchical levels of the graph. Moreover, the results of the algorithm depend on the order of the sequential sweep over the vertices.

We conclude by stressing that, despite the improvements and refinements of the last years, the accuracy of greedy optimization is not that good, as compared with other techniques.

### *6.1.2. Simulated annealing*

Simulated annealing [195] is a probabilistic procedure for global optimization used in different fields and problems. It consists of performing an exploration of the space of possible states, looking for the global optimum of a function $F$, say its maximum. Transitions from one state to another occur with probability 1 if $F$ increases after the change, otherwise with a probability $\exp(\beta \Delta F)$, where $\Delta F$ is the decrease of the function and $\beta$ is an index of stochastic noise, a sort of inverse temperature, which increases after each iteration. The noise reduces the risk that the system gets trapped in local optima.

At some stage, the system converges to a stable state, which can be an arbitrarily good approximation of the maximum of $F$, depending on how many states were explored and how slowly $\beta$ is varied. Simulated annealing was first employed for modularity optimization by Guimerà et al. [185]. Its standard implementation [27] combines two types of "move": local moves, where a single vertex is shifted from one cluster to another, taken at random; global moves, consisting of mergers and splits of communities. Splits can be carried out in several distinct ways. The best performance is achieved if one optimizes the modularity of a bipartition of the cluster, taken as an isolated graph. This is done again with simulated annealing, where one considers only individual vertex movements, and the temperature is decreased until it reaches the running value for the global optimization. Global moves reduce the risk of getting trapped in local minima and they have proven to lead to much better optima than using simply local moves [186,187]. In practical applications, one typically combines $n^2$ local moves with $n$ global ones in one iteration. The method can potentially come very close to the true modularity maximum, but it is slow. The actual complexity cannot be estimated, as it heavily depends on the parameters chosen for the optimization (initial temperature, cooling factor), not only on the graph size. Simulated annealing can be used for small graphs, with up to about $10^4$ vertices.

*6.1.3. Extremal optimization*

Extremal optimization (EO) is a heuristic search procedure proposed by Boettcher and Percus [196], in order to achieve an accuracy comparable with simulated annealing, but with a substantial gain in computer time. It is based on the optimization of local variables, expressing the contribution of each unit of the system to the global function at study. This technique was used for modularity optimization by Duch and Arenas [188]. Modularity can be indeed written as a sum over the vertices: the local modularity of a vertex is the value of the corresponding term in this sum. A fitness measure for each vertex is obtained by dividing the local modularity of the vertex by its degree, as in this case the measure does not depend on the degree of the vertex and is suitably normalized. One starts from a random partition of the graph in two groups with the same number of vertices. At each iteration, the vertex with the lowest fitness is shifted to the other cluster. The move changes the partition, so the local fitnesses of many vertices need to be recalculated. The process continues until the global modularity $Q$ cannot be improved any more by the procedure. This technique reminds one of the Kernighan–Lin [124] algorithm for graph partitioning (Section 4.1), but here the sizes of the communities are determined by the process itself, whereas in graph partitioning they are fixed from the beginning. After the bipartition, each cluster is considered as a graph on its own and the procedure is repeated, as long as $Q$ increases for the partitions found. The procedure, as described, proceeds deterministically from the given initial partition, as one shifts systematically the vertex with lowest fitness, and is likely to get trapped in local optima. Better results can be obtained if one introduces a probabilistic selection, in which vertices are ranked based on their fitness values and one picks the vertex of rank $q$ with the probability $P(q) \sim q^{-\tau}$ ($\tau$-EO, [196]). The algorithm finds very good estimates of the modularity maximum, and performs very well on the benchmark of Girvan and Newman [12] (Section 15.1). Ranking the fitness values has a cost $O(n \log n)$, which can be reduced to $O(n)$ if heap data structures are used. Choosing the vertex to be shifted can be done with a binary search, which amounts to an additional factor $O(\log n)$. Finally, the number of steps needed to verify whether the running modularity maximum can be improved or not is also $O(n)$. The total complexity of the method is then $O(n^2 \log n)$. We conclude that EO represents a good tradeoff between accuracy and speed, although the use of recursive bisectioning may lead to poor results on large networks with many communities.

*6.1.4. Spectral optimization*

Modularity can be optimized using the eigenvalues and eigenvectors of a special matrix, the modularity matrix $\mathbf{B}$, whose elements are

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{28}$$

Here the notation is the same used in Eq. (14). Let $\mathbf{s}$ be the vector representing any partition of the graph in two clusters $\mathcal{A}$ and $\mathcal{B}$: $s_i = +1$ if vertex $i$ belongs to $\mathcal{A}$, $s_i = -1$ if $i$ belongs to $\mathcal{B}$. Modularity can be written as

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j) = \frac{1}{4m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) (s_i s_j + 1) \\ &= \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}. \end{aligned} \tag{29}$$

The last expression indicates standard matrix products. The vector $\mathbf{s}$ can be decomposed on the basis of eigenvectors $\mathbf{u}_i$ ($i = 1, \ldots, n$) of the modularity matrix $\mathbf{B}$: $\mathbf{s} = \sum_i a_i \mathbf{u}_i$, with $a_i = \mathbf{u}_i^T \cdot \mathbf{s}$. By plugging this expression of $\mathbf{s}$ into Eq. (29) one finally gets

$$Q = \frac{1}{4m} \sum_i a_i \mathbf{u}_i^T \mathbf{B} \sum_j a_j \mathbf{u}_j = \frac{1}{4m} \sum_{i=1}^{n} (\mathbf{u}_i^T \cdot \mathbf{s})^2 \beta_i, \tag{30}$$

where $\beta_i$ is the eigenvalue of $\mathbf{B}$ corresponding to the eigenvector $\mathbf{u}_i$. Eq. (30) is analogous to Eq. (19) for the cut size of the graph partitioning problem. This suggests that one can optimize modularity on bipartitions via spectral bisection

(Section 4.1), by replacing the Laplacian matrix with the modularity matrix [190,118]. Like the Laplacian matrix, **B** has always the trivial eigenvector $(1, 1, \ldots, 1)$ with eigenvalue zero, because the sum of the elements of each row/column of the matrix vanishes. From Eq. (30) we see that, if **B** has no positive eigenvalues, the maximum coincides with the trivial partition consisting of the graph as a single cluster (for which $Q = 0$), i.e. it has no community structure. Otherwise, one has to look for the eigenvector of $B$ with largest (positive) eigenvalue, $\mathbf{u}_1$, and group the vertices according to the signs of the components of $\mathbf{u}_1$, just like in Section 4.1. Here, however, one does not need to specify the sizes of the two groups: the vertices with positive components are all in one group, the others in the other group. If, for example, the component of $\mathbf{u}_1$ corresponding to vertex $i$ is positive, but we set $s_i = -1$, the modularity is lower than by setting $s_i = +1$. The values of the components of $\mathbf{u}_1$ are also informative, as they indicate the level of the participation of the vertices to their communities. In particular, components whose values are close to zero lie at the border between the two clusters and can be considered as belonging to both of them. The result obtained from the spectral bipartition can be further improved by shifting single vertices from one community to the other, such to have the highest increase (or lowest decrease) of the modularity of the resulting graph partition. This refinement technique, similar to the Kernighan–Lin algorithm (Section 4.1), can be also applied to improve the results of other optimization techniques (e.g. greedy algorithms, extremal optimization, etc.). The procedure is repeated for each of the clusters separately, and the number of communities increases as long as modularity does. At variance with graph partitioning, where one needs to fix the number of clusters and their size beforehand, here there is a clear-cut stopping criterion, represented by the fact that cluster subdivisions are admitted only if they lead to a modularity increase. We stress that modularity needs to be always computed from the full adjacency matrix of the original graph.[11] The drawback of the method is similar as for spectral bisection, i.e. the algorithm gives the best results for bisections, whereas it is less accurate when the number of communities is larger than two. Recently, Sun et al. [198] have added a step after each bipartition of a cluster, in that single vertices can be moved from one cluster to another and even form the seeds of new clusters. We remark that the procedure is different from the Kernighan–Lin-like refining steps, as here the number of clusters can change. This variant, which does not increase the complexity of the original spectral optimization, leads to better modularity maxima. Moreover, one does not need to stick to bisectioning, if other eigenvectors with positive eigenvalues of the modularity matrix are used. Given the first $p$ eigenvectors, one can construct $n$ $p$-dimensional vectors, each corresponding to a vertex, just like in spectral partitioning (Section 4.4). The components of the vector of vertex $i$ are proportional to the $p$ entries of the eigenvectors in position $i$. Then one can define *community vectors*, by summing the vectors of vertices in the same community. It is possible to show that, if the vectors of two communities form an angle larger that $\pi/2$, keeping the communities separate yields larger modularity than if they are merged (Fig. 13). In this way, in a $p$-dimensional space the modularity maximum corresponds to a partition in at most $p + 1$ clusters. Community vectors were used by Wang et al. to obtain high-modularity partitions into a number of communities smaller than a given maximum [199]. In particular, if one takes the eigenvectors corresponding to the two largest eigenvalues, one can obtain a split of the graph in three clusters: in a recent work, Richardson et al. presented a fast technique to obtain graph tripartitions with large modularity along these lines [197]. The eigenvectors with the most negative eigenvalues can also be used to extract useful information, like the presence of a possible multipartite structure of the graph, as they give the most relevant contribution to the modularity minimum.

The spectral optimization of modularity is quite fast. The leading eigenvector of the modularity matrix can be computed with the power method, by repeatedly multiplying **B** by an arbitrary vector (not orthogonal to $\mathbf{u}_1$). The number of required iterations to reach convergence is $O(n)$. Each multiplication seems to require a time $O(n^2)$, as **B** is a complete matrix, but the peculiar form of **B** allows for a much quicker calculation, taking time $O(m + n)$. So, a graph bipartition requires a time $O[n(m+n)]$, or $O(n^2)$ on a sparse graph. To find the modularity optimum one needs a number of subsequent bipartitions that equals the depth $d$ of the resulting hierarchical tree. In the worst-case scenario, $d = O(n)$, but in practical cases the procedure usually stops much before reaching the leaves of the dendrogram, so one could go for the average value $\langle d \rangle \sim \log n$, for a total complexity of $O(n^2 \log n)$. The algorithm is faster than extremal optimization and it is also slightly more accurate, especially for large graphs. The modularity matrix and the corresponding spectral optimization can be trivially extended to weighted graphs.

A different spectral approach had been previously proposed by White and Smyth [189]. Let **W** indicate the weighted adjacency matrix of a graph $\mathcal{G}$. A partition of $\mathcal{G}$ in $k$ clusters can be described through an $n \times k$ assignment matrix **X**, where $x_{ic} = 1$ if vertex $i$ belongs to cluster $c$, otherwise $x_{ic} = 0$. It can be easily shown that, up to a multiplicative constant, modularity can be rewritten in terms of the matrix **X** as

$$Q \propto \mathrm{tr}[\mathbf{X}^T(\boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{D}})\mathbf{X}] = -\mathrm{tr}[\mathbf{X}^T\mathbf{L_Q}\mathbf{X}], \tag{31}$$

where $\boldsymbol{\mathcal{W}}$ is a diagonal matrix with identical elements, equal to the sum of all edge weights, and the entries of $\boldsymbol{\mathcal{D}}$ are $\boldsymbol{\mathcal{D}}_{ij} = k_i k_j$, where $k_i$ is the degree of vertex $i$. The matrix $\mathbf{L}_Q = \boldsymbol{\mathcal{D}} - \boldsymbol{\mathcal{W}}$ is called the $Q$-Laplacian. Finding the assignment matrix **X** that maximizes $Q$ is an *NP*-complete problem, but one can get a good approximation by relaxing the constraint that the elements of **X** have to be discrete. By doing so $Q$ becomes a sort of continuous functional of **X** and one can determine

[11] Richardson et al. [197] have actually shown that if one instead seeks the optimization of modularity for each cluster, taken as an independent graph, the combination of spectral bisectioning and the post-processing technique may yield better results for the final modularity optima.

further increases so they will be even more likely to be chosen in the future. In the asymptotic limit of infinite number of vertices, this *rich-gets-richer* strategy generates a graph with a degree distribution characterized by a power-law tail with exponent 3. In Fig. 42 (bottom) we show an example of Barabási–Albert (BA) graph. The clustering coefficient of a BA graph decays with the size of the graph, and it is much lower than in real networks. Moreover, the power law decays of the degree distributions observed in real networks are characterized by a range of exponents' values (usually between 2 and 3), whereas the BA model yields a fixed value. However, many refinements of the BA model as well as plenty of different models have been later introduced to account more closely for the features observed in real systems (for details see [5–10]).

## References


[1] L. Euler, Solutio problematis ad geometriam situs pertinentis, Commentarii Academiae Petropolitanae 8 (1736) 128–140.
[2] B. Bollobas, Modern Graph Theory, Springer Verlag, New York, USA, 1998.
[3] S. Wasserman, K. Faust, Social Network Analysis, Cambridge University Press, Cambridge, UK, 1994.
[4] J. Scott, Social Network Analysis: A Handbook, SAGE Publications, London, UK, 2000.
[5] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. 74 (1) (2002) 47–97.
[6] J.F.F. Mendes, S.N. Dorogovtsev, Evolution of Networks: From Biological Nets to the Internet and WWW, Oxford University Press, Oxford, UK, 2003.
[7] M.E.J. Newman, The structure and function of complex networks, SIAM Rev. 45 (2) (2003) 167–256.
[8] R. Pastor-Satorras, A. Vespignani, Evolution and Structure of the Internet: A Statistical Physics Approach, Cambridge University Press, New York, NY, USA, 2004.
[9] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.U. Hwang, Complex networks: Structure and dynamics, Phys. Rep. 424 (4-5) (2006) 175–308.
[10] A. Barrat, M. Barthélemy, A. Vespignani, Dynamical Processes on Complex Networks, Cambridge University Press, Cambridge, UK, 2008.
[11] P. Erdös, A. Rényi, On random graphs. I., Publ. Math. Debrecen 6 (1959) 290–297.
[12] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (12) (2002) 7821–7826.
[13] M.E.J. Newman, Detecting community structure in networks, Eur. Phys. J. B 38 (2004) 321–330.
[14] L. Danon, J. Duch, A. Arenas, A. Díaz-Guilera, in: C. G., V. A. (Eds.), Large Scale Structure and Dynamics of Complex Networks: From Information Technology to Finance and Natural Science, World Scientific, Singapore, 2007, pp. 93–114.
[15] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (1) (2007) 27–64.
[16] S. Fortunato, C. Castellano, Community structure in graphs, in: R.A. Meyers (Ed.), Encyclopedia of Complexity and Systems Science, vol. 1, Springer, Berlin, Germany, 2009, eprint arXiv:0712.2716.
[17] M.A. Porter, J.-P. Onnela, P.J. Mucha, Communities in networks, Notices of the American Mathematical Society 56 (2009) 1082–1166.
[18] J.S. Coleman, An Introduction to Mathematical Sociology, Collier-Macmillan, London, UK, 1964.
[19] L.C. Freeman, The Development of Social Network Analysis: A Study in the Sociology of Science, BookSurge Publishing, 2004.
[20] C.P. Kottak, Cultural Anthropology, McGraw-Hill, New York, USA, 2004.
[21] J. Moody, D.R. White, Structural cohesion and embeddedness: A hierarchical concept of social groups, Am. Sociol. Rev. 68 (1) (2003) 103–127.
[22] A.W. Rives, T. Galitski, Modular organization of cellular networks, Proc. Natl. Acad. Sci. USA 100 (3) (2003) 1128–1133.
[23] V. Spirin, L.A. Mirny, Protein complexes and functional modules in molecular networks, Proc. Natl. Acad. Sci. USA 100 (21) (2003) 12123–12128.
[24] J. Chen, B. Yuan, Detecting functional modules in the yeast protein–protein interaction network, Bioinformatics 22 (18) (2006) 2283–2290.
[25] G.W. Flake, S. Lawrence, C. Lee Giles, F.M. Coetzee, Self-organization and identification of web communities, IEEE Computer 35 (2002) 66–71.
[26] Y. Dourisboure, F. Geraci, M. Pellegrini, Extraction and classification of dense communities in the web, in: WWW '07: Proceedings of the 16th International Conference on the World Wide Web, ACM, New York, NY, USA, 2007, pp. 461–470.
[27] R. Guimerà, L.A.N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (2005) 895–900.
[28] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814–818.
[29] S.L. Pimm, The structure of food webs, Theoret. Popul. Biol. 16 (1979) 144–158.
[30] A.E. Krause, K.A. Frank, D.M. Mason, R.E. Ulanowicz, W.W. Taylor, Compartments revealed in food-web structure, Nature 426 (2003) 282–285.
[31] B. Krishnamurthy, J. Wang, On network-aware clustering of web clients, SIGCOMM Comput. Commun. Rev. 30 (4) (2000) 97–110.
[32] K.P. Reddy, M. Kitsuregawa, P. Sreekanth, S.S. Rao, A graph based approach to extract a neighborhood customer community for collaborative filtering, in: DNIS '02: Proceedings of the Second International Workshop on Databases in Networked Information Systems, Springer-Verlag, London, UK, 2002, pp. 188–200.
[33] R. Agrawal, H.V. Jagadish, Algorithms for searching massive graphs, Knowl. Data Eng. 6 (2) (1994) 225–238.
[34] A.Y. Wu, M. Garland, J. Han, Mining scale-free networks using geodesic clustering, in: KDD '04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, NY, USA, 2004, pp. 719–724.
[35] C.E. Perkins, Ad Hoc Networking, 1st edition, Addison-Wesley, Reading, USA, 2001.
[36] M. Steenstrup, Cluster-Based Networks, Addison Wesley, Reading, USA, 2001, (Chapter 4).
[37] P. Csermely, Creative elements: Network-based predictions of active centres in proteins and cellular and social networks, Trends Biochem. Sci. 33 (12) (2008) 569–576.
[38] M. Granovetter, The strength of weak ties, Am. J. Sociol. 78 (1973) 1360–1380.
[39] R.S. Burt, Positions in networks, Soc. Forces 55 (1976) 93–122.
[40] L.C. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (1977) 35–41.
[41] D. Gfeller, P. De Los Rios, Spectral coarse graining of complex networks, Phys. Rev. Lett. 99 (3) (2007) 038701.
[42] D. Gfeller, P. De Los Rios, Spectral coarse graining and synchronization in oscillator networks, Phys. Rev. Lett. 100 (17) (2008) 174104.
[43] P. Pollner, G. Palla, T. Vicsek, Preferential attachment of communities: The same principle, but a higher level, Europhys. Lett. 73 (2006) 478–484.
[44] H. Simon, The architecture of complexity, Proc. Am. Phil. Soc. 106 (6) (1962) 467–482.
[45] R.S. Weiss, E. Jacobson, A method for the analysis of the structure of complex organizations, Am. Sociol. Rev. 20 (1955) 661–668.
[46] S.A. Rice, The identification of blocs in small political bodies, Am. Polit. Sci. Rev. 21 (1927) 619–627.
[47] G.C. Homans, The Human Groups, Harcourt, Brace & Co, New York, 1950.
[48] L. Donetti, M.A. Muñoz, Detecting network communities: a new systematic and efficient algorithm, J. Stat. Mech. P10012 (2004).
[49] A. Arenas, A. Fernández, S. Gómez, Analysis of the structure of complex networks at different resolution levels, New J. Phys. 10 (5) (2008) 053039.
[50] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (1977) 452–473.
[51] D. Lusseau, The emergent properties of a dolphin social network, Proc. R. Soc. London B 270 (2003) S186–S188.
[52] A. Zhang, Protein Interaction Networks, Cambridge University Press, Cambridge, UK, 2009.
[53] P.F. Jonsson, T. Cavanna, D. Zicha, P.A. Bates, Cluster analysis of networks generated through homology: Automatic identification of important protein communities involved in cancer metastasis, BMC Bioinf. 7 (2006) 2.
[54] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[55] R. Albert, H. Jeong, A.-L. Barabási, Internet: Diameter of the world-wide web, Nature 401 (1999) 130–131.
[56] S. Brin, L.E. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN 30 (1998) 107–117.
[57] E.A. Leicht, M.E.J. Newman, Community structure in directed networks, Phys. Rev. Lett. 100 (11) (2008) 118703.

[58] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. USA 105 (2008) 1118–1123.
[59] D.L. Nelson, C.L. McEvoy, T.A. Schreiber, The university of south florida word association, rhyme, and word fragment norms (1998).
[60] M.J. Barber, Modularity and community detection in bipartite networks, Phys. Rev. E 76 (6) (2007) 066102.
[61] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.
[62] A. Davis, B.B. Gardner, M.R. Gardner, Deep South, University of Chicago Press, Chicago, USA, 1941.
[63] G. Gan, C. Ma, J. Wu, Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability), Society for Industrial and Applied Mathematics, Philadelphia, USA, 2007.
[64] M.R. Garey, D.S. Johnson, Computers and Intractability : A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, USA, 1990.
[65] C.M. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, USA, 1994.
[66] S. Mancoridis, B.S. Mitchell, C. Rorres, Y. Chen, E.R. Gansner, Using automatic clustering to produce high-level system organizations of source code, in: IWPC '98: Proceedings of the 6th International Workshop on Program Comprehension, IEEE Computer Society, Washington, DC, USA, 1998.
[67] R.D. Luce, A.D. Perry, A method of matrix analysis of group structure, Psychometrika 14 (2) (1949) 95–116.
[68] I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in: D.-Z. Du, P. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Norwell, USA, 1999, pp. 1–74.
[69] C. Bron, J. Kerbosch, Finding all cliques on an undirected graph, Commun. ACM 16 (1973) 575–577.
[70] R.D. Luce, Connectivity and generalized cliques in sociometric group structure, Psychometrika 15 (2) (1950) 169–190.
[71] R.D. Alba, A graph-theoretic definition of a sociometric clique, J. Math. Sociol. 3 (1973) 113–126.
[72] R.J. Mokken, Cliques, clubs and clans, Qual. Quant. 13 (2) (1979) 161–173.
[73] S.B. Seidman, B.L. Foster, A graph theoretic generalization of the clique concept, J. Math. Sociol. 6 (1978) 139–154.
[74] S.B. Seidman, Network structure and minimum degree, Soc. Netw. 5 (1983) 269–287.
[75] V. Batagelj, M. Zaversnik, An o(m) algorithm for cores decomposition of networks. arXiv:cs.DS/0310049.
[76] H. Matsuda, T. Ishihara, A. Hashimoto, Classifying molecular sequences using a linkage graph with their pairwise similarities, Theoret. Comput. Sci. 210 (1999) 305–325.
[77] F. Luccio, M. Sami, IEEE Trans. Circuit Th. CT 16 (1969) 184–188.
[78] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proc. Natl. Acad. Sci. USA 101 (2004) 2658–2663.
[79] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, Y. Fan, Comparative definition of community and corresponding identifying algorithm, Phys. Rev. E 78 (2) (2008) 026121.
[80] S. Borgatti, M. Everett, P. Shirey, LS sets, lambda sets, and other cohesive subsets, Soc. Netw. 12 (1990) 337–358.
[81] U. Feige, D. Peleg, G. Kortsarz, The dense $k$-subgraph problem, Algorithmica 29 (3) (2001) 410–421.
[82] Y. Asahiro, R. Hassin, K. Iwama, Complexity of finding dense subgraphs, Discrete Appl. Math. 121 (1–3) (2002) 15–26.
[83] K. Holzapfel, S. Kosub, M.G. Maa, H. Täubig, The complexity of detecting fixed-density clusters, in: R. Petreschi, G. Persiano, R. Silvestri (Eds.), CIAC, in: Lecture Notes in Computer Science, vol. 2653, Springer, 2003, pp. 201–212.
[84] J. Šíma, S.E. Schaeffer, On the NP-completeness of some graph cluster measures, in: J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, J. Štuller (Eds.), Proceedings of the Thirty-second International Conference on Current Trends in Theory and Practice of Computer Science (Sofsem 06), in: Lecture Notes in Computer Science, vol. 3831, Springer-Verlag, Berlin, Heidelberg, Germany, 2006, pp. 530–537.
[85] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection, Phys. Rev. E 74 (1) (2006) 016110.
[86] F. Lorrain, H. White, Structural equivalence of individuals in social networks, J. Math. Sociol. 1 (1971) 49–80.
[87] P. Elias, A. Feinstein, C.E. Shannon, Note on maximum flow through a network, IRE Trans. Inf. Theory IT- 2 (1956) 117–119.
[88] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewood Cliffs, USA, 1993.
[89] E. Estrada, N. Hatano, Communicability in complex networks, Phys. Rev. E 77 (3) (2008) 036111.
[90] E. Estrada, N. Hatano, Communicability graph and community structures in complex networks, Appl. Math. Comput. 214 (2009) 500–511.
[91] M. Saerens, F. Fouss, L. Yen, P. Dupont, The principal component analysis of a graph and its relationships to spectral clustering, in: Proc. Eur. Conf. on Machine Learning, 2004. URL citeseer.ist.psu.edu/saerens04principal.html.
[92] F. Fouss, J.-M. Renders, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Trans. Knowl. Data Eng. 19 (3) (2007) 355–369. member-Pirotte, Alain and Member-Saerens, Marco.
[93] L. Yen, F. Fouss, C. Decaestecker, P. Francq, M. Saerens, Graph nodes clustering with the sigmoid commute-time kernel: A comparative study, Data Knowl. Eng. 68 (3) (2009) 338–361.
[94] L. Yen, F. Fouss, C. Decaestecker, P. Francq, M. Saerens, Graph nodes clustering based on the commute-time kernel, in: PAKDD, 2007, pp. 1037–1045.
[95] A.K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky, The electrical resistance of a graph captures its commute and cover times, in: STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing, ACM, New York, NY, USA, 1989, pp. 574–586.
[96] D.J. Klein, M. Randic, Resistance distance, J. Math. Chem. 12 (1993) 81–95.
[97] S. White, P. Smyth, Algorithms for estimating relative importance in networks, in: KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2003, pp. 266–275.
[98] H. Zhou, Distance, dissimilarity index, and network community structure, Phys. Rev. E 67 (6) (2003) 061901.
[99] D. Harel, Y. Koren, On clustering using random walks, in: FST TCS '01: Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science, Springer-Verlag, London, UK, 2001, pp. 18–41.
[100] M. Latapy, P. Pons, Lect. Notes Comput. Sci. 3733 (2005) 284–293.
[101] B. Nadler, S. Lafon, R.R. Coifman, I.G. Kevrekidis, Diffusion maps, spectral clustering and reaction coordinates of dynamical systems, Appl. Comput. Harmon. Anal. 21 (1) (2006) 113–127.
[102] C.R. Palmer, C. Faloutsos, Electricity based external similarity of categorical attributes, in: Proceedings of PAKDD 2003, 2003, pp. 486–500.
[103] H. Tong, C. Faloutsos, J.-Y. Pan, Random walk with restart: Fast solutions and applications, Knowl. Inf. Syst. 14 (3) (2008) 327–346.
[104] M. Gori, A. Pucci, Itemrank: A random-walk based scoring algorithm for recommender engines, in: IJCAI'07: Proceedings of the 20th International Joint Conference on Artifical Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 2766–2771.
[105] G.E. Andrews, The Theory of Partitions, Addison-Wesley, Boston, USA, 1976.
[106] L. Lovász, Combinatorial Problems and Exercises, North-Holland, Amsterdam, The Netherlands, 1993.
[107] G. Pólya, G. Szegö, Problems and Theorems in Analysis I, Springer-Verlag, Berlin, Germany, 1998.
[108] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, A.-L. Barabási, Hierarchical organization of modularity in metabolic networks, Science 297 (5586) (2002) 1551–1555.
[109] E. Ravasz, A.-L. Barabási, Hierarchical organization in complex networks, Phys. Rev. E 67 (2) (2003) 026112.
[110] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (3) (2009) 033015.
[111] J. Kleinberg, An impossibility theorem for clustering, in: Advances in NIPS 15, MIT Press, Boston, USA, 2002, pp. 446–453.
[112] M. Tumminello, F. Lillo, R.N. Mantegna, Correlation, hierarchies, and networks in financial markets, eprint arXiv:0809.4615.
[113] S.N. Dorogovtsev, J.F.F. Mendes, Evolution of networks, Adv. Phys. 51 (2002) 1079–1187.
[114] T. Łuczak, Sparse random graphs with a given degree sequence, in: Proceedings of the Symposium on Random Graphs, Poznań 1989, John Wiley & Sons, New York, USA, 1992, pp. 165–182.
[115] M. Molloy, B. Reed, A critical point for random graphs with a given degree sequence, Random Struct. Algor. 6 (1995) 161–179.

[116] H. Djidjev, A scalable multilevel algorithm for graph clustering and community structure detection, in: W. Aiello, A.Z. Broder, J.C.M. Janssen, E.E. Milios (Eds.), WAW, in: Lecture Notes in Computer Science, vol. 4936, Springer-Verlag, Berlin, Germany, 2007, pp. 117–128.
[117] B.H. Good, Y. de Montjoye, A. Clauset, The performance of modularity maximization in practical contexts, eprint arXiv:0910.0165.
[118] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[119] E. Holmström, N. Bock, J. Brännlund, Modularity density of network community divisions, Physica D 238 (2009) 1161–1167.
[120] C.P. Massen, J.P.K. Doye, Thermodynamics of Community Structure, eprint arXiv:cond-mat/0610077.
[121] A. Noack, Modularity clustering is force-directed layout, Phys. Rev. E 79 (2) (2009) 026102.
[122] A. Arenas, J. Duch, A. Fernández, S. Gómez, Size reduction of complex networks preserving modularity, New J. Phys. 9 (2007) 176.
[123] A. Pothen, Graph partitioning algorithms with applications to scientific computing, Tech. rep., Norfolk, VA, USA, 1997.
[124] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J. 49 (1970) 291–307.
[125] P.R. Suaris, G. Kedem, An algorithm for quadrisection and its application to standard cellplacement, IEEE Trans. Circuits Syst. 35 (1988) 294–303.
[126] E.R. Barnes, An algorithm for partitioning the nodes of a graph, SIAM J. Algebr. Discrete Methods 3 (1982) 541–550.
[127] M. Fiedler, Algebraic connectivity of graphs, Czech. Math. J. 23 (98) (1973) 298–305.
[128] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand. 45 (1950) 255–282.
[129] G.H. Golub, C.F.V. Loan, Matrix Computations, John Hopkins University Press, Baltimore, USA, 1989.
[130] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, Canad. J. Math. 8 (1956) 399–404.
[131] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem, J. ACM 35 (1988) 921–940.
[132] G.W. Flake, S. Lawrence, C.L. Giles, Efficient identification of web communities, in: Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, Boston, USA, 2000, pp. 150–160.
[133] Y.-C. Wei, C.-K. Cheng, Towards efficient hierarchical designs by ratio cut partitioning, in: Proceedings of IEEE International Conference on Computer Aided Design, Institute of Electrical and Electronics Engineers, New York, 1989, pp. 298–301.
[134] J. Shi, J. Malik, Normalized cuts and image segmentation, in: CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition, CVPR '97, IEEE Computer Society, Washington, DC, USA, 1997, p. 731.
[135] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[136] D.W. Matula, F. Shahrokhi, Sparsest cuts and bottlenecks in graphs, Discrete Appl. Math. 27 (1–2) (1990) 113–123.
[137] A. Blake, A. Zisserman, Visual Reconstruction, MIT Press, Cambridge, USA, 1987.
[138] L. Hagen, A.B. Kahng, New spectral methods for ratio cut partitioning and clustering, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 11 (9) (1992) 1074–1085.
[139] P.K. Chan, M.D.F. Schlag, J.Y. Zien, Spectral k-way ratio-cut partitioning and clustering, in: Proceedings of the 30th International Conference on Design Automation, ACM Press, New York, USA, 1993, pp. 749–754.
[140] T. Hastie, R. Tibshirani, J.H. Friedman, The Elements of Statistical Learning, Springer, Berlin, Germany, 2001.
[141] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L.M.L. Cam, J. Neyman (Eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, Berkeley, USA, 1967, pp. 281–297.
[142] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (2) (1982) 129–137.
[143] A. Schenker, M. Last, H. Bunke, A. Kandel, Graph representations for web document clustering, in: F.J.P. López, A.C. Campilho, N.P. de la Blanca, A. Sanfeliu (Eds.), IbPRIA'03: Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis, in: Lecture Notes in Computer Science, vol. 2652, Springer, 2003, pp. 935–942.
[144] A. Hlaoui, S. Wang, A direct approach to graph clustering, Neural Networks Computational Intelligence (2004) 158–163.
[145] M.J. Rattigan, M. Maier, D. Jensen, Graph clustering with network structure indices, in: ICML '07: Proceedings of the 24th International Conference on Machine Learning, ACM, New York, NY, USA, 2007, pp. 783–790.
[146] J.C. Dunn, A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters, J. Cybernet. 3 (1974) 32–57.
[147] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Kluwer Academic Publishers, Norwell, USA, 1981.
[148] W. Donath, A. Hoffman, Lower bounds for the partitioning of graphs, IBM J. Res. Dev. 17 (5) (1973) 420–425.
[149] D.A. Spielman, S.-H. Teng, Spectral partitioning works: Planar graphs and finite element meshes, in: IEEE Symposium on Foundations of Computer Science, 1996, pp. 96–105.
[150] U. von Luxburg, A tutorial on spectral clustering, Tech. Rep. 149, Max Planck Institute for Biological Cybernetics, August 2006.
[151] G.W. Stewart, J.-G. Sun, Matrix Perturbation Theory, Academic Press, New York, USA, 1990.
[152] R. Bhatia, Matrix Analysis, Springer-Verlag, New York, USA, 1997.
[153] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, vol. 14, MIT Press, Cambridge, USA, 2001.
[154] M. Meilă, J. Shi, A random walks view of spectral segmentation, in: AI and STATISTICS (AISTATS) 2001.
[155] J.M. Anthonisse, The rush in a directed graph, Tech. rep., Stichting Mathematisch Centrum, 2e Boerhaavestraat 49 Amsterdam, The Netherlands, 1971.
[156] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 (2001) 163–177.
[157] T. Zhou, J.-G. Liu, B.-H. Wang, Notes on the calculation of node betweenness, Chin. Phys. Lett. 23 (2006) 2327–2329.
[158] M.E.J. Newman, A measure of betweenness centrality based on random walks, Soc. Netw. 27 (2005) 39–54.
[159] M.E.J. Newman, Analysis of weighted networks, Phys. Rev. E 70 (5) (2004) 056131.
[160] J.R. Tyler, D.M. Wilkinson, B.A. Huberman, Email as spectroscopy: Automated discovery of community structure within organizations, in: Communities and Technologies, Kluwer, B.V., Deventer, The Netherlands, 2003, pp. 81–96.
[161] D.M. Wilkinson, B.A. Huberman, A method for finding communities of related genes, Proc. Natl. Acad. Sci. U.S.A. 101 (2004) 5241–5248.
[162] M.J. Rattigan, M. Maier, D. Jensen, Using structure indices for efficient approximation of network properties, in: T. Eliassi-Rad, L.H. Ungar, M. Craven, D. Gunopulos (Eds.), SIGKDD'06: Proceedings of the 12th international conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 357–366.
[163] U. Brandes, M. Gaertler, D. Wagner, Experiments on graph clustering algorithms, in: Proceedings of ESA, Springer-Verlag, Berlin, Germany, 2003, pp. 568–579.
[164] P. Holme, M. Huss, H. Jeong, Subnetwork hierarchies of biochemical pathways, Bioinformatics 19 (4) (2003) 532–538.
[165] J.W. Pinney, D.R. Westhead, Betweenness-based decomposition methods for social and biological networks, in: Interdisciplinary Statistics and Bioinformatics, Leeds University Press, Leeds, UK, 2006, pp. 87–90.
[166] S. Gregory, An algorithm to find overlapping community structure in networks, in: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2007, Springer-Verlag, Berlin, Germany, 2007, pp. 91–102.
[167] D. Watts, S. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442.
[168] C. Castellano, F. Cecconi, V. Loreto, D. Parisi, F. Radicchi, Self-contained algorithms to detect communities in networks, Eur. Phys. J. B 38 (2) (2004) 311–319.
[169] P. Zhang, J. Wang, X. Li, Z. Di, Y. Fan, The clustering coefficient and community structure of bipartite networks, eprint arXiv:0710.0117.
[170] V. Latora, M. Marchiori, Efficient behavior of small-world networks, Phys. Rev. Lett. 87 (19) (2001) 198701.
[171] S. Fortunato, V. Latora, M. Marchiori, Method to find community structures based on information centrality, Phys. Rev. E 70 (5) (2004) 056104.
[172] I. Vragović, E. Louis, Network community structure and loop coefficient method, Phys. Rev. E 74 (1) (2006) 016105.
[173] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2004) 066133.
[174] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.

[175] L. Danon, A. Díaz-Guilera, A. Arenas, The effect of size heterogeneity on community identification in complex networks, J. Stat. Mech. 11 (2006) 10.
[176] J.M. Pujol, J. Béjar, J. Delgado, Clustering algorithm for determining community structure in large networks, Phys. Rev. E 74 (1) (2006) 016107.
[177] H. Du, M.W. Feldman, S. Li, X. Jin, An algorithm for detecting community structure of social networks based on prior knowledge and modularity, Complexity 12 (3) (2007) 53–60.
[178] K. Wakita, T. Tsurumi, Finding community structure in mega-scale social networks, eprint arXiv:cs/0702048.
[179] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. P10008 (2008).
[180] P. Schuetz, A. Caflisch, Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement, Phys. Rev. E 77 (4) (2008) 046112.
[181] P. Schuetz, A. Caflisch, Multistep greedy algorithm identifies community structure in real-world and computer-generated networks, Phys. Rev. E 78 (2) (2008) 026112.
[182] A. Noack, R. Rotta, Multi-level algorithms for modularity clustering, in: SEA '09: Proceedings of the 8th International Symposium on Experimental Algorithms, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 257–268.
[183] B. Xiang, E.-H. Chen, T. Zhou, Finding community structure based on subgraph similarity, in: S. Fortunato, R. Menezes, G. Mangioni, V. Nicosia (Eds.), Complex Networks, in: Studies in Computational Intelligence, vol. 207, Springer Verlag, Berlin/Heidelberg, Germany, 2009, pp. 73–81.
[184] J. Mei, S. He, G. Shi, Z. Wang, W. Li, Revealing network communities through modularity maximization by a contraction-dilation method, New J. Phys. 11 (4) (2009) 043025.
[185] R. Guimerà, M. Sales-Pardo, L.A.N. Amaral, Modularity from fluctuations in random graphs and complex networks, Phys. Rev. E 70 (2) (2004) 025101 (R).
[186] C.P. Massen, J.P.K. Doye, Identifying communities within energy landscapes, Phys. Rev. E 71 (4) (2005) 046101.
[187] A. Medus, G. Acuña, C.O. Dorso, Detection of community structures in networks via global optimization, Physica A 358 (2005) 593–604.
[188] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Phys. Rev. E 72 (2) (2005) 027104.
[189] S. White, P. Smyth, A spectral clustering approach to finding communities in graphs, in: Proceedings of SIAM International Conference on Data Mining, 2005, pp. 76–84.
[190] M.E.J. Newman, From the cover: Modularity and community structure in networks, Proc. Natl. Acad. Sci. USA 103 (2006) 8577–8582.
[191] S. Lehmann, L.K. Hansen, Deterministic modularity optimization, Eur. Phys. J. B 60 (2007) 83–88.
[192] J. Ruan, W. Zhang, An efficient spectral algorithm for network community discovery and its applications to biological and social networks, in: ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, 2007, pp. 643–648.
[193] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikolski, D. Wagner, On modularity — np-completeness and beyond. URL http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/3255.
[194] Z. Ye, S. Hu, J. Yu, Adaptive clustering algorithm for community detection in complex networks, Phys. Rev. E 78 (4) (2008) 046115.
[195] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[196] S. Boettcher, A.G. Percus, Optimization with extremal dynamics, Phys. Rev. Lett. 86 (2001) 5211–5214.
[197] T. Richardson, P.J. Mucha, M.A. Porter, Spectral tripartitioning of networks, Phys. Rev. E 80 (3) (2009) 036111.
[198] Y. Sun, B. Danila, K. Josic, K.E. Bassler, Improved community structure detection using a modified fine-tuning strategy, Europhys. Lett. 86 (2) (2009) 28004.
[199] G. Wang, Y. Shen, M. Ouyang, A vector partitioning approach to detecting community structure in complex networks, Comput. Math. Appl. 55 (12) (2008) 2746–2752.
[200] J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Society for Industrial and Applied Mathematics, Philadelphia, USA, 2000.
[201] P. Gleiser, L. Danon, Community structure in jazz, Adv. Complex Syst. 6 (2003) 565.
[202] M.E.J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA 98 (2) (2001) 404–409.
[203] G. Agarwal, D. Kempe, Modularity-maximizing network communities via mathematical programming, Eur. Phys. J. B 66 (2008) 409–418.
[204] H. Karloff, Linear Programming, Birkhäuser Verlag, Basel, Switzerland, 1991.
[205] G. Xu, S. Tsoka, L.G. Papageorgiou, Finding community structures in complex networks using mixed integer optimisation, Eur. Phys. J. B 60 (2) (2007) 231–239.
[206] W.Y.C. Chen, A.W.M. Dress, W.Q. Yu, Community structures of networks, Math. Comput. Sci. 1 (3) (2008) 441–457.
[207] F.S. Hillier, G.J. Lieberman, Introduction to Operations Research, McGraw-Hill, New York, USA, 2004.
[208] J.W. Berry, B. Hendrickson, R.A. LaViolette, V.J. Leung, C.A. Phillips, Community detection via facility location, eprint arXiv:0710.3800.
[209] C. Peterson, J.R. Anderson, A mean field theory learning algorithm for neural networks, Complex Systems 1 (1987) 995–1019.
[210] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, USA, 1992.
[211] M. Tasgin, A. Herdagdelen, H. Bingol, Community detection in complex networks using genetic algorithms, eprint arXiv:0711.0491.
[212] X. Liu, D. Li, S. Wang, Z. Tao, Effective algorithm for detecting community structure in complex networks based on GA and clustering, in: ICCS '07: Proceedings of the 7th International Conference on Computational Science, Part II, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 657–664.
[213] Z. Feng, X. Xu, N. Yuruk, T. Schweiger, A novel similarity-based modularity function for graph partitioning, Lect. Notes Comp. Sci. 4654 (2007) 385–396.
[214] R. Ghosh, K. Lerman, Community detection using a measure of global influence, eprint arXiv:0805.4606.
[215] Y. Kim, S.-W. Son, H. Jeong, Link rank: Finding communities in directed networks, eprint arXiv:0902.3728.
[216] H. Shen, X. Cheng, K. Cai, M.-B. Hu, Detect overlapping and hierarchical community structure in networks, Physica A 388 (2009) 1706–1712.
[217] V. Nicosia, G. Mangioni, V. Carchiolo, M. Malgeri, Extending the definition of modularity to directed graphs with overlapping communities, J. Stat. Mech. P03024 (2009).
[218] H.-W. Shen, X.-Q. Cheng, J.-F. Guo, Quantifying and identifying the overlapping community structure in networks, J. Stat. Mech. P07042 (2009).
[219] M. Gaertler, R. Görke, D. Wagner, Significance-driven graph clustering, in: M.-Y. Kao, X.-Y. Li (Eds.), AAIM, in: Lecture Notes in Computer Science, vol. 4508, Springer, Berlin, Germany, 2007, pp. 11–26.
[220] S. Muff, F. Rao, A. Caflisch, Local modularity measure for network clusterizations, Phys. Rev. E 72 (5) (2005) 056107.
[221] M. Mezard, G. Parisi, M. Virasoro, Spin Glass Theory and Beyond, World Scientific Publishing Company, Singapore, 1987.
[222] A. Arenas, A. Fernández, S. Fortunato, S. Gómez, Motif-based communities in complex networks, J. Phys. A 41 (22) (2008) 224001.
[223] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: Simple building blocks of complex networks, Science 298 (5594) (2002) 824–827.
[224] R.N. Mantegna, H.E. Stanley, An Introduction to Econophysics: Correlations and Complexity in Finance, Cambridge University Press, New York, USA, 2000.
[225] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Mach. Learn. 56 (1–3) (2004) 89–113.
[226] S. Gómez, P. Jensen, A. Arenas, Analysis of community structure in networks of correlated data, Phys. Rev. E 80 (1) (2009) 016114.
[227] T.D. Kaplan, S. Forrest, A dual assortative measure of community structure, eprint arXiv:0801.3290.
[228] V.A. Traag, J. Bruggeman, Community detection in networks with positive and negative links, Phys. Rev. E 80 (3) (2009) 036115.
[229] R.J. Williams, N.D. Martinez, Simple rules yield complex food webs, Nature 404 (2000) 180–183.
[230] R. Guimerà, M. Sales-Pardo, L.A.N. Amaral, Module identification in bipartite and directed networks, Phys. Rev. E 76 (3) (2007) 036102.
[231] M.J. Barber, M. Faria, L. Streit, O. Strogan, Searching for communities in bipartite networks, in: C.C. Bernido, M.V. Carpio-Bernido (Eds.), Stochastic and Quantum Dynamics of Biomolecular Systems, in: American Institute of Physics Conference Series, vol. 1021, American Institute of Physics, Melville, USA, 2008, pp. 171–182.
[232] J. Reichardt, S. Bornholdt, When are networks truly modular? Physica D 224 (2006) 20–26.

[233] J. Reichardt, S. Bornholdt, Partitioning and modularity of graphs with arbitrary degree distribution, Phys. Rev. E 76 (1) (2007) 015102 (R).
[234] Y. Fu, P. Anderson, Application of statistical mechanics to NP-complete problems in combinatorial optimisation, J. Phys. A 19 (1986) 1605–1620.
[235] D. Sherrington, S. Kirkpatrick, Solvable model of a spin-glass, Phys. Rev. Lett. 35 (1975) 1792–1796.
[236] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. USA 104 (2007) 36–41.
[237] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (6) (2003) 065103 (R).
[238] L. Danon, A. Díaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. P09008 (2005).
[239] S. Fortunato, Quality functions in community detection, in: Noise and Stochastics in Complex Systems and Finance, in: SPIE Conference Series, vol. 6601, 2007, p. 660108.
[240] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, L. Chen, Quantitative function for community detection, Phys. Rev. E 77 (3) (2008) 036109.
[241] J. Ruan, W. Zhang, Identifying network communities with a high resolution, Phys. Rev. E 77 (1) (2008) 016104.
[242] J.M. Kumpula, J. Saramäki, K. Kaski, J. Kertész, Limited resolution in complex network community detection with Potts model approach, Eur. Phys. J. B 56 (2007) 41–45.
[243] J.W. Berry, B. Hendrickson, R.A. LaViolette, C.A. Phillips, Tolerating the community detection resolution limit with edge weighting, eprint arXiv:0903.1072.
[244] K.A. Eriksen, I. Simonsen, S. Maslov, K. Sneppen, Modularity and extreme edges of the internet, Phys. Rev. Lett. 90 (14) (2003) 148701.
[245] I. Simonsen, K.A. Eriksen, S. Maslov, K. Sneppen, Diffusion on complex networks: A way to probe their large-scale topological structures, Physica A 336 (2004) 163–173.
[246] F. Slanina, Y.-C. Zhang, Referee networks and their spectral properties, Acta Phys. Polon. B 36 (2005) 2797–2804.
[247] M. Mitrović, B. Tadić, Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities, Phys. Rev. E 80 (2) (2009) 026123.
[248] I. Simonsen, Diffusion and networks: A powerful combination! Physica A 357 (2) (2005) 317–330.
[249] N.A. Alves, Unveiling community structures in weighted networks, Phys. Rev. E 76 (3) (2007) 036101.
[250] A. Capocci, V.D.P. Servedio, G. Caldarelli, F. Colaiori, Detecting communities in large networks, Physica A 352 (2005) 669–676.
[251] B. Yang, J. Liu, Discovering global network communities based on local centralities, ACM Trans. Web 2 (1) (2008) 1–32.
[252] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, J. Math. Sociol. 2 (1) (1972) 113–120.
[253] P. Bonacich, Power and centrality: A family of measures, Amer. J. Sociol. 92 (5) (1987) 1170–1182.
[254] F.Y. Wu, The Potts model, Rev. Mod. Phys. 54 (1) (1982) 235–268.
[255] M. Blatt, S. Wiseman, E. Domany, Superparamagnetic clustering of data, Phys. Rev. Lett. 76 (1996) 3251–3254.
[256] J. Reichardt, S. Bornholdt, Detecting Fuzzy community structures in complex networks with a Potts model, Phys. Rev. Lett. 93 (21) (2004) 218701.
[257] I. Ispolatov, I. Mazo, A. Yuryev, Finding mesoscopic communities in sparse networks, J. Stat. Mech. P09014 (2006).
[258] S.-W. Son, H. Jeong, J.D. Noh, Random field Ising model and community structure in complex networks, Eur. Phys. J. B 50 (3) (2006) 431–437.
[259] A.A. Middleton, D.S. Fisher, Three-dimensional random-field Ising magnet: Interfaces, scaling, and the nature of states, Phys. Rev. B 65 (13) (2002) 134411.
[260] J.D. Noh, H. Rieger, Disorder-driven critical behavior of periodic elastic media in a crystal potential, Phys. Rev. Lett. 87 (17) (2001) 176102.
[261] J.D. Noh, H. Rieger, Numerical study of the disorder-driven roughening transition in an elastic manifold in a periodic potential, Phys. Rev. E 66 (3) (2002) 036117.
[262] B.D. Hughes, Random Walks and Random Environments: Random Walks, vol. 1, Clarendon Press, Oxford, UK, 1995.
[263] H. Zhou, Network landscape from a brownian particle's perspective, Phys. Rev. E 67 (4) (2003) 041908.
[264] H. Zhou, R. Lipowsky, Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities, Lect. Notes Comput. Sci. 3038 (2004) 1062–1069.
[265] J.H. Ward, Hierarchical grouping to optimize an objective function, J. Am. Assoc. 58 (301) (1963) 236–244.
[266] Y. Hu, M. Li, P. Zhang, Y. Fan, Z. Di, Community detection by signaling on complex networks, Phys. Rev. E 78 (1) (2008) 016115.
[267] J.C. Delvenne, S.N. Yaliraki, M. Barahona, Stability of graph communities across time scales, eprint arXiv:0812.1811.
[268] M. Fiedler, A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory, Czechoslovak Math. J. 25 (1975) 619–633.
[269] R. Lambiotte, J. Delvenne, M. Barahona, Laplacian dynamics and multiscale modular structure in networks, eprint arXiv:0812.1770.
[270] E. Weinan, T. Li, E. Vanden-Eijnden, Optimal partition and effective dynamics of complex networks, Proc. Natl. Acad. Sci. USA 105 (2008) 7907–7912.
[271] S. van Dongen, Graph clustering by flow simulation, Ph.D. Thesis, Dutch National Research Institute for Mathematics and Computer Science, University of Utrecht, Netherlands, 2000.
[272] A. Pikovsky, M.G. Rosenblum, J. Kurths, Synchronization : A Universal Concept in Nonlinear Sciences, Cambridge University Press, Cambridge, UK, 2001.
[273] A. Arenas, A. Díaz-Guilera, C.J. Pérez-Vicente, Synchronization reveals topological scales in complex networks, Phys. Rev. Lett. 96 (11) (2006) 114102.
[274] Y. Kuramoto, Chemical Oscillations, Waves and Turbulence, Springer-Verlag, Berlin, Germany, 1984.
[275] A. Arenas, A. Díaz-Guilera, Synchronization and modularity in complex networks, Eur. Phys. J. Special Topics 143 (2007) 19–25.
[276] S. Boccaletti, M. Ivanchenko, V. Latora, A. Pluchino, A. Rapisarda, Detecting complex network modularity by dynamical clustering, Phys. Rev. E 75 (4) (2007) 045102.
[277] A. Pluchino, V. Latora, A. Rapisarda, Changing opinions in a changing world, Int. J. Mod. Phys. C 16 (2005) 515–531.
[278] D. Li, I. Leyva, J.A. Almendral, I. Sendiña-Nadal, J.M. Buldú, S. Havlin, S. Boccaletti, Synchronization interfaces and overlapping communities in complex networks, Phys. Rev. Lett. 101 (16) (2008) 168701.
[279] D.J.C. Mackay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, Cambridge, UK, 2003.
[280] R.L. Winkler, Introduction to Bayesian Inference and Decision, Probabilistic Publishing, Gainesville, USA, 2003.
[281] P. Doreian, V. Batagelj, A. Ferligoj, Generalized Blockmodeling, Cambridge University Press, New York, USA, 2005.
[282] K.P. Burnham, D.R. Anderson, Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, Springer, New York, USA, 2002.
[283] M.S. Handcock, A.E. Raftery, J.M. Tantrum, Model based clustering for social networks, J. Roy. Statist. Soc. A 170 (2007) 1–22. Working Paper no. 46.
[284] J.H. Koskinen, T.A.B. Snijders, Bayesian inference for dynamic social network data, J. Stat. Plan. Infer. 137 (12) (2007) 3930–3938.
[285] C.J. Rhodes, E.M.J. Keefe, Social network topology: A bayesian approach, J. Oper. Res. Soc. 58 (12) (2007) 1605–1611.
[286] M. Rowicka, A. Kudlicki, Bayesian modeling of protein interaction networks, in: R. Fischer, R. Preuss, U. von Toussaint (Eds.), Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, vol. 735, American Institute of Physics, Melville, USA, 2004, pp. 283–288.
[287] J. Berg, M. Lässig, Cross-species analysis of biological networks by bayesian alignment, Proc. Natl. Acad. Sci. USA 103 (29) (2006) 10967–10972.
[288] M.B. Hastings, Community detection as an inference problem, Phys. Rev. E 74 (3) (2006) 035102.
[289] R.G. Gallager, Low Density Parity Check Codes, MIT Press, Cambridge, USA, 1963.
[290] M.E.J. Newman, E.A. Leicht, Mixture models and exploratory analysis in networks, Proc. Natl. Acad. Sci. USA 104 (2007) 9564–9569.
[291] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Stat. Soc. B 39 (1977) 1–38.
[292] T.A.B. Snijders, K. Nowicki, Estimation and prediction for stochastic blockmodels for graphs with latent block structure, J. Classification 14 (1997) 75–100.
[293] K. Nowicki, T.A.B. Snijders, Estimation and prediction for stochastic blockstructures, J. Am. Assoc. 96 (455).
[294] M. Mungan, J.J. Ramasco, Who is keeping you in that community? eprint arXiv:0809.1398.

[295] A. Vazquez, Population stratification using a statistical model on hypergraphs, Phys. Rev. E 77 (6) (2008) 066106.
[296] A. Vazquez, Bayesian approach to clustering real value, categorical and network data: Solution via variational methods, eprint arXiv:0805.2689.
[297] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, L.K. Saul, An introduction to variational methods for graphical models, Mach. Learn. 37 (2) (1999) 183–233.
[298] M.J. Beal, Variational algorithms for approximate bayesian inference, Ph.D. Thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.
[299] J.J. Ramasco, M. Mungan, Inversion method for content-based networks, Phys. Rev. E 77 (3) (2008) 036122.
[300] W. Ren, G. Yan, X. Liao, L. Xiao, Simple probabilistic algorithm for detecting community structure, Phys. Rev. E 79 (3) (2009) 036111.
[301] J. Čopič, M.O. Jackson, A. Kirman, Identifying community structures from network data. URL http://www.hss.caltech.edu/~jernej/netcommunity.pdf.
[302] H. Zanghi, C. Ambroise, V. Miele, Fast online graph clustering via Erdös-Rényi mixture, Pattern Recognit. 41 (12) (2008) 3592–3599.
[303] C. Biernacki, G. Celeux, G. Govaert, Assessing a mixture model for clustering with the integrated completed likelihood, IEEE Trans. Pattern Anal. Mach. Intell. 22 (7) (2000) 719–725.
[304] J.M. Hofman, C.H. Wiggins, Bayesian Approach to Network Modularity, Phys. Rev. Lett. 100 (25) (2008) 258701.
[305] D.R. White, K.P. Reitz, Graph and semigroup homomorphisms on networks and relations, Soc. Netw. 5 (1983) 193–234.
[306] M.G. Everett, S.P. Borgatti, Regular equivalence: General theory, J. Math. Soc. 19 (1) (1994) 29–52.
[307] S.E. Fienberg, S. Wasserman, Categorical data analysis of single sociometric relations, Sociol. Methodol. 12 (1981) 156–192.
[308] P. Holland, K.B. Laskey, S. Leinhardt, Stochastic blockmodels: Some first steps, Soc. Netw. 5 (1983) 109–137.
[309] J. Reichardt, D.R. White, Role models for complex networks, Eur. Phys. J. B 60 (2007) 217–224.
[310] H. Akaike, A new look at the statistical model identification, IEEE Trans. Automat. Control 19 (6) (1974) 716–723.
[311] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (2) (1978) 461–464.
[312] J. Rissanen, Modelling by shortest data descriptions, Automatica 14 (1978) 465–471.
[313] P.D. Grünwald, I.J. Myung, M.A. Pitt, Advances in Minimum Description Length: Theory and Applications, MIT Press, Cambridge, USA, 2005.
[314] C.S. Wallace, D.M. Boulton, An information measure for classification, Comput. J. 11 (2) (1968) 185–194.
[315] M. Rosvall, C.T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, Proc. Natl. Acad. Sci. USA 104 (2007) 7327–7331.
[316] J. Sun, C. Faloutsos, S. Papadimitriou, P.S. Yu, Graphscope: Parameter-free mining of large time-evolving graphs, in: KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, USA, 2007, pp. 687–696.
[317] D.A. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE 40 (9) (1952) 1098–1101.
[318] M. Rosvall, D. Axelsson, C.T. Bergstrom, The map equation, eprint arXiv:0906.1405.
[319] D. Chakrabarti, Autopart: Parameter-free graph partitioning and outlier detection, in: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), PKDD, in: Lect. Notes Comp. Sci., vol. 3202, Springer, 2004, pp. 112–124.
[320] E. Ziv, M. Middendorf, C.H. Wiggins, Information-theoretic approach to network modularity, Phys. Rev. E 71 (4) (2005) 046117.
[321] N. Tishby, F. Pereira, W. Bialek, The information bottleneck method, in: Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing, 1999, pp. 368–377.
[322] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (3) (2007) 036106.
[323] G. Tibély, J. Kertész, On the equivalence of the label propagation method of community detection and a Potts model approach, Physica A 387 (2008) 4982–4984.
[324] M.J. Barber, J.W. Clark, Detecting network communities by propagating labels under constraints, Phys. Rev. E 80 (2) (2009) 026129.
[325] I.X.Y. Leung, P. Hui, P. Liò, J. Crowcroft, Towards real-time community detection in large networks, Phys. Rev. E 79 (6) (2009) 066107.
[326] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.
[327] J.P. Bagrow, E.M. Bollt, Local method for detecting communities, Phys. Rev. E 72 (4) (2005) 046108.
[328] L. da Fontoura Costa, Hub-based community finding, eprint arXiv:cond-mat/0405022.
[329] M.A. Porter, P.J. Mucha, M.E.J. Newman, A.J. Friend, Community structure in the United States House of Representatives, Physica A 386 (2007) 414–438.
[330] F.A. Rodrigues, G. Travieso, L. da F. Costa, Fast community identification by hierarchical growth, Int. J. Mod. Phys. C 18 (2007) 937–947.
[331] S. Papadopoulos, A. Skusa, A. Vakali, Y. Kompatsiaris, N. Wagner, Bridge bounding: A local approach for efficient community discovery in complex networks, eprint arXiv:0902.0871.
[332] A. Clauset, Finding local community structure in networks, Phys. Rev. E 72 (2) (2005) 026132.
[333] P. Hui, E. Yoneki, S.-Y. Chan, J. Crowcroft, Distributed community detection in delay tolerant networks, in: Proc. MobiArch, 2007.
[334] J.-P. Eckmann, E. Moses, Curvature of co-links uncovers hidden thematic layers in the World Wide Web, Proc. Natl. Acad. Sci. USA 99 (2002) 5825–5829.
[335] B. Long, X. Xu, Z. Zhang, P.S. Yu, Community learning by graph approximation, in: ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, 2007, pp. 232–241.
[336] F. Wu, B.A. Huberman, Finding communities in linear time: A physics approach, Eur. Phys. J. B 38 (2004) 331–338.
[337] P. Orponen, S.E. Schaeffer, Local clustering of large graphs by approximate Fiedler vectors, in: S. Nikoletseas (Ed.), Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms, WEA'05, Santorini, Greece, May 2005, in: Lect. Notes Comp. Sci., vol. 3503, Springer-Verlag, Berlin-Heidelberg, 2005, pp. 524–533.
[338] J. Ohkubo, K. Tanaka, Nonadditive volume and community detection problem in complex networks, J. Phys. Soc. Japan 75 (11) (2006) 115001-1–115001-2.
[339] M. Zarei, K.A. Samani, Eigenvectors of network complement reveal community structure more accurately, Physica A 388 (2009) 1721–1730.
[340] A. Condon, R.M. Karp, Algorithms for graph partitioning on the planted partition model, Random Struct. Algor. 18 (2001) 116–140.
[341] V. Gudkov, V. Montealegre, S. Nussinov, Z. Nussinov, Community detection in complex networks by dynamical simplex evolution, Phys. Rev. E 78 (1) (2008) 016113.
[342] M.J. Krawczyk, K. Kulakowski, Communities in networks — A continuous approach, eprint arXiv:0709.0923.
[343] M.J. Krawczyk, Differential equations as a tool for community identification, Phys. Rev. E 77 (6) (2008) 065701.
[344] A. Narasimhamurthy, D. Greene, N. Hurley, P. Cunningham, Community finding in large social networks through problem decomposition, in: Proc. 19th Irish Conference on Artificial Intelligence and Cognitive Science, AICS'08, 2008.
[345] I.S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors: A multilevel approach, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007) 1944–1957.
[346] M.G. Everett, S.P. Borgatti, Analyzing clique overlap, Connections 21 (1) (1998) 49–61.
[347] I. Derényi, G. Palla, T. Vicsek, Clique percolation in random networks, Phys. Rev. Lett. 94 (16) (2005) 160202.
[348] B. Adamcsek, G. Palla, I.J. Farkas, I. Derényi, T. Vicsek, Cfinder: Locating cliques and overlapping modules in biological networks, Bioinformatics 22 (8) (2006) 1021–1023.
[349] I. Farkas, D. Ábel, G. Palla, T. Vicsek, Weighted network modules, New J. Phys. 9 (2007) 180.
[350] S. Lehmann, M. Schwartz, L.K. Hansen, Biclique communities, Phys. Rev. E 78 (1) (2008) 016108.
[351] R. Peeters, The maximum edge biclique problem is NP-complete, Discrete Appl. Math. 131 (2003) 651–654.
[352] N. Du, B. Wang, B. Wu, Y. Wang, Overlapping community detection in bipartite networks, in: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, Los Alamitos, CA, USA, 2008, pp. 176–179.
[353] J.M. Kumpula, M. Kivelä, K. Kaski, J. Saramäki, Sequential algorithm for fast clique percolation, Phys. Rev. E 78 (2) (2008) 026109.

[354] J. Baumes, M.K. Goldberg, M.S. Krishnamoorthy, M.M. Ismail, N. Preston, Finding communities by clustering a graph into overlapping subgraphs, in: N. Guimaraes, P.T. Isaias (Eds.), IADIS AC, IADIS, 2005, pp. 97–104.
[355] J. Baumes, M. Goldberg, M. Magdon-Ismail, Efficient identification of overlapping communities, in: IEEE International Conference on Intelligence and Security Informatics, ISI, 2005, pp. 27–36.
[356] S. Zhang, R.-S. Wang, X.-S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Physica A 374 (2007) 483–490.
[357] T. Nepusz, A. Petróczi, L. Négyessy, F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks, Phys. Rev. E 77 (1) (2008) 016107.
[358] T.S. Evans, R. Lambiotte, Line graphs, link partitions, and overlapping communities, Phys. Rev. E 80 (1) (2009) 016105.
[359] Y.-Y. Ahn, J.P. Bagrow, S. Lehmann, Communities and hierarchical organization of links in complex networks, eprint arXiv:0903.3178.
[360] V.K. Balakrishnan, Schaum's Outline of Graph Theory, McGraw-Hill, New York, USA, 1997.
[361] S. Gregory, Finding overlapping communities using disjoint community detection algorithms, in: S. Fortunato, R. Menezes, G. Mangioni, V. Nicosia (Eds.), Complex Networks, in: Studies on Computational Intelligence, vol. 207, Springer, Berlin, Germany, 2009, pp. 47–62.
[362] T. Heimo, J.M. Kumpula, K. Kaski, J. Saramäki, Detecting modules in dense weighted networks with the Potts method, J. Stat. Mech. (2008) P08007.
[363] P. Pons, Post-processing hierarchical community structures: Quality improvements and multi-scale view, eprint arXiv:cs/0608050.
[364] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (5) (1986) 533–549.
[365] J.M. Kumpula, J. Saramäki, K. Kaski, J. Kertész, Limited resolution and multiresolution methods in complex network community detection, in: Noise and Stochastics in Complex Systems and Finance, in: SPIE Conference Series, vol. 6601, 2007, p. 660116.
[366] P. Ronhovde, Z. Nussinov, A highly accurate and resolution-limit-free Potts model for community detection, eprint arXiv:0803.2548.
[367] P. Ronhovde, Z. Nussinov, Multiresolution community detection for megascale networks by information-based replica correlations, Phys. Rev. E 80 (1) (2009) 016109.
[368] M. Sales-Pardo, R. Guimerà, A.A. Moreira, L.A.N. Amaral, Extracting the hierarchical organization of complex systems, Proc. Natl. Acad. Sci. USA 104 (2007) 15224–15229.
[369] A. Barrat, M. Barthélemy, R. Pastor-Satorras, A. Vespignani, The architecture of complex weighted networks, Proc. Natl. Acad. Sci. USA 101 (11) (2004) 3747–3752.
[370] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, U. Alon, Coarse-graining and self-dissimilarity of complex networks, Phys. Rev. E 71 (1) (2005) 016127.
[371] R. Guimerà, M. Sales-Pardo, L.A.N. Amaral, A network-based method for target selection in metabolic networks, Bioinformatics 23 (13) (2007) 1616–1622.
[372] A. Clauset, C. Moore, M.E.J. Newman, Structural inference of hierarchies in networks, in: E.M. Airoldi, D.M. Blei, S.E. Fienberg, A. Goldenberg, E.P. Xing, A.X. Zheng (Eds.), Statistical Network Analysis: Models, Issues, and New Directions, in: Lect. Notes Comput. Sci., vol. 4503, Springer, Berlin, Germany, 2007, pp. 1–13.
[373] A. Clauset, C. Moore, M.E.J. Newman, Hierarchical structure and the prediction of missing links in networks, Nature 453 (7191) (2008) 98–101.
[374] M.E.J. Newman, T. Barkema, Monte Carlo Methods in Statistical Physics, Oxford University Press, Oxford, UK, 1999.
[375] D. Liben-Nowell, J. Kleinberg, The link prediction problem for social networks, in: CIKM '03: Proceedings of the Twelfth International Conference on Information and Knowledge Management, ACM, New York, NY, USA, 2003, pp. 556–559.
[376] R. Kumar, J. Novak, P. Raghavan, A. Tomkins, On the bursty evolution of blogspace, in: WWW '03: Proceedings of the Twelfth International Conference on the World Wide Web, ACM Press, 2003, pp. 568–576.
[377] R. Kumar, J. Novak, A. Tomkins, Structure and evolution of online social networks, in: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2006, pp. 611–617.
[378] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: Densification laws, shrinking diameters and possible explanations, in: KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, ACM, New York, NY, USA, 2005, pp. 177–187.
[379] J. Leskovec, L. Backstrom, R. Kumar, A. Tomkins, Microscopic evolution of social networks, in: KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2008, pp. 462–470.
[380] G. Palla, A.-L. Barabási, T. Vicsek, Quantifying social group evolution, Nature 446 (2007) 664–667.
[381] J. Hopcroft, O. Khan, B. Kulis, B. Selman, Tracking evolving communities in large linked networks, Proc. Natl. Acad. Sci. USA 101 (2004) 5249–5253.
[382] C.L. Giles, K. Bollacker, S. Lawrence, CiteSeer: An automatic citation indexing system, in: I. Witten, R. Akscyn, F.M. Shipman III (Eds.), Digital Libraries 98 — The Third ACM Conference on Digital Libraries, ACM Press, Pittsburgh, PA, 1998, pp. 89–98.
[383] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, Boston, USA, 1999.
[384] S. Asur, S. Parthasarathy, D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs, in: KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2007, pp. 913–921.
[385] D.J. Fenn, M.A. Porter, M. McDonald, S. Williams, N.F. Johnson, N.S. Jones, Dynamic communities in multichannel data: An application to the foreign exchange market during the 2007–2008 credit crisis, Chaos 19 (3) (2009) 033119.
[386] D. Chakrabarti, R. Kumar, A. Tomkins, Evolutionary clustering, in: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2006, pp. 554–560.
[387] Y. Chi, X. Song, D. Zhou, K. Hino, B.L. Tseng, Evolutionary spectral clustering by incorporating temporal smoothness, in: KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2007, pp. 153–162.
[388] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, Facetnet: a framework for analyzing communities and their evolutions in dynamic networks, in: WWW '08: Proceedings of the 17th International Conference on the World Wide Web, ACM, New York, NY, USA, 2008, pp. 685–694.
[389] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Statist. 22 (1951) 49–86.
[390] M.-S. Kim, J. Han, A particle-and-density based evolutionary clustering method for dynamic networks, in: Proceedings of 2009 Int. Conf. on Very Large Data Bases, Lyon, France, 2009.
[391] L. Backstrom, D. Huttenlocher, J. Kleinberg, X. Lan, Group formation in large social networks: membership, growth, and evolution, in: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2006, pp. 44–54.
[392] D. Gfeller, J.-C. Chappelier, P. De Los Rios, Finding instabilities in the community structure of complex networks, Phys. Rev. E 72 (5) (2005) 056135.
[393] B. Karrer, E. Levina, M.E.J. Newman, Robustness of community structure in networks, Phys. Rev. E 77 (4) (2008) 046119.
[394] M. Rosvall, C.T. Bergstrom, Mapping change in large networks, eprint arXiv:0812.1242.
[395] B. Efron, R.J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York, USA, 1993.
[396] D.J. Earl, M.W. Deem, Parallel tempering: Theory, applications, and new perspectives, Phys. Chem. Chem. Phys. 7 (2005) 3910–3916.
[397] G. Bianconi, P. Pin, M. Marsili, Assessing the relevance of node features for network structure, Proc. Natl. Acad. Sci. USA 106 (28) (2009) 11433–11438.
[398] G. Bianconi, The entropy of randomized network ensembles, Europhys. Lett. 81 (2008) 28005.
[399] G. Bianconi, A.C.C. Coolen, C.J. Perez Vicente, Entropies of complex networks with hierarchically constrained topologies, Phys. Rev. E 78 (1) (2008) 016114.
[400] A. Lancichinetti, F. Radicchi, J.J. Ramasco, Statistical significance of communities in networks, eprint arXiv:0907.3708.
[401] H.A. David, H.N. Nagaraja, Order Statistics, Wiley-Interscience, Hoboken, USA, 2003.
[402] J. Beirlant, Y. Goegebeur, J. Segers, J. Teugels, Statistics of Extremes: Theory and Applications, 1st edition, Wiley & Sons, Chichester, UK, 2004.
[403] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phys. Rev. E 80 (1) (2009) 016118.
[404] J. Reichardt, M. Leone, (Un)detectable cluster structure in sparse networks, Phys. Rev. Lett. 101 (7) (2008) 078701.

[405] M. Mézard, G. Parisi, The cavity method at zero temperature, J. Stat. Phys. 111 (2003) 1–34.
[406] Y. Fan, M. Li, P. Zhang, J. Wu, Z. Di, Accuracy and precision of methods for community identification in weighted networks, Physica A 377 (2007) 363–372.
[407] J.P. Bagrow, Evaluating local community methods in networks, J. Stat. Mech. P05001.
[408] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.
[409] D.J. Watts, Small Worlds : The Dynamics of Networks between Order and Randomness, Princeton University Press, Princeton, USA, 2003.
[410] E.N. Sawardecker, M. Sales-Pardo, L.A.N. Amaral, Detection of node group membership in networks with group overlap, Eur. Phys. J. B 67 (2009) 277–284.
[411] M. Meilă, Comparing clusterings — An information based distance, J. Multivariate Anal. 98 (5) (2007) 873–895.
[412] D.L. Wallace, A method for comparing two hierarchical clusterings: Comment, J. Am. Assoc. 78 (1983) 569–576.
[413] E.B. Fowlkes, C.L. Mallows, A method for comparing two hierarchical clusterings, J. Am. Assoc. 78 (1983) 553–569.
[414] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Assoc. 66 (336) (1971) 846–850.
[415] B. Mirkin, Mathematical Classification and Clustering, Kluwer Academic Press, Norwell, USA, 1996.
[416] M. Meilă, D. Heckerman, An experimental comparison of model-based clustering methods, Mach. Learn. 42 (1) (2001) 9–29.
[417] S. van Dongen, Performance criteria for graph clustering and Markov cluster experiments, Tech. rep., National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, The Netherlands, 2000.
[418] M. Gustafsson, M. Hörnquist, A. Lombardi, Comparison and validation of community structures in complex networks, Physica A 367 (2006) 559–576.
[419] D. Gusfield, Partition-distance: A problem and class of perfect graphs arising in clustering, Inform. Process. Lett. 82 (3) (2002) 159–164.
[420] R.P. Stanley, Enumerative Combinatorics, vol. I, Cambridge University Press, Cambridge, UK, 1997.
[421] P. Zhang, M. Li, J. Wu, Z. Di, Y. Fan, The analysis and dissimilarity comparison of community structure, Physica A 367 (2006) 577–585.
[422] A.L. Traud, E.D. Kelsic, P.J. Mucha, M.A. Porter, Community structure in online collegiate social networks, eprint arXiv:0809.0690.
[423] L. Donetti, M.A. Muñoz, Improved spectral algorithm for the detection of network communities, in: P. Garrido, J. Maroo, M.A. Muñoz (Eds.), Modeling Cooperative Behavior in the Social Sciences, in: American Institute of Physics Conference Series, vol. 779, 2005, pp. 104–107.
[424] A. Lancichinetti, S. Fortunato, Community detection algorithms: A comparative analysis, eprint arXiv:0908.1062.
[425] D. Delling, M. Gaertler, R. Görke, Z. Nikoloski, D. Wagner, How to evaluate clustering techniques., Tech. rep., Universität Karlsruhe, Germany, 2007.
[426] J. Leskovec, K.J. Lang, A. Dasgupta, M.W. Mahoney, Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, eprint arXiv:0810.1355.
[427] R. Dunbar, Grooming, Gossip, and the Evolution of Language, Harvard Univ Press, Cambridge, USA, 1998.
[428] R. Guimerà, L.A.N. Amaral, Cartography of complex networks: Modules and universal roles, J. Stat. Mech. P02001 (2005).
[429] B.H. Junker, F. Schreiber, Analysis of Biological Networks, Wiley-Interscience, New York, USA, 2008.
[430] R. Dunn, F. Dudbridge, C.M. Sanderson, The use of edge-betweenness clustering to investigate biological function in protein interaction networks, BMC Bioinf. 6 (2005) 39.
[431] V. Farutin, K. Robison, E. Lightcap, V. Dancik, A. Ruttenberg, S. Letovsky, J. Pradines, Edge-count probabilities for the identification of local protein communities and their organization, Proteins 62 (3) (2006) 800–818.
[432] T.Z. Sen, A. Kloczkowski, R.L. Jernigan, Functional clustering of yeast proteins from the protein–protein interaction network, BMC Bioinf. 7 (1) (2006) 355.
[433] A.C.F. Lewis, N.S. Jones, M.A. Porter, C.M. Deane, The function of communities in protein interaction networks, eprint arXiv:0904.0989.
[434] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, G. Sherlock, Gene ontology: Tool for the unification of biology, Nature Genetics 25 (1) (2000) 25–29.
[435] K. Yuta, N. Ono, Y. Fujiwara, A gap in the community-size distribution of a large-scale social networking site, eprint arXiv:physics/0701168.
[436] D.J. de Solla Price, Networks of scientific papers, Science 169 (1965) 510–515.
[437] J. Reichardt, S. Bornholdt, Clustering of sparse data via network communities—a prototype study of a large online market, J. Stat. Mech. 2007 (06) (2007) P06016.
[438] R.K.-X. Jin, D.C. Parkes, P.J. Wolfe, Analysis of bidding networks in ebay: Aggregate preference identification through community detection, in: Proc. AAAI Workshop on Plan, Activity and Intent Recognition, PAIR, 2007, pp. 66–73.
[439] M.A. Porter, P.J. Mucha, M.E.J. Newman, C.M. Warmbrand, A network analysis of committees in the US House of Representatives, Proc. Natl. Acad. Sci. USA 102 (2005) 7057–7062.
[440] Y. Zhang, A.J. Friend, A.L. Traud, M.A. Porter, J.H. Fowler, P.J. Mucha, Community structure in congressional cosponsorship networks, Physica A 387 (7) (2008) 1705–1712.
[441] R.N. Mantegna, Hierarchical structure in financial markets, Eur. Phys. J. B 11 (1999) 193–197.
[442] G. Bonanno, N. Vandewalle, R.N. Mantegna, Taxonomy of stock market indices, Phys. Rev. E 62 (6) (2000) R7615–R7618.
[443] G. Bonanno, G. Caldarelli, F. Lillo, R.N. Mantegna, Topology of correlation-based minimal spanning trees in real and model markets, Phys. Rev. E 68 (4) (2003) 046130.
[444] J.-P. Onnela, A. Chakraborti, K. Kaski, J. Kertész, Dynamic asset trees and portfolio analysis, Eur. Phys. J. B 30 (3) (2002) 285–288.
[445] J.-P. Onnela, A. Chakraborti, K. Kaski, J. Kertész, A. Kanto, Dynamics of market correlations: Taxonomy and portfolio analysis, Phys. Rev. E 68 (5) (2003) 056110.
[446] A.E. Allahverdyan, A. Galstyan, Community detection with and without prior information, eprint arXiv:0907.4803.
[447] F. Chung, Spectral Graph Theory, Number 92 in CBMS Regional Conference Series in Mathematics, American Mathematical Society, Providence, USA, 1997.
[448] M. Barahona, L.M. Pecora, Synchronization in small-world systems, Phys. Rev. Lett. 89 (5) (2002) 054101.
[449] T. Nishikawa, A.E. Motter, Y.-C. Lai, F.C. Hoppensteadt, Heterogeneity in oscillator networks: Are smaller worlds easier to synchronize? Phys. Rev. Lett. 91 (1) (2003) 014101.
[450] R. Solomonoff, A. Rapoport, Connectivity of random nets, Bull. Math. Biophys. 13 (1951) 107–117.
[451] S. Milgram, The small world problem, Psychol. Today 2 (1967) 60–67.
[452] J. Travers, S. Milgram, An experimental study of the small world problem, Sociometry 32 (1969) 425–443.
[453] R. Albert, H. Jeong, A.-L. Barabási, Error and attack tolerance of complex networks, Nature 406 (2000) 378–382.
[454] R. Cohen, K. Erez, D. ben Avraham, S. Havlin, Resilience of the internet to random breakdowns, Phys. Rev. Lett. 85 (21) (2000) 4626–4628.
[455] R. Pastor-Satorras, A. Vespignani, Epidemic spreading in scale-free networks, Phys. Rev. Lett. 86 (14) (2001) 3200–3203.
[456] D.J. de Solla Price, A general theory of bibliometric and other cumulative advantage processes, J. Am. Soc. Inform. Sci. 27 (5) (1976) 292–306.
[457] H.A. Simon, On a class of skew distribution functions, Biometrika 42 (1955) 425–440.