# Community detection in graphs

Santo Fortunato*

*Complex Networks Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino, I-ITALY.*

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



## I. INTRODUCTION

The origin of graph theory dates back to Euler's solution of the puzzle of Königsberg's bridges in 1736 (Euler,

*Electronic address: fortunato@isi.it

1736). Since then a lot has been learned about graphs and their mathematical properties (Bollobas, 1998). In the 20th century they have also become extremely useful as 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 (Scott, 2000; Wasserman and Faust, 1994). 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 (Albert and Barabási, 2002; Barrat *et al.*, 2008; Boccaletti *et al.*, 2006; Mendes and Dorogovtsev, 2003; Newman, 2003; Pastor-Satorras and Vespignani, 2004).

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 P. Erdös and A. Rényi (Erdös and Rényi, 1959). 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 neighbours 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 structure* (Girvan and Newman, 2002), or *clustering*, and is the topic of this review (for earlier reviews see Refs. (Danon *et al.*, 2007; Fortunato and Castellano, 2009; Newman, 2004a; Schaeffer, 2007)). 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 (Coleman, 1964; Freeman, 2004; Kottak, 2004; Moody and White, 2003). Communities also occur in many networked systems from biology,

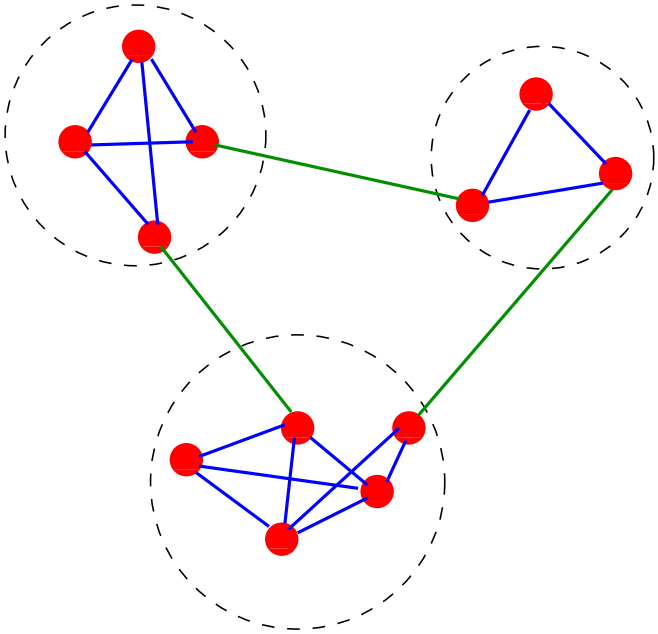

FIG. 1 A simple graph with three communities, enclosed by the dashed circles. Reprinted figure with permission from (Fortunato and Castellano, 2009). ©2009 by Springer.

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 (Chen and Yuan, 2006; Rives and Galitski, 2003; Spirin and Mirny, 2003), in the graph of the World Wide Web they may correspond to groups of pages dealing with the same or related topics (Flake *et al.*, 2002), in metabolic networks they may be related to functional modules such as cycles and pathways (Guimerà and Amaral, 2005; Palla *et al.*, 2005), in food webs they may identify compartments (Krause *et al.*, 2003; Pimm, 1979), and so on.

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. Such classification seems to be meaningful in social (Burt, 1976; Freeman, 1977; Granovetter, 1973) and metabolic networks (Guimerà and Amaral, 2005). 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. Recent studies indi-

cate that networks of communities have a different degree distribution with respect to the full graphs (Palla *et al.*, 2005); however, the origin of their structures can be explained by the same mechanism (Pollner *et al.*, 2006).

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, who 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 (Simon, 1962). 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 (Weiss and Jacobson, 1955), 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 V). 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 (Rice, 1927). 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 (Homans, 1950). This procedure is now standard. Meanwhile, traditional techniques to find communities in social networks are hierarchical clustering and partitional clustering (Sections IV.B and IV.C), 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 IV.A). 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 a few iterations leads to the isolation of the communities (Girvan and Newman, 2002). 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 a 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 II introduces communities in real networks, and is supposed to make the reader acquainted with the problem and its relevance. In Section III 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 and partitional clustering are reviewed in Section IV. Modern methods, divided into categories based on the type of approach, are presented in Sections V to X. Algorithms to find overlapping communities, multiresolution and hierarchical techniques, are separately described in Sections XI and XII, respectively. 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 XIII and XIV are devoted to the issues of defining when community structure is significant, and deciding about the quality of algorithms' performances. In Sections XV and XVI we describe general properties of clusters found in real networks, and specific applications of clustering algorithms. Section XVII 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

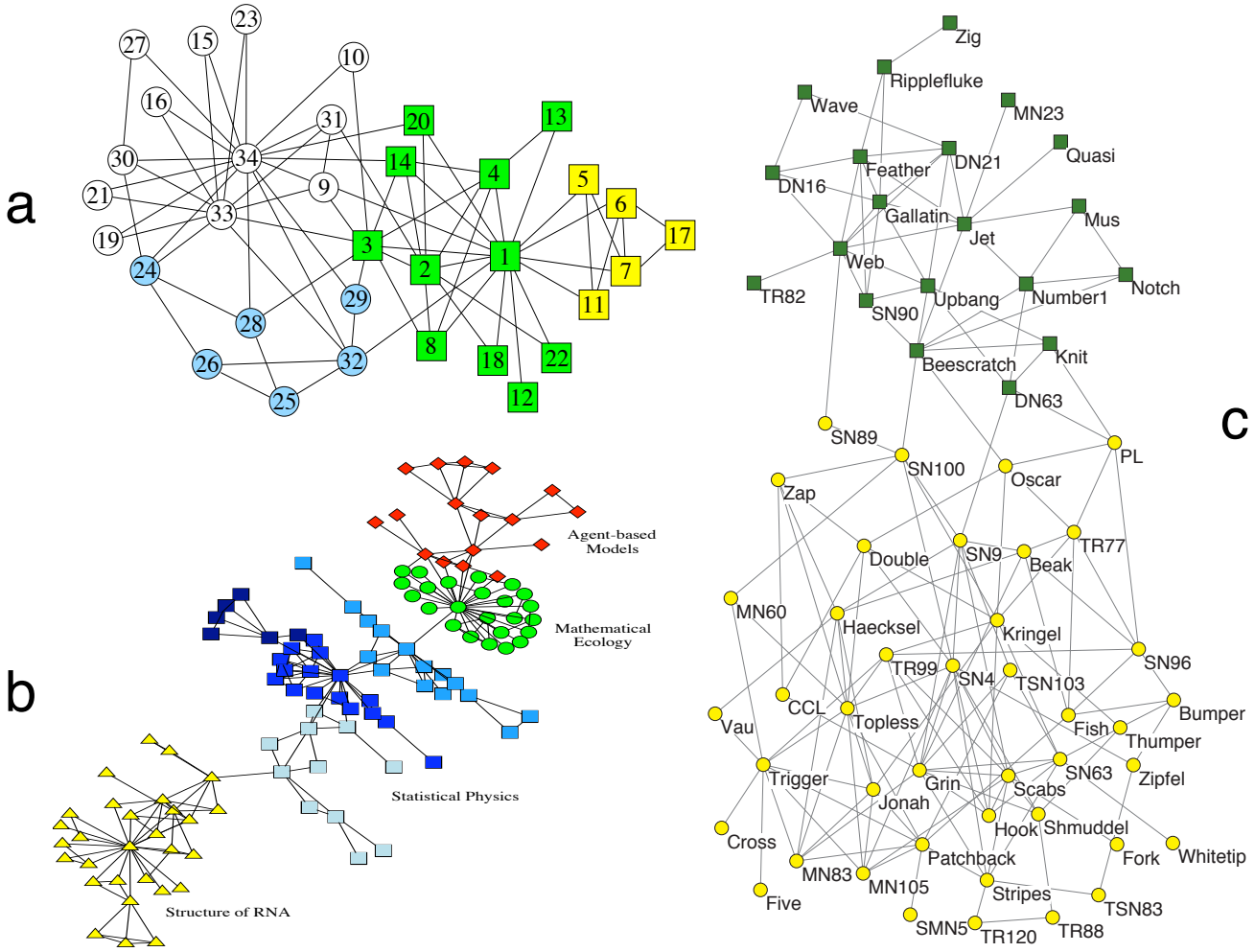


FIG. 2 Community structure in social networks. a) Zachary's karate club, a standard benchmark in community detection. The colors correspond to the best partition found by optimizing the modularity of Newman and Girvan (Section VI.A). Reprinted figure with permission from (Donetti and Muñoz, 2004). ©2004 by IOP Publishing and SISSA. 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 V.A) 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 (Girvan and Newman, 2002). ©2002 by the National Academy of Science of the USA. 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. (Arenas *et al.*, 2008b) (Section XII.A). The partition matches the biological classification of the dolphins proposed by Lusseau. Reprinted figure with permission from (Arenas *et al.*, 2008b). ©2008 by IOP Publishing.

explained in the Appendix. Readers not acquainted with these concepts are urged to read the Appendix first.

## II. 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 (Zachary, 1977), a well-known graph regularly used as a benchmark to test community detection algorithms (Section XIV.A). 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 in two separate groups, supporting the instructor and the president, respectively (indicated by squares

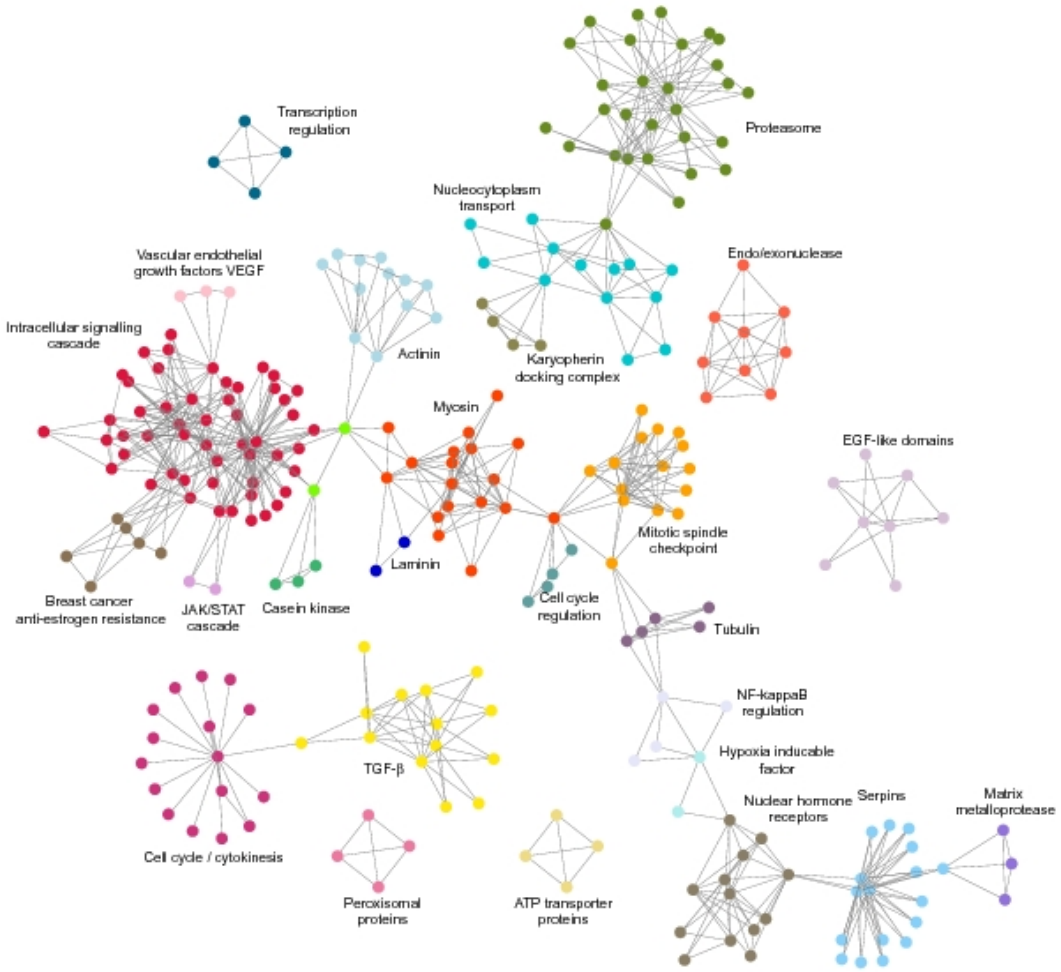


FIG. 3 Community structure in protein-protein interaction networks. The graph pictures the interactions between proteins in cancerous cells of a rat. Communities, labeled by colors, were detected with the k-clique percolation method by Palla et al. (Section XI.A). Reprinted figure with permission from (Jonsson *et al.*, 2006). ©2006 by PubMed Central.

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 (Lusseau, 2003). 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.

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 (Zhang, 2009). Fig. 3 illustrates a PPI network of the rat proteome (Jonsson *et al.*, 2006).

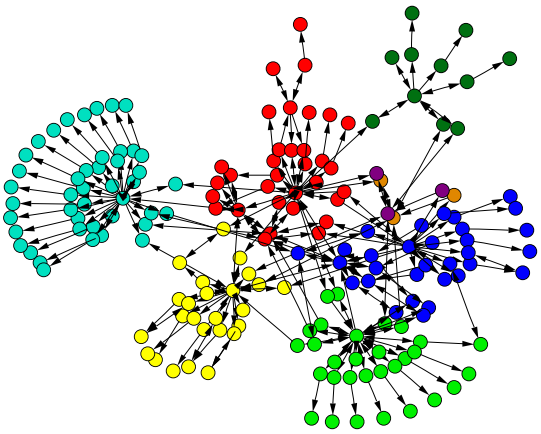

FIG. 4 Community structure in technological networks. Sample of the web graph consisting of the pages of a web site and their mutual hyperlinks, which are directed. Communities, indicated by the colors, were detected with the algorithm of Girvan and Newman (Section V.A), by neglecting the directedness of the edges. Reprinted figure with permission from (Newman and Girvan, 2004). ©2004 by the American Physical Society.

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.

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 (Albert *et al.*, 1999). 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 (Brin and Page, 1998) 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 (Leicht and Newman, 2008; Rosvall and Bergstrom, 2008). 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 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 (Nelson *et al.*, 1998). An edge 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 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

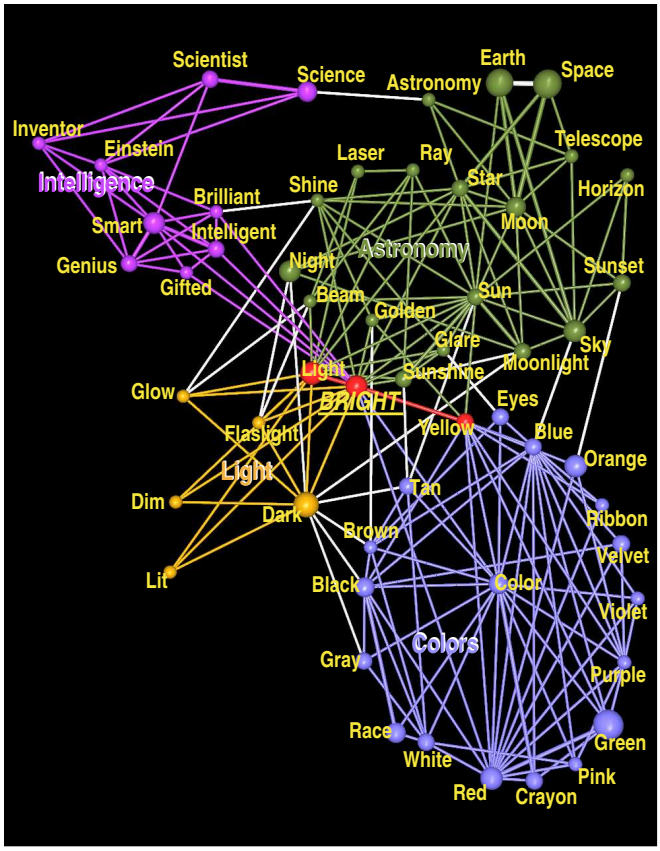


FIG. 5 Overlapping communities in a network of word association. The groups, labeled by the colors, were detected with the $k$-clique percolation method by Palla et al. (Section XI.A). Reprinted figure with permission from (Palla *et al.*, 2005). ©2005 by the Nature Publishing Group.

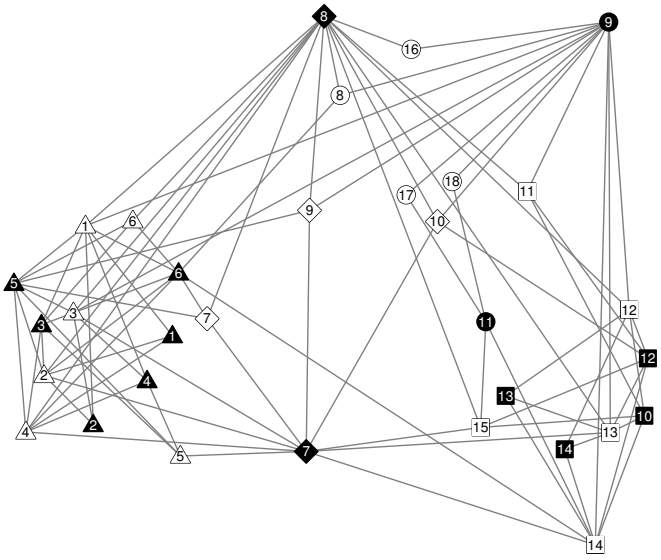

FIG. 6 Community structure in multipartite networks. This bipartite graph refers to the Southern Women Event Participation data set. Women are represented as open symbols with black labels, events as filled symbols with white labels. The illustrated vertex partition has been obtained by maximizing a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

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 coauthorship. 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 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 (Adomavicius and Tuzhilin, 2005). 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. (Davis *et al.*, 1941). 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, including in community detection. In many cases methods working on unweighted graphs can be simply extended to the weighted case.

## III. 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[1]. In this case the problem turns into something rather different, close to data clustering (Gan *et al.*, 2007), 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* and *partitional clustering* will be discussed later in the review (Sections IV.B and IV.C), 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* (Gan *et al.*, 2007).

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.

### A. Computational complexity

The massive amount of data on real networks currently available makes the issue of the speed 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 $O(n)$.

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 Travelling Salesman, Boolean Satisfiability (SAT), Linear Programming, etc. (Garey and Johnson, 1990; Papadimitriou, 1994). 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 **NP**=**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

[1] 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.

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.

## B. Communities

### 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. From intuition and the examples of Section II 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 probably be 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.

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

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

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

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 $\delta(\mathcal{G})$.

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 (Moody and White, 2003; Scott, 2000; Wasserman and Faust, 1994). 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.

### 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 (Wasserman and Faust, 1994). There, four types of criteria 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 (Luce and Perry, 1949) (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 call cliques also 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 hi-

erarchy 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 $k$-clique percolation method of Palla et al. (Palla *et al.*, 2005) (see Section XI.A). From a practical point of view, finding cliques in a graph is an **NP**-complete problem (Bomze *et al.*, 1999). The Bron-Kerbosch method (Bron and Kerbosch, 1973) 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 subgraph such that the distance of each pair of its vertices is not larger than $n$ (Alba, 1973; Luce, 1950). 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 (Mokken, 1979) 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 minumum 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 (Seidman and Foster, 1978). Similarly, a *$k$-core* is a maximal subgraph in which each vertex is adjacent to at least $k$ other vertices of the subgraph (Seidman, 1983). 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 (Batagelj and Zaversnik, 2003). 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 (Matsuda *et al.*, 1999). Determining whether a graph has a $1/2$-quasi complete graph 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* (Luccio and Sami, 1969), or *strong community* (Radicchi *et al.*, 2004), 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 (Radicchi *et al.*, 2004), 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. (Hu *et al.*, 2008) 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 XIV). An $LS$-set is a strong community also 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 cluster 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 (Borgatti *et al.*, 1990). 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 subraph 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 III.C.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 $\xi = 1$ (Garey and Johnson, 1990). 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 (Asahiro *et al.*, 2002; Feige *et al.*, 2001; Holzapfel *et al.*, 2003). 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 (Šíma and Schaeffer, 2006). 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 Section 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 IV.A.

### 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 (Section 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 (although it has recently been shown that this is not true (Guimerà *et al.*, 2004)). 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 each vertex keeps its degree (Newman and Girvan, 2004). This null model is the basic concept behind the definition of *modularity*, a function which evaluates the goodness of partitions of a graph into modules. 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 VI.B). A general class of null models, including modularity as a special case, has been designed by Reichardt and Bornholdt (Reichardt and Bornholdt, 2006a) (Section VI.B).

### 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 the method of hierarchical clustering, to be discussed in Section IV.B. Here we discuss some popular measures used in the literature.

If it is 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 $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$, 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 (Burt, 1976; Wasserman and Faust, 1994) 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 (F.Lorrain and White, 1971). Two vertices are

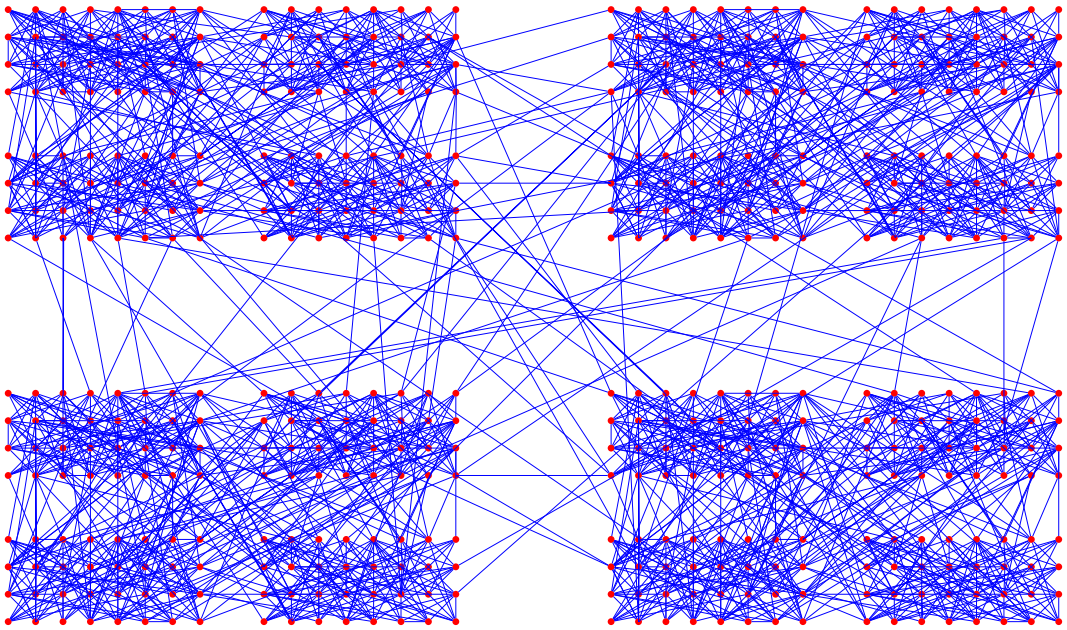

FIG. 7 Schematic example of a hierarchical graph. Sixteen modules with 32 vertices each clearly form four larger clusters. All vertices have degree 64. Reprinted figure with permission from (Lancichinetti *et al.*, 2009). ©2009 by IOP Publishing.

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 neighbours are considered very "far" from each other. Alternatively, one could measure the *overlap* between the neighborhoods of the 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 $\sigma_i = \sum_j (A_{ij} - \mu_i)^2$.

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 (Elias *et al.*, 1956)). The maximum flow can be calculated in a time $O(m)$, for a graph with $m$ edges, using techniques like the augmenting path algorithm (Ahuja *et al.*, 1993). 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, where paths of length $l$ are weighted by the factor $\alpha^l$, with $\alpha < 1$. So, the weights of long paths are exponentially suppressed and the sum converges.

### C. Partitions

#### 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 II, in real systems vertices may be shared among different communities. A division of a graph into overlapping (or *fuzzy*) communities is called *cover*.

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 $k$ (Ravasz and Barabási, 2003; Ravasz *et al.*, 2002).

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

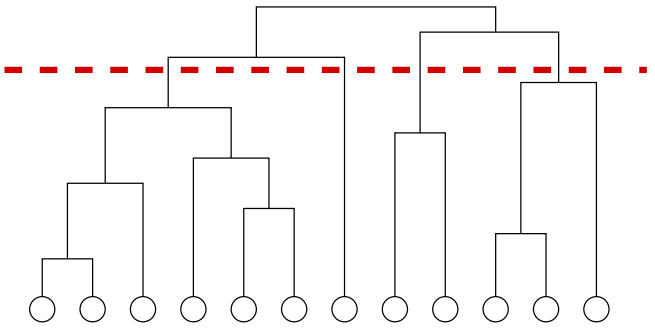

FIG. 8 A dendrogram, or hierarchical tree. Horizontal cuts correspond to partitions of the graph in communities. Reprinted figure with permission from (Newman and Girvan, 2004). ©2004 by the Americal Physical Society.

module (the "leaves" of the tree). By moving upwards, groups of vertices are successively aggregated. Merges 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 IV.B, lends itself naturally to this kind of representation.

### 2. Quality functions: modularity

The number of possible partitions grows faster than exponentially with the size of the graph. 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 (Kleinberg, 2002), 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 (Tumminello *et al.*, 2008), 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[2], 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.

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 IV.B), 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

$$Q(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} q(\mathcal{C}), \tag{10}$$

where $\mathcal{C}$ is a generic cluster of partition $\mathcal{P}$. Eq. 10 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 III.B.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

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

an edge. The definition of performance, for a partition $\mathcal{P}$, is

$$P(\mathcal{P}) = \frac{|\{(i,j) \in E, C_i = C_j| + |\{(i,j) \notin E, C_i \neq C_j|}{n(n-1)/2}. \quad (11)$$

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 (Newman and Girvan, 2004). 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

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta(C_i, C_j), \quad (12)$$

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. 12 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 (Albert and Barabási, 2002; Barrat *et al.*, 2008; Boccaletti *et al.*, 2006; Dorogovtsev and Mendes, 2002; Newman, 2003; Pastor-Satorras and Vespignani, 2004), 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 [3] to the *configuration model*, which has been subject of intense investigations in the recent literature on networks (Łuczak, 1992; Molloy and Reed, 1995). 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

$$Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j). \quad (13)$$

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

$$Q = \sum_{c=1}^{n_c} \left[ \frac{l_c}{m} - \left( \frac{d_c}{2m} \right)^2 \right]. \quad (14)$$

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. 14, 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. 14, and in terms of the inter-cluster edges (Djidjev, 2006). In fact, the maximum of modularity can be expressed as

$$\begin{aligned} Q_{max} &= \max_{\mathcal{P}} \left\{ \sum_{c=1}^{n_c} \left[ \frac{l_c}{m} - \left( \frac{d_c}{2m} \right)^2 \right] \right\} \\ &= \frac{1}{m} \max_{\mathcal{P}} \left\{ \sum_{c=1}^{n_c} \left[ l_c - \mathrm{Ex}(l_c) \right] \right\} \\ &= -\frac{1}{m} \min_{\mathcal{P}} \left\{ -\sum_{c=1}^{n_c} \left[ l_c - \mathrm{Ex}(l_c) \right] \right\}, \end{aligned} \quad (15)$$

[3] 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.

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

$$\begin{aligned} Q_{max} &= -\frac{1}{m}\min_{\mathcal{P}}\left[\left(m - \sum_{c=1}^{n_c} l_c\right) - \left(m - \sum_{c=1}^{n_c} \mathrm{Ex}(l_c)\right)\right] \\ &= -\frac{1}{m}\min_{\mathcal{P}}(|\mathrm{Cut}_{\mathcal{P}}| - \mathrm{ExCut}_{\mathcal{P}}). \end{aligned} \quad (16)$$

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

According to Eq. 14, 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[4]. 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*) (Newman, 2006a). Modularity has been employed as quality function in many algorithms, like some of the divisive algorithms of Section V. In addition, modularity optimization is itself a popular method for community detection (see Section VI.A). Modularity also allows to assess the stability of partitions (Massen and Doye, 2006) (Section XIII), it can be used to design layouts for graph visualization (Noack, 2009) and to perform a sort of renormalization of a graph, by transforming a graph into a smaller one with the same community structure (Arenas *et al.*, 2007).

## IV. TRADITIONAL METHODS

### A. Graph partitioning

The problem of graph partitioning consists in 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.

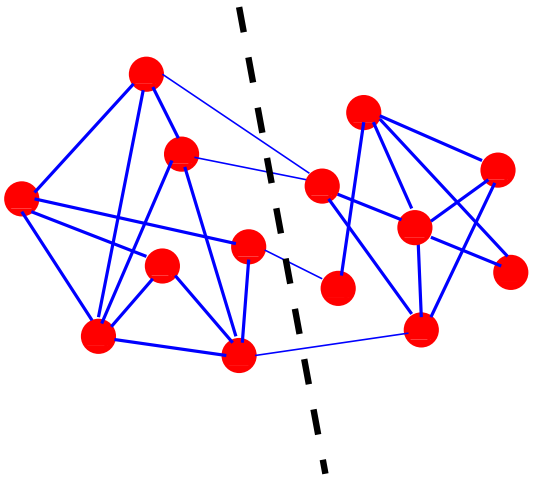

FIG. 9 Graph partitioning. The dashed line shows the solution of the minimum bisection problem for the graph illustrated, i. e. the partition in two groups of equal size with minimal number of edges running between the groups. Reprinted figure with permission from (Fortunato and Castellano, 2009). ©2009 by Springer.

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.

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 (Pothen, 1997). 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* (Kernighan and Lin, 1970) 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 in-

[4] This is not necessarily true, as we will see in Section VI.C.

side 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 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 (Suaris and Kedem, 1988), however the run-time and storage costs increase rapidly with the number of clusters.

Another popular technique is the *spectral bisection method* (Barnes, 1982), which is based on the properties of the Laplacian matrix. In Section A.2 we have seen that the Laplacian of a graph with $g$ connected components has $g$ 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 $g$ 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 $(1, 1, 1, ..., 1, 1)$. Therefore, there are $g$ degenerate eigenvectors with equal non-vanishing components in correspondence of 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.

If the graph is connected, but consists of $g$ subgraphs which are weakly linked to each other, the spectrum will have one zero eigenvalue and $g-1$ eigenvalues which are close to zero. If the groups are two, the second lowest eigenvalue will be close to zero and the corresponding eigenvector, also called *Fiedler vector*, can be used to identify the two clusters as shown below.

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

$$R = \frac{1}{4}\mathbf{s}^T \mathbf{L} \mathbf{s}, \tag{17}$$

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 $\mathbf{v}_i$, $i = 1, ..., n$ are the eigenvectors of the Laplacian. If $\mathbf{s}$ is properly normalized, then

$$R = \sum_i a_i^2 \lambda_i, \tag{18}$$

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. 18. 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 to the Fiedler vector $\mathbf{v}_2$: this would reduce the sum to $\lambda_2$, which is a small number. But the index vector cannot be perfectly parallel to $\mathbf{v}_2$ 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 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 (Lanczos, 1950), that scales as $m/(\lambda_3 - \lambda_2)$, where $m$ is the number of edges of the graph. If the eigenvalues $\lambda_2$ and $\lambda_3$ 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 (Ford and Fulkerson, 1956) 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 (Goldberg and Tarjan, 1988). Flake et al. (Flake *et al.*, 2000; Flake *et al.*, 2002) 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 at. 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 (Radicchi *et al.*, 2004). 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 (Pothen, 1997).

Graphs can be also partitioned by minimizing measures that are affine to the cut size, like *conductance* (Bollobas, 1998). The conductance $\Phi(\mathcal{C})$ of the subgraph $\mathcal{C}$ of a graph $\mathcal{G}$ is defined as

$$\Phi(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{min(k_{\mathcal{C}}, k_{\mathcal{G}\setminus\mathcal{C}})}, \tag{19}$$

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. 19 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. 19, which peaks when the total degrees of the two clusters 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 (Šíma and Schaeffer, 2006). A similar measure is the *cut ratio* (Wei and Cheng, 1989), which is defined as

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

where $n_{\mathcal{C}}$ and $n_{\mathcal{G}\setminus\mathcal{C}}$ are the number of vertices of the two subgraphs. As for the conductance, minimizing the cut ratio favors partitions into clusters of approximately equal size. On the other hand, graph partitioning requires preliminary assumptions on the cluster sizes, whereas the minimization of both the conductance and the cut ratio do not. The optimization of the cut ratio is an **NP**-hard problem (Matula and Shahrokhi, 1990). The cut ratio has been used in several spectral methods for graph partitioning (Chan *et al.*, 1993; Hagen and Kahng, 1992).

Algorithms for graph partitioning are not good for community detection, because it is necessary to provide as input both the number of groups and their size, about which in principle one knows nothing. Instead, one would like an algorithm capable to produce 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.

### B. 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 at hand 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 III.C.1). In such cases, one may use *hierarchical clustering algorithms* (Hastie *et al.*, 2001), 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 III.B.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 are rarely used, 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 III.C.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 (Newman, 2004a). 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 a distance is not trivially defined, the complexity can become much heavier if the calculation of the chosen similarity measure is costly.

### C. 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* (MacQueen, 1967). Here the cost function is the total intra-cluster distance, or squared error function

$$\sum_{i=1}^{k} \sum_{\mathbf{x_j} \in S_i} ||\mathbf{x_j} - \mathbf{c_i}||^2, \tag{21}$$

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 (Lloyd, 1982). 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.

Another popular technique, similar in spirit to $k$-means clustering, is *fuzzy k-means clustering* (Bezdek, 1981; Dunn, 1973). 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

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^m ||\mathbf{x_i} - \mathbf{c_j}||^2, \tag{22}$$

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 $j$

$$\mathbf{c_j} = \frac{\sum_{i=1}^{n} u_{ij}^m \mathbf{x_i}}{\sum_{i=1}^{n} u_{ij}^m}. \quad (23)$$

The matrix $u_{ij}$ is normalized so that the sum of the memberships of every point in all clusters yields one. 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

$$u_{ij} = \frac{1}{\sum_{l=1}^{k} \left( \frac{||\mathbf{x_i} - \mathbf{c_j}||}{||\mathbf{x_i} - \mathbf{c_l}||} \right)^{\frac{2}{m-1}}}. \quad (24)$$

The cost function $J_m$ can be minimized by iterating Eqs. 23 and 24. One starts from some initial guess for $u_{ij}$ and uses Eq. 23 to compute the centers, which are then plugged back into Eqs. 24, 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. 22. 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.

## V. 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 IV.B). 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* 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.

### A. The algorithm of Girvan and Newman

The most popular algorithm is that proposed by Girvan and Newman (Girvan and Newman, 2002; Newman and Girvan, 2004). 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 (Freeman, 1977) and expresses the importance of edges in processes like information spreading, where information usually flows through shortest paths. Historically edge betweenness was introduced before site betweenness in a never published technical report by Anthonisse (Anthonisse, 1971). 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 (Brandes, 2001; Newman and Girvan, 2004; Zhou *et al.*, 2006).

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

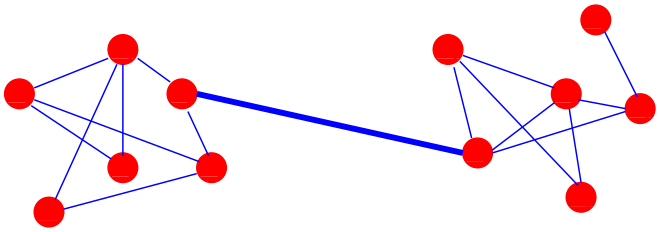

FIG. 10 Edge betweenness is highest for edges connecting communities. In the figure, the edge in the middle has a much higher betweenness than all other edges, because all shortest paths connecting vertices of the two communities run through it. Reprinted figure with permission from (Fortunato and Castellano, 2009). ©2009 by Springer.

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 (Newman, 2005). 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 (Newman and Girvan, 2004). 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 $n \sim 10000$ vertices, with current computational resources. In the original version of Girvan-Newman's algorithm (Girvan and Newman, 2002), 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 (Newman and Girvan, 2004), they selected the partition with the largest value of modularity (see Section III.C.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 (Newman, 2004). There have been countless applications of the Girvan-Newman method: the algorithm is now integrated in well known libraries of network analysis programs.

Tyler, Wilkinson and Huberman proposed a modification of the Girvan-Newman algorithm, to improve the speed of the calculation (Tyler *et al.*, 2003; Wilkinson and Huberman, 2004). 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 (Brandes, 2001; Newman and Girvan, 2004; Zhou *et al.*, 2006). 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 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 structures 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 (Wilkinson and Huberman, 2004), with a substantial gain of computer time. The technique has been also applied to a network of people corresponding via email (Tyler *et al.*, 2003). 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 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. 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 (Chen and Yuan, 2006). Holme et al. have used a modified version of the algorithm in which vertices, rather than edges, are removed (Holme *et al.*, 2003). 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 (Pinney and Westhead, 2006). 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 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 endvertices 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 (Gregory, 2007) 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`.

### B. 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 (Radicchi *et al.*, 2004). The edge clustering coefficient generalizes to edges the notion of clustering coefficient introduced by Watts and Strogatz for vertices (Watts and Strogatz, 1998) (Fig. 11). In Section A.1 we have seen that 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. The edge clustering coefficient is defined as

$$\tilde{C}_{i,j}^{(g)} = \frac{z_{i,j}^{(g)} + 1}{s_{i,j}^{(g)}}, \tag{25}$$

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

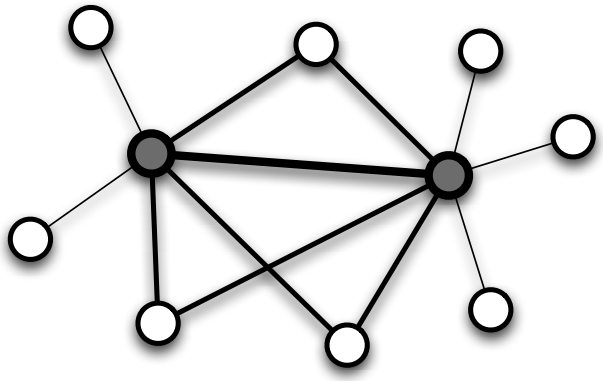

FIG. 11 Schematic illustration of the edge clustering coefficient introduced by Radicchi et al. (Radicchi *et al.*, 2004). The two grey vertices have five and six other neighbors, respectively. Of the five possible triangles based on the edge connecting the grey vertices, three are actually there, yielding an edge clustering coefficient $C^3 = 3/5$. Courtesy by F. Radicchi.

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 III.B.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 more 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 (C. Castellano *et al.*, 2004) the authors extended the method to the case of weighted networks, by modifying the edge clustering coefficient of Eq. 25, 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 (Zhang *et al.*, 2007), 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 all 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 (Latora and Marchiori, 2001), 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, Latora and Marchiori (Fortunato *et al.*, 2004), 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 (Watts and Strogatz, 1998), has been introduced by Vragovîc and Louis (Vragović and Louis, 2006). 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. Vragovîc 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. (Fortunato *et al.*, 2004). 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.

## VI. MODULARITY-BASED METHODS

Newman-Girvan modularity $Q$ (Section III.C.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 (Clauset *et al.*, 2004; Newman, 2004b; Noack and Rotta, 2008; Schuetz and Caflisch, 2008a,a; Wakita and Tsurumi, 2007); more accurate methods, which are computationally demanding (Guimerà *et al.*, 2004; Massen and Doye, 2005; Medus *et al.*, 2005); algorithms giving a good tradeoff between high accuracy and low complexity (Duch and Arenas, 2005; Lehmann and Hansen, 2007; Newman, 2006b). We shall also point out other properties of modularity, discuss some extensions/modifications of it, as well as highlight its limits.

### A. Modularity optimization

By assumption, high values of modularity indicate good partitions[5]. 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 maximization, perhaps 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 (Brandes *et al.*, 2006), 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.

#### 1. Greedy techniques

The first algorithm devised to maximize modularity was a greedy method of Newman (Newman, 2004b). 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

[5] This is not true in general, as we shall discuss in Section VI.C.

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 100000 vertices with current computers). In a later paper (Clauset *et al.*, 2004), 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 to analyse 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 (Danon *et al.*, 2006). 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 (Wakita and Tsurumi, 2007) 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 (Schuetz and Caflisch, 2008a,b). 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 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 (Du *et al.*, 2007; Pujol *et al.*, 2006). Higher-quality modularities can be also achieved by applying refinement strategies based on local search at various steps of the greedy agglomeration (Noack and Rotta, 2008). Such refinement procedures are similar to the technique proposed by Newman to improve the results of his spectral optimization of modularity ((Newman, 2006b) and Section VI.A.4).

A different greedy approach has been introduced by Blondel et al. (Blondel *et al.*, 2008), 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. 33) 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 en-

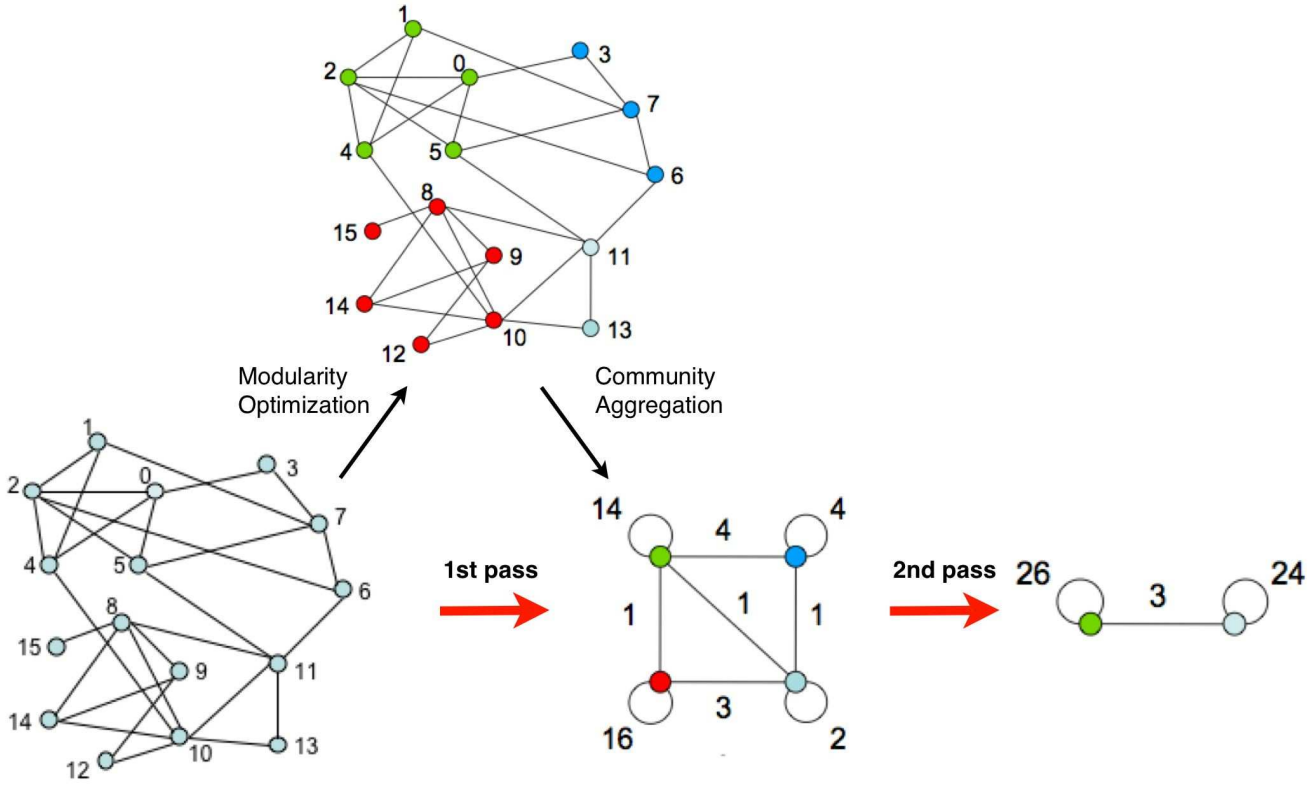


FIG. 12 Hierarchical optimization of modularity by Blondel et al. (Blondel *et al.*, 2008). The diagram shows two iterations of the method, starting from the graph on the left. Each iteration consists of a step, in which every vertex is assigned to the (local) cluster that produces the largest modularity increase, followed by a successive transformation of the clusters into vertices of a (weighted) graph, representing the next higher hierarchical level. Reprinted figure with permission from (Blondel *et al.*, 2008). ©2008 by IOP Publishing and SISSA.

ables 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 anymore, 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. (Clauset *et al.*, 2004) and Wakita and Tsurumi (Wakita and Tsurumi, 2007). 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.

### 2. Simulated annealing

Simulated annealing (Kirkpatrick *et al.*, 1983) is a probabilistic procedure for global optimization used in different fields and problems. It consists in 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. (Guimerà *et al.*, 2004). Its standard implementation (Guimerà and Amaral, 2005) combines two types of "moves": 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 (Massen and Doye, 2005; Medus *et al.*, 2005). 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.

### 3. Extremal optimization

Extremal optimization (EO) is a heuristic search procedure proposed by Boettcher and Percus (Boettcher and Percus, 2001), 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 (Duch and Arenas, 2005). 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 (Kernighan and Lin, 1970) algorithm for graph partitioning (Section IV.A), 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, (Boettcher and Percus, 2001)). The algorithm finds very good estimates of the modularity maximum, and performs very well on the benchmark of Girvan and Newman (Girvan and Newman, 2002) (Section XIV.A) . 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.

### 4. Spectral optimization

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

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{26}$$

Here the notation is the same used in Eq. 13. 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

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j) \\ &= \frac{1}{4m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) (s_i s_j + 1) \\ &= \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}. \end{aligned} \tag{27}$$

The last expression indicates standard matrix products. The vector $\mathbf{s}$ can be decomposed on the basis of eigenvectors $\mathbf{u}_i$ ($i = 1, ..., n$) 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. 27 one finally gets

$$Q = \frac{1}{4m} \sum_i a_i \mathbf{u}_i^T \mathbf{B} \sum_j a_j \mathbf{u}_j = \frac{1}{4m} \sum_{i=1}^{n} (\mathbf{u}_i^T \cdot \mathbf{s})^2 \beta_i, \tag{28}$$

where $\beta_i$ is the eigenvalue of $\mathbf{B}$ corresponding to the eigenvector $\mathbf{u}_i$. Eq. 28 is analogous to Eq. 18 for the cut size of the graph partitioning problem. This suggests that one can optimize modularity on bipartitions via spectral bisection (Section IV.A), by replacing the Laplacian matrix with the modularity matrix (Newman, 2006a,b). Like the Laplacian matrix, $\mathbf{B}$ has always the

trivial eigenvector $(1, 1, ..., 1)$ with eigenvalue zero, because the sum of the elements of each row/column of the matrix vanishes. From Eq. 28 we see that, if $\mathbf{B}$ 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 IV.A. 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 well 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 IV.A), 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[6]. 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. (Sun *et al.*, 2009) 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

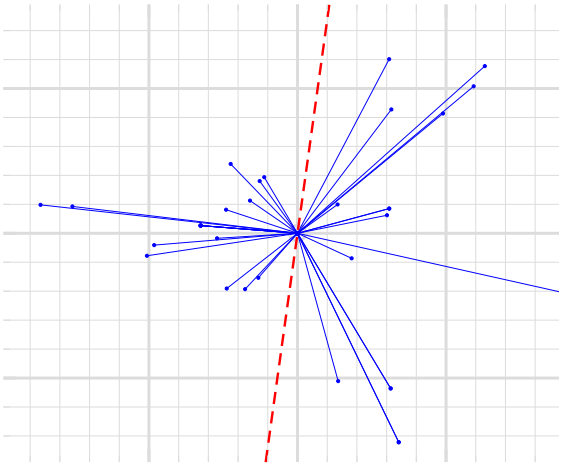

FIG. 13 Spectral optimization of modularity by Newman (Newman, 2006a,b). By using the first two eigenvectors of the modularity matrix, vertices can be represented as points on a plane. By cutting the plane with a line passing through the origin (like the dotted line in the figure) one obtains bipartitions of the graph with possibly high modularity values. Reprinted figure with permission from (Newman, 2006a). ©2006 by the American Physical Society.

matrix are used. Given the first $p$ eigenvectors, one can construct $n$ $p$-dimensional vectors, each corresponding to a vertex: 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. 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 (Richardson *et al.*, 2008). 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 $\mathbf{B}$ 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

[6] Richardson et al. (Richardson *et al.*, 2008) 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.

time $O(n^2)$, as $\mathbf{B}$ is a complete matrix, but the peculiar form of $\mathbf{B}$ 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 (White and Smyth, 2005). Let $\mathbf{W}$ indicate the weighted adjancency matrix of a graph $\mathcal{G}$. A partition of $\mathcal{G}$ in $k$ clusters can be described through an $n \times k$ assignment matrix $\mathbf{X}$, 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

$$Q \propto \mathrm{tr}[\mathbf{X}^T(\mathcal{W} - \mathcal{D})\mathbf{X}] = -\mathrm{tr}[\mathbf{X}^T\mathbf{L_Q}\mathbf{X}], \qquad (29)$$

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

$$\mathbf{L_Q}\mathbf{X} = \mathbf{X}\mathbf{\Lambda}. \qquad (30)$$

Here $\mathbf{\Lambda}$ is a diagonal matrix. A nice feature of the $Q$-Laplacian is that, for graphs which are not too small, it can be approximated by the transition matrix $\tilde{\mathcal{W}}$, obtained by normalizing $\mathcal{W}$ such that the sum of the elements of each row equals one. Eq. 30 is at the basis of the algorithms developed by White and Smyth, which search for partitions with at most $K$ clusters, where $K$ is a predefined input parameter that may be suggested by preliminary information on the graph cluster structure. The first $K-1$ eigenvectors of the transition matrix $\tilde{\mathcal{W}}$ can be computed with a variant of the Lanczos method (Demmel *et al.*, 2000). Since the eigenvector components are not integer, the eigenvectors do not correspond directly to a partition of the graph in clusters. However, the components of the eigenvectors can be used as coordinates of the graph vertices in an Euclidean space and $k$-means clustering is applied to obtain the desired partition. White and Smyth proposed two methods to derive the clustering after embedding the graph in space. Both methods have a worst-case complexity $O(K^2 n + Km)$, which is essentially linear in the number of vertices of the graph if the latter is sparse and $K \ll n$.

### 5. Other optimization strategies

Agarwal and Kempe have suggested to maximize modularity within the framework of mathematical programming (Agarwal and Kempe, 2008). In fact, modularity optimization can be formulated both as a linear and as a quadratic program. In the first case, the variables are defined on the links: $x_{ij} = 0$ if $i$ and $j$ are in the same cluster, otherwise $x_{ij} = 1$. The modularity of a partition, up to a multiplicative constant, can then be written as

$$Q \propto \sum_{ij} B_{ij}(1 - x_{ij}), \qquad (31)$$

where $\mathbf{B}$ is the modularity matrix defined by Newman (see Section VI.A.4). Eq. 31 is linear in the variables $\{x\}$, which must obey the constraint $x_{ij} \leq x_{ik} + x_{kj}$, because, if $i$ and $j$ are in the same cluster, and so are $i$ and $k$, then $j$ and $k$ must be in that cluster too. Maximizing the expression in Eq. 31 under the above constraint is $NP$-hard, if the variables have to be integer as required. However, if one relaxes this condition by using real-valued $\{x\}$, the problem can be solved in polynomial time (Karloff, 1991). On the other hand, the solution does not correspond to an actual partition, as the $x$ variables are fractional. To get clusters out of the $\{x\}$ one needs a rounding step. The values of the $x$ variables are used as sort of distances in a metric space (the triangular inequality is satisfied by construction): clusters of vertices "close" enough to each other (i.e. whose mutual $x$ variables are close to zero) are formed and removed until each vertex is assigned to a cluster. The resulting partition is further refined with the same post-processing technique used by Newman for the spectral optimization of modularity, i.e. by a sequence of steps similar to those of the algorithm by Kernighan and Lin (see Section VI.A.4). Quadratic programming can be used to get bisections of graphs with high modularity, that can be iterated to get a whole hierarchy of partitions as in Newman's spectral optimization. One starts from one of the identities in Eq. 27

$$Q = \frac{1}{2m}\sum_{ij} B_{ij}(1 + s_i s_j), \qquad (32)$$

where $s_i = \pm 1$, depending on whether the vertex belongs to the first or the second cluster. Since the optimization of the expression 32 is $NP$-complete, one must relax again the constraint on the variables $s$ being integer. A possibility is to transform each $s$ into an $n$-dimensional vector $\mathbf{s}$ and each product in the scalar product between vectors. The vectors are normalized so that their tips lie on the unit-sphere of the $n$-dimensional space. This vector problem is polynomially solvable, but

one needs a method to associate a bipartition to the set of $n$ vectors of the solution. Any $(n-1)$-dimensional hyperplane centered at the origin cuts the space in two halves, separating the vectors in two subsets. One can then choose multiple random hyperplanes and pick the one which delivers the partition with highest modularity. As in the linear program, a post-processing technique á la Newman (see Section VI.A.4) is used to improve the results of the procedure. The two methods proposed by Agarwal and Kempe are strongly limited by their high computational complexity, due mostly to the large storage demands, making graphs with more than $10^4$ vertices intractable. On the other hand, the idea of applying mathematical programming to graph clustering is promising. The code of the algorithms can be downloaded from `http://www-scf.usc.edu/~gaurava/`. In a recent work (G. Xu *et al.*, 2007), Xu et al. have optimized modularity using mixed-integer mathematical programming, with both integer and continuous variables, obtaining very good approximations of the modularity optimum, at the price of high computational costs. Chen et al. have used integer linear programming to transform the initial graph into an optimal target graph consisting of disjoint cliques, which effectively yields a partition (Chen *et al.*, 2008). Berry et al. have formulated the problem of graph clustering as a *facility location problem* (Hillier and Lieberman, 2004), consisting in the minimization of a cost function based on a local variation of modularity (Berry *et al.*, 2007).

Lehmann and Hansen (Lehmann and Hansen, 2007) optimized modularity via mean field annealing (Peterson and Anderson, 1987), a deterministic alternative to simulated annealing (Kirkpatrick *et al.*, 1983). The method uses Gibbs probabilities to compute the conditional mean value for the variable of a vertex, which indicates its community membership. By making a mean field approximation on the variables of the other vertices in the Gibbs probabilities one derives a self-consistent set of non-linear equations, that can be solved by iteration in a time $O(m+n)n$. The method yields better modularity maxima than the spectral optimization by Newman (Section VI.A.4), at least on artificial graphs with built-in community structure, similar to the benchmark graphs by Girvan and Newman (Section XIV.A).

Genetic algorithms (Holland, 1992) have also been used to optimize modularity. In a standard genetic algorithm one has a set of candidate solutions to a problem, which are numerically encoded as chromosomes, and an objective function to be optimized on the space of solutions. The objective function plays the role of biological fitness for the chromosomes. One usually starts from a random set of candidate solutions, which are progressively changed through manipulations inspired by biological processes regarding real chromosomes, like point mutation (random variations of some parts of the chromosome) and crossing over (generating new chromosomes by merging parts of existing chromosomes). Then, the fitness of the new pool of candidates is computed and the chromosomes with the highest fitness have the greatest chances to survive in the next generation. After several iterations only solutions with large fitness survive. In a work by Tasgin et al. (Tasgin *et al.*, 2007), partitions are the chromosomes and modularity is the fitness function. With a suitable choice of the algorithm parameters, like the number of chromosomes and the rates of mutation and crossing over, Tasgin et al. could obtain results of comparative quality as greedy modularity optimization on Zachary's karate club (Zachary, 1977), the college football network (Girvan and Newman, 2002) and the benchmark by Girvan and Newman (Section XIV.A).

In Section III.C.2 we have seen that the modularity maximum is obtained for the partition that minimizes the difference between the cut size and the expected cut size of the partition (Eq. 16). In the complete weighted graph $\mathcal{G}_w$ such that the weight $w_{ij}$ of an edge is $1 - k_i k_j/2m$, if $i$ and $j$ are connected in $\mathcal{G}$, and $-k_i k_j/2m$ if they are not, the difference $|\mathrm{Cut}_{\mathcal{P}}| - \mathrm{ExCut}_{\mathcal{P}}$ is simply the cut size of partition $\mathcal{P}$. So, maximizing modularity for $\mathcal{G}$ is equivalent to the problem of finding the partition with minimal cut size of the weighted graph $\mathcal{G}_w$, i.e. to a graph partitioning problem. The problem can then be efficiently solved by using existing software for graph partitioning (Djidjev, 2006).

### B. Modifications of modularity

In the most recent literature on graph clustering several modifications and extensions of modularity can be found. They are usually motivated by specific classes of clustering problems and/or graphs that one may want to analyze.

Modularity can be easily extended to graphs with weighted edges (Newman, 2004). One needs to replace the degrees $k_i$ and $k_j$ in Eq. 13 with the strengths $s_i$ and $s_j$ of vertices $i$ and $j$. We remind that the strength of a vertex is the sum of the weights of edges adjacent to the vertex (Section A.1). For a proper normalization, the number of edges $m$ in Eq. 13 has to be replaced by the sum $W$ of the weights of all edges. The product $s_i s_j/2W$ is now the expected weight of the edge $ij$ in the null model of modularity, which has to be compared with the actual weight $W_{ij}$ of that edge in the original graph. This can be understood if we consider the case in which all weights are multiples of a unit weight, so they can be rewritten as integers. The weight of the connection between two nodes can then be replaced by as many edges between the nodes as expressed by the number of weight units. For the resulting multigraph we can use the same procedure as in the case of unweighted graphs, which leads to the formally identical expression

$$Q_w = \frac{1}{2W} \sum_{ij} \left( W_{ij} - \frac{s_i s_j}{2W} \right) \delta(C_i, C_j), \qquad (33)$$

which can be also written as a sum over the modules

$$Q = \sum_{c=1}^{n_c} \left[ \frac{W_c}{W} - \left( \frac{S_c}{2W} \right)^2 \right], \qquad (34)$$

where $W_c$ is the sum of the weights of the internal edges of module $c$ and $S_c$ is the sum of the strengths of the vertices of $c$. If edge weights are not mutually commensurable, one can always represent them as integers with good approximation, provided a sufficiently small weight unit is adopted, so the expressions for weighted modularity of Eqs. 33,34 are generally valid.

Modularity has also a straightforward extension to the case of directed graphs (Arenas *et al.*, 2007; Leicht and Newman, 2008). If an edge is directed, the probability that it will be oriented in either of the two possible directions depends on the in- and out-degrees of the endvertices. For instance, taken two vertices $A$ and $B$, where $A$ ($B$) has a high (low) indegree and low (high) outdegree, in the null model of modularity an edge will be much more likely to point from $B$ to $A$ than from $A$ to $B$. Therefore, the expression of modularity for directed graphs reads

$$Q_d = \frac{1}{m} \sum_{ij} \left( A_{ij} - \frac{k_i^{out} k_j^{in}}{m} \right) \delta(C_i, C_j), \qquad (35)$$

where the factor 2 in the denominator of the second summand has been dropped because the sum of the indegrees (outdegrees) equals $m$, whereas the sum of the degrees of the vertices of an undirected graph equals $2m$; the factor 2 in the denominator of the prefactor has been dropped because the number of non-vanishing elements of the adjacency matrix is $m$, not $2m$ as in the symmetric case of an undirected graph. If a graph is both directed and weighted, formulas 33 and 35 can be combined as

$$Q_{gen} = \frac{1}{W} \sum_{ij} \left( W_{ij} - \frac{s_i^{out} s_j^{in}}{W} \right) \delta(C_i, C_j), \qquad (36)$$

which is the most general (available) expression of modularity (Arenas *et al.*, 2007). Kim et al. (Kim *et al.*, 2009) have remarked that the directed modularity of Eq. 35 may not properly account for the directedness of the edges (Fig. 14), and proposed a different definition based on diffusion on directed graphs, inspired by Google's PageRank algorithm (Brin and Page, 1998).

If vertices may belong to more clusters, it is not obvious how to find a proper generalization of modularity. In fact, there is no unique recipe. Shen et al. (Shen *et al.*, 2009), for instance, suggested the simple definition

$$Q = \frac{1}{2m} \sum_{ij} \frac{1}{O_i O_j} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j). \qquad (37)$$

Here $O_i$ is the number of communities including vertex $i$. The contribution of each edge to modularity is then the smaller, the larger the number of communities including its endvertices. Nicosia et al. (Nicosia *et al.*, 2009) have made some more general considerations on the problem of extending modularity to the case of overlapping communities. They considered the case of directed unweighted networks, starting from the following general expression

$$Q_{ov} = \frac{1}{m} \sum_{c=1}^{n_c} \sum_{i,j} \left[ r_{ijc} A_{ij} - s_{ijc} \frac{k_i^{out} k_j^{in}}{m} \right], \qquad (38)$$

where $k_i^{in}$ and $k_j^{out}$ are the indegree and outdegree of vertices $i$ and $j$, the index $c$ labels the communities and $r_{ijc}$, $s_{ijc}$ express the contributions to the sum corresponding to the edge $ij$ in the network and in the null model, due to the multiple memberships of $i$ and $j$. If there is no overlap between the communities, $r_{ijc} = s_{ijc} = \delta_{c_i c_j c}$, where $c_i$ and $c_j$ correspond to the communities of $i$ and $j$. In this case, the edge $ij$ contributes to the sum only if $c_i = c_j$, as in the original definition of modularity. For overlapping communities, the coefficients $r_{ijc}$, $s_{ijc}$ must depend on the membership coefficients $\alpha_{i,c}$, $\alpha_{j,c}$ of vertices $i$ and $j$. One can assume that $r_{ijc} = \mathcal{F}(\alpha_{i,c}, \alpha_{j,c})$, where $\mathcal{F}$ is some function. The term $s_{ijc}$ is related to the null model of modularity, and it must be handled with care. In modularity's original null model edges are formed by joining two random stubs, so one needs to define the membership of a random stub in the various communities. If we assume that there is no correlation *a priori* between the membership coefficients of any two vertices we can assign to a stub originating from a vertex $i$ in community $c$ the average membership corresponding to all edges which can be formed with $i$. On a directed

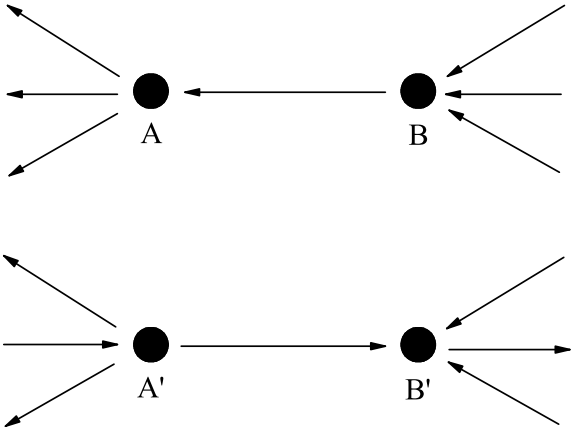


FIG. 14 Problem of the directed modularity introduced by Arenas et al. (Arenas *et al.*, 2007). The two situations illustrated are equivalent for modularity, as vertices $A$ and $A'$, as well as $B$ and $B'$, have identical indegrees and outdegrees. In this way, the optimization of directed modularity is not able to distinguish a situation in which there is directed flow (top) or not (bottom). Reprinted figure with permission from (Kim *et al.*, 2009).

graph we have to distinguish between outgoing and incoming stubs, so one has

$$\beta^{out}_{i\rightarrow,c} = \frac{\sum_j \mathcal{F}(\alpha_{i,c},\alpha_{j,c})}{n}, \quad (39)$$

$$\beta^{in}_{i\leftarrow,c} = \frac{\sum_j \mathcal{F}(\alpha_{j,c},\alpha_{i,c})}{n}, \quad (40)$$

and one can write the following general expression for modularity

$$Q_{ov} = \frac{1}{m}\sum_{c=1}^{n_c}\sum_{i,j}\left[r_{ijc}A_{ij} - \frac{\beta^{out}_{i\rightarrow,c}k^{out}_i\beta^{in}_{j\leftarrow,c}k^{in}_j}{m}\right]. \quad (41)$$

The question now concerns the choice of the function $\mathcal{F}(\alpha_{i,c},\alpha_{j,c})$. If the formula of Eq. 41 is to be an extension of modularity to the case of overlapping communities, it has to satisfy some general properties of classical modularity. For instance, the modularity value of a cover consisting of a single cluster the whole network should be zero. It turns out that a large class of functions yield an expression for modularity that fulfills this requirement. Otherwise, the choice of $\mathcal{F}$ is rather arbitrary and good choices can be only tested *a posteriori*, based on the results of the optimization.

Gaertler et al. have introduced quality measures based on modularity's principle of the comparison between a variable relative to the original graph and the corresponding variable of a null model (Gaertler *et al.*, 2007). They remark that modularity is just the difference between the coverage of a partition and the expected coverage of the partition in the null model. We remind that the coverage of a partition is the ratio between the number of edges within clusters and the total number of edges (Section III.C.2). Based on this observation, Gaertler et al. suggest that the comparison between the two terms can be done with other binary operations as well. For instance, one could consider the ratio

$$S^{\div}_{cov} = \frac{\sum_{c=1}^{n_c} l_c/m}{\sum_{c=1}^{n_c}(d_c/2m)^2}, \quad (42)$$

where the notation is the same as in Eq. 14. This can be done as well for any variable other than coverage. By using performance, for instance, (Section III.C.2) one obtains two new quality functions $S^{-}_{perf}$ and $S^{\div}_{perf}$, corresponding to taking the difference or the ratio between performance and its null model expectation value, respectively. Gaertler et al. compared the results obtained with the four functions $S^{-}_{cov} = Q$, $S^{\div}_{cov}$, $S^{-}_{perf}$ and $S^{\div}_{perf}$, on a class of benchmark graphs with built-in cluster structure (Section XIV.A) and social networks. They found that the "absolute" variants $S^{-}_{cov}$ and $S^{-}_{perf}$ are better than the "relative" variants $S^{\div}_{cov}$ and $S^{\div}_{perf}$ on the artificial benchmarks, whereas $S^{\div}_{perf}$ is better on social networks[7]. Furthermore $S^{-}_{perf}$ is better than the standard modularity $S^{-}_{cov}$.

Modifications of modularity's null model have been introduced by Massen and Doye (Massen and Doye, 2005) and Muff et al. (Muff *et al.*, 2005). Massen and Doye's null model is still a graph with the same expected degree sequence as the original, and with edges rewired at random among the vertices, but one imposes the additional constraint that there can be neither multiple edges between a pair of vertices nor edges joining a vertex with itself (loops or self-edges). This null model is more realistic, as multiple edges and loops are usually absent in real graphs. The maximization of the corresponding modified modularity yields partitions with smaller average cluster size than standard modularity. The latter tends to disfavor small communities, because the actual densities of edges inside small communities hardly exceed the null model densities, which are appreciably enhanced by the contributions from multiple connections and loops. Muff et al. proposed a local version of modularity, in which the expected number of edges within a module is not calculated with respect to the full graph, but considering just a portion of it, namely the subgraph including the module and its neighbouring modules. Their motivation is the fact that modularity's null model implicitly assumes that each vertex could be attached to any other, whereas in real cases a cluster is usually connected to few other clusters. On a directed graph, their *localized modularity* $LQ$ reads

$$LQ = \sum_{c=1}^{n_c}\left[\frac{l_c}{L_{c_n}} - \frac{d^{in}_c d^{out}_c}{L^2_{c_n}}\right]. \quad (43)$$

In Eq. 43 $l_c$ is the number of edges inside cluster $c$, $d^{in}_c$ ($d^{out}_c$) the total internal (external) degree of cluster $c$ and $L_{c_n}$ the total number of edges in the subgraph comprising cluster $c$ and its neighbor clusters. The localized modularity is not bounded by 1, but can take any value. Its maximization delivers more accurate partitions than standard modularity optimization on a model network describing the social interactions between children in a school (school network) and on the metabolic and protein-protein interaction networks of *E. coli*.

Reichardt and Bornholdt have shown that it is possible to reformulate the problem of community detection as the problem of finding the ground state of a spin glass model (Reichardt and Bornholdt, 2006a). Each vertex $i$ is labeled by a Potts spin variable $\sigma_i$, which indicates the cluster including the vertex. The basic principle of the model is that edges should connect vertices of the same class (i.e. same spin state), whereas vertices of different classes (i.e. different spin states) should be disconnected (ideally). So, one has to energetically favor edges between vertices in the same class, as well as non-edges between

[7] The comparison was done by computing the values of significance indices like coverage and performance on the final partitions.

vertices in different classes, and penalize edges between vertices of different classes, along with non-edges between vertices in the same class. The resulting Hamiltonian of the spin model is

$$\begin{aligned}\mathcal{H}(\{\sigma\}) &= -\sum_{i<j} J_{ij}\delta(\sigma_i,\sigma_j) \\ &= -\sum_{i<j} J(A_{ij}-\gamma p_{ij})\delta(\sigma_i,\sigma_j), \end{aligned} \quad (44)$$

where $J$ is a constant expressing the coupling strength, $A_{ij}$ are the elements of the adjacency matrix of the graph, $\gamma > 0$ a parameter expressing the relative contribution to the energy from existing and missing edges, and $p_{ij}$ is the expected number of links connecting $i$ and $j$ for a null model graph with the same total number of edges $m$ of the graph considered. The system is a spin glass (Mezard *et al.*, 1987), as the couplings $J_{ij}$ between spins are both ferromagnetic (on the edges of the graph, provided $\gamma p_{ij} < 1$) and antiferromagnetic (between disconnected vertices, as $A_{ij} = 0$ and $J_{ij} = -J\gamma p_{ij} < 0$). The multiplicative costant $J$ is irrelevant for practical purposes, so in the following we set $J = 1$. The range of the spin-spin interaction is infinite, as there is a non-zero coupling between any pair of spins. Eq. 44 bears a strong resemblance with the expression of modularity of Eq. 13. In fact, if $\gamma = 1$ and $p_{ij} = k_i k_j/2m$ we recover exactly modularity, up to a factor $-1/m$. In this case, finding the spin configuration for which the Hamiltonian is minimal is equivalent to maximizing modularity. Eq. 44 is much more general than modularity, though, as both the null model and the parameter $\gamma$ can be arbitrarily chosen. In particular, the value of $\gamma$ determines the importance of the null model term $p_{ij}$ in the quality function. Eq. 44 can be rewritten as

$$\begin{aligned}\mathcal{H}(\{\sigma\}) &= -\sum_{s} \left[l_s - \gamma(l_s)_{p_{ij}}\right] = -\sum_{s=1} c_{ss} \\ &= \sum_{s<r} \left[l_{rs} - \gamma(l_{rs})_{p_{ij}}\right] = \sum_{s<r} a_{rs}. \end{aligned} \quad (45)$$

Here, the sums run over the clusters: $l_s$ and $l_{rs}$ indicate the number of edges within cluster $s$ and between clusters $r$ and $s$, respectively; $(l_s)_{p_{ij}}$ and $(l_{rs})_{p_{ij}}$ are the corresponding null model expectation values. Eq. 45 defines the coefficients $c_{ss}$ of *cohesion* and $a_{rs}$ of *adhesion*. If a subset of a cluster $s$ has a larger coefficient of adhesion with another cluster $r$ than with its complement in $s$, the energy can be reduced by merging the subset with cluster $r$. In the particular case in which the coefficient of adhesion of a subset $\mathcal{G}'$ of a cluster $s$ with its complement in the cluster exactly matches the coefficient of adhesion of $\mathcal{G}'$ with another cluster $r$, the partitions in which $\mathcal{G}'$ stays within $s$ or is merged with $r$ have the same energy. In this case one can say that clusters $r$ and $s$ are overlapping. In general, the partition with minimum energy has the following properties: 1) every subset of each cluster has a coefficient of adhesion with its complement in the cluster not smaller than with any other cluster; 2) every cluster has non-negative coefficient of cohesion; 3) the coefficient of adhesion between any two clusters is non-positive.

By tuning the parameter $\gamma$ one can vary the number of clusters in the partition with minimum energy, going from a single cluster comprising all vertices ($\gamma = 0$), to $n$ clusters with a single vertex ($\gamma \to \infty$). So, $\gamma$ is a resolution parameter that allows to explore the cluster structure of a graph at different scales (see Section VI.C). The authors used single spin heatbath simulated annealing algorithms to find the ground state of the Hamiltonian of Eq. 44.

Another generalization of modularity was recently suggested by Arenas et al. (Arenas *et al.*, 2008a). They remarked that the fundamental unit to define modularity is the edge, but that high edge densities inside clusters usually imply the existence of long-range topological correlations between vertices, which are revealed by the presence of *motifs* (Milo *et al.*, 2002), i.e. connected undirected subgraphs, like cycles (Section A.1). For instance, a high edge density inside a cluster usually means that there are also several triangles in the cluster, and comparatively few between clusters, a criterion that has inspired on its own popular graph clustering algorithms (Palla *et al.*, 2005; Radicchi *et al.*, 2004). Modularity can then be simply generalized by comparing the density of motifs inside clusters with the expected density in modularity's null model (*motif modularity*). As a particular case, the *triangle modularity* of a partition $\mathcal{C}$ reads

$$Q_{\triangle}(\mathcal{C}) = \frac{\sum_{ijk} A_{ij}(\mathcal{C})A_{jk}(\mathcal{C})A_{ki}(\mathcal{C})}{\sum_{ijk} A_{ij}A_{jk}A_{ki}} - \frac{\sum_{ijk} n_{ij}(\mathcal{C})n_{jk}(\mathcal{C})n_{ki}(\mathcal{C})}{\sum_{ijk} n_{ij}n_{jk}n_{ki}} \quad (46)$$

where $A_{ij}(\mathcal{C}) = A_{ij}\delta(C_i, C_j)$ ($C_i$ is the label of the cluster $i$ belongs to), $n_{ij} = k_i k_j$ ($k_i$ is the degree of vertex $i$) and $n_{ij}(\mathcal{C}) = n_{ij}\delta(C_i, C_j)$. If one chooses as motifs paths with even length, and removes the constraint that all vertices of the motif/path should stay inside the same cluster, maximizing motif modularity could reveal the existence of multipartite structure. For example, if a graph is bipartite, one expects to see many 2-paths starting from one vertex class and returning to it from the other class. Motif modularity can be trivially extended to the case of weighted graphs.

Several graphs representing real systems are built out of correlation data between elements. Correlation matrices are very common in the study of complex systems: well-known examples are the correlations of price returns, which are intensively studied by economists and econophysicists (Mantegna and Stanley, 2000). Correlations may be positive as well as negative, so the corresponding weighted edges indicate both attraction and repulsion between pairs of vertices. Usually the correlation values are filtered or otherwise transformed such to eliminate the weakest correlations and anticorrelations and to maintain strictly positive weights for the edges, yielding graphs

that can be treated with standard techniques. However, ignoring negative correlations means to give up useful information on the relationships between vertices. Finding clusters in a graph with both positive and negative weights is called *correlation clustering problem* (Bansal *et al.*, 2004). According to intuition, one expects that vertices of the same cluster are linked by positive edges, whereas vertices of different clusters are linked by negative edges. The best cluster structure is the partition that maximizes the sum of the strengths (in absolute value) of positive edges within clusters and negative edges between clusters, or, equivalently, the partition that minimizes the sum of the strengths (in absolute value) of positive edges between clusters and negative edges within clusters. This can be formulated by means of modularity, if one accounts for the contribution of the negative edges. A natural way to proceed is to create two copies of the graph at study: in one copy only the weights of the positive edges are kept, in the other only the weights of the negative edges (in absolute value). By applying Eq. 33 to the same partition of both graphs, one derives the contributions $Q^+$ and $Q^-$ to the modularity of that partition for the original graph. Gómez et al. define the global modularity as a linear combination of $Q^+$ and $Q^-$, that accounts for the relative total strengths of positive and negative edge weights (Gómez *et al.*, 2008). Kaplan and Forrest (Kaplan and Forrest, 2008) have proposed a similar expression, with two important differences. First, they have used the total strength of the graph, i.e. the sum of the absolute values of all weights, to normalize $Q^+$ and $Q^-$; Gómez et al. instead have used the positive and the negative strengths, for $Q^+$ and $Q^-$, respectively, which seems to be the more natural choice looking at Eq. 33. Second, Kaplan and Forrest have given equal weight to the contributions of $Q^+$ and $Q^-$ to their final expression of modularity, which is just the difference $Q^+ - Q^-$. In another work, Traag and Bruggeman (Traag and Bruggeman, 2008) have introduced negative links in the general spin glass formulation of modularity of Reichardt and Bornholdt (Reichardt and Bornholdt, 2006a). Here the relative importance of the contribution of positive and negative edge weights is a free parameter, the tuning of which allows to detect communities of various sizes and densities of positive/negative edges.

Some authors have pointed out that the original expression of modularity is not ideal to detect communities in bipartite graphs, which describe several real systems, like food webs (Williams and Martinez, 2000), scientific (Newman, 2001) and artistic (Gleiser and Danon, 2003) collaboration networks, etc.. Expressions of modularity for bipartite graphs were suggested by Guimerà et al. (Guimerà *et al.*, 2007) and Barber (Barber, 2007; Barber *et al.*, 2008). Guimerà et al. call the two classes of vertices actors and teams, and indicate with $t_i$ the degree of actor $i$ and $m_a$ the degree of team $a$. The null model graphs are random graphs with the same expected degrees for the vertices, as usual. The bipartite modularity $\mathcal{M}_B(\mathcal{P})$ for a partition $\mathcal{P}$ (of the actors) has the following expression

$$\mathcal{M}_B(\mathcal{P}) = \sum_{c=1}^{n_c} \left[ \frac{\sum_{i \neq j \in c} c_{ij}}{\sum_a m_a(m_a - 1)} - \frac{\sum_{i \neq j \in c} t_i t_j}{\left(\sum_a m_a\right)^2} \right] . \quad (47)$$

Here, $c_{ij}$ is the number of teams in which actors $i$ and $j$ are together and the sum $\sum_a m_a(m_a - 1)$ gives the number of ordered pairs of actors in the same team. The second ratio of each summand is the null model term, indicating the expected (normalized) number of teams for pairs of actors in cluster $c$. The bipartite modularity can also be applied to (unipartite) directed graphs: each vertex can be duplicated and assigned to both classes, based on its twofold role of source and target for the edges.

Another interesting alternative was introduced by Barber (Barber, 2007; Barber *et al.*, 2008) and is a simple extension of Eq. 13. Let us suppose that the two vertex classes (red and blue) are made out of $p$ and $q$ vertices, respectively. The degree of a red vertex $i$ is indicated with $k_i$, that of a blue vertex $j$ with $d_j$. The adjacency matrix $\mathbf{A}$ of the graph is in block off-diagonal form, as there are edges only between red and blue vertices. Because of that, Barber assumes that the null model matrix $\mathbf{P}$, whose element $P_{ij}$ indicates as usual the expected number of edges between vertices $i$ and $j$ in the null model, also has the block off-diagonal form

$$\mathbf{P} = \begin{bmatrix} \mathbf{O}_{p\times p} & \tilde{\mathbf{P}}_{p\times q} \\ \tilde{\mathbf{P}}^{\mathbf{T}}_{q\times p} & \mathbf{O}_{q\times q} \end{bmatrix} , \quad (48)$$

where the $\mathbf{O}$ are square matrices with all zero elements and $\tilde{P}_{ij} = k_i d_j / m$, as in the null model of standard modularity (though other choices are possible). The modularity maximum can be computed through the modularity matrix $\mathbf{B} = \mathbf{A} - \mathbf{P}$, as we have seen in Section VI.A.4. However, spectral optimization of modularity gives excellent results for bipartitions, while its performance worsens when the number of clusters is unknown, as it is usually the case. Barber has proposed a different optimization technique, called Bipartite Recursively Induced Modules (BRIM), based on the bipartite nature of the graph. The algorithm is based on the special expression of modularity for the bipartite case, for which once the partition of the red or the blue vertices is known, it is easy to get the partition of the other vertex class that yields the maximum modularity. Therefore, one starts from an arbitrary partition in $c$ clusters of, say, the blue vertices, and recovers the partition of the red vertices, which is in turn used as input to get a better partition of the blue vertices, and so on until modularity converges. BRIM does not predict the number of clusters $c$ of the graph, but one can obtain good estimates for it by exploring different values with a simple bisection approach. Typically, for a given $c$ the algorithm needs a few steps to converge, each step having a complexity $O(m)$. An expression of the number of convergence steps in terms of $n$ and/or $m$ still needs to be derived.

### C. Limits of modularity

In this Section we shall discuss some features of modularity, which are crucial to identify the domain of its applicability and ultimately to assess the issue of the reliability of the measure for the problem of graph clustering.

An important question concerns the value of the maximum modularity $Q_{max}$ for a graph. We know that it must be non-negative, as there is always at least a partition with zero modularity, consisting in a single cluster with all vertices (Section III.C.2). However, a large value for the modularity maximum does not necessarily mean that a graph has community structure. Random graphs are supposed to have no community structure, as the linking probability between vertices is either constant or a function of the vertex degrees, so there is no bias *a priori* towards special groups of vertices. Still, random graphs may have partitions with large modularity values (Guimerà *et al.*, 2004; Reichardt and Bornholdt, 2006a). This is due to fluctuations in the distribution of edges in the graph, which in many graph realizations is not homogeneous even if the linking probability is constant, like in Erdös-Rényi graphs. The fluctuations determine concentrations of links in some subsets of the graph, which then appear like communities. According to the definition of modularity, a graph has community structure with respect to a random graph with equal size and expected degree sequence. Therefore, the modularity maximum of a graph reveals a significant community structure only if it is appreciably larger than the modularity maximum of random graphs of the same size and expected degree sequence. The significance of the modularity maximum $Q_{Max}$ for a graph can be estimated by calculating the maximum modularity for many realizations of the null model, obtained from the original graph by randomly rewiring its edges. One then computes the average $\langle Q \rangle_{NM}$ and the standard deviation $\sigma_Q^{NM}$ of the results. The statistical significance of $Q_{max}$ is indicated by the distance of $Q_{max}$ from the null model average $\langle Q \rangle_{NM}$ in units of the standard deviation $\sigma_Q^{NM}$, i.e. by the $z$-score

$$z = \frac{Q_{max} - \langle Q \rangle_{NM}}{\sigma_Q^{NM}}. \tag{49}$$

If $z \gg 1$, $Q_{max}$ indicates strong community structure. Cutoff values of $2-3$ for the $z$-scores are customary. This approach has problems, though. It can generate both false positives and false negatives: a few graphs that most people would consider without a significant community structure have a large $z$-score; on the other hand, some graphs that are agreed to display cluster structure have very low values for the $z$-score. Besides, the distribution of the maximum modularity values of the null model, though peaked, is not Gaussian. Therefore, one cannot attribute to the values of the $z$-score the significance corresponding to a Gaussian distribution, and one would need instead to compute the statistical significance for the right distribution.

Reichardt and Bornholdt have studied the issue of the modularity values for random graphs in some depth (Reichardt and Bornholdt, 2006b, 2007), using their general spin glass formulation of the clustering problem (Section VI.B). They considered the general case of a random graph with arbitrary degree distribution $P(k)$ and without degree-degree correlations. They set $\gamma = 1$, so that the energy of the ground state coincides with modularity (up to a constant factor). For modularity's null model graphs, the modularity maximum corresponds to an equipartition of the graph, i.e. the magnetization of the ground state of the spin glass is zero, a result confirmed by numerical simulations (Reichardt and Bornholdt, 2006b, 2007). This is because the distribution of the couplings has zero mean, and the mean is only coupled to magnetization (Fu and Anderson, 1986). For a partition of any graph with $n$ vertices and $m$ edges in $q$ clusters with equal numbers of vertices, there is a simple linear relation between the cut size $\mathcal{C}_q$ of the partition and its modularity $Q_q$: $\mathcal{C}_q = m[(q-1)/q - Q_q]$. We remind that the cut size $\mathcal{C}_q$ is the total number of inter-cluster edges of the partition (Section IV.A). In this way, the partition with maximum modularity is also the one with minimum cut size, and community detection becomes equivalent to graph partitioning. Reichardt and Bornholdt derived analytically the ground state energy for Ising spins ($q = 2$), which corresponds to the following expression of the expected maximum modularity $Q_2^{max}$ for a bipartition (Reichardt and Bornholdt, 2007)

$$Q_2^{max} = U_0 J \frac{\langle k^{1/2} \rangle}{\langle k \rangle}. \tag{50}$$

Here $\langle k^\alpha \rangle = \int P(k) k^\alpha dk$ and $U_0$ is the ground state energy of the Sherrington-Kirkpatrick model (Sherrington and Kirkpatrick, 1975). The most interesting feature of Eq. 50 is the simple scaling with $\langle k^{1/2} \rangle / \langle k \rangle$. Numerical calculations show that this scaling holds for both Erdös-Rényi and scale-free graphs (Section A.3). Interestingly, the result is valid for partitions in $q$ clusters, where $q$ is left free, not only for $q = 2$. The number of clusters of the partition with maximum modularity decreases if the average degree $\langle k \rangle$ increases, and tends to 5 for large values of $\langle k \rangle$, regardless of the degree distribution and the size of the graph. From Eq. 50 we also see that the expected maximum modularity for a random graph increases when $\langle k \rangle$ decreases, i. e. if the graph gets sparser. So it is particularly hard to detect communities in sparse graphs by using modularity optimization. As we shall see in Section XIII, the sparsity of a graph is generally a serious obstacle for graph clustering methods, no matter if one uses modularity or not.

A more fundamental issue concerns the capability of modularity to detect "good" partitions. If a graph has a clear cluster structure, one expects that the maximum modularity of the graph reveals it. The null model of modularity assumes that any vertex $i$ "sees" any other vertex $j$, and the expected number of edges between them

is $p_{ij} = k_i k_j/2m$. Similarly, the expected number of edges between two clusters $\mathcal{A}$ and $\mathcal{B}$ with total degrees $K_\mathcal{A}$ and $K_\mathcal{B}$, respectively, is $P_{\mathcal{AB}} = K_\mathcal{A} K_\mathcal{B}/2m$. The variation of modularity determined by the merger of $\mathcal{A}$ and $\mathcal{B}$ with respect to the partition in which they are separate clusters is $\Delta Q_{\mathcal{AB}} = l_{\mathcal{AB}}/m - K_\mathcal{A} K_\mathcal{B}/2m^2$, with $l_{\mathcal{AB}}$ number of edges connecting $\mathcal{A}$ to $\mathcal{B}$. If $l_{\mathcal{AB}} = 1$, i.e. there is a single edge joining $\mathcal{A}$ to $\mathcal{B}$, we expect that often the two subgraphs will be kept separated. Instead, if $K_\mathcal{A} K_\mathcal{B}/2m < 1$, $\Delta Q_{\mathcal{AB}} > 0$. Let us suppose for simplicity that $K_\mathcal{A} \sim K_\mathcal{B} = K$, i.e. that the two subgraphs are of about the same size, measured in terms of edges. We conclude that, if $K <\sim \sqrt{2m}$ and the two subgraphs $\mathcal{A}$ and $\mathcal{B}$ are connected, modularity is greater if they are in the same cluster (Fortunato and Barthélemy, 2007). The reason is intuitive: if there are more edges than expected between $\mathcal{A}$ and $\mathcal{B}$, there is a strong topological correlation between the subgraphs. If the subgraphs are sufficiently small (in degree), the expected number of edges for the null model can be smaller than one, so even the weakest possible connection (a single edge) suffices to keep the subgraphs together. Interestingly, this result holds independently of the structure of the subgraphs. In particular it remains true if the subgraphs are cliques, which are the subgraphs with the largest possible density of internal edges, and represent the strongest possible communities. In Fig. 15 a graph is made out of $n_c$ identical cliques, with $l$ vertices each, connected by single edges. It is intuitive to think that the clusters of the best partition are the individual cliques: instead, if $n_c$ is larger than about $l^2$, modularity would be higher for the partition in which pairs of consecutive cliques are parts of the same cluster (indicated by the dashed lines in the figure).

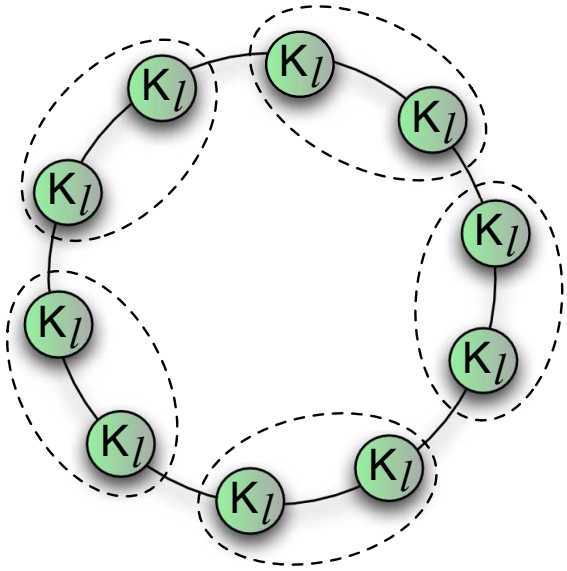


FIG. 15 Resolution limit of modularity optimization. The natural community structure of the graph, represented by the individual cliques (circles), is not recognized by optimizing modularity, if the cliques are smaller than a scale depending on the size of the graph. In this case, the maximum modularity corresponds to a partition whose clusters include two or more cliques (like the groups indicated by the dashed contours). Reprinted figure with permission from (Fortunato and Barthélemy, 2007). ©2007 from the National Academy of Science of the USA.

The conclusion is striking: modularity optimization has a resolution limit that may prevent it from detecting clusters which are comparatively small with respect to the graph as a whole, even when they are well defined communities like cliques. So, if the partition with maximum modularity includes clusters with total degree of the order of $\sqrt{m}$ or smaller, one cannot know *a priori* whether the clusters are single communities or combinations of smaller weakly interconnected communities. This resolution problem has a large impact in practical applications. Real graphs with community structure usually contain communities which are very diverse in size (Clauset *et al.*, 2004; Danon *et al.*, 2005; Guimerà *et al.*, 2003; Palla *et al.*, 2005), so many (small) communities may remain undetected. Besides, modularity is extremely sensitive to even individual connections. Many real graphs, in biology and in the social sciences, are reconstructed through experiments and surveys, so edges may occasionally be false positives: if two small subgraphs happen to be connected by a few false edges, modularity will put them in the same cluster, inferring a relationship between entities that in reality may have nothing to do with each other.

The resolution limit comes from the very definition of modularity, in particular from its null model. The weak point of the null model is the implicit assumption that each vertex can interact with every other vertex, which implies that each part of the graph knows about everything else. This is however questionable, and certainly wrong for large systems like, e.g., the Web graph. It is certainly more reasonable to assume that each vertex has a limited horizon within the graph, and interacts just with a portion of it. However, nobody knows how to define such local territories for the graph vertices. The null model of the localized modularity of Muff et al. (Section VI.B) is a possibility, since it limits the horizon of a vertex to a local neighborhood, comprising the cluster of the vertex and the clusters linked to it by at least one edge (neighboring clusters). However, there are many other possible choices. In this respect, the null model of Girvan and Newman, though unrealistic, is the simplest one can think of, which partly explains its success. Quality functions that, like modularity, are based on a null model such that the horizon of vertices is of the order of the size of the whole graph, are likely to be affected by a resolution limit (Fortunato, 2007). The problem is more general, though. For instance, Li et al. (Li *et al.*, 2008) have introduced a quality function, called *modularity density*, which consists in the sum over the clusters

of the ratio between the difference of the internal and external degrees of the cluster and the cluster size. The modularity density does not require a null model, and delivers better results than modularity optimization (e. g. it correctly recovers the natural partition of the graph in Fig. 15 for any number/size of the cliques). However, it is still affected by a resolution limit. To avoid that, Li et al. proposed a more general definition of their measure, including a tunable parameter that allows to explore the graph at different resolutions, in the spirit of the methods of Section XII.

A way to go around the resolution limit problem could be to perform further subdivisions of the clusters obtained from modularity optimization, in order to eliminate possible artificial mergers of communities. For instance, one could recursively optimize modularity for each single cluster, taking the cluster as a separate entity (Fortunato and Barthélemy, 2007; Ruan and Zhang, 2008). However, this is not a reliable procedure, for two reasons: 1) the local modularities used to find partitions within the clusters have different null models, as they depend on the cluster sizes, so they are inconsistent with each other; 2) one needs to define a criterion to decide when one has to stop partitioning a cluster, but there is no obvious prescription, so any choice is necessarily based on arbitrary assumptions.

Resolution limits arise as well in the more general formulation of community detection by Reichardt and Bornholt (Kumpula *et al.*, 2007b). Here the limit scale for the undetectable clusters is $\sqrt{\gamma m}$. We remind that $\gamma$ weighs the contribution of the null model term in the quality function. For $\gamma = 1$ one recovers the resolution limit of modularity. By tuning the parameter $\gamma$ it is possible to arbitrarily vary the resolution scale of the corresponding quality function. This in principle solves the problem of the resolution limit, as one could adjust the resolution of the method to the actual scale of the communities to detect. The problem is that usually one has no information about the community sizes, so it is not possible to decide *a priori* the proper value(s) of $\gamma$ for a specific graph. In the most recent literature on graph clustering quite a few *multiresolution methods* have been introduced, addressing this problem in several ways. We will discuss them in some detail in Section XII.

## VII. SPECTRAL ALGORITHMS

Spectral properties of graph matrices are frequently used to find partitions. A paradigmatic example is spectral graph partitioning, which makes use of eigenvectors of the Laplacian matrix (Section IV.A). In the same spirit, Newman-Girvan modularity can be optimized by using the eigenvectors of the modularity matrix (Section VI.A.4). The main idea is to infer structural relationships between vertices from the similarity of the corresponding components of eigenvectors of special graph matrices. Here we review the main techniques.

Early works have shown that the eigenvectors of the *transfer matrix* $\mathbf{T}$ (Section A.2) can be used to extract useful information on community structure. The transfer matrix acts as a time propagator for the process of random walk on a graph. Given the eigenvector $\mathbf{c}^\alpha$ of the transposed transfer matrix $T^\dagger$, corresponding to the eigenvalue $\lambda_\alpha$, $c_i^\alpha$ is the outgoing current flowing from vertex $i$, corresponding to the eigenmode $\alpha$. The *participation ratio* (PR)

$$\chi_\alpha = \left[ \sum_{i=1}^{n} (c_i^\alpha)^4 \right]^{-1} \tag{51}$$

indicates the effective number of vertices contributing to eigenvector $\mathbf{c}^\alpha$. If $\chi_\alpha$ receives contributions only from vertices of the same cluster, i.e. eigenvector $\mathbf{c}^\alpha$ is "localized", the value of $\chi_\alpha$ indicates the size of that cluster (Eriksen *et al.*, 2003; Simonsen *et al.*, 2004). The significance of the cluster can be assessed by comparing $\chi_\alpha$ with the corresponding participation ratio for a random graph with the same expected degree sequence as the original graph. Eigenvectors of the adjacency matrix may be localized as well if the graph has a clear community structure (Slanina and Zhang, 2005).

Donetti and Muñoz have devised an elegant method based on the eigenvectors of the Laplacian matrix (Donetti and Muñoz, 2004). The idea is simple: since the values of the eigenvector components are close for vertices in the same community, one can use them as coordinates, such that vertices turn into points in a metric space. So, if one uses $M$ eigenvectors, one can embed the vertices in an $M$-dimensional space. Communities appear as groups of points well separated from each other, as illustrated in Fig. 16. The separation is the more visible, the larger the number of dimensions/eigenvectors $M$. The points are grouped in communities by hierarchical clustering (see Section III), but one merges only pairs of connected clusters. Among all partitions of the resulting dendrogram, the one with largest modularity is chosen. For the similarity measure between vertices, Donetti and Muñoz used both the Euclidean distance and the angle distance. The angle distance between two points is the angle between the vectors going from the origin of the $M$-dimensional space to either point. Tests on the benchmark by Girvan and Newman (Section XIV.A) show that the best results are obtained with complete-linkage clustering. The most computationally expensive part of the algorithm is the calculation of the Laplacian eigenvectors. Since a few eigenvectors suffice to get good partitions, one can determine them with the Lanczos method (Lanczos, 1950), which has complexity $m/(\lambda_3 - \lambda_2)$, $\lambda_2$ and $\lambda_3$ being the two smallest (nonzero) eigenvalues of the Laplacian. The number $M$ of eigenvectors that are needed to have a clean separation of the clusters is not known *a priori*, but one can compute a number $M_0 > 1$ of them and search for the highest modularity partition among those delivered by the method for all $1 \leq M \leq M_0$. In a related work, Simonsen has embedded graph vertices in space by

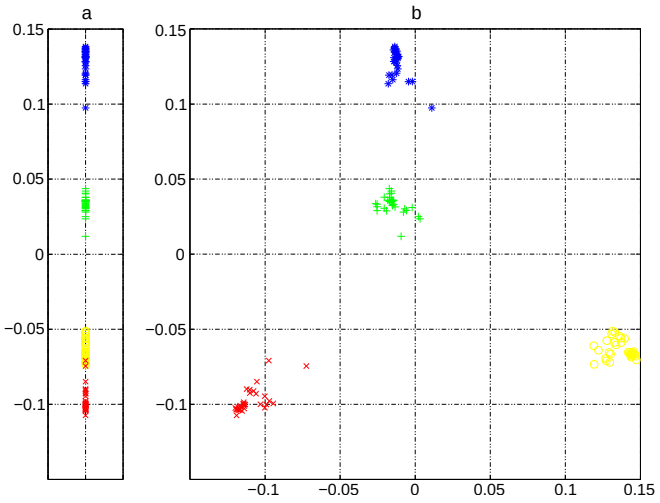


FIG. 16 Spectral algorithm by Donetti and Muñoz. Vertex $i$ is represented by the values of the $i$th components of Laplacian eigenvectors. In this example, the graph has an ad-hoc division in four communities, indicated by the colours. The communities are better separated in two dimensions (b) than in one (a). Reprinted figure with permission from (Donetti and Muñoz, 2004). ©2004 by IOP Publishing and SISSA.

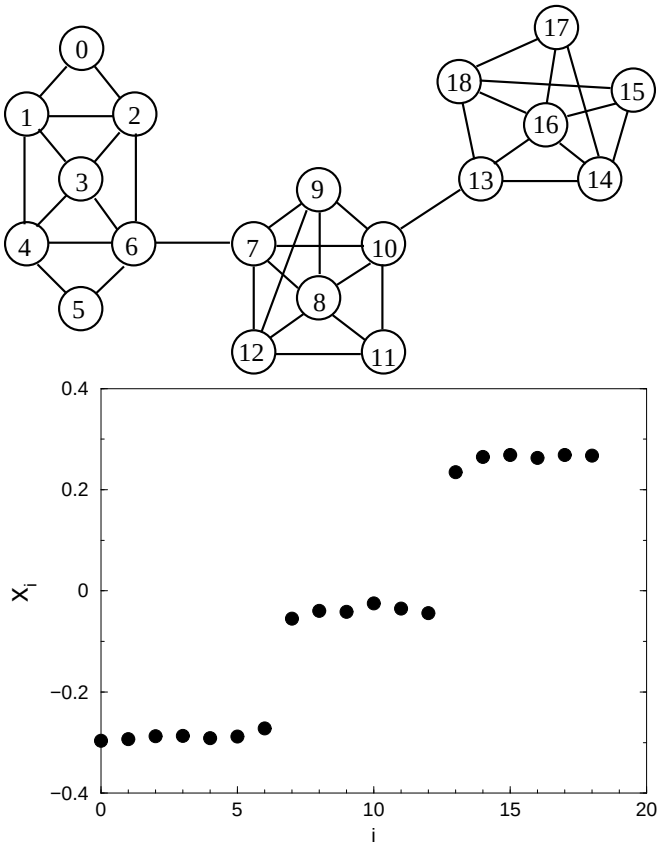


FIG. 17 Basic principle of the spectral algorithm by Capocci et al. (Capocci *et al.*, 2005). The bottom diagram shows the values of the components of the second eigenvector of the right stochastic matrix for the graph drawn on the top. The three plateaus of the eigenvector components correspond to the three evident communities of the graph. Reprinted figures with permission from (Capocci *et al.*, 2005). ©2005 by Elsevier.

using as coordinates the components of the eigenvector of the right stochastic matrix (Simonsen, 2005).

Eigenvalues and eigenvectors of the Laplacian matrix have been used by Alves to compute the effective conductances for pairs of vertices in a graph, assuming that the latter is an electric network with edges of unit resistance (Alves, 2007). The conductances enable one to compute the transition probabilities for a random walker moving on the graph, and from the transition probabilities one builds a similarity matrix between vertex pairs. Hierarchical clustering is applied to join vertices in groups. The method can be trivially extended to the case of weighted graphs. The algorithm by Alves is rather slow, as one needs to compute the whole spectrum of the Laplacian, which requires a time $O(n^3)$. Moreover, there is no criterion to select which partition(s) of the dendrogram is (are) the best.

Capocci et al. also used eigenvector components to identify communities (Capocci *et al.*, 2005). In this case the eigenvectors are those of the *right stochastic matrix* $\mathbf{R}$ (Section A.2), that is derived from the adjacency matrix by dividing each row by the sum of its elements. The right stochastic matrix has similar properties as the Laplacian. If the graph has $g$ connected components, the largest $g$ eigenvalues are equal to 1, with eigenvectors characterized by having equal-valued components for vertices belonging to the same component. In this way, by listing the vertices according to the connected components they belong to, the components of any eigenvector of $\mathbf{R}$, corresponding to eigenvalue 1, display a step-wise profile, with plateaus indicating vertices in the same connected component. For connected graphs with cluster structure, one can still see plateaus, if communities are only loosely connected to each other (Fig. 17). Here the communities can be immediately deducted by an inspection of the components of any eigenvector with eigenvalue 1. In practical cases, plateaus are not clearly visible, and one eigenvector is not enough. However, one expects that there should be a strong correlation between eigenvector components corresponding to vertices in the same cluster. Capocci et al. derived a similarity matrix, where the similarity between vertices $i$ and $j$ is the Pearson correlation coefficient between their corresponding eigenvector components, where the averages are taken over a small set of eigenvectors. The eigenvectors can be calculated by performing a constrained optimization of a suitable cost function. The method can be extended to weighted and directed graphs. It is useful to estimate vertex similarities, however it does not provide a well-defined partition of the graph.

## VIII. DYNAMIC ALGORITHMS

This Section describes methods employing processes running on the graph, focusing on spin-spin interactions, random walks and synchronization.

### A. Spin models

The Potts model is among the most popular models in statistical mechanics (Wu, 1982). It describes a system of spins that can be in $q$ different states. The interaction is ferromagnetic, i.e. it favours spin alignment, so at zero temperature all spins are in the same state. If antiferromagnetic interactions are also present, the ground state of the system may not be the one where all spins are aligned, but a state where different spin values coexist, in homogeneous clusters. If Potts spin variables are assigned to the vertices of a graph with community structure, and the interactions are between neighbouring spins, it is likely that the structural clusters could be recovered from like-valued spin clusters of the system, as there are many more interactions inside communities than outside. Based on this idea, inspired by an earlier paper by Blatt, Wiseman and Domany (Blatt *et al.*, 1996), Reichardt and Bornholdt proposed a method to detect communities that maps the graph onto a $q$-Potts model with nearest-neighbours interactions (Reichardt and Bornholdt, 2004). The Hamiltonian of the model, i.e. its energy, reads

$$\mathcal{H} = -J \sum_{i,j} A_{ij} \delta(\sigma_i, \sigma_j) + \gamma \sum_{s=1}^{q} \frac{n_s(n_s-1)}{2}, \qquad (52)$$

where $A_{ij}$ is the element of the adjacency matrix, $\delta$ is the Kronecker's function, $n_s$ the number of spins in state $s$, $J$ and $\gamma$ are coupling parameters. The energy $\mathcal{H}$ is the sum of two competing terms: the first is the classical ferromagnetic Potts model energy, and favors spin alignment; the second term instead peaks when the spins are homogeneously distributed. The ratio $\gamma/J$ expresses the relative importance of the two terms: by tuning $\gamma/J$ one can explore different levels of modularity of the system, from the whole graph seen as a single cluster to clusters consisting of individual vertices. If $\gamma/J$ is set to the value $\delta(\mathcal{G})$ of the average density of edges of the graph $\mathcal{G}$, the energy of the system is smaller if spins align within subgraphs such that their internal edge density exceeds $\delta(\mathcal{G})$, whereas the external edge density is smaller than $\delta(\mathcal{G})$, i.e. if the subgraphs are clusters (Section III.B.1). The minimization of $\mathcal{H}$ is carried out via simulated annealing ((Kirkpatrick *et al.*, 1983) and Section VI.A.2), starting from a configuration where spins are randomly assigned to the vertices and the number of states $q$ is very high. The procedure is quite fast and the results do not depend on $q$ (provided $q$ is sufficiently high). The method also allows to identify vertices shared between communities, from the comparison of partitions corresponding to global and local energy minima. The Hamiltonian $\mathcal{H}$ can be rewritten as

$$\mathcal{H} = \sum_{i<j} \delta(\sigma_i, \sigma_j)(\gamma - A_{ij}), \qquad (53)$$

which is the energy of an infinite-range Potts spin glass, as all pairs of spins are interacting (neighboring or not) and there may be both positive and negative couplings. Eq. 53 is at the basis of the successive generalization of modularity with arbitrary null models proposed by Reichardt and Bornholdt, that we have discussed in Section VI.B. The method can be simply extended to the analysis of weighted graphs, by introducing spin couplings proportional to the edge weights, which amounts to replacing the adjacency matrix $\mathbf{A}$ with the weight matrix $\mathbf{W}$ in Eq. 52.

In another work (S.-W. Son *et al.*, 2006), Son et al. have presented a clustering technique based on the Ferromagnetic Random Field Ising Model (FRFIM). Given a weighted graph with weight matrix $\mathbf{W}$, the Hamiltonian of the FRFIM on the graph is

$$\mathcal{H} = -\frac{1}{2} \sum_{i,j} W_{ij} \sigma_i \sigma_j - \sum_i B_i \sigma_i. \qquad (54)$$

In Eq. 54 $\sigma_i = \pm 1$ and $B_i$ are the spin and the random magnetic field of vertex $i$, respectively. The FRFIM has been studied to understand the nature of the spin glass phase transition (Middleton and Fisher, 2002) and the disorder-driven roughening transition of interfaces in disordered media (Noh and Rieger, 2001, 2002). The behavior of the model depends on the choice of the magnetic fields. Son et al. set to zero the magnetic fields of all vertices but two, say $s$ and $t$, for which the field has infinite strength and opposite signs. This amounts to fix the spins of $s$ and $t$ to opposite values, introducing frustration in the system. The idea is that, if $s$ and $t$ are central vertices of different communities, they impose their spin state to the other community members. So, the state of minimum energy is a configuration in which the graph is polarized into a subgraph with all positive spins and a subgraph with all negative spins, coinciding with the communities, if they are well defined. Finding the minimum of $\mathcal{H}$ is equivalent to solving a maximum-flow/minimum-cut problem, which can be done through well known techniques of combinatorial optimization, like the augmenting path algorithm (Ahuja *et al.*, 1993). For a given choice of $s$ and $t$, many ground states can be found. The vertices that end up in the same cluster in all ground states represent the cores of the clusters, which are called *coteries*. Possible vertices not belonging to the coteries indicate that the two clusters overlap. In the absence of information about the cluster structure of the graph, one needs to repeat the procedure for any pair of vertices $s$ and $t$. Picking vertices of the same cluster, for instance, would not give meaningful partitions. Son et al. distinguish relevant clusters if they are of about the same size. The procedure can be iteratively applied

to each of the detected clusters, considered as a separate graph, until all clusters have no community structure any more. On sparse graphs, the algorithm has complexity $O(n^{2+\theta})$, where $\theta \sim 1.2$, so it is very slow and can be currently used for graphs of up to few thousands vertices. If one happens to know which are the important vertices of the clusters, e.g. by computing appropriate centrality values (like degree or site betweenness (Freeman, 1977)), the choices for $s$ and $t$ are constrained and the complexity can become as low as $O(n^{\theta})$, which enables the analysis of systems with millions of vertices. Tests on Barabási-Albert graphs (Section A.3) show that the latter have no community structure, as expected.

### B. Random walk

Random walks (Hughes, 1995) can also be useful to find communities. If a graph has a strong community structure, a random walker spends a long time inside a community due to the high density of internal edges and consequent number of paths that could be followed. Here we describe the most popular clustering algorithms based on random walks. All of them can be trivially extended to the case of weighted graphs.

Zhou used random walks to define a distance between pairs of vertices (Zhou, 2003a): the distance $d_{ij}$ between $i$ and $j$ is the average number of edges that a random walker has to cross to reach $j$ starting from $i$. Close vertices are likely to belong to the same community. Zhou defines a "global attractor" of a vertex $i$ to be a closest vertex to $i$ (i.e. any vertex lying at the smallest distance from $i$), whereas the "local attractor" of $i$ is its closest neighbour. Two types of communities are defined, according to local or global attractors: a vertex $i$ has to be put in the same community of its attractor and of all other vertices for which $i$ is an attractor. Communities must be minimal subgraphs, i.e. they cannot include smaller subgraphs which are communities according to the chosen criterion. Applications to real networks, like Zachary's karate club (Zachary, 1977) and the college football network compiled by Girvan and Newman (Girvan and Newman, 2002) (Section XIV.A), along with artificial graphs like the benchmark by Girvan and Newman (Girvan and Newman, 2002) (Section XIV.A), show that the method can find meaningful partitions. The method can be refined, in that vertex $i$ is associated to its attractor $j$ only with a probability proportional to $\exp(-\beta d_{ij})$, $\beta$ being a sort of inverse temperature. The computation of the distance matrix requires solving $N$ linear-algebra equations, which requires a time $O(n^3)$. On the other hand, an exact computation of the distance matrix is not necessary, as the attractors of a vertex can be identified by considering only a localized portion of the graph around the vertex; therefore the method can be applied to large graphs as well. In a successive paper (Zhou, 2003b), Zhou introduced a measure of dissimilarity between vertices based on the distance defined above. The measure resembles the definition of distance based on structural equivalence of Eq. 7, where the elements of the adjacency matrix are replaced by the corresponding distances. Graph partitions are obtained with a divisive procedure that, starting from the graph as a single community, performs successive splits based on the criterion that vertices in the same cluster must be less dissimilar than a running threshold, which is decreased during the process. The hierarchy of partitions derived by the method is representative of actual community structures for several real and artificial graphs, including Zachary's karate club (Zachary, 1977), the college football network (Girvan and Newman, 2002) and the benchmark by Girvan and Newman (Girvan and Newman, 2002) (Section XIV.A). The time complexity of the procedure is again $O(n^3)$. The code of the algorithm can be downloaded from `http://www.mpikg-golm.mpg.de/theory/people/zhou/networkcommunity.html`.

In another work (Zhou and Lipowsky, 2004), Zhou and Lipowsky adopted biased random walkers, where the bias is due to the fact that walkers move preferentially towards vertices sharing a large number of neighbours with the starting vertex. They defined a proximity index, which indicates how close a pair of vertices is to all other vertices. Communities are detected with a procedure called *NetWalk*, which is an agglomerative hierarchical clustering method (Section IV.B), where the similarity between vertices is expressed by their proximity. The method has a time complexity $O(n^3)$: however, the proximity index of a pair of vertices can be computed with good approximation by considering just a small portion of the graph around the two vertices, with a considerable gain in time. The performance of the method is comparable with that of the algorithm of Girvan and Newman (Section V.A).

A different distance measure between vertices based on random walks was introduced by Latapy and Pons (Latapy and Pons, 2005). The distance is calculated from the probabilities that the random walker moves from a vertex to another in a fixed number of steps. The number of steps has to be large enough to explore a significant portion of the graph, but not too long, as otherwise one would approach the stationary limit in which transition probabilities trivially depend on the vertex degrees. Vertices are then grouped into communities through an agglomerative hierarchical clustering technique based on Ward's method (Ward, 1963). Modularity (Section III.C.2) is used to select the best partition of the resulting dendrogram. The algorithm runs to completion in a time $O(n^2 d)$ on a sparse graph, where $d$ is the depth of the dendrogram. Since $d$ is often small for real graphs ($O(\log n)$), the expected complexity in practical computations is $O(n^2 \log n)$. The software of the algorithm can be found at `http://www-rp.lip6.fr/~latapy/PP/walktrap.html`.

Hu et al. (Hu *et al.*, 2008) designed a graph clustering technique based on a signaling process between vertices, somewhat resembling diffusion. Initially a vertex $s$ is as-

signed one unit of signal, all the others have no signal. In the first step, the source vertex $s$ sends one unit of signal to each of its neighbors. Next, all vertices send as many units of signals they have to each of their neighbors. The process is continued until a given number of iterations $T$ is reached. The intensity of the signal at vertex $i$, normalized by the total amount of signal, is the $i$-th entry of a vector $\mathbf{u_s}$, representing the source vertex $s$. The procedure is then repeated by choosing each vertex as source. In this way one can associate an $n$-dimensional vector to each vertex, which correspons to a point in an Euclidean space. The vector $\mathbf{u_s}$ is actually the $s$-th column of the matrix $(\mathbf{I}+\mathbf{A})^T$, where $\mathbf{I}$ and $\mathbf{A}$ are the unit and adjacency matrix, respectively. The idea is that the vector $\mathbf{u_s}$ describes the influence that vertex $s$ exerts on the graph through signaling. Vertices of the same community are expected to have similar influence on the graph and thus to correspond to vectors which are "close" in space. The vectors are finally grouped via fuzzy $k$-means clustering (Section IV.C). The optimal number of clusters corresponds to the partition with the shortest average distance between vectors in the same community and the largest average distance between vectors of different communities. The signaling process is similar to diffusion, but with the important difference that here there is no flow conservation, as the amount of signal at each vertex is not distributed among its neighbors but transferred entirely to each neighbor (as if the vertex sent multiple copies of the same signal). The complexity of the algorithm is $O(T(\langle k\rangle+1)n^2)$, where $\langle k\rangle$ is the average degree of the graph. Like in the previous algorithm by Latapy and Pons (Latapy and Pons, 2005), finding an optimal value for the number of iterations $T$ is non-trivial.

Delvenne et al. (Delvenne *et al.*, 2008) have shown that random walks enable one to introduce a general quality function, expressing the persistence of clusters in time. A cluster is persistent with respect to a random walk after $t$ time steps if the probability that the walker escapes the cluster before $t$ steps is low. Such probability is computed via the *clustered autocovariance matrix* $\mathbf{R}_t$, that, for a partition of the graph in $c$ clusters, is defined as

$$\mathbf{R}_t = \mathbf{H}^T(\mathbf{\Pi}\mathbf{M}^t - \pi^T\pi)\mathbf{H}. \tag{55}$$

Here, $\mathbf{H}$ is the $n \times c$ membership matrix, whose element $H_{ij}$ equals one if vertex $i$ is in cluster $j$, zero otherwise; $\mathbf{M}$ is the transition matrix of the random walk; $\mathbf{\Pi}$ the diagonal matrix whose elements are the stationary probabilities of the random walk, i. e. $\Pi_{ii} = k_i/2m$, $k_i$ being the degree of vertex $i$; $\pi$ is the vector whose entries are the diagonal elements of $\mathbf{\Pi}$. The element $(R_t)_{ij}$ expresses the probability for the walk to start in cluster $i$ and end up in cluster $j$ after $t$ steps, minus the stationary probability that two independent random walkers are in $i$ and $j$. In this way, the persistence of a cluster $i$ is related to the diagonal element $(R_t)_{ii}$. Delvenne et al. defined the *stability of the clustering*

$$r(t;\mathbf{H}) = \min_{0\le s\le t}\sum_{i=1}^{c}(R_s)_{ii} = \min_{0\le s\le t}\mathrm{trace}[R_s]. \tag{56}$$

The aim is then, for a given time $t$, finding the partition with the largest value for $r(t;\mathbf{H})$. For $t = 0$, the most stable partition is that in which all vertices are their own clusters. Interestingly, for $t = 1$, maximizing stability is equivalent to maximizing Newman-Girvan modularity (Section III.C.2) and the cut size (Section IV.A) equals $(r(0) - r(1))$, so it is also a one-step measure. In the limit $t \to \infty$, the most stable partition coincides with the Fiedler partition (Fiedler, 1973, 1975), i. e. the bipartition where vertices are put in the same class according to the signs of the corresponding component of the Fiedler eigenvector (Section IV.A). Therefore, the measure $r(t;\mathbf{H})$ is very general, and gives a unifying interpretation in the framework of the random walk of several measures that were defined in different contexts. In particular, modularity has a natural interpretation in this dynamic picture (Lambiotte *et al.*, 2008). Since the size of stable clusters increases with $t$, time can be considered as a resolution parameter. Resolution can be fine tuned by taking time as a continuous variable (the extension of the formalism is straightforward); the linearization of the stability measure at small (continuous) times delivers previously introduced multiresolution versions of modularity (Arenas *et al.*, 2008b; Reichardt and Bornholdt, 2006a) (Section XII.A).

We conclude this section by describing the *Markov Cluster Algorithm (MCL)*, which was invented by van Dongen (van Dongen, 2000a). This method simulates a peculiar process of flow diffusion in a graph. One starts from the *right stochastic matrix* of the graph $\mathbf{R}$, which we have defined in Section A.2. The element $R_{ij}$ of the stochastic matrix gives the probability that a random walker, sitting at vertex $i$, moves to $j$. The sum of the elements of each column of $R$ is one. Each iteration of the algorithm consists of two steps. In the first step, called *expansion*, the stochastic matrix of the graph is raised to an integer power $p$ (usually $p = 2$). The entry $M_{ij}$ of the resulting matrix gives the probability that a random walker, starting from vertex $i$, reaches $j$ in $p$ steps (diffusion flow). The second step, which has no physical counterpart, consists in raising each single entry of the matrix $M$ to some power $\alpha$, where $\alpha$ is now real-valued. This operation, called *inflation*, enhances the weights between pairs of vertices with large values of the diffusion flow, which are likely to be in the same community. Next, the elements of each column must be divided by their sum, such that the sum of the elements of the column equals one and a new right stochastic matrix is recovered. After some iterations, the process delivers a stable matrix, with some remarkable properties. Its elements are either zero or one, so it is a sort of adjacency matrix. Most importantly, the graph described by the matrix is disconnected, and its connected components are the com-

munities of the original graph. The method is really simple to implement, which is the main reason of its success: as of now, the MCL is one of the most used clustering algorithms in bioinformatics. The code can be downloaded from `http://www.micans.org/mcl/`. Due to the matrix multiplication of the expansion step, the algorithm should scale as $O(n^3)$, even if the graph is sparse, as the running matrix becomes quickly dense after a few steps of the algorithm. However, while computing the matrix multiplication, MCL keeps only a maximum number $k$ of non-zero elements per column, where $k$ is usually much smaller than $n$. So, the actual worst-case running time of the algorithm is $O(nk^2)$ on a sparse graph. A problem of the method is the fact that the final partition is sensitive to the parameter $\alpha$ used in the inflation step. Therefore several partitions can be obtained, and it is not clear which are the most meaningful or representative.

### C. Synchronization

Synchronization (Pikovsky *et al.*, 2001) is an emergent phenomenon occurring in systems of interacting units and is ubiquitous in nature, society and technology. In a synchronized state, the units of the system are in the same or similar state(s) at every time. Synchronization has also been applied to find communities in graphs. If oscillators are placed at the vertices, with initial random phases, and have nearest-neighbour interactions, oscillators in the same community synchronize first, whereas a full synchronization requires a longer time. So, if one follows the time evolution of the process, states with synchronized clusters of vertices can be quite stable and long-lived, so they can be easily recognized. This was first shown by Arenas, Díaz-Guilera and Pérez-Vicente (Arenas *et al.*, 2006). They used Kuramoto oscillators (Kuramoto, 1984), which are coupled two-dimensional vectors endowed with a proper frequency of oscillations. In the Kuramoto model, the phase $\theta_i$ of oscillator $i$ evolves according to the following dynamics

$$\frac{d\theta_i}{dt} = \omega_i + \sum_j K \sin(\theta_j - \theta_i), \tag{57}$$

where $\omega_i$ is the natural frequency of $i$, $K$ the strength of the coupling between oscillators and the sum runs over all oscillators (mean field regime). If the interaction coupling exceeds a threshold, depending on the width of the distribution of natural frequencies, the dynamics leads to synchronization. If the dynamics runs on a graph, each oscillator is coupled only to its nearest neighbors. In order to reveal the effect of local synchronization, Arenas et al. introduced the local order parameter

$$\rho_{ij}(t) = \langle \cos(\theta_i(t) - \theta_j(t)) \rangle, \tag{58}$$

measuring the average correlation between oscillators $i$ and $j$. The average is computed over different initial conditions. By visualizing the correlation matrix $\rho(t)$ at a given time $t$, one may distinguish groups of vertices that synchronize together. The groups can be identified by means of the *dynamic connectivity matrix* $\mathcal{D}_{\mathbf{t}}(T)$, which is a binary matrix obtained from $\rho(t)$ by thresholding its entries. The dynamic connectivity matrix embodies information about both the synchronization dynamics and the underlying graph topology. From the spectrum of $\mathcal{D}_{\mathbf{t}}(T)$ it is possible to derive the number of disconnected components at time $t$. By plotting the number of components as a function of time, plateaus may appear at some characteristic time scales, indicating structural scales of the graph with robust communities (Fig. 18). Partitions corresponding to long plateaus are characterized by high values of the modularity of Newman and Girvan (Section III.C.2) on graphs with homogeneous degree distributions, whereas such correlation is poor in the presence of hubs (Arenas and Díaz-Guilera, 2007). Indeed, it has been proven that the stability (Eq. 56) of the dynamics associated to the standard Laplacian matrix, which describes the convergence towards synchronization of the Kuramoto model with equal intrinsic frequencies, coincides with modularity only for graphs whose vertices have the same degree (Lambiotte *et al.*, 2008). The appearance of plateaus at different time scales hints to a hierarchical organization of the graph. After a sufficiently long $t$ all oscillators are synchronized and the whole system behaves as a single component. Interestingly, Arenas et al. found that the structural scales revealed by synchronization correspond to groups of eigenvalues of the Laplacian matrix of the graph, separated by gaps.

Based on the same principle, Boccaletti et al. designed a community detection method based on synchronization (Boccaletti *et al.*, 2007). The synchronization dynamics is a variation of Kuramoto's model, the opinion changing rate (OCR) model (Pluchino *et al.*, 2005). Here the interaction coupling between adjacent vertices is weighted by a term proportional to a (negative) power of the betweenness of the edge connecting the vertices (Section V.A), with exponent $\alpha$. The evolution equations of the model are solved by decreasing the value of $\alpha$ during the evolution of the dynamics, starting from a configuration in which the system is fully synchronized ($\alpha = 0$). The graph tends to get split into clusters of synchronized elements, because the interaction strengths across inter-cluster edges get suppressed due to their high betweenness scores. By varying $\alpha$, different partitions are recovered, from the graph as a whole until the vertices as separate communities: the partition with the largest value of modularity is taken as the most relevant. The algorithm scales in a time $O(mn)$, or $O(n^2)$ on sparse graphs, and gives good results in practical examples, including Zachary's karate club (Zachary, 1977) and the benchmark by Girvan and Newman (Girvan and Newman, 2002) (Section XIV.A). The method can be refined by homogeneizing the natural frequencies of the oscillators during the evolution of the system. In this way, the system becomes more stable and partitions with higher modularity values can be recovered.

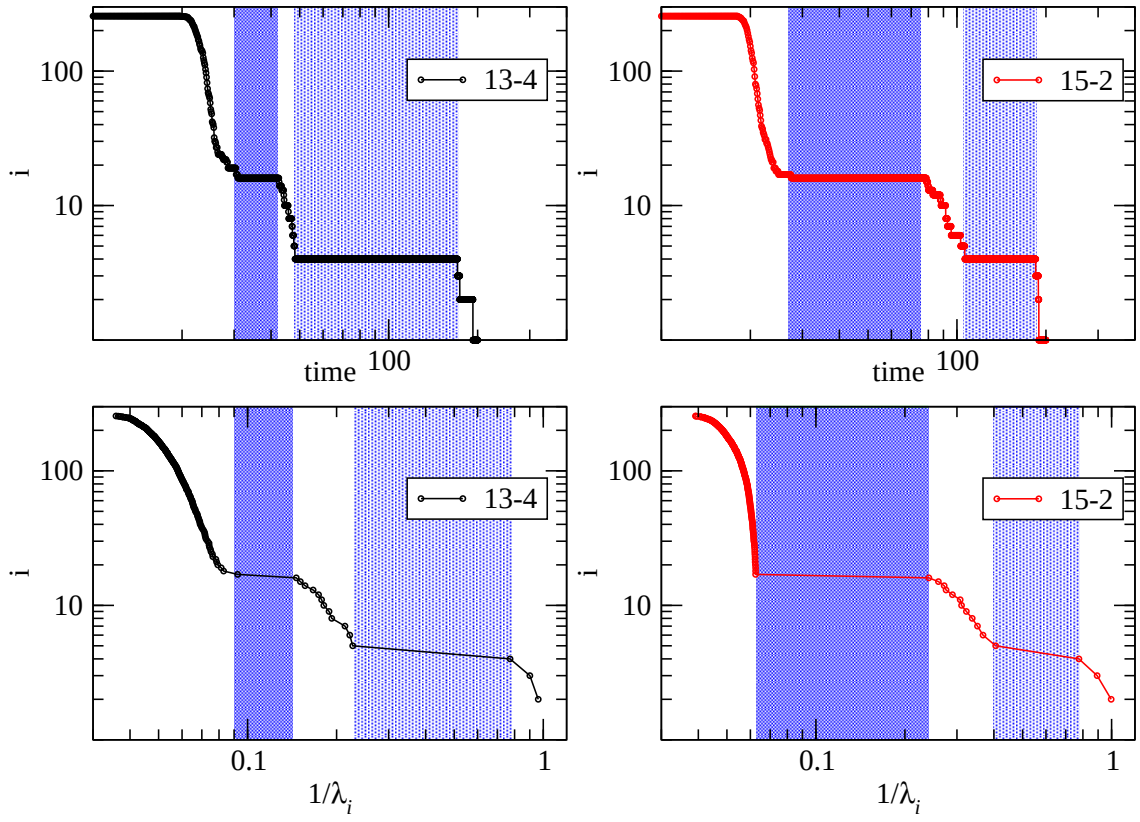


FIG. 18 Synchronization of Kuramoto oscillators on graphs with two hierarchical levels of communities. (Top) The number of different synchronized components is plotted versus time for two graphs with different densities of edges within the clusters. (Bottom) The rank index of the eivenvalues of the Laplacian matrices of the same two graphs of the upper panels is plotted versus the inverse eigenvalues. (the ranking goes from the highest to the smallest eigenvalue). The two types of communities are revealed by the plateaus. Reprinted figure with permission from (Arenas *et al.*, 2006). ©2006 by the American Physical Society.

Synchronization-based algorithms may not be reliable when communities are very different in size; tests in this direction are still missing.

## IX. METHODS BASED ON STATISTICAL INFERENCE

*Statistical inference* (Mackay, 2003) aims at deducing properties of data sets, starting from a set of observations and model hypotheses. If the data set is a graph, the model, based on hypotheses on how vertices are connected to each other, has to *fit* the actual graph topology. In this section we review those clustering techniques attempting to find the best fit of a model to the graph, where the model assumes that vertices have some sort of classification, based on their connectivity patterns. We mainly focus on methods adopting *Bayesian inference* (Winkler, 2003), in which the best fit is obtained through the maximization of a likelihood (*generative models*), but we also discuss related techniques, based on *blockmodeling* (Doreian *et al.*, 2005), *model selection* (Burnham and Anderson, 2002) and *information theory* (Mackay, 2003).

### A. Generative models

Bayesian inference uses observations to estimate the probability that a given hypothesis is true. It consists of two ingredients: the evidence, expressed by

the information $D$ one has about the system (e.g., through measurements); a statistical model with parameters $\{\theta\}$. Bayesian inference starts by writing the likelihood $P(D|\{\theta\})$ that the observed evidence is produced by the model for a given set of parameters $\{\theta\}$. The aim is to determine the choice of $\{\theta\}$ that maximizes the posterior distribution $P(\{\theta\}|D)$ of the parameters given the model and the evidence. By using Bayes' theorem one has

$$P(\{\theta\}|D) = \frac{1}{Z} P(D|\{\theta\}) P(\{\theta\}), \tag{59}$$

where $P(\{\theta\})$ is the prior distribution of the model parameters and

$$Z = \int P(D|\{\theta\}) P(\{\theta\}) d\theta. \tag{60}$$

Unfortunately, computing the integral 60 is a major challenge. Moreover, the choice of the prior distribution $P(\{\theta\})$ is non-obvious. Generative models differ from each other by the choice of the model and the way they address these two issues.

Bayesian inference is frequently used in the analysis and modeling of real graphs, including social (Handcock *et al.*, 2007; Koskinen and Snijders, 2007; Rhodes and Keefe, 2007) and biological networks (Berg and Lässig, 2006; Rowicka and Kudlicki, 2004). Graph clustering can be considered a specific example of inference problem. Here, the evidence is represented by the graph structure (adjacency or weight matrix) and there is an additional ingredient, represented by the classification of the vertices in groups, which is a *hidden* (or *missing*) information that one wishes to infer along with the parameters of the model which is supposed to be responsible for the classification. This idea is at the basis of several recent papers, which we discuss here. In all these works, one essentially maximizes the likelihood $P(D|\{\theta\})$ that the model is consistent with the observed graph structure, with different constraints. We specify the set of parameters $\{\theta\}$ as the triplet $(\{q\}, \{\pi\}, k)$, where $\{q\}$ indicates the community assignment of the vertices, $\{\pi\}$ the model parameters, and $k$ the number of clusters. In the following we shall stick to the notation of the papers, so the variables above may be indicated by different symbols. However, to better show what each method specifically does we shall refer to our general notation at the end of the section.

Hastings (Hastings, 2006) chooses as a model of network with communities the *planted partition model* (Section XIV). In it, $n$ vertices are assigned to $q$ groups: vertices of the same group are linked with a probability $p_{in}$, while vertices of different groups are linked with a probability $p_{out}$. If $p_{in} > p_{out}$, the model graph has a built-in community structure. The vertex classification is indicated by the set of labels $\{q_i\}$. The probability that, given a graph, the classification $\{q_i\}$ is the right one according to the model is[8]

$$p(\{q_i\}) \propto \{\exp[-\sum_{\langle ij \rangle} J\delta_{q_i q_j} - \sum_{i \neq j} J'\delta_{q_i q_j}/2]\}^{-1}, \tag{61}$$

where $J = \log\{[p_{in}(1-p_{out})]/[p_{out}(1-p_{in})]\}$, $J' = \log[(1-p_{in})/(1-p_{out})]$ and the first sum runs over nearest neighboring vertices. Maximizing $p(\{q_i\})$ is equivalent to minimizing the argument of the exponential, which is the Hamiltonian of a Potts model with short- and long-range interactions. For $p_{in} > p_{out}$, $J > 0$ and $J' < 0$, so the model is a spin glass with ferromagnetic nearest-neighbor interactions and antiferromagnetic long-range interactions, similar to the model proposed by Reichardt and Bornholdt to generalize Newman-Girvan modularity (Reichardt and Bornholdt, 2006a) (Section VI.B). Hastings used belief propagation (Gallager, 1963) to find the ground state of the spin model. On sparse graphs, the complexity of the algorithm is expected to be $O(n \log^{\alpha} n)$, where $\alpha$ needs to be estimated numerically. In principle one needs to input the parameters $p_{in}$ and $p_{out}$, which are usually unknown in practical applications. However, it turns out that they can be chosen rather arbitrarily, and that bad choices can be recognized and corrected.

Newman and Leicht (Newman and Leicht, 2007) have recently proposed a similar method based on a mixture model and the expectation-maximization technique (Dempster *et al.*, 1977). The method bears some resemblance with an *a posteriori* blockmodel previously introduced by Snijders and Nowicki (Nowicki and Snijders, 2001; Snijders and Nowicki, 1997). They start from a directed graph with $n$ vertices, whose vertices fall into $c$ classes. The group of vertex $i$ is indicated by $g_i$, $\pi_r$ the fraction of vertices in group $r$, and $\theta_{ri}$ the probability that there is a directed edge from vertices of group $r$ to vertex $i$. By definition, the sets $\{\pi_i\}$ and $\{\theta_{ri}\}$ satisfy the normalization conditions $\sum_{r=1}^{c} \pi_i = 1$ and $\sum_{i=1}^{n} \theta_{ri} = 1$. Apart from normalization, the probabilities $\{\theta_{ri}\}$ are assumed to be independent of each other. The best classification of the vertices corresponds to the maximum of the average log-likelihood $\bar{\mathcal{L}}$ that the model, described by the values of the parameters $\{\pi_i\}$ and $\{\theta_{ri}\}$ fits the adjacency matrix $\mathbf{A}$ of the graph. The expression of the average log-likelihood $\bar{\mathcal{L}}$ requires the definition of the probability $q_{ir} = Pr(g_i = r|A, \pi, \theta)$, that vertex $i$ belongs to group $g$. By applying Bayes' theorem the probabilities $\{q_{ir}\}$ can be computed in terms of the $\{\pi_i\}$ and the $\{\theta_{ri}\}$, as

$$q_{ir} = \frac{\pi_r \prod_j \theta_{rj}^{A_{ij}}}{\sum_s \pi_s \prod_j \theta_{sj}^{A_{ij}}}, \tag{62}$$

while the maximization of the average log-likelihood $\bar{\mathcal{L}}$,

[8] The actual likelihood includes an additional factor expressing the a priori probability of the community sizes. Hastings assumes that this probability is constant.

under the normalization constraints of the model variables $\{\pi_i\}$ and $\{\theta_{ri}\}$, yields the relations

$$\pi_r = \frac{1}{n}\sum_i q_{ir}, \quad \theta_{rj} = \frac{\sum_i A_{ij} q_{ir}}{\sum_i k_i q_{ir}}, \tag{63}$$

where $k_i$ is the outdegree of vertex $i$. Equations 62 and 63 are self-consistent, and can be solved by iterating them to convergence, starting from a suitable set of initial conditions. Convergence is fast, so the algorithm could be applied to fairly large graphs, with up to about $10^6$ vertices.

The method, designed for directed graphs, can be easily extended to the undirected case, whereas an extension to weighted graphs is not straightforwad. A nice feature of the method is that it does not require any preliminary indication on what type of structure to look for; the resulting structure is the most likely classification based on the connectivity patterns of the vertices. Therefore, various types of structures can be detected, not necessarily communities. For instance, multipartite structure could be uncovered, or mixed patterns where multipartite subgraphs coexist with communities, etc.. In this respect, it is more powerful than most methods of community detection, which are bound to focus only on proper communities, i.e. subgraphs with more internal than external edges. In addition, since partitions are defined by assigning probability values to the vertices, expressing the extent of their membership in a group, it is possible that some vertices are not clearly assigned to a group, but to more groups, so the method is able to deal with overlapping communities. The main drawback of the algorithm is the fact that one needs to specify the number of groups $c$ at the beginning of the calculation, a number that is typically unknown for real networks. It is possible to derive this information self-consistently by maximizing the probability that the data are reproduced by partitions with a given number of clusters. But this procedure involves some degree of approximation, and the results are often not good.

In a recent study it has been shown that the method by Newman and Leicht enables one to rank vertices based on their degree of influence on other vertices, which allows to identify the vertices responsible for the group structure and its stability (Mungan and Ramasco, 2008). A very similar technique has also been applied by Vázquez (Vazquez, 2008) to the problem of population stratification, where animal populations and their attributes are represented as hypergraphs (Section A.1). Vázquez also suggested an interesting criterion to decide the optimal number of clusters, namely picking the number $\bar{c}$ whose solution has the greatest similarity with solutions obtained at different values of $c$. The similarity between two partitions can be estimated in various ways, for instance by computing the normalized mutual information (Section XIV). In a successive paper (Vazquez, 2008), Vázquez showed that better results are obtained if the classification likelihood is maximized by using Variational Bayes (Beal, 2003; Jordan *et al.*, 1999).

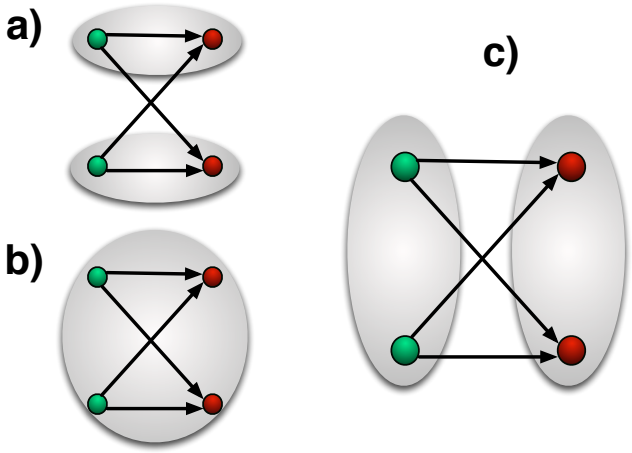


FIG. 19 Problem of method by Newman and Leicht. By applying the method to the illustrated complete bipartite graph (colors indicate the vertex classes) the natural group structure c) is not recovered; instead, the most likely classifications are a) and b). Reprinted figure with permission from (Ramasco and Mungan, 2008).©2008 by the American Physical Society.

Ramasco and Mungan (Ramasco and Mungan, 2008) remarked that the normalization condition on the probabilities $\{\theta_{ri}\}$ implies that each group $r$ must have nonzero outdegree and that therefore the method fails to detect the intuitive group structure of (directed) bipartite graphs (Fig. 19). To avoid this problem, they proposed a modification, that consists in introducing three sets for the edge probabilities $\{\theta_{ri}\}$, relative to edges going from group $r$ to vertex $i$ (as before), from $i$ to $r$ and in both directions, respectively. Furthermore, they used the average entropy of the classification $S_q = -(\sum_{i,r} q_{ir} \ln q_{ir})/n$, where the $q_{ir}$ are the analogs of the probabilities in Eq. 62, to infer the optimal number of groups, that the method of Newman and Leicht is unable to provide. Another technique similar to that by Newman and Leicht has been designed by Ren et al. (Ren *et al.*, 2007). The model is based on the group fractions $\{\pi_i\}$, defined as above, and a set of probabilities $\{\beta_{r,i}\}$, expressing the relevance of vertex $i$ for group $r$; the basic assumption is that the probability that two vertices of the same group are connected by an edge is proportional to the product of the relevances of the two vertices. In this way, there is an explicit relation between group membership and edge density, and the method can only detect community structure. The community assignments are recovered through an expectation-maximization procedure that closely follows that by Newman and Leicht.

Maximum likelihood estimation has been used by Čopič et al. to define an axiomatization of the problem of graph clustering and its related concepts (Čopič *et al.*, 2005). The starting point is again the planted partition model (Section XIV), with probabilities $p_{in}$ and $p_{out}$. A novelty of the approach is the introduction of the *size matrix* $\mathbf{S}$, whose element $S_{ij}$ indicates the max-

imum strength of interaction between vertices $i$ and $j$. For instance, in a graph with unweighted connections, all elements of $\mathbf{S}$ equal 1. In this case, the probability that the graph conceals a community structure coincides with the expression (61) by Hastings. Čopič et al. used this probability as a quality function to define rankings between graph partitions (*likelihood rankings*). The authors show that the likelihood rankings satisfy a number of general properties, which should be satisfied by any reasonable ranking. They also propose an algorithm to find the maximum likelihood partition, by using the auxiliary concept of *pseudo-community structure*, i. e. a grouping of the graph vertices in which it is specified which pairs of vertices stay in the same community and which pairs instead stay in different communities. A pseudo-community may not be a community because the transitive property is not generally valid, as the focus is on pairwise vertex relationships: it may happen that $i$ and $j$ are classified in the same group, and that $j$ and $k$ are classified in the same group, but that $i$ and $k$ are not classified as belonging to the same group. We believe that the work by Čopič et al. is an important first step towards a more rigorous formalization of the problem of graph clustering.

Zanghi et al. (Zanghi *et al.*, 2008) have designed a clustering technique that lies somewhat in between the method by Hastings and that by Newman and Leicht. As in (Hastings, 2006), they use the planted partition model to represent a graph with community structure; as in (Newman and Leicht, 2007), they maximize the classification likelihood using an expectation-maximization algorithm (Dempster *et al.*, 1977). The algorithm runs for a fixed number of clusters $q$, like that by Newman and Leicht; however, the optimal number of clusters can be determined by running the algorithm for a range of $q$-values and selecting the solution that maximizes the Integrated Classification Likelihood introduced by Biernacki et al. (Biernacki *et al.*, 2000). The time complexity of the algorithm is $O(n^2)$.

Hofman and Wiggins have proposed a general Bayesian approach to the problem of graph clustering (Hofman and Wiggins, 2008). Like Hastings (Hastings, 2006), they model a graph with community structure as in the planted partition problem (Section XIV), in that there are two probabilities $\theta_c$ and $\theta_d$ that there is an edge between vertices of the same or different clusters, respectively. The unobserved community structure is indicated by the set of labels $\vec{\sigma}$ for the vertices; $\pi_r$ is again the fraction of vertices in group $r$. The conjugate prior distributions $p(\vec{\theta})$ and $p(\vec{\pi})$ are chosen to be Beta and Dirichlet distributions. The most probable number of clusters $K^*$ maximizes the conditional probability $p(K|\mathbf{A})$ that there are $K$ clusters, given the matrix $\mathbf{A}$. Like Hastings, Hofman and Wiggins assume that the prior probability $p(K)$ on the number of clusters is a smooth function, therefore maximizing $p(K|\mathbf{A})$ amounts to maximizing the Bayesian evidence $p(\mathbf{A}|K) \propto p(K|\mathbf{A})/p(K)$, obtained by integrating the joint distribution $p(\mathbf{A}|\vec{\sigma},\vec{\pi},\vec{\theta}|K)$, which is factorizable, over the model parameters $\vec{\theta}$ and $\vec{\pi}$. The integration can be performed exactly only for small graphs. Hofman and Wiggins used Variational Bayes (Beal, 2003; Jordan *et al.*, 1999), in order to compute controlled approximations of $p(\mathbf{A}|K)$. The complexity of the algorithm was estimated numerically on synthetic graphs, yielding $O(n^\alpha)$, with $\alpha = 1.44$. In fact, the main limitation comes from high memory requirements. The method is more powerful than the one by Hastings (Hastings, 2006), in that the edge probabilities $\vec{\theta}$ are inferred by the procedure itself and need not be specified (or guessed) at the beginning. It also includes the expectation-maximization approach by Newman and Leicht (Newman and Leicht, 2007) as a special case, with the big advantage that the number of clusters need not be given as an input, but is an output of the method. The software of the algorithm can be found at `http://www.columbia.edu/~chw2/`.

We conclude with a brief summary on the techniques described above, coming back to our notation at the beginning of the section. In the method by Hastings, one maximizes the likelihood $P(D|\{q\},\{\pi\},k)$ over the set of all possible community assignments $\{q\}$, given the number of clusters $k$ and the model parameters (i.e. the linking probabilities $p_{in}$ and $p_{out}$). Newman and Leicht maximize the likelihood $P(D|\{q\},\{\pi\},k)$ for a given number of clusters, over the possible choices for the model parameters and community assignments, by deriving the optimal choices for both variables with a self-consistent procedure. Hofman and Wiggins maximize the likelihood $P_{HW}(k) = \sum_{\{q\}} \int P(D|\{q\},\{\pi\},k)P(\{q\}|\{\pi\})P(\{\pi\})d\pi$ over the possible choices for the number of clusters.

### B. Blockmodeling, model selection & information theory

Block modeling is a common approach in statistics and social network analysis to decompose a graph in classes of vertices with common properties. In this way, a simpler description of the graph is attained. Vertices are usually grouped in classes of equivalence. There are two main definitions of topological equivalence for vertices: *structural equivalence* (F.Lorrain and White, 1971) (Section III.B.4), in which vertices are equivalent if they have the same neighbors[9]; *regular equivalence* (Everett and Borgatti, 1994; White and Reitz, 1983), in which vertices of a class have similar connection patterns to vertices of the other classes (ex. parents/children). Regular equivalence does not require that ties/edges are restricted to specific target vertices, so it is a more general concept than structural equivalence. Indeed, vertices which are

[9] More generally, if they have the same ties/edges to the same vertices, as in a social network there may be different types of ties/edges.)

structurally equivalent are also regularly equivalent, but the inverse is not true. The concept of structural equivalence can be generalized to probabilistic models, in which one compares classes of graphs, not single graphs, characterized by a set of linking probabilities between the vertices. In this case, vertices are organized in classes such that the linking probabilities of a vertex with all other vertices of the graph are the same for vertices in the same class, which are called *stochastically equivalent* (Fienberg and Wasserman, 1981; Holland *et al.*, 1983).

A thorough discussion of blockmodeling is beyond the scope of this review: we point the reader to (Doreian *et al.*, 2005). Here we discuss a recent work by Reichardt and White (Reichardt and White, 2007). Let us suppose to have a directed graph with $n$ vertices and $m$ edges. A classification of the graph is indicated by the set of labels $\{\sigma\}$, where $\sigma_i = 1, 2, ..., q$ is the class of vertex $i$. The corresponding blockmodel, or *image graph*, is expressed by a $q \times q$ adjacency matrix $\mathbf{B}$: $B_{q_1 q_2} = 1$ if edges between class $q_1$ and $q_2$ are allowed, otherwise it is zero. The aim is finding the classification $\{\sigma\}$ and the matrix $B$ that best fit the adjacency matrix $\mathbf{A}$ of the graph. The goodness of the fit is expressed by the quality function

$$\mathcal{Q}^B(\{\sigma\}) = \frac{1}{m}\sum_{i\neq j}[a_{ij}A_{ij}B_{\sigma_i\sigma_j} + b_{ij}(1-A_{ij})(1-B_{\sigma_i\sigma_j})], \tag{64}$$

where $a_{ij}$ ($b_{ij}$) reward the presence (absence) of edges between vertices if there are edges (non-edges) between the corresponding classes, and $m$ is the number of edges of the graph, as usual. Eq. 64 can be rewritten as a sum over the classes

$$\mathcal{Q}^B(\{\sigma\}) = \sum_{r,s}^{q}(e_{rs} - [e_{rs}])B_{rs}, \tag{65}$$

by setting $e_{rs} = (1/m)\sum_{i\neq j}(a_{ij} + b_{ij})A_{ij}\delta_{\sigma_i r}\delta_{\sigma_j s}$ and $[e_{rs}] = (1/m)\sum_{i\neq j} b_{ij}\delta_{\sigma_i r}\delta_{\sigma_j s}$. If one sets $a_{ij} = 1 - p_{ij}$ and $b_{ij} = p_{ij}$, $p_{ij}$ can be interpreted as the linking probability between $i$ and $j$, in some null model. Thereof, $e_{rs}$ becomes the number of edges running between vertices of class $r$ and $s$, and $[e_{rs}]$ the expected number of edges in the null model. Reichardt and White set $p_{ij} = k_i^{out}k_j^{in}/m$, which defines the same null model of Newman-Girvan modularity for directed graphs (Section VI.B). In fact, if the image graph has only self-edges, i.e. $B_{rs} = \delta_{rs}$, the quality function $\mathcal{Q}^B(\{\sigma\})$ exactly matches modularity. Other choices for the image graph are possible, however. For instance, a matrix $B_{rs} = 1 - \delta_{rs}$ describes the classes of a $q$-partite graph (Section A.1). From Eq. 65 we see that, for a given classification $\{\sigma\}$, the image graph that yields the largest value of the quality function $\mathcal{Q}^B(\{\sigma\})$ is that in which $B_{rs} = 1$ when the term $e_{rs} - [e_{rs}]$ is non-negative, and $B_{rs} = 0$ when the term $e_{rs} - [e_{rs}]$ is non-positive. So, the best classification is the one maximizing the quality function

$$\mathcal{Q}^*(\{\sigma\}) = \frac{1}{2}\sum_{r,s}^{q}||e_{rs} - [e_{rs}]||, \tag{66}$$

where all terms of the sum are taken in absolute value. The function $\mathcal{Q}^*(\{\sigma\})$ is maximized via simulated annealing. The absolute maximum $Q_{max}$ is obtained by construction when $q$ matches the number $q^*$ of structural equivalence classes of the graph. However, the absolute maximum $Q_{max}$ does not have a meaning by itself, as one can achieve fairly high values of $\mathcal{Q}^*(\{\sigma\})$ also for null model instances of the original graph, i. e. if one randomizes the graph by keeping the same expected indegree and outdegree sequences. In practical applications, the optimal number of classes is determined by comparing the ratio $Q^*(q)/Q^{max}$ ($Q^*(q)$ is the maximum of $\mathcal{Q}^*(\{\sigma\})$ for $q$ classes) with the expected ratio for the null model. Since classifications for different $q$-values are not hierarchically ordered, overlaps between classes may be detected. The method can be trivially extended to the case of weighted graphs.

Model selection (Burnham and Anderson, 2002) aims at finding models which are at the same time simple and good at describing a system/process. A basic example of a model selection problem is curve fitting. There is no clear-cut recipe to select a model, but a bunch of heuristics, like Akaike Information Criterion (AIC) (Akaike, 1974), Bayesian Information Criterion (BIC) (Schwarz, 1978), Minimum Description Length (MDL) (Grünwald *et al.*, 2005; Rissanen, 1978), Minimum Message Length (MML) (Wallace and Boulton, 1968), etc..

The modular structure of a graph can be considered as a compressed description of the graph to approximate the whole information contained in its adjacency matrix. Based on this idea, Rosvall and Bergstrom (Rosvall and Bergstrom, 2007) envisioned a communication process in which a partition of a graph in communities represents a synthesis $Y$ of the full structure that a signaler sends to a receiver, who tries to infer the original graph topology $X$ from it (Fig. 20). The same idea is at the basis of an earlier method by Sun et al. (Sun *et al.*, 2007), which was originally designed for bipartite graphs evolving in time and will be described in Section XV.B. The best partition corresponds to the signal $Y$ that contains the most information about $X$. This can be quantitatively assessed by the minimization of the conditional information $H(X|Y)$ of $X$ given $Y$,

$$H(X|Y) = \log\left[\prod_{i=1}^{q}\binom{n_i(n_i-1)/2}{l_{ii}}\prod_{i>j}\binom{n_i n_j}{l_{ij}}\right], \tag{67}$$

where $q$ is the number of clusters, $n_i$ the number of vertices in cluster $i$, $l_{ij}$ the number of edges between clusters $i$ and $j$. We remark that, if one imposes no constraints on $q$, $H(X|Y)$ is minimal in the trivial case in which $X = Y$ ($H(X|X) = 0$). This solution is not acceptable

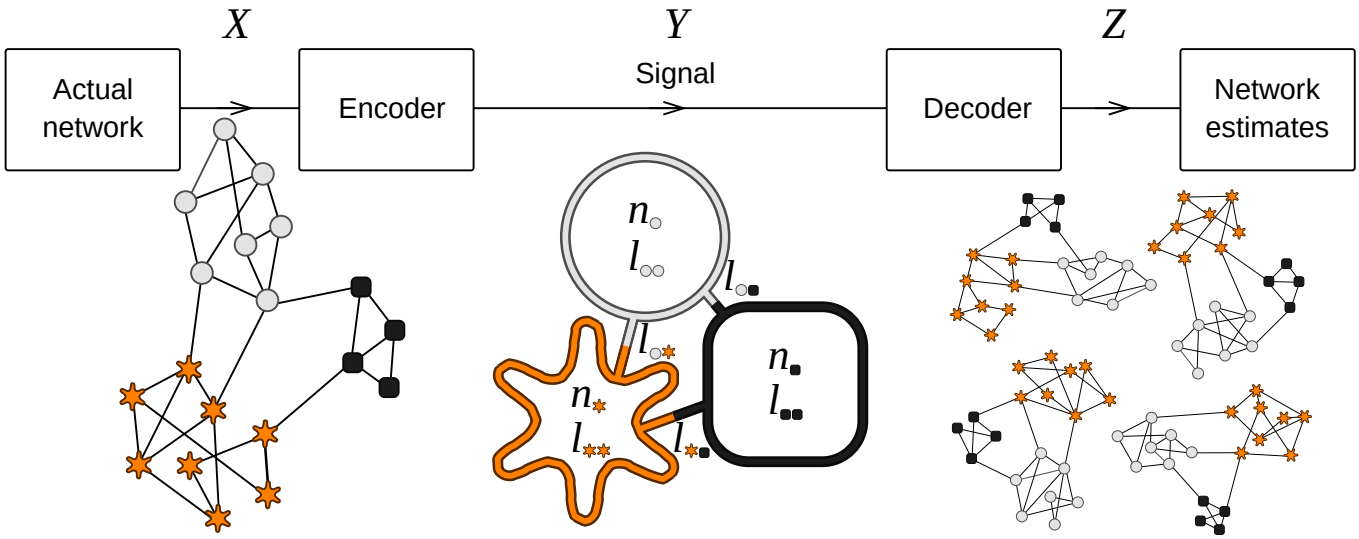


FIG. 20 Basic principle of the method by Rosvall and Bergstrom (Rosvall and Bergstrom, 2007). An encoder sends to a decoder a compressed information about the topology of the graph on the left. The information gives a coarse description of the graph, which is used by the decoder to deduce the original graph structure. Reprinted figure with permission from (Rosvall and Bergstrom, 2007). ©2007 by the National Academy of Science of the USA.

because it does not correspond to a compression of information with respect to the original data set. One has to look for the ideal tradeoff between a good compression and a small enough information $H(X|Y)$. The Minimum Description Length (MDL) principle (Grünwald *et al.*, 2005; Rissanen, 1978) provides a solution to this problem, which amounts to the minimization of a function given by $H(X|Y)$ plus a function of the number $n$ of vertices, $m$ of edges and $q$ of clusters. The optimization is performed by simulated annealing, so the method is rather slow and can be applied to graphs with up to about $10^4$ vertices. However, faster techniques may in principle be used, even if they imply a loss in accuracy. The method appears superior than modularity optimization, especially when communities are of different sizes. This comes from tests performed on the benchmark of Girvan and Newman (Girvan and Newman, 2002) (Section XIV.A), both in its original version and in asymmetric versions, proposed by the authors, where the clusters have different sizes or different average degrees. In addition, it can detect other types of vertex classifications than communities, as in Eq. 67 there are no constraints on the relative importance of the edge densities within communities with respect to the edge densities between communities. The software of the algorithm can be found at `http://www.tp.umu.se/~rosvall/code.html`.

In a recent paper (Rosvall and Bergstrom, 2008), Rosvall and Bergstrom pursued the same idea of describing a graph by using less information than that encoded in the full adjacency matrix. The goal is to optimally compress the information needed to describe the process of information diffusion across the graph. Random walk is chosen as a proxy of information diffusion. A two-level description, in which one gives unique names to important structures of the graph and to vertices within the same structure, but the vertex names are recycled among different structures, leads to a more compact description than by simply coding all vertices with different names. This is similar to the procedure usually adopted in geographic maps, where the structures are cities and one usually chooses the same names for streets of different cities, as long as there is only one street with a given name in the same city. Huffman coding (Huffman, 1952) is used to name vertices. For the random walk, the above-mentioned structures are communities, as it is intuitive that walkers will spend a lot of time within them, so they play a crucial role in the process of information diffusion. Graph clustering turns then into the following coding problem: finding the partition that yields the minimum description length of an infinite random walk. Such description length consists of two terms, expressing the Shannon entropy of the random walk within and between clusters. The optimum is computed by combining greedy search with simulated annealing. The method can be applied to weighted graphs, both undirected and directed. In the latter case, the random walk process is modified by introducing a jump probability $\tau$, to guarantee ergodicity, just like in Google's Pagerank algorithm (Brin and Page, 1998). The partitions of directed graphs obtained by the method differ from those derived by optimizing the directed version of Newman-Girvan modularity (Section VI.B): this is due to the fact that modularity focuses on pairwise relationships between vertices, so it does not capture flows. The code of the method is available at `http://www.tp.umu.se/~rosvall/code.html`.

Information theory has also been used to detect communities in graphs. Ziv et al. (Ziv *et al.*, 2005) have designed a method in which the information contained in

the graph topology is compressed such to preserve some predefined information. This is the basic principle of the information bottleneck method (Tishby *et al.*, 1999). To understand this criterion, we need to introduce an important measure, the *mutual information* $I(X,Y)$ (Mackay, 2003) of two random variables $X$ and $Y$. It is defined as

$$I(X,Y) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}, \qquad (68)$$

where $P(x)$ indicates the probability that $X = x$ (similarly for $P(y)$) and $P(x,y)$ is the joint probability of $X$ and $Y$, i. e. $P(x,y) = P(X = x, Y = y)$. The measure $I(X,Y)$ tells how much we learn about $X$ if we know $Y$, and viceversa. If $X$ is the input variable, $Z$ the variable specifying the partition and $Y$ the variable encoding the information we want to keep, which is called *relevant variable*, the goal is to minimize the mutual information between $X$ and $Z$ (to achieve the largest possible data compression), under the constraint that the information on $Y$ extractable from $Z$ be accurate. The optimal tradeoff between the values of $I(X,Z)$ and $I(Y,Z)$ (i.e. compression versus accuracy) is expressed by the minimization of a functional, where the relative weight of the two contributions is given by a parameter playing the role of a temperature. In the case of graph clustering, the question is what to choose as relevant information variable. Ziv et al. proposed to adopt the structural information encoded in the process of diffusion on the graph. They also introduce the concept of *network modularity*, which characterizes the graph as a whole, not a specific partition like the modularity by Newman and Girvan (Section III.C.2). The network modularity is defined as the area under the *information curve*, which essentially represents the relation between the extent of compression and accuracy for all solutions found by the method and all possible numbers of clusters. The software of the algorithm by Ziv et al. can be found at `http://www.columbia.edu/~chw2/`.

## X. OTHER METHODS

In this section we describe some algorithms that do not fit in the previous categories, although some overlap is possible.

Raghavan et al. (Raghavan *et al.*, 2007) have designed a simple and fast method based on *label propagation*. Vertices are initially given unique labels (e.g. their vertex labels). At each iteration, a sweep over all vertices, in random sequential order, is performed: each vertex takes the label shared by the majority of its neighbors. If there is no unique majority, one of the majority labels is picked at random. In this way, labels propagate across the graph: most labels will disappear, others will dominate. The process reaches convergence when each vertex has the majority label of its neighbors. Communities are defined as groups of vertices having identical labels at convergence. By construction, each vertex has more neighbors in its community than in any other community. This resembles the strong definition of community we have discussed in Section III.B.2, although the latter is stricter, in that each vertex must have more neighbors in its community than in the rest of the graph. The algorithm does not deliver a unique solution. Due to the many ties encountered along the process it is possible to derive different partitions starting from the same initial condition, with different random seeds. Tests on real graphs show that all partitions found are similar to each other, though. The most precise information that one can extract from the method is contained by aggregating the various partitions obtained, which can be done in various ways. The authors proposed to label each vertex with the set of all labels it has in different partitions. Aggregating partitions enables one to detect possible overlapping communities. The main advantage of the method is the fact that it does not need any information on the number and the size of the clusters. It does not need any parameter, either. The time complexity of each iteration of the algorithm is $O(m)$, the number of iterations to convergence appears independent of the graph size, or growing very slowly with it. So the technique is really fast and could be used for the analysis of large systems. In a recent paper (Tibély and Kertész, 2008), Tibély and Kertész showed that the method is equivalent to finding the local energy minima of a simple zero-temperature kinetic Potts model, and that the number of such energy minima is considerably larger than the number of vertices of the graph. Aggregating partitions as Raghavan et al. suggest leads to a fragmentation of the resulting partition in clusters that are the smaller, the larger the number of aggregated partitions. This is potentially a serious problem of the algorithm by Raghavan et al., especially when large graphs are investigated.

Bagrow and Bollt designed an agglomerative technique, called *L-shell method* (Bagrow and Bollt, 2005). It is a procedure that finds the community of any vertex, although the authors also presented a more general procedure to identify the full community structure of the graph. Communities are defined locally, based on a simple criterion involving the number of edges inside and outside a group of vertices. One starts from a vertex-origin and keeps adding vertices lying on successive shells, where a shell is defined as a set of vertices at a fixed geodesic distance from the origin. The first shell includes the nearest neighbours of the origin, the second the next-to-nearest neighbours, and so on. At each iteration, one calculates the number of edges connecting vertices of the new layer to vertices inside and outside the running cluster. If the ratio of these two numbers ("emerging degree") exceeds some predefined threshold, the vertices of the new shell are added to the cluster, otherwise the process stops. The idea of closing a community by expanding a shell has been previously introduced by Costa (da Fontoura Costa, 2004), in which shells are centered on hubs. However, in this procedure the number of clusters is preassigned and no cluster can contain more than one hub.

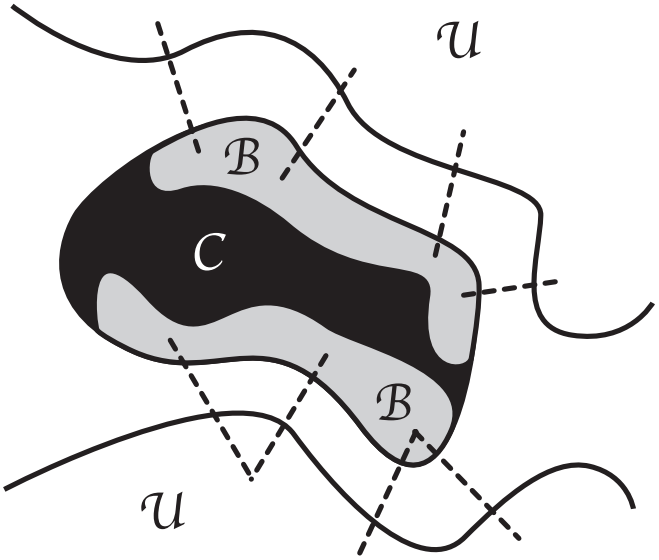


FIG. 21 Schematic picture of a community $\mathcal{C}$ used in the definition of localized modularity by Clauset (Clauset, 2005). The black area indicates the subgraph of $\mathcal{C}$ including all vertices of $\mathcal{C}$, whose neighbors are also in $\mathcal{C}$. The boundary $\mathcal{B}$ entails the vertices of $\mathcal{C}$ with at least one neighbor outside the community. Reprinted figure with permission from (Clauset, 2005). ©2005 by the American Physical Society.

Because of the local nature of the process, the L-shell method is very fast and can identify communities very quickly. By repeating the process starting from every vertex, one could derive a *membership matrix* $M$: the element $M_{ij}$ is one if vertex $j$ belongs to the community of vertex $i$, otherwise it is zero. The membership matrix can be rewritten by suitably permutating rows and columns based on their mutual distances. The distance between two rows (or columns) is defined as the number of entries whose elements differ. If the graph has a clear community structure, the membership matrix takes a block-diagonal form, where the blocks identify the communities. The method enables one to detect overlaps between communities as well (Porter *et al.*, 2007). Unfortunately, the rearrangement of the matrix requires a time $O(n^3)$, so it is quite slow. In a different algorithm by Clauset, local communities are discovered through greedy maximization of a local modularity measure (Clauset, 2005). Given a community $\mathcal{C}$, the boundary $\mathcal{B}$ is the set of vertices of $\mathcal{C}$ with at least one neighbor outside $\mathcal{C}$ (Fig. 21). The localized modularity $R$ by Clauset is the ratio of the number of edges having both endpoints in $\mathcal{C}$ (but at least one in $\mathcal{B}$), with the number of edges having at least one endpoint in $\mathcal{B}$. It is a measure of the sharpness of the community boundary. Its optimization consists of a local exploration of the community starting from a source vertex: at each step the neighboring vertex yielding the largest increase (smallest decrease) of $R$ is added, until the community has reached a predefinite size $n_c$. This greedy optimization takes a time $O(n_c^2\langle k\rangle)$, where $\langle k\rangle$ is the average degree of the graph.

Another method, where communities are defined based on a local criterion, was presented by Eckmann and Moses (Eckmann and Moses, 2002). The idea is to use the clustering coefficient (Watts and Strogatz, 1998) of a vertex as a quantity to distinguish tightly connected groups of vertices. Many edges mean many loops inside a community, so the vertices of a community are likely to have a large clustering coefficient. The latter can be related to the average distance between pairs of neighbours of the vertex. The possible values of the distance are 1 (if neighbors are connected) or 2 (if they are not), so the average distance lies between 1 and 2. The more triangles there are in the subgraph, the shorter the average distance. Since each vertex always has distance 1 from its neighbours, the fact that the average distance between its neighbours is different from 1 reminds what happens when one measures segments on a curved surface. Endowed with a metric, represented by the geodesic distance between vertices/points, and a curvature, the graph can be embedded in a geometric space. Communities appear as portions of the graph with a large curvature. The algorithm was applied to the graph representation of the World Wide Web, where vertices are Web pages and edges are the hyperlinks that take users from a page to the other. The authors found that communities correspond to Web pages dealing with the same topic.

A fast algorithm by Wu and Huberman identifies communities based on the properties of resistor networks (Wu and Huberman, 2004). It is essentially a method for partitioning graphs in two parts, similar to spectral bisection, although partitions in an arbitrary number of communities can be obtained by iterative applications. The graph is transformed into a resistor network where each edge has unit resistance. A unit potential difference is set between two randomly chosen vertices. The idea is that, if there is a clear division in two communities of the graph, there will be a visible gap between voltage values for vertices at the borders between the clusters. The voltages are calculated by solving Kirchoff's equations: an exact resolution would be too time consuming, but it is possible to find a reasonably good approximation in a linear time for a sparse graph with a clear community structure, so the more time consuming part of the algorithm is the sorting of the voltage values, which takes time $O(n \log n)$. Any possible vertex pair can be chosen to set the initial potential difference, so the procedure should be repeated for all possible vertex pairs. The authors showed that this is not necessary, and that a limited number of sampling pairs is sufficient to get good results, so the algorithm scales as $O(n \log n)$ and is very fast. An interesting feature of the method is that it can quickly find the natural community of any vertex, without determining the complete partition of the graph. For that, one uses the vertex as source voltage and places the sink at an arbitrary vertex. The same feature is present in an older algorithm by Flake et al. (Flake *et al.*, 2002), where one uses max-flow instead of current flow (Section IV.A).

The limit of the method is the fact that one has to give as input the number of clusters, which is usually not known beforehand.

Ohkubo and Tanaka (Ohkubo and Tanaka, 2006) pointed out that, since communities are rather compact structures, they should have a small volume, where the volume of a community is defined as the ratio of the number of vertices by the internal edge density of the community. Ohkubo and Tanaka assumed that the sum $V_{total}$ of the volumes of the communities of a partition is a reliable index of the goodness of the partition. So, the most relevant partition is the one minimizing $V_{total}$. The optimization is carried out with simulated annealing.

Zarei and Samani (Zarei and Samani, 2009) remarked that there is a symmetry between community structure and anti-community (multipartite) structure, when one considers a graph and its complement, whose edges are the missing edges of the original graph. In fact, if a graph has a well identified communities, the same groups of vertices would be strong anti-communities in the complement graph, i. e. they should have a few intra-cluster edges and many inter-cluster edges. Based on this remark, the communities of a graph can be identified by looking for anticommunities in the complement graph, which can sometimes be easier. Zarei and Samani devised a spectral method using matrices of the complement graph. The results of this technique appear good as compared to other spectral methods on artificial graphs generated with the planted $\ell$-partition model (Condon and Karp, 2001), as well as on Zachary's karate club (Zachary, 1977), Lusseau's dolphins' network (Lusseau, 2003) and a network of protein-protein interactions. However, the authors have used very small graphs for testing. Communities make sense on sparse graphs, but the complements of large sparse graphs would not be sparse, but very dense, and their community (multipartite) structure basically invisible.

Gudkov and Montealegre detected communities by means of dynamical simplex evolution (Gudkov *et al.*, 2008). Graph vertices are represented as points in an $(n-1)$-dimensional space. Each point initially sits on the $n$ vertices of a simplex, and then moves in space due to forces exerted by the other points. If vertices are neighbors, the mutual force acting on their representative points is attractive, otherwise it is repulsive. If the graph has a clear community structure, the corresponding spatial clusters repel each other because of the few connections between them (repulsion dominates over attraction). If communities are more mixed with each other, clusters are not well separated and they could be mistakenly aggregated in larger structures. To avoid that, Gudkov and Montealegre defined clusters as groups of points such that the distance between each pair of points does not exceed a given threshold, which can be arbitrarily tuned, to reveal structures at different resolutions (Section XII.A). The algorithm consists in solving first-order differential equations, describing the dynamics of mass points moving in a viscous medium. The complexity of the procedure is $O(n^2)$. Differential equations are also at the basis of a recent method designed by Krawczyk and Kułakowski (Krawczyk, 2008; Krawczyk and Kulakowski, 2007). Here the equations describe a dynamic process, in which the original graph topology evolves to a disconnected graph, whose components are the clusters of the original graph.

## XI. METHODS TO FIND OVERLAPPING COMMUNITIES

Most of the methods discussed in the previous sections aim at detecting standard partitions, i.e. partitions in which each vertex is assigned to a single community. However, in real graphs vertices are often shared between communities (Section II), and the issue of detecting overlapping communities has become quite popular in the last few years. We devote this section to the main techniques to detect overlapping communities.

### A. Clique percolation

The most popular technique is the Clique Percolation Method (CPM) by Palla et al. (Palla *et al.*, 2005). It is based on the concept that the internal edges of community are likely to form cliques due to their high density. On the other hand, it is unlikely that intercommunity edges form cliques: this idea was already used in the divisive method of Radicchi et al. (Section V.B). Palla et al. use the term $k$-clique to indicate a complete graph with $k$ vertices[10]. Notice that a $k$-clique is different from the $n$-clique (see Section III.B.2) used in social science. If it were possible for a clique to move on a graph, in some way, it would probably get trapped inside its original community, as it could not cross the bottleneck formed by the intercommunity edges. Palla et al. introduced a number of concepts to implement this idea. Two $k$-cliques are *adjacent* if they share $k-1$ vertices. The union of adjacent $k$-cliques is called *$k$-clique chain*. Two $k$-cliques are connected if they are part of a $k$-clique chain. Finally, a *$k$-clique community* is the largest connected subgraph obtained by the union of a $k$-clique and of all $k$-cliques which are connected to it. Examples of $k$-clique communities are shown in Fig. 22. One could say that a $k$-clique community is identified by making a $k$-clique "roll" over adjacent $k$-cliques, where rolling means rotating a $k$-clique about the $k-1$ vertices it shares with any adjacent $k$-clique. By construction, $k$-clique communities can share vertices, so they can be overlapping. There may be vertices belonging to non-adjacent $k$-cliques, which could be reached by different paths and end up in different clusters. In order to find $k$-clique communities, one searches

[10] In graph theory the k-clique by Palla et al. is simply called clique, or complete graph, with $k$ vertices (Section A.1).

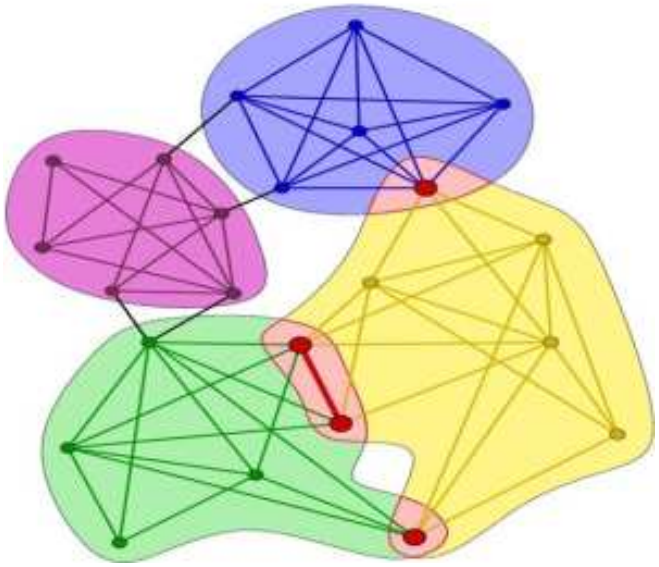

FIG. 22 Clique Percolation Method. The example shows communities spanned by adjacent 3-cliques (triangles). Overlapping vertices are shown by the bigger dots. Reprinted figure with permission from (Palla *et al.*, 2005). ©2005 by the Nature Publishing Group.

first for maximal cliques. Then a clique-clique overlap matrix **O** is built (Everett and Borgatti, 1998), which is an $n_c \times n_c$ matrix, $n_c$ being the number of cliques; $O_{ij}$ is the number of vertices shared by cliques $i$ and $j$. To find $k$-cliques, one needs simply to keep the entries of **O** which are larger than or equal to $k-1$, set the others to zero and find the connected components of the resulting matrix. Detecting maximal cliques is known to require a running time that grows exponentially with the size of the graph. However, the authors found that, for the real networks they analyzed, the procedure is quite fast, allowing to analyze graphs with up to $10^5$ vertices in a reasonably short time. The actual scalability of the algorithm depends on many factors, and cannot be expressed in closed form. An interesting aspect of $k$-clique communities is that they allow to make a clear distinction between random graphs and graphs with community structure. This is a rather delicate issue: we have seen in Section VI.C that Newman-Girvan modularity can attain large values on random graphs. Derényi et al. (Derényi *et al.*, 2005) have studied the percolation properties of $k$-cliques on random graphs, when the edge probability $p$ varies. They found that the threshold $p_c(k)$ for the emergence of a giant $k$-clique community, i.e. a community occupying a macroscopic portion of the graph, is $p_c(k) = [(k-1)n]^{-1/(k-1)}$, $n$ being the number of vertices of the graph, as usual. For $k = 2$, for which the $k$-cliques reduce to edges, one recovers the known expression for the emergence of a giant connected component in Erdös-Rényi graphs (Section A.3). This percolation transition is quite sharp: if the edge probability $p < p_c(k)$, $k$-clique communities are rather small; if $p > p_c(k)$ there is a giant component and many small communities. To assess the significance of the clusters found with the CPM, one can compare the detected cover[11] with the cover found on a null model graph, which is random but preserves the expected degree sequence of the original graph. The modularity of Newman and Girvan is based on the same null model (Section III.C.2). The null models of real graphs seem to display the same two scenarios found for Erdös-Rényi graphs, characterized by the presence of very small $k$-clique communities, with or without a giant cluster. Therefore, covers with $k$-clique communities of large or appreciable size can hardly be due to random fluctuations. Palla and coworkers (Adamcsek *et al.*, 2006) have designed a software package implementing the CPM, called *CFinder*, which is freely available (`www.cfinder.org`).

The algorithm has been extended to the analysis of weighted, directed and bipartite graphs. For weighted graphs, in principle one can follow the standard procedure of thresholding the weights, and apply the method on the resulting graphs, treating them as unweighted. Farkas et al. (Farkas *et al.*, 2007) proposed instead to threshold the weight of cliques, defined as the geometric mean of the weights of all edges of the clique. The value of the threshold is chosen slightly above the critical value at which a giant $k$-clique community emerges, in order to get the richest possible variety of clusters. On directed graphs, Palla et al. defined *directed k-cliques* as complete graphs with $k$ vertices, such that there is an ordering among the vertices, and each edge goes from a vertex with higher order to one with lower order. The ordering is determined from the *restricted outdegree* of the vertex, expressing the fraction of outgoing edges pointing to the other vertices of the clique versus the total outdegree. The method has been extended to bipartite graphs by Lehmann et al. (Lehmann *et al.*, 2008). In this case one uses bipartite cliques, or *bicliques*: a subgraph $K_{a,b}$ is a biclique if each of $a$ vertices of one class are connected with each of $b$ vertices of the other class. Two cliques $K_{a,b}$ are adjacent if they share a clique $K_{a-1,b-1}$, and a $K_{a,b}$ clique community is the union of all $K_{a,b}$ cliques that can be reached from each other through a path of adjacent $K_{a,b}$ cliques. Finding all $N_c$ bicliques of a graph is an **NP**-complete problem (Peeters, 2003), mostly because the number of bicliques tends to grow exponentially with the size of the graph. The algorithm designed by Lehmann et al. to find biclique communities is similar to the original CPM, and has a total complexity of $O(N_c^2)$. On sparse graphs, $N_c$ often grows linearly with the number of edges $m$, yielding a time complexity $O(m^2)$. Bicliques are also the main ingredients of *BiTector*, a recent algorithm to detect community structure in

[11] We remind that *cover* is the equivalent of partition for overlapping communities.

bipartite graphs (Du *et al.*, 2008).

Kumpula et al. have developed a fast implementation of the CPM, called Sequential Clique Percolation algorithm (SCP) (Kumpula *et al.*, 2008). It consists in detecting $k$-clique communities by sequentially inserting the edges of the graph at study, one by one, starting from an initial empty graph. Whenever a new edge is added, one checks whether new $k$-cliques are formed, by searching for $(k-2)$-cliques in the subset of neighboring vertices of the endpoints of the inserted edge. The procedure requires to build a graph $\Gamma^*$, in which the vertices are $(k-1)$-cliques and edges are set between vertices corresponding to $(k-1)$-cliques which are subgraphs of the same $k$-clique. At the end of the process, the connected components of $\Gamma^*$ correspond to the searched $k$-clique communities. The technique has a time complexity which is linear in the number of $k$-cliques of the graph, so it can vary a lot in practical applications. Nevertheless, it turns out to be much faster than the original implementation of the CPM. The big advantage of the SCP, however, consists of its implementation for weighted graphs. By inserting edges in decreasing order of weight, one recovers in a single run the community structure of the graph for all possible weight thresholds, by storing every cover detected after the addition of each edge. The standard CPM, instead, needs to be applied once for each threshold. If, instead of edge weight thresholding, one performs $k$-clique weight thresholding, as prescribed by Farkas et al. (Farkas *et al.*, 2007), the SCP remains much faster than the CPM, if one applies a simple modification to it, consisting in detecting and storing all $k$-cliques on the full graph, sorting them based on their weights, and finding the communities by sequentially adding the $k$-cliques in decreasing order of weight.

The CPM has the same limit as the algorithm of Radicchi et al. (Radicchi *et al.*, 2004) (Section V.B): it assumes that the graph has a large number of cliques, so it may fail to give meaningful covers for graphs with just a few cliques, like technological networks and some social networks. On the other hand, if there are many cliques, the method may deliver trivial community structure, like a cover consisting of the whole graph as a single cluster. Furthermore it is not clear *a priori* which value of $k$ one has to choose to identify meaningful structures. Finally, the criterion to choose the threshold for weighted graphs and the definition of directed $k$-cliques are rather arbitrary.

### B. Other techniques

One of the first methods to find overlapping communities was designed by Baumes et al. (Baumes *et al.*, 2005b). A community is defined as a subgraph which locally optimizes a given function $W$, typically some measure related to the edge density of the cluster[12]. Different overlapping subsets may all be locally optimal, so vertices can be shared between communities. Detecting the cluster structure of a graph amounts to finding the set of all locally optimal clusters. Two efficient heuristics are proposed, called Iterative Scan (`IS`) and Rank Removal (`RaRe`). `IS` performs a greedy optimization of the function $W$. One starts from a random seed vertex/edge and adds/deletes vertices one by one as long as $W$ increases. Then another seed is randomly picked and the procedure is repeated. The algorithm stops when, by picking any seed, one recovers a previously identified cluster. `RaRe` consists in removing important vertices such to disconnect the graphs in small components representing the cores of the clusters. The importance of vertices is determined by their centrality scores (e.g. degree, betweenness centrality (Freeman, 1977)), PageRank (Brin and Page, 1998)). Vertices are removed until one fragments the graph into components of a given size. After that, the removed vertices are added again to the graph, and are associated to those clusters for which doing so increases the value of the function $W$. The complexity of `IS` and `RaRe` is $O(n^2)$ on sparse graphs. The best performance is achieved by using `IS` to refine results obtained from `RaRe`. In a successive paper (Baumes *et al.*, 2005a), Baumes et al. further improved such two-step procedure, in that the removed vertices in `RaRe` are reinserted in decreasing order of their centrality scores, and the optimization of $W$ in `IS` is only extended to neighboring vertices of the running cluster. The new recipe maintains time complexity $O(n^2)$, but on sparse graphs it requires a time lower by an order of magnitude than the old one, while the quality of the detected clustering is comparable.

A different method, combining spectral mapping, fuzzy clustering and the optimization of a quality function, has been presented by Zhang et al. (Zhang *et al.*, 2007). The membership of vertex $i$ in cluster $k$ is expressed by $u_{ik}$, which is a number between 0 and 1. The sum of the $u_{ik}$ over all communities $k$ of a cover is 1, for any vertex. This normalization is suggested by the fact that the entry $u_{ik}$ can be thought of as the probability that $i$ belongs to community $k$, so the sum of the $u_{ik}$ represents the probability that the vertex belongs to any community of the cover, which is necessarily 1. If there were no overlaps, $u_{ik} = \delta_{k_i k}$, where $k_i$ represents the unique community of vertex $i$. The algorithm consists of three phases: 1) embedding vertices in Euclidean space; 2) grouping the corresponding vertex points in a given number $n_c$ of clusters; 3) maximizing a modularity function over the set of covers found in step 2), corresponding to different values of $n_c$. This scheme has been used in other techniques as well, like in the algorithm of Donetti and

[12] Community definitions based on local optimization are adopted in other algorithms as well, like that by Lancichinetti et al. (Lancichinetti *et al.*, 2009) (Section XII.A).

Muñoz (Donetti and Muñoz, 2004) (Section VII). The first step builds upon a spectral technique introduced by White and Smyth (White and Smyth, 2005), that we have discussed in Section VI.A.4. Graph vertices are embedded in a $d$-dimensional Euclidean space by using the top $d$ eigenvectors of the matrix $\mathbf{W}$, derived from the adjacency matrix $\mathbf{A}$ by dividing each element by the sum of the elements of the same row. The spatial coordinates of vertex $i$ are the $i$-th components of the eigenvectors. In the second step, the vertex points are associated to $n_c$ clusters by using fuzzy $k$-means clustering (Bezdek, 1981; Dunn, 1973) (Section IV.C). The number of clusters $n_c$ varies from 2 to a maximum $K$, so one obtains $K-1$ covers. The best cover is the one that yields the largest value of the modularity $Q_{ov}^{zh}$, defined as

$$Q_{ov}^{zh} = \sum_{c=1}^{n_c} \left[ \frac{\bar{W}_c}{W} - \left( \frac{\bar{S}_c}{2W} \right)^2 \right], \tag{69}$$

where

$$\bar{W}_c = \sum_{i,j \in V_c} \frac{u_{ic} + u_{jc}}{2} w_{ij}, \tag{70}$$

and

$$\bar{S}_c = \bar{W}_c + \sum_{i \in V_c, j \in V \setminus V_c} \frac{u_{ic} + (1 - u_{jc})}{2} w_{ij}. \tag{71}$$

The sets $V_c$ and $V$ include the vertices of module $c$ and of the whole network, respectively. Eq. 69 is an extension of Eq. 34, obtained by weighing the contribution of the edges' weights to the sums in $W_c$ and $S_c$ by the (average) membership coefficients of the vertices of the edge. We remark that Eq. 69 is the expression of modularity for the general case of a weighted graph. The determination of the eigenvectors is the most computationally expensive part of the method, so the time complexity is the same as that of the algorithm by White and Smyth (see Section VI.A.4), i.e. $O(K^2 n + Km)$, which is essentially linear in $n$ if the graph is sparse and $K \ll n$.

Nepusz et al. proposed a different approach based on vertex similarity (Nepusz *et al.*, 2008). One starts from the membership matrix $\mathbf{U}$, defined as in the previous method by Zhang et al. From $\mathbf{U}$ a matrix $\mathbf{S}$ is built, where $s_{ij} = \sum_{k=1}^{n_c} u_{ki} u_{kj}$, expressing the similarity between vertices ($n_c$ is the number of clusters). If one assumes to have information about the actual vertex similarity, corresponding to the matrix $\tilde{\mathbf{S}}$, the best cover is obtained by choosing $\mathbf{U}$ such that $\mathbf{S}$ approximates as closely as possible $\tilde{\mathbf{S}}$. This amounts to minimize the function

$$D_{\mathcal{G}}(\mathbf{U}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (\tilde{s}_{ij} - s_{ij})^2, \tag{72}$$

where the $w_{ij}$ weigh the importance of the approximation for each entry of the similarity matrices. In the absence of any information on the community structure of the graph, one sets $w_{ij} = 1, \forall i, j$ (equal weights) and $\tilde{\mathbf{S}}$ equal to the adjacency matrix $\mathbf{A}$, by implicitly assuming that vertices are similar if they are neighbors, dissimilar otherwise. On weighted graphs, one can set the $w_{ij}$ equal to the edge weights. Minimizing $D_{\mathcal{G}}(\mathbf{U})$ is a nonlinear constrained optimization problem, that can be solved with a gradient-based iterative optimization method, like simulated annealing. The optimization procedure adopted by Nepusz et al., for a fixed number of clusters $n_c$, has a time complexity $O(n^2 n_c h)$, where $h$ is the number of iterations leading to convergence, so the method can only be applied to fairly small graphs. If $n_c$ is unknown, as is often the case, the best cover is the one corresponding to the largest value of the modularity

$$Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) s_{ij}. \tag{73}$$

Eq. 73 is very similar to the expression of Newman-Girvan modularity (Eq. 12): the difference is that the Kronecker's $\delta$ is replaced by the vertices' similarity, to account for overlapping communities. Once the best cover is identified, one can use the entries of the partition matrix $\mathbf{U}$ to evaluate the participation of each vertex in the $n_c$ clusters of the cover. Nepusz et al. defined the *bridgeness* $b_i$ of a vertex $i$ as

$$b_i = 1 - \sqrt{\frac{n_c}{n_c - 1} \sum_{j=1}^{c} \left( u_{ji} - \frac{1}{n_c} \right)^2}. \tag{74}$$

If $i$ belongs to a single cluster, $b_i = 0$. If, for a vertex $i$, $u_{ik} = 1/n_c$, $\forall k$, $b_i = 1$ and $i$ is a perfect *bridge*, as it lies exactly between all clusters. However, a vertex with low $b_i$ may be simply an outlier, not belonging to any cluster. Since real bridges are usually rather central vertices, one can identify them by checking for large values of the *centrality-corrected bridgeness*, obtained by multiplying the bridgeness of Eq. 74 by the centrality of the vertex (expressed by, e.g., degree, betweenness (Freeman, 1977), etc.). A variant of the algorithm by Nepusz et al. can be downloaded from `http://www.cs.rhul.ac.uk/home/tamas/assets/files/fuzzyclust-static.tar.gz`.

In real networks it is often easier to discriminate between intercluster and intracluster edges than recognizing overlapping vertices. For instance, in social networks, even though many people may belong to more groups, their social ties within each group can be easily spotted. Besides, it may happen that communities are joined to each other through their overlapping vertices (Fig. 23), without intercluster edges. For these reasons, it has been recently suggested that defining clusters as sets of edges, rather than vertices, may be a promising strategy to analyze graphs with overlapping communities (Ahn *et al.*, 2009; Evans and Lambiotte, 2009). One has to focus on the *line graph* (Balakrishnan, 1997), i. e. the graph

whose vertices are the edges of the original graph; vertices of the line graph are linked if the corresponding edges in the original graph are adjacent, i. e. if they share one of their endvertices. Partitioning the line graph means grouping the edges of the starting graph[13]. Evans and Lambiotte (Evans and Lambiotte, 2009) introduced a set of quality functions, similar to Newman-Girvan modularity (Eq. 12), expressing the stability of partitions against random walks taking place on the graph, following the work of Delvenne et al. (Delvenne *et al.*, 2008) (Section VIII.B). They considered a projection of the traditional random walk on the line graph, along with two other diffusion processes, where walkers move between adjacent edges (rather than between neighboring vertices). Evans and Lambiotte optimized the three corresponding modularity functions to look for partitions in two real networks, Zachary's karate club (Zachary, 1977) (Section XIV.A) and the network of word associations derived from the University of South Florida Free Association Norms (Nelson *et al.*, 1998) (Section II). The optimization was carried out with the hierarchical technique by Blondel et al. (Blondel *et al.*, 2008) and the multi-level algorithm by Noack and Rotta (Noack and Rotta, 2008). While the results for the word association network are reasonable, the test on the karate club yields partitions in more than two clusters. However, the modularities used by Evans et Lambiotte can be modified to include longer random walks (just like in (Delvenne *et al.*, 2008)), and the length of the walk represents a resolution parameter that can be tuned to get better results. Ahn et al. (Ahn *et al.*, 2009) proposed to group edges with an agglomerative hierarchical clustering technique, called *hierarchical link clustering* (Section IV.B). They use a similarity measure for a pair of (adjacent) edges that expresses the size of the overlap between the neighborhoods of the non-coincident endvertices, divided by the total number of (different) neighbors of such endvertices. Groups of edges are merged pairwise in descending order of similarity, until all edges are together in the same cluster. The resulting dendrogram provides the most complete information on the community structure of the graph. However, as usual, most of this information is redundant and is an artefact of the procedure itself. So, Ahn et al. introduced a quality function to select the most meaningful partition(s), called *partition density*, which is essentially the average edge density within the clusters. The method is able to find meaningful clusters in biological networks, like protein-protein and metabolic networks, as well as in a social network of mobile phone communication. It can also be extended to multipartite and weighted graphs.

The idea of grouping edges is surely interesting. However it is not *a priori* better than grouping vertices.

[13] Ideally one wants to put together only the edges lying within clusters, and exclude the others. Therefore partitioning does not necessarily mean assigning each vertex of the line graph to a group, as standard clustering techniques would do.

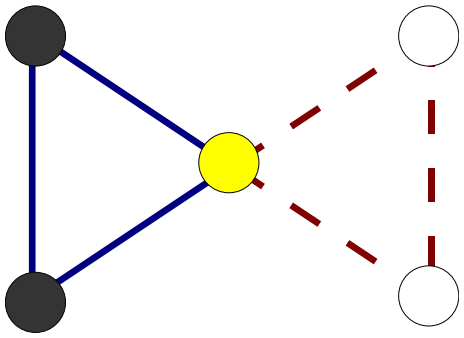

FIG. 23 Communities as sets of edges. In the figure, the graph has a natural division in two triangles, with the central vertex shared between them. If communities are identified by their internal edges, detecting the triangles and their overlapping vertex becomes easier than by using methods that group vertices. Reprinted figure with permission from (Evans and Lambiotte, 2009).

In fact, the two situations are somewhat symmetric. Edges connecting vertices of different clusters are "overlapping", but they will be assigned just to one cluster (or else the clusters would be merged).

The possibility of having overlapping communities makes most standard clustering methods inadequate, and enforces the design of new *ad hoc* techniques, like the ones we have described so far. On the other hand, if it were possible to identify the overlapping vertices and "separate" them among the clusters they belong to, the overlaps would be removed and one could then apply any of the traditional clustering methods to the resulting graph. This idea is at the basis of a recent method proposed by Gregory (Gregory, 2009). It is a three-stages procedure: first, one transforms the graph into a larger graph without overlapping vertices; second, a clustering technique is applied to the resulting graph; third, one maps the partition obtained into a cover by replacing the vertices with those of the original graph. The transformation step, called *Peacock*, is performed by identifying the vertices with highest *split betweenness* (Section V.A) and splitting them in multiple parts, connected by edges. This is done as long as the split betweenness of the vertices is sufficiently high, which is determined by a parameter $s$. In this way, most vertices of the resulting graph are exactly the same one had initially, the others are multiple copies of the overlapping vertices of the initial graph. The overlaps of the final cover are obtained by checking if copies of the same initial vertex end up in different disjoint clusters. The complexity is dominated by the Peacock algorithm, if one computes the exact values of the split betweenness for the vertices, which requires a time $O(n^3)$ on a sparse graph[14]. Gregory proposed an approximate local compu-

[14] The split betweenness needs to be recalculated after each vertex

tation, which scales as $O(n \log n)$: in this way the total complexity of the method becomes competitive, if one chooses a fast algorithm for the identification of the clusters. The goodness of the results depends on the specific method one uses to find the clusters after the graph transformation. The software of the version of the method used by Gregory in his applications can be found at `http://www.cs.bris.ac.uk/~steve/networks/peacockpaper/`. The idea of Gregory is interesting, as it allows to exploit traditional methods even in the presence of overlapping communities. The weakness is represented by the choice of the parameter $s$, which determines whether a vertex is overlapping or not. Choosing a range of "good" values for $s$ can be done *a posteriori*, by testing the methods on artificial graphs with built-in overlapping communities; however, there is no guarantee that there is a unique range, the choice may strongly depend on the specific graph one wants to study.

## XII. MULTIRESOLUTION METHODS AND CLUSTER HIERARCHY

The existence of a resolution limit for Newman-Girvan modularity (Section VI.C) implies that the straight optimization of quality functions yields a coarse description of the cluster structure of the graph, at a scale which has *a priori* nothing to do with the actual scale of the clusters. In the absence of information on the cluster sizes of the graph, a method should be able to explore all possible scales, to make sure that it will eventually identify the right communities. Multiresolution methods are based on this principle. However, many real graphs display *hierarchical* cluster structures, with clusters inside other clusters (Simon, 1962). In these cases, there are more levels of organization of vertices in clusters, and more relevant scales. In principle, clustering algorithms should be able to identify them. Multiresolution methods can do the trick, in principle, as they scan continuously the range of possible cluster scales. Recently other methods have been developed, where partitions are by construction hierarchically nested in each other. In this section we discuss both classes of techniques.

### A. Multiresolution methods

In general, multiresolution methods have a freely tunable parameter, that allows to set the characteristic size of the clusters to be detected. The general spin glass framework by Reichardt and Bornholdt ((Reichardt and Bornholdt, 2006a) and Section VI.B) is a typical example, where $\gamma$ is the resolution parameter. The extension of the method to weighted graphs has been recently discussed (Heimo *et al.*, 2008).

Pons has proposed a method (Pons, 2006) consisting of the optimization of multiscale quality functions, including the *multiscale modularity*

$$Q_\alpha^{\mathcal{M}} = \sum_{c=1}^{n_c} \left[ \alpha \frac{l_c}{m} - (1-\alpha) \left( \frac{d_c}{2m} \right)^2 \right], \qquad (75)$$

and two other additive quality functions, derived from the *performance* (Eq. 11) and a measure based on the similarity of vertex pairs. In Eq. 75 $0 \leq \alpha \leq 1$ is the resolution parameter and the notation is otherwise the same as in Eq. 13. We see that, for $\alpha = 1/2$, one recovers standard modularity. However, since multiplicative factors in $Q_\alpha^{\mathcal{M}}$ do not change the results of the optimization, we can divide $Q_\alpha^{\mathcal{M}}$ by $\alpha$, recovering the same quality function as in Eq. 44, with $\gamma = (1-\alpha)/\alpha$, up to an irrelevant multiplicative constant. To evaluate the relevance of the partitions, for any given multiscale quality function, Pons suggested that the length of the $\alpha$-range $[\alpha_{min}(\mathcal{C}), \alpha_{max}(\mathcal{C})]$, for which a community $\mathcal{C}$ "lives" in the maximum modularity partition, is a good indicator of the stability of the community. He then defined the *relevance function* of a community $\mathcal{C}$ at scale $\alpha$ as

$$\begin{aligned} R_\alpha(\mathcal{C}) &= \frac{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}{2} \\ &+ \frac{2(\alpha_{max}(\mathcal{C}) - \alpha)(\alpha - \alpha_{min}(\mathcal{C}))}{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}. \end{aligned} \qquad (76)$$

The relevance $R(\alpha)$ of a partition $\mathcal{P}$ at scale $\alpha$ is the average of the relevances of the clusters of the partitions, weighted by the cluster sizes. Peaks in $\alpha$ of $R(\alpha)$ reveal the most meanigful partitions.

Another interesting technique has been devised by Arenas et al. (Arenas *et al.*, 2008b), and consists of a modification of the original expression of modularity. The idea is to make vertices contribute as well to the computation of the edge density of the clusters, by adding a self-loop of strength $r$ to each vertex. Arenas et al. remarked that the parameter $r$ does not affect the structural properties of the graph in most cases, which are usually determined by an adjacency matrix without diagonal elements. With the introduction of the vertex strength $r$, modularity reads

$$Q_r = \sum_{c=1}^{n_c} \left[ \frac{2W_c + N_c r}{2W + nr} - \left( \frac{S_c + N_c r}{2W + nr} \right)^2 \right], \qquad (77)$$

for the general case of a weighted graph. The notation is the same as in Eq. 34, $N_c$ is the number of vertices in cluster $c$. We see that now the relative importance of the two terms in each summand depends on $r$, which can take any value in $]-2W/n, \infty[$. Arenas et al. made a sweep in the range of $r$, and determined for each $r$ the maximum modularity with extremal optimization (Sec-

split, just as one does for the edge betweenness in the Girvan-Newman algorithm (Girvan and Newman, 2002). Therefore both computations have the same complexity.

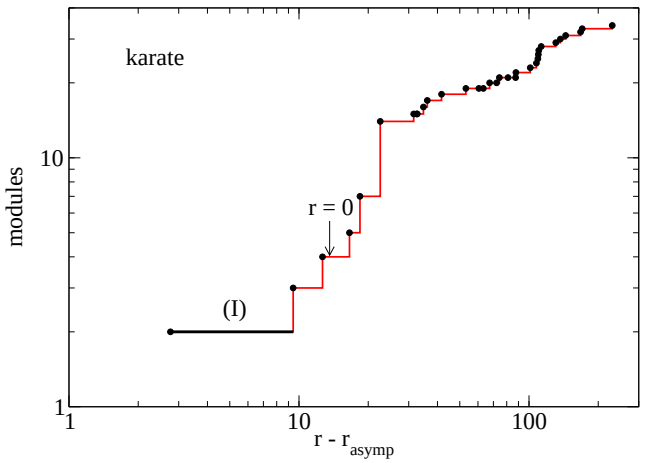


FIG. 24 Analysis of Zachary's karate club with the multiresolution method by Arenas et al. (Arenas *et al.*, 2008b). The plot shows the number of clusters obtained in correspondence of the resolution parameter $r$. The longest plateau (I) indicates the most stable partition, which exactly matches the social fission observed by Zachary. The partition obtained with straight modularity optimization ($r = 0$) consists of four clusters and is much less stable with respect to $I$, as suggested by the much shorter length of its plateau. Reprinted figure with permission from (Arenas *et al.*, 2008b). ©2008 by IOP Publishing.

tion VI.A.3) and *tabu search*[15] (Glover, 1986). Meaningful cluster structures correspond to plateaus in the plot of the number of clusters versus $r$ (Fig. 24). The length of a plateau gives a measure of the *stability* of the partition against the variation of $r$. The procedure is able to disclose the community structure of a number of real benchmark graphs. As expected, the most relevant partitions can be found in intervals of $r$ not including the value $r = 0$, which corresponds to the case of standard modularity (Fig. 24). A drawback of the method is that it is very slow, as one has to compute the modularity maximum for many values of $r$ in order to discriminate between relevant and irrelevant partitions. If the modularity maximum is computed with precise methods like simulated annealing and/or extremal optimization, as in (Arenas *et al.*, 2008b), only graphs with a few hundred vertices can be analyzed on a single processor. On the other hand the algorithm can be trivially parallelized by running the optimization for different values of $r$ on different processors. This is a common feature of all multiresolution methods discussed in this Section. In spite of the different formal expressions of modularity, the methods by Arenas et al. and Reichardt and Bornholdt are somewhat related to each other and yield similar results (Kumpula *et al.*, 2007a) on Zachary's karate club (Zachary, 1977) (Section XIV.A), synthetic graphs á la Ravasz-Barabási (Ravasz and Barabási, 2003) and on a model graph with the properties of real weighted social networks[16]. In fact, their modularities can be both recovered from the continuous-time version of the stability of clustering under random walk, introduced by Delvenne et al. (Delvenne *et al.*, 2008) (Section VIII.B).

Lancichinetti et al. have designed a multiresolution method which is capable of detecting both the hierarchical structure of graphs and overlapping communities (Lancichinetti *et al.*, 2009). It is based on the optimization of a fitness function, which estimates the strength of a cluster and entails a resolution parameter $\alpha$. The function could in principle be arbitrary, in their applications the authors chose a simple ansatz based on the tradeoff between the internal and the total degree of the cluster. The optimization procedure starts from a cluster with a single vertex, arbitrarily selected. Given a cluster core, one keeps adding and removing neighboring vertices of the cluster as long as its fitness increases. The fitness is recalculated after each addition/removal of a vertex. At some point one reaches a local maximum and the cluster is "closed". Then, another vertex is chosen at random, among those not yet assigned to a cluster, a new cluster is built, and so on, until all vertices have been assigned to clusters. During the buildup of a cluster, vertices already assigned to other clusters may be included, i.e. communities may overlap. The computational complexity of the algorithm, estimated on sparse Erdös-Rényi random graphs, is $O(n^\beta)$, with $\beta \sim 2$ for small values of the resolution parameter $\alpha$, and $\beta \sim 1$ if $\alpha$ is large. For a complete analysis, the worst-case computational complexity is $O(n^2 \log n)$, where the factor $\log n$ comes from the minimum number of different $\alpha$-values which are needed to resolve the actual community structure of the graph. Relevant partitions are revealed by pronounced spikes in the histogram of the fitness values of covers obtained for different $\alpha$-values, where the fitness of a cover is defined as the average fitness of its clusters.

A technique based on the Potts model, similar to that of Reichardt and Bornholdt (Reichardt and Bornholdt, 2006a), has been suggested by Ronhovde and Nussinov (Ronhovde and Nussinov, 2008a). The energy of

[15] Tabu search consists in moving single vertices from one community to another, chosen at random, or to new communities, starting from some initial partition. After a sweep over all vertices, the best move, i. e. the one producing the largest increase of modularity, is accepted and applied to the graph, yielding a new partition. The procedure is repeated until modularity does not increase further. To escape local optima, a list of recent accepted moves is kept and updated, so that those moves are not accepted in the next update of the configuration (tabu list). The cost of the procedure is about the same of other stochastic optimization techniques like, e. g. simulated annealing.

[16] Related does not mean equivalent, though. Arenas et al. have shown that their method is better than that by Reichardt and Bornholdt when the graph at hand includes communities of different sizes (Arenas *et al.*, 2008b).

their spin model is

$$\mathcal{H}(\{\sigma\}) = -\frac{1}{2}\sum_{i\neq j}[A_{ij} - \gamma(1 - A_{ij})]\delta(\sigma_i, \sigma_j). \qquad (78)$$

The big difference with Eq. 44 is the absence of a null model term. The model considers pairs of vertices in the same community: edges between vertices are energetically rewarded, whereas missing edges are penalized. The parameter $\gamma$ fixes the tradeoff between the two contributions. The energy is minimized by sequentially shifting single vertices/spins to the communities which yield the largest decrease of the system's energy, until convergence. If, for each vertex, one just examines the communities of its neighbors, the energy is minimized in a time $O(m^\beta)$, where $\beta$ turns out to be slightly above 1 in most applications, enabling the analysis of large graphs. This essentially eliminates the problem of limited resolution, as the criterion to decide about the merger or the split of clusters only depends on local parameters. Still, for the detection of possible hierarchical levels tuning $\gamma$ is mandatory. In a successive paper (Ronhovde and Nussinov, 2008b), the authors have introduced a new stability criterion for the partitions, consisting of the computation of the similarity of partitions obtained for the same $\gamma$ and different initial conditions. The idea is that, if a partition is robust in a given range of $\gamma$-values, most replicas delivered by the algorithm will be very similar. On the other hand, if one explores a region of resolutions in between two strong partitions, the algorithm will deliver the one or the other partition and the individual replicas will be, on average, not so similar to each other. So, by plotting the similarity as a function of the resolution parameter $\gamma$, stable communities are revealed by peaks. Ronhovde and Nussinov adopted similarity measures borrowed from information theory (Section XIV.B). Their criterion of stability can be adopted to determine the relevance of partitions obtained with any multiresolution algorithm.

A general problem of multiresolution methods is how to assess the stability of partitions for large graphs. The rapidly increasing number of partitions, obtained by minimal shifts of vertices between clusters, introduces a large amount of noise, that blurs signatures of stable partitions like plateaus, spikes, etc. that one can observe in small systems. In this respect, it seems far more reliable focusing on correlations between partitions (like the average similarity used by Ronhovde and Nussinov (Ronhovde and Nussinov, 2008a,b)) than on properties of the individual partitions (like the measures of occurrence used by Arenas et al. (Arenas *et al.*, 2008b) and by Lancichinetti et al. (Lancichinetti *et al.*, 2009)).

### B. Hierarchical methods

The natural procedure to detect the hierarchical structure of a graph is hierarchical clustering, that we have discussed in Section IV.B. There we have emphasized the main weakness of the procedure, which consists of the necessity to introduce a criterion to identify relevant partitions (hierarchical levels) out of the full dendrogram produced by the given algorithm. Furthermore, there is no guarantee that the results indeed reflect the actual hierarchical structure of the graph, and that they are not mere artefacts of the algorithm itself. Scholars have just started to deal with these problems.

Sales-Pardo et al. have proposed a top-down approach (Sales-Pardo *et al.*, 2007). Their method consists of two steps: 1) measuring the similarity between vertices; 2) deriving the hierarchical structure of the graph from the similarity matrix. The similarity measure, named *node affinity*, is based on Newman-Girvan modularity. Basically the affinity between two vertices is the frequency with which they coexist in the same community in partitions corresponding to local optima of modularity. The latter are configurations for which modularity is stable, i.e. it cannot increase if one shifts one vertex from one cluster to another or by merging or splitting clusters. The set of these partitions is called $\mathcal{P}_{max}$. Before proceeding with the next step, one verifies whether the graph has a significant community structure or not. This is done by calculating the $z$-score (Eq. 49) for the average modularity of the partitions in $\mathcal{P}_{max}$ with respect to the average modularity of partitions with local modularity optima of the equivalent ensemble of null model graphs, obtained as usual by randomly rewiring the edges of the original graph under the condition that the expected degree sequence is the same as the degree sequence of the graph. Large $z$-scores indicate meaningful cluster structure: Sales-Pardo et al. used a threshold corresponding to the 1% significance level[17]. If the graph has a relevant cluster structure, one proceeds with the second step, which consists in putting the affinity matrix in block-diagonal form, by minimizing a cost function expressing the average distance of connected vertices from the diagonal. The blocks correspond to the communities and the recovered partition represents the uppermost organization level. To determine lower levels, one iterates the procedure for each subgraph identified at the previous level, which is treated as an independent graph. The procedure stops when all blocks found do not have a relevant cluster structure, i.e. their $z$-scores are lower than the threshold. The partitions delivered by the method are hierarchical by construction, as communities at each level are nested within communities at higher levels. However, the method may find no relevant partition (no community structure), a single partition (community structure but no hierarchy) or more (hierarchy) and in this respect it is better than most existing methods. The algorithm is not fast, as both the search of local optima for modularity and the rearrangement of the similarity matrix are

[17] We remind that the significance of the $z$-score has to be computed with respect to the actual distribution of the maximum modularity for the null model graphs, as the latter is not Gaussian (Section VI.C).

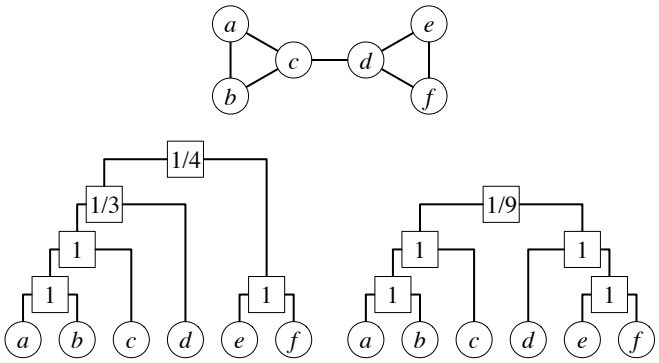


FIG. 25 Hierarchical random graphs by Clauset et al. (Clauset *et al.*, 2008). The picture shows two possible dendrograms for the simple graph on the top. The linking probabilities on the internal nodes of the dendrograms yield the best fit of the model graphs to the graph at study. Reprinted figure with permission from (Clauset *et al.*, 2008). ©2008 by the Nature Publishing Group.

performed with simulated annealing[18], but delivers good results for computer generated networks, and meaningful partitions for some real networks, like the world airport network (Barrat *et al.*, 2004), an email exchange network of a Catalan university (Guimerà *et al.*, 2003), a network of electronic circuits (Itzkovitz *et al.*, 2005) and metabolic networks of *E. coli* (Guimerà *et al.*, 2007).

Clauset et al. (Clauset *et al.*, 2007; Clauset *et al.*, 2008) described the hierarchical organization of a graph by introducing a class of *hierarchical random graphs*. A hierarchical random graph is defined by a dendrogram $\mathcal{D}$, which is the natural representation of the hierarchy, and by a set of probabilities $\{p_r\}$ associated to the $n-1$ internal nodes of the dendrogram. An *ancestor* of a vertex $i$ is any internal node of the dendrogram that is encountered by starting from the "leaf" vertex $i$ and going all the way up to the top of the dendrogram. The probability that vertices $i$ and $j$ are linked to each other is given by the probability $p_r$ of the lowest common ancestor of $i$ and $j$. Clauset et al. searched for the model $(\mathcal{D}, \{p_r\})$ that best fits the observed graph topology, by using Bayesian inference (Section IX.A). The probability that the model fits the graph is proportional to the likelihood

$$\mathcal{L}(\mathcal{D}, \{p_r\}) = \prod_{r \in \mathcal{D}} p_r^{E_r} (1 - p_r)^{L_r R_r - E_r}. \qquad (79)$$

Here, $E_r$ is the number of edges connecting vertices whose lowest common ancestor is $r$, $L_r$ and $R_r$ are the numbers of graph vertices in the left and right subtrees descending from the dendrogram node $r$, and the product runs over all internal dendrogram nodes. For a given dendrogram $\mathcal{D}$, the maximum likelihood $\mathcal{L}(\mathcal{D})$ corresponds to the set of probabilities $\{\bar{p}_r\}$, where $\bar{p}_r$ equals the actual density of edges $E_r/(L_r R_r)$ between the two subtrees of $r$ (Fig. 25). One can define the statistical ensemble of hierarchical random graphs describing a given graph $\mathcal{G}$, by assigning to each model graph $(\mathcal{D}, \{\bar{p}_r\})$ a probability proportional to the maximum likelihood $\mathcal{L}(\mathcal{D})$. The ensemble can be sampled by a Markov chain Monte Carlo method (Newman and Barkema, 1999). The procedure suggested by Clauset et al. seems to converge to equilibrium roughly in a time $O(n^2)$, although the actual complexity may be much higher. Still, the authors were able to investigate graphs with a few thousand vertices. From sufficiently large sets of model configurations sampled at equilibrium, one can compute average properties of the model, e. g. degree distributions, clustering coefficients. etc., and compare them with the corresponding properties of the original graph. Tests on real graphs reveal that the model is indeed capable to describe closely the graph properties. Furthermore, the model enables one to predict missing connections between vertices of the original graph. This is a very important problem (Liben-Nowell and Kleinberg, 2003): edges of real graphs are the result of observations/experiments, that may fail to discover some relationships between the units of the system. From the ensemble of the hierarchical random graphs one can derive the average linking probability between all pairs of graph vertices. By ranking the probabilities corresponding to vertex pairs which are disconnected in the original graph, one may expect that the pairs with highest probabilities are likely to be connected in the system, even if such connections are not observed. Clauset et al. pointed out that their method does not deliver a sharp hierarchical organization for a given graph, but a class of possible organizations, with well-defined probabilities. It is certainly reasonable to assume that many structures are compatible with a given graph topology. In the case of community structure, it is not clear which information one can extract from averaging over the ensemble of hierarchical random graphs. Moreover, since the hierarchical structure is represented by a dendrogram, it is impossible to rank partitions according to their relevance. In fact, the work by Clauset et al. questions the concept of "relevant partition", and opens a debate in the scientific community about the meaning itself of graph clustering. The software of the method can be found at `http://www.santafe.edu/~aaronc/hierarchy/`.

## XIII. SIGNIFICANCE OF CLUSTERING

Given a network, many partitions could represent meaningful clusterings in some sense, and it could be difficult for some methods to discriminate between them. Quality functions evaluate the goodness of a partition (Section III.C.2), so one could say that high quality corresponds to meaningful partitions. But this is not nec-

[18] The reordering of the matrix is by far the most time-consuming part of the method. The situation improves if one adopts faster optimization strategies than simulated annealing, at the cost of less accurate results.

essarily true. In Section VI.C we have seen that high values of the modularity of Newman and Girvan do not necessarily indicate that a graph has a definite cluster structure. In particular we have seen that partitions of random graphs may also achieve considerably large values of $Q$, although we do not expect them to have community structure, due to the lack of correlations between the linking probabilities of the vertices. The optimization of quality functions, like modularity, delivers the best partition according to the criterion underlying the quality function. But is the optimal clustering also *significant*, i.e. a relevant feature of the graph, or is it just a byproduct of randomness and basic structural properties, like, e. g. the degree sequence? Very little effort has been devoted to this crucial issue, that we discuss here.

In some works the concept of significance has been related to that of *robustness* or *stability* of a partition against random perturbations of the graph structure. The basic idea is that, if a partition is significant, it will be recovered even if the structure of the graph is modified, as long as the modification is not too extensive. Instead, if a partition is not significant, one expects that minimal modifications of the graph will suffice to disrupt the partition, so other clusterings are recovered. A nice feature of this approach is the fact that it can be applied for any clustering technique. Gfeller et al. (Gfeller *et al.*, 2005) considered the general case of weighted graphs. A graph is modified, in that its edge weights are increased or decreased by a relative amount $0 < \sigma < 1$. This choice also allows to account for the possible effects of uncertainties in the values of the edge weights, which result from measurements/experiments carried out on a given system. After fixing $\sigma$ (usually to 0.5), multiple realizations of the original graph are generated. The best partition for each realization is identified and, for each pair of adjacent vertices $i$ and $j$, the *in-cluster* probability $p_{ij}$ is computed, i.e. the fraction of realizations in which $i$ and $j$ were classified in the same cluster. Edges with in-cluster probability smaller than a threshold $\theta$ (usually 0.8) are called *external edges*. The stability of a partition is estimated through the *clustering entropy*

$$S = -\frac{1}{m} \sum_{(i,j):A_{ij}=1} [p_{ij} \log_2 p_{ij} - (1-p_{ij}) \log_2 (1-p_{ij})], \tag{80}$$

where $m$ is, as usual, the number of graph edges, and the sum runs over all edges. The most stable partition has $p_{ij} = 0$ along inter-cluster edges and $p_{ij} = 1$ along intra-cluster edges, which yields $S = 0$; the most unstable partition has $p_{ij} = 1/2$ on all edges, yielding $S = 1$. The absolute value of $S$ is not meaningful, though, and needs to be compared with the corresponding value for a null model graph, similar to the original graph, but with supposedly no cluster structure. Gfeller et al. adopted the same null model of Newman-Girvan modularity, i.e. the class of graphs with expected degree sequence coinciding with that of the original graph. Since the null model is defined on unweighted graphs, the significance of $S$ can be assessed only in this case, although it would not be hard to think of a generalization to weighted graphs. The approach enables one as well to identify *unstable vertices*, i.e. vertices lying at the boundary between clusters. In order to do that, the external edges are removed and the connected components of the resulting disconnected graph are associated with the clusters detected in the original graph, based on their relative overlap (computed through Eq. 93). Unstable vertices end up in components that are not associated to any of the initial clusters. A weakness of the method by Gfeller et al. is represented by the two parameters $\sigma$ and $\theta$, whose values are in principle arbitrary.

More recently, Karrer et al. (Karrer *et al.*, 2008) adopted a similar strategy to unweighted graphs. Here one performs a sweep over all edges: the perturbation consists in removing each edge with a probability $\alpha$ and replacing it with another edge between a pair of vertices $(i, j)$, chosen at random with probability $p_{ij} = k_i k_j / 2m$, where $k_i$ and $k_j$ are the degrees of $i$ and $j$. We recognize the probability of the null model of Girvan-Newman's modularity. Indeed, by varying the probability $\alpha$ from 0 to 1 one smoothly interpolates between the original graph (no perturbation) and the null model (maximal perturbation). The degree sequence of the graph remains invariant (on average) along the whole process, by construction. The idea is that the perturbation affects solely the organization of the vertices, keeping the basic structural properties. For a given value of $\alpha$, many realizations of the perturbed graph are generated, their cluster structures are identified with some method (Karrer et al. used modularity optimization) and compared with the partition obtained from the original unperturbed graph. The partitions are compared by computing the variation of information $V$ (Section XIV.B). From the plot of the average $\langle V \rangle$ versus $\alpha$ one can assess the stability of the cluster structure of the graph. If $\langle V(\alpha) \rangle$ changes rapidly for small values of $\alpha$ the partition is likely to be unstable. As in the approach by Gfeller et al. the behaviour of the function $\langle V(\alpha) \rangle$ does not have an absolute meaning, but needs to be compared with the corresponding curve obtained for a null model. For consistency, the natural choice is again the null model of modularity, which is already used in the process of graph perturbation. The approaches by Gfeller et al. and Karrer et al., with suitable modifications, can also be used to check for the stability of the cluster structure in parts of a graph, up to the level of individual communities. This is potentially important as it may happen that some parts of the graph display a strong community structure and other parts weak or no community structure at all.

Rosvall and Bergstrom (Rosvall and Bergstrom, 2008) defined the significance of clusters with the bootstrap method (Efron and Tibshirani, 1993), which is a standard procedure to check for the accuracy of a measurement/estimate based on resampling from the empirical data. The graph at study is supposed to be generated by a parametric model, which is used to create many sam-

ples. This is done by assigning to each edge a weight taken by a Poisson distribution with mean equal to the original edge weight. For the initial graph and each sample one identifies the community structure with some method, that can be arbitrary. For each cluster of the partition of the original graph one determines the largest subset of vertices that are classified in the same cluster in at least 95% of all bootstrap samples. Identifying such cluster cores enables one to track the evolution in time of the community structure, as we will explain in Section XV.B.

A different approach has been proposed by Massen and Doye (Massen and Doye, 2006). They analyzed an equilibrium canonical ensemble of partitions, with $-Q$ playing the role of the energy, $Q$ being Newman-Girvan modularity. This means that the probability of occurrence of a partition at temperature $T$ is proportional to $\exp(Q/T)$. The idea is that, if a graph has a significant cluster structure, at low temperatures one would recover essentially the same partition, corresponding to the modularity maximum, which is separated by an appreciable gap from the modularity values of the other partitions. On the contrary, graphs with no community structure, e. g. random graphs, have many competing (local) maxima, and the corresponding configurations will emerge already at low temperatures, since their modularity values are close to the absolute maximum. These distinct behaviors can manifest themselves in various ways. For instance, if one considers the variation of the specific heat $C = -dQ/dT$ with $T$, the gap in the modularity landscape is associated to a sharp peak of $C$ around some temperature value, like it happens in a phase transition. If the gap is small and there are many partitions with similar modularity values, the peak of $C$ becomes broad. Another strategy to assess the significance of the maximum modularity partition consists of the investigation of the similarity between partitions recovered at a given temperature $T$. This similarity can be expressed by the *frequency matrix*, whose element $f_{ij}$ indicates the relative number of times vertices $i$ and $j$ have been classified in the same cluster. If the graph has a clear community structure, at low temperatures the frequency matrix can be put in block-diagonal form, with the blocks corresponding to the communities of the best partition; if there is no significant community structure, the frequency matrix is rather homogeneous. The Fiedler eigenvalue $\lambda_2$, the second smallest eigenvalue of the Laplacian matrix associated to the frequency matrix, allows to estimate how "block-diagonal" the matrix is (see Section IV.A). At low temperatures $\lambda_2 \sim 0$ if there is one (a few) partitions with maximum or near to maximum modularity; if there are many (almost) degenerate partitions, $\lambda_2$ is appreciably different from zero even when $T \to 0$. A sharp transition from low to high values of $\lambda_2$ by varying temperature indicates significant community structure. Another clear signature of significant community structure is the observation of a rapid drop of the average community size with $T$, as "strong" communities break up in many small pieces for a modest temperature increase, while the disintegration of "weak" communities takes place more slowly. In scale-free graphs (Section A.3) clusters are often not well separated, due to the presence of the hubs; in these cases the above-mentioned transitions of ensemble variables are not so sharp and take place over a broader temperature range. The canonical ensemble of partitions is generated through single spin heatbath simulated annealing (Reichardt and Bornholdt, 2006a), combined with parallel tempering (Earl and Deem, 2005). The approach by Massen and Doye is useful to recognize graphs without cluster structure, if the modularity landscape is characterized by many maxima with close values. However, it can happen that gaps between the absolute modularity maximum and the rest of the modularity values are created by fluctuations, and the method is unable to identify these situations. Furthermore, the approach heavily relies on modularity and on a costly technique like simulated annealing: extensions to other quality functions and/or optimization procedures do not appear straightforward.

In a recent work by Bianconi et al. (Bianconi *et al.*, 2008b) the notion of entropy of graph ensembles (Bianconi, 2008; Bianconi *et al.*, 2008a) is employed to find out how likely it is for a cluster structure to occur on a graph with a given degree sequence. The entropy is computed from the number of graph configurations which are compatible with a given classification of the vertices in $q$ groups. The clustering is quantitatively described by fixing the number of edges $A(q_1, q_2)$ running between clusters $q_1$ and $q_2$, for all choices of $q_1 \neq q_2$. Bianconi et al. proposed the following indicator of clustering significance

$$\Theta_{\vec{k},\vec{q}} = \frac{\Sigma_{\vec{k},\vec{q}} - \langle \Sigma_{\vec{k},\pi(\vec{q})} \rangle_\pi}{\sqrt{\langle \delta\Sigma^2_{\vec{k},\pi(\vec{q})} \rangle_\pi}}, \tag{81}$$

where $\Sigma_{\vec{k},\vec{q}}$ is the entropy of the graph configurations with given degree sequence $\vec{k}$ and clustering $\vec{q}$ (with fixed numbers of inter-cluster edges $A(q_1, q_2)$), and $(\Sigma_{\vec{k},\pi(\vec{q})})_\pi$ is the average entropy of the configurations with the same degree sequence and a random permutation $\pi(\vec{q})$ of the cluster labels. The absolute value of the entropy $\Sigma_{\vec{k},\vec{q}}$ is not meaningful, so the comparison of $\Sigma_{\vec{k},\vec{q}}$ and $\langle \Sigma_{\vec{k},\pi(\vec{q})} \rangle_\pi$ is crucial, as it tells how relevant the actual cluster structure is with respect to a random classification of the vertices. However, different permutations of the assignments $\vec{q}$ yield different values of the entropy, which can fluctuate considerably. Therefore one has to compute the standard deviation $\langle \delta\Sigma^2_{\vec{k},\pi(\vec{q})} \rangle_\pi$ of the entropy corresponding to all random permutations $\pi(\vec{q})$, to estimate how significant the difference between $\Sigma_{\vec{k},\vec{q}}$ and $(\Sigma_{\vec{k},\pi(\vec{q})})_\pi$ is. In this way, if $\Theta_{\vec{k},\vec{q}} \leq 1$, the entropy of the given cluster structure is of the same order as the entropy of some random permutation of the cluster labels, so it is not relevant. Instead, if $\Theta_{\vec{k},\vec{q}} \gg 1$, the cluster structure is far more likely than a random classification of the vertices, so the

clustering is relevant. The indicator $\Theta_{\vec{k},\vec{q}}$ can be simply generalized to the case of directed and weighted graphs.

We conclude with a general issue which is related to the significance of community structure. The question is: given a cluster structure in a graph, can it be recovered *a priori* by an algorithm? In a recent paper (Reichardt and Leone, 2008), Reichardt and Leone studied under which conditions a special built-in cluster structure can be recovered. The clusters have equal size and a pair of vertices is connected with probability $p$ if they belong to the same cluster, with probability $r < p$ otherwise. In computer science this is known as the *planted partitioning problem* (Condon and Karp, 2001). The goal is to propose algorithms that recover the planted partition for any choice of $p$ and $r$. For dense graphs, i.e. graphs whose average degree grows with the number $n$ of vertices, algorithms can be designed that find the solution with a probability which equals 1 minus a term that vanishes in the limit of infinite graph size, regardless of the difference $p - r$, which can then be chosen arbitrarily small. Since many real networks are not dense graphs, as their average degree $\langle k \rangle$ is usually much smaller than $n$ and does not depend on it, Reichardt and Leone investigated the problem in the case of fixed $\langle k \rangle$ and infinite graph size. We indicate with $q$ the number of clusters and with $p_{in}$ the probability that a randomly selected edge of the graph lies within any of the $q$ clusters. In this way, if $p_{in} = 1/q$, the inter-cluster edge density matches the intra-cluster edge density (i.e. $p = r$), and the planted partition would not correspond to a recoverable clustering, whereas for $p_{in} = 1$, there are no inter-cluster edges and the partition can be trivially recovered. The value of $p_{in}$ is in principle unknown, so one has to detect the cluster structure ignoring this information. Reichardt and Leone proposed to look for a minimum cut partition, i.e. for the partition that minimizes the number of inter-cluster edges, as it is usually done in the graph partitioning problem (discussed in Section IV.A). Clearly, for $p_{in} = 1$ the minimum cut partition trivially coincides with the planted partition, whereas for $1/q < p_{in} < 1$ there should be some overlap, which is expected to vanish in the limit case $p_{in} = 1/q$. The minimum cut partition corresponds to the minimum of the following ferromagnetic Potts model Hamiltonian

$$\mathcal{H}_{part} = -\sum_{i<j} J_{ij}\delta_{\sigma_i,\sigma_j}, \tag{82}$$

over the set of all spin configurations with zero magnetization. Here the spin $\sigma_i$ indicates the cluster vertex $i$ belongs to, and the coupling matrix $J_{ij}$ is just the adjacency matrix of the graph. The constraint of zero magnetization ensures that the clusters have all the same size, as required by the planted partitioning problem. The energy of a spin configuration, expressed by Eq. 82, is the negative of the number of edges that lie within clusters: the minimum energy corresponds to the maximum number of intra-cluster edges, which is coupled to the minimum number of inter-cluster edges. The minimum energy can be computed with the cavity method, or belief propagation, at zero temperature (Mézard and Parisi, 2003). The accuracy of the solution with respect to the planted solution is expressed by the fraction of vertices which are put in the same class in both partitions. The analysis yields a striking result: the planted clustering is accurately recovered for $p_{in}$ larger than a critical threshold $p_{in}^c > 1/q$. So, there is a range of values of $p_{in}$, $1/q < p_{in} < p_{in}^c$, in which the clustering is not recoverable, as the minimum cut partition is uncorrelated with it. The threshold $p_{in}^c$ depends on the degree distribution $p(k)$ of the graph.

## XIV. TESTING ALGORITHMS

When a clustering algorithm is designed, it is necessary to test its performance, and compare it with that of other methods. In the previous sections we have said very little about the performance of the algorithms, other than their computational complexity. Indeed, the issue of testing algorithms has received very little attention in the literature on graph clustering. This is a serious limit of the field. Because of that, it is still impossible to state which method (or subset of methods) is the most reliable in applications, and people rely blindly on some algorithms instead of others for reasons that have nothing to do with the actual performance of the algorithms, like. e.g. popularity (of the method or of its inventor). This lack of control is also the main reason for the proliferation of graph clustering techniques in the last few years. Virtually in any paper, where a new method is introduced, the part about testing consists in applying the method to a small set of simple benchmark graphs, whose cluster structure is fairly easy to recover. Because of that, the freedom in the design of a clustering algorithm is basically infinite, whereas it is not clear what a new procedure is adding to the field, if anything.

In this section we discuss at length the issue of testing. First, we describe the fundamental ingredients of any testing procedure, i.e. benchmark graphs with built-in community structure, that methods have to identify (Section XIV.A). We proceed by reviewing measures to compare graph partitions with each other (Section XIV.B). In Section XIV.C we present the comparative evaluations of different methods that have been performed in the literature. We conclude by addressing the important issue of the robustness of community structure, i. e. of its stability against perturbations, which is closely related to the problem of defining when partitions are "significant".

### A. Benchmarks

Testing an algorithm essentially means applying it to a specific problem whose solution is known and compare such solution with that delivered by the algorithm. In the case of graph clustering, a problem with a well-

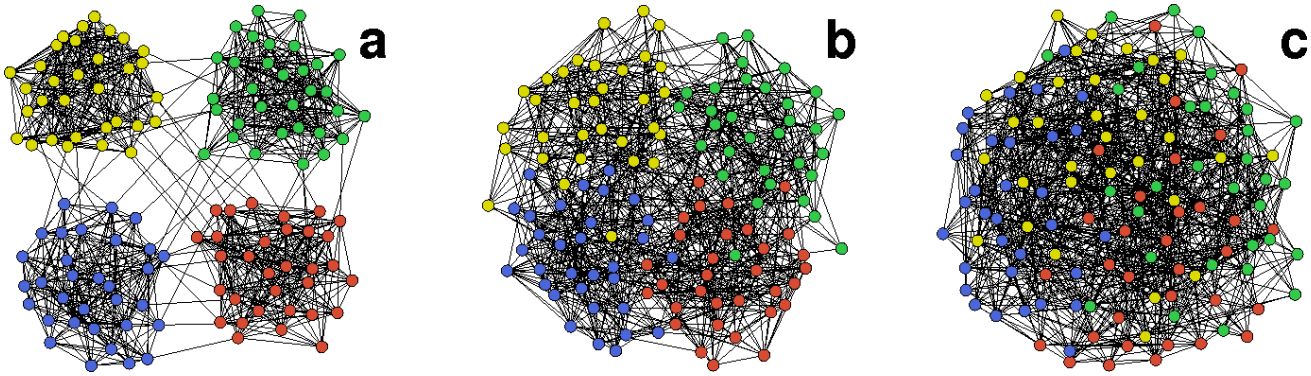


FIG. 26 Benchmark of Girvan and Newman. The three pictures correspond to $z_{in} = 15$ (a), $z_{in} = 11$ (b) and $z_{in} = 8$ (c). In (c) the four groups are hardly visible. Reprinted figure with permission from (Guimerà and Amaral, 2005). ©2005 by the Nature Publishing Group.

defined solution is a graph with a clear community structure. This concept is not trivial, however. Many clustering algorithms are based on similar intuitive notions of what a community is, but different implementations. So it is crucial that the scientific community agrees on a set of reliable benchmark graphs. This mostly applies to computer-generated graphs, where the built-in cluster structure can be arbitrarily designed. In the literature real networks are used as well, in those cases in which communities are well defined because of information about the system.

We start our survey from computer-generated benchmarks. A special class of graphs has become quite popular in the last years. They are generated with the so-called *planted $\ell$-partition model* (Condon and Karp, 2001). The model partitions a graph with $n = g \cdot \ell$ vertices in $\ell$ groups with $g$ vertices each. Vertices of the same group are linked with a probability $p_{in}$, whereas vertices of different groups are linked with a probability $p_{out}$. Each subgraph corresponding to a group is then a random graph á la Erdös-Rényi with connection probability $p = p_{in}$ (Section A.3). The average degree of a vertex is $\langle k \rangle = p_{in}(g-1) + p_{out}g(\ell-1)$. If $p_{in} > p_{out}$ the intra-cluster edge density exceeds the inter-cluster edge density and the graph has a community structure. This idea is quite intuitive and we have encountered it in several occasions in the previous sections. Girvan and Newman considered a special case of the planted $\ell$-partition model (Girvan and Newman, 2002). They set $\ell = 4$, $g = 32$ (so the number of graph vertices is $n = 128$) and fixed the average total degree $\langle k \rangle$ to 16. This implies that $p_{in} + 3p_{out} \approx 1$, so the probabilities $p_{in}$ and $p_{out}$ are not independent parameters. In calculation it is common to use as parameters $z_{in} = p_{in}(g-1) = 15p_{in}$ and $z_{out} = p_{out}g(\ell-1) = 48p_{out}$, indicating the expected internal and external degree of a vertex, respectively. These particular graphs have by now gained the status of standard benchmarks (Girvan and Newman, 2002) (Fig. 26). In the first applications of the graphs one assumed that communities are well defined when $z_{out} < 8$, corresponding to the situation in which the internal degree exceeds the external degree. However, the threshold $z_{out} = z_{in} = 8$ implies $p_{in} \approx 1/2$ and $p_{out} = 1/6$, so it is not the actual threshold of the model, where $p_{in} = p_{out} = 1/4$, corresponding to $z_{out} \approx 12$. So, one expects to be able to detect the planted partition up until $z_{out} \approx 12$.

Testing a method against the Girvan-Newman benchmark consists in calculating the similarity between the partitions determined by the method and the natural partition of the graph in the four equal-sized groups. Several measures of partitions' similarity may be adopted; we describe them in Section XIV.B. One usually builds many graph realizations for a particular value of $z_{out}$ and computes the average similarity between the solutions of the method and the built-in solution. The procedure is then iterated for different values of $z_{out}$. The results are usually represented in a plot, where the average similarity is drawn as a function of $z_{out}$. Most algorithms usually do a good job for small $z_{out}$ and start to fail when $z_{out}$ approaches 8. Fan et al. (Fan *et al.*, 2007) have designed a weighted version of the benchmark of Girvan and Newman, in that one gives different weights to edges inside and between communities. One could pick just two values, one for intra- and the other for inter-community edges, or uniformly distributed values in two different ranges. For this benchmark there are then two parameters that can be varied, $z_{out}$ and the relative importance of the internal and the external weights. Typically one fixes the topological structure and varies the weights. This is particularly insightful when $z_{out} = 4$, which delivers graphs without topological cluster structure: in this case, the question whether there are clusters or not depends entirely on the weights.

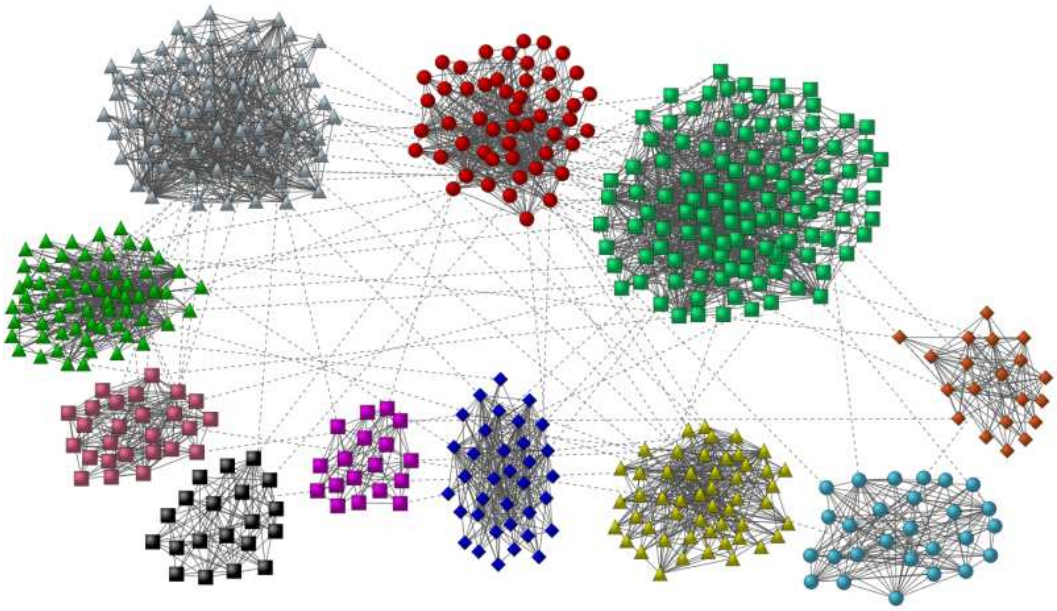

FIG. 27 A realization of the benchmark graphs by Lancichinetti et al. (Lancichinetti *et al.*, 2008), with 500 vertices. The distributions of the vertex degree and of the community size are both power laws. Such benchmark is a more faithful approximation of real-world networks with community structure than simpler benchmarks like, e. g. that by Girvan and Newman (Girvan and Newman, 2002). Reprinted figure with permission from (Lancichinetti *et al.*, 2008). ©2008 by the American Physical Society.

As we have remarked above, the planted $\ell$-partition model generates mutually interconnected random graphs á la Erdös-Rényi. Therefore, all vertices have approximately the same degree. Moreover, all communities have exactly the same size by construction. These two features are at odds with what is observed in graph representations of real systems. Degree distributions are usually skewed, with many vertices with low degree coexisting with a few vertices with high degree. A similar heterogeneity is also observed in the distribution of cluster sizes, as we shall see in Section XV. So, the planted $\ell$-partition model is not a good description of a real graph with community structure. However, the model can be modified to account for the heterogeneity of degrees and community sizes. A modified version of the model, called *Gaussian random partition generator*, was designed by Brandes et al. (Brandes *et al.*, 2003). Here the cluster sizes have a Gaussian distribution, so they are not the same, although they do not differ much from each other. The heterogeneity of the cluster sizes introduces a heterogeneity in the degree distribution as well, as the expected degree of a vertex depends on the number of vertices of its cluster. Still, the variability of degree and cluster size is not appreciable. Besides, vertices of the same cluster keep having approximately the same degree. A better job in this direction has been recently done by Lancichinetti et al. (Lancichinetti *et al.*, 2008). They assume that the distributions of degree and community sizes are power laws, with exponents $\gamma$ and $\beta$, respectively. Each vertex shares a fraction $1-\mu$ of its edges with the other vertices of its community and a fraction $\mu$ with the vertices of the other communities; $0 \leq \mu \leq 1$ is the *mixing parameter*. The graphs are built as follows: 1) a sequence of degrees and a sequence of community sizes obeying the prescribed power-law distributions are extracted; 2) each vertex is given a degree of the sequence, as a set of adjacent stubs, that are randomly attached to stubs of the other vertices, so to create a graph preserving the degree sequence (on average); 3) vertices are assigned to communities such that minimal topological constraints are satisfied; 4) the internal degree of each vertex $v$ in its community is adjusted as close as possible to the prescribed value $(1-\mu)k_v$, $k_v$ being the total degree of $v$, by a series of rewiring steps, preserving the degree sequence (on average). Numerical tests show that this procedure has a complexity $O(m)$, where $m$ is as usual the number of edges of the graph, so it can be used to create graphs of sizes spanning several orders of magnitude. Fig. 27 shows an example of a benchmark graph. Recently the benchmark has been extended to directed and weighted graphs with overlapping communities (Lancichinetti and Fortunato, 2009). The software to create the benchmark graphs can be freely downloaded at `http://santo.fortunato.googlepages.com/inthepress2`.

A class of benchmark graphs with power law degree distributions had been previously introduced by Bagrow (Bagrow, 2008). The construction process starts from a graph with a power-law degree distribution.

Bagrow used Barabási-Albert scale free graphs (Barabási and Albert, 1999). Then, vertices are randomly assigned to one of four equally-sized communities. Finally, pairs of edges between two communities are rewired so that either edge ends up within the same community, without altering the degree sequence (on average). This is straightforward: suppose that the edges join the vertex pairs $a_1$, $b_1$ and $a_2$, $b_2$, where $a_1$, $a_2$ belong to community $A$ and $b_1$, $b_2$ to community $B$. It the edges are replaced by $a_1$-$a_2$ and $b_1$-$b_2$ (provided they do not exist already), all vertices keep their degrees. With this rewiring procedure one can arbitrarily vary the edge density within and, accordingly, between clusters. In this class of benchmarks, however, communities are all of the same size by construction, although one can in principle relax this condition.

A (seemingly) different benchmark is represented by the class of *relaxed caveman graphs*, which were originally introduced to explain the clustering properties of social networks (Watts, 2003). The starting point is a set of disconnected cliques. With some probability edges are rewired to link different cliques. Such model graphs are interesting as they are smooth variations of the ideal graph with "perfect" communities, i.e. disconnected cliques. On the other hand the model is equivalent to the planted $\ell$-partition model, where $p_{in} = 1 - p$ and $p_{out}$ is proportional to $p$, with coefficient depending on the size of the clusters.

Benchmark graphs have also been introduced to deal with special types of graphs and/or cluster structures. For instance, Arenas et al. (Arenas *et al.*, 2006) have introduced a class of benchmark graphs with embedded hierarchical structure, which extend the class of graphs by Girvan and Newman. Here there are 256 vertices and two hierarchical levels, corresponding to a partition in 16 groups (microcommunities) with 16 vertices and a partition in 4 larger groups of 64 vertices (macrocommunities), comprising each 4 of the smaller groups. The edge densities within and between the clusters are indicated by three parameters $z_{in_1}$, $z_{in_2}$ and $z_{out}$: $z_{in_1}$ is the expected internal degree of a vertex within its microcommunity; $z_{in_2}$ is the expected number of edges that the vertex shares with the vertices of the other microcommunities within its macrocommunity; $z_{out}$ is the expected number of edges connecting the vertex with vertices of the other three macrocommunities. The average degree $\langle k \rangle = z_{in_1} + z_{in_2} + z_{out}$ of a vertex is fixed to 18. Fig. 7 shows an example of hierarchical graph constructed based on the same principle, with 512 vertices and an average degree of 32.

Guimerà et al. (Guimerà *et al.*, 2007) have proposed a model of bipartite graphs with built-in communities. They considered a bipartite graph of actors and teams, here we describe how to build the benchmarks for general bipartite graphs. One starts from a bipartite graph whose vertex classes $A$ and $B$ are partitioned into $n_c$ groups, $\mathcal{C}_i^A$ and $\mathcal{C}_i^B$ $(i = 1, 2, ..., n_c)$. Each cluster $\mathcal{C}_i$ comprises all vertices of the subgroups $\mathcal{C}_i^A$ and $\mathcal{C}_i^B$, respectively. With probability $p$ edges are placed between vertices of subgroups $\mathcal{C}_i^A$ and $\mathcal{C}_i^B$ $(i = 1, 2, ..., n_c)$, i.e. within clusters. With probability $1 - p$, edges are placed between vertices of subgroups $\mathcal{C}_i^A$ and $\mathcal{C}_j^B$, where $i$ and $j$ are chosen at random, so they can be equal or different. By construction, a non-zero value of the probability $p$ indicates a preference by vertices to share links with vertices of the same cluster, i.e. for $p > 0$ the graph has a built-in community structure. For $p = 1$ there would be edges only within clusters, i. e. the graph has a perfect cluster structure.

Finally, Sawardecker et al. introduced a general model, that accounts for the possibility that clusters overlap (Sawardecker *et al.*, 2009). The model is based on the reasonable assumption that the probability $p_{ij}$ that two vertices are connected by an edge grows with the number $n_0$ of communities both vertices belong to. For vertices in different clusters, $p_{ij} = p_0$, if they are in the same cluster (and only in that one) $p_{ij} = p_1$, if they belong to the same two clusters $p_{ij} = p_2$, etc.. By hypothesis, $p_0 < p_1 \leq p_2 \leq p_3$.... The planted $\ell$-partition model is recovered when $p_1 = p_2 = p_3$....

As we have seen, nearly all existing benchmark graphs are inspired by the planted $\ell$-partition model, to some extent. However, the model needs to be refined to provide a good description of real graphs with community structure. The hypothesis that the linking probabilities of each vertex with the vertices of its community or of the other communities are constant is not realistic. It is more plausible that each pair of vertices $i$ and $j$ has its own linking probability $p_{ij}$, and that such probabilities are correlated for vertices in the same cluster.

Tests on real networks usually focus on a limited number of examples, for which one has precise information about the vertices and their properties.

In Section II we have introduced two popular real networks with known community structure, i. e. the social network of Zachary's karate club and the social network of bottlenose dolphins living in Doubtful Sound (New Zealand), studied by Lusseau. Here, the question is whether the actual separation in two social groups could be predicted from the graph topology. Zachary's karate club is by far the most investigated system. Several algorithms are actually able to identify the two classes, modulo a few intermediate vertices, which may be misclassified. Other methods are less successful: for instance, the maximum of Newman-Girvan modularity corresponds to a split of the network in four groups (Donetti and Muñoz, 2004; Duch and Arenas, 2005). Another well known example is the network of American college football teams, derived by Girvan and Newman (Girvan and Newman, 2002). There are 115 vertices, representing the teams, and two vertices are connected if their teams play against each other. The teams are divided into 12 conferences. Games between teams in the same conference are more frequent than games between teams of different conferences, so one has a natural partition where the communities correspond to the conferences.

When dealing with real networks, it is useful to re-

solve their community structure with different clustering techniques, to cross-check the results and make sure that they are consistent with each other, as in some cases the answer may strongly depend on the specific algorithm adopted. However, one has to keep in mind that there is no guarantee that "reasonable" communities, defined on the basis of non-structural information, must coincide with those detected by methods based only on the graph structure.

### B. Comparing partitions: measures

Checking the performance of an algorithm involves defining a criterion to establish how "similar" the partition delivered by the algorithm is to the partition one wishes to recover. Several measures for the similarity of partitions exist. In this section we present and discuss the most popular measures. A thorough introduction of similarity measures for graph partitions has been given by Meilă (Meilă, 2007) and we will follow it closely.

Let us consider two generic partitions $\mathcal{X} = (X_1, X_2, ..., X_{n_X})$ and $\mathcal{Y} = (Y_1, Y_2, ..., Y_{n_Y})$ of a graph $\mathcal{G}$, with $n_X$ and $n_Y$ clusters, respectively. We indicate with $n$ the number of graph vertices, with $n_i^X$ and $n_j^Y$ the number of vertices in clusters $X_i$ and $Y_j$ and with $N_{ij}$ the number of vertices shared by clusters $X_i$ and $Y_j$: $N_{ij} = |X_i \bigcap Y_j|$.

In the first tests using the benchmark graphs by Girvan and Newman (Section XIV.A) scholars used a measure proposed by Girvan and Newman themselves, the *fraction of correctly classified vertices*. A vertex is correctly classified if it is in the same cluster with at least half of its "natural" partners. If the partition found by the algorithm has clusters given by the merging of two or more natural groups, all vertices of the cluster are considered incorrectly classified. The number of correctly classified vertices is then divided by the total size of the graph, to yield a number between 0 and 1. The recipe to label vertices as correctly or incorrectly classified is somewhat arbitrary, though.

Apart from the fraction of correctly classified vertices, which is somewhat *ad hoc* and distinguishes the roles of the natural partition and of the algorithm's partition, most similarity measures can be divided in three categories: measures based on *pair counting*, *cluster matching* and *information theory*.

Measures based on pair counting depend on the number of pairs of vertices which are classified in the same (different) clusters in the two partitions. In particular $a_{11}$ indicates the number of pairs of vertices which are in the same community in both partitions, $a_{01}$ ($a_{10}$) the number of pairs of elements which are put in the same community in $\mathcal{X}$ ($\mathcal{Y}$) and in different communities in $\mathcal{Y}$ ($\mathcal{X}$) and $a_{00}$ the number of pairs of vertices that are in different communities in both partitions. Wallace (Wallace, 1983) proposed the two indices

$$W_I = \frac{a_{11}}{\sum_k n_k^X (n_k^X - 1)/2}; \qquad W_{II} = \frac{a_{11}}{\sum_k n_k^Y (n_k^Y - 1)/2}. \tag{83}$$

$W_I$ and $W_{II}$ represent the probability that vertex pairs in the same cluster of $\mathcal{X}$ are also in the same cluster for $\mathcal{Y}$, and viceversa. These indices are asymmetrical, as the role of the two partitions is not the same. Fowlkes and Mallows (Fowlkes and Mallows, 1983) suggested to use the geometric mean of $W_I$ and $W_{II}$, which is symmetric.

The *Rand index* (Rand, 1971) is the ratio of the number of vertex pairs correctly classified in both partitions (i.e. either in the same or in different clusters), by the total number of pairs

$$R(\mathcal{X}, \mathcal{Y}) = \frac{a_{11} + a_{00}}{a_{11} + a_{01} + a_{10} + a_{00}}. \tag{84}$$

A measure equivalent to the Rand index is the *Mirkin metric* (Mirkin, 1996)

$$M(\mathcal{X}, \mathcal{Y}) = 2(a_{01} + a_{10}) = n(n-1)[1 - R(\mathcal{X}, \mathcal{Y})]. \tag{85}$$

The *Jaccard index* is the ratio of the number of vertex pairs classified in the same cluster in both partitions, by the number of vertex pairs which are classified in the same cluster in at least one partition, i.e.

$$J(\mathcal{X}, \mathcal{Y}) = \frac{a_{11}}{a_{11} + a_{01} + a_{10}}. \tag{86}$$

Adjusted versions of both the Rand and the Jaccard index exist, in that a null model is introduced, corresponding to the hypothesis of independence of the two partitions (Meilă, 2007). The null model expectation value of the measure is subtracted from the unadjusted version, and the result is normalized by the range of this difference, yielding 1 for identical partitions and 0 as expected value for independent partitions (negative values are possible as well). Unadjusted indices have the drawback that they are not local, i.e. the result depends on how the whole graph is partitioned, even when the partitions differ only in a small region of the graph.

Similarity measures based on cluster matching aim at finding the largest overlaps between pairs of clusters of different partitions. For instance, the *classification error* $H(\mathcal{X}, \mathcal{Y})$ is defined as (Meilă and Heckerman, 2001)

$$H(\mathcal{X}, \mathcal{Y}) = 1 - \frac{1}{n} \max_{\pi} \sum_{k=1}^{n_X} n_{k\pi(k)}, \tag{87}$$

where $\pi$ is an injective mapping from the cluster indices of partition $\mathcal{Y}$ to the cluster indices of partition $\mathcal{X}$. The maximum is taken over all possible injections $\{\pi\}$. In this way one recovers the maximum overlap between the clusters of the two partitions. An alternative measure is the *normalized Van Dongen metric*, defined as (van Dongen, 2000b)

$$D(\mathcal{X}, \mathcal{Y}) = 1 - \frac{1}{2n} \left[ \sum_{k=1}^{n_X} \max_{k'} n_{kk'} + \sum_{k'=1}^{n_Y} \max_{k} n_{kk'} \right]. \tag{88}$$

A common problem of this type of measures is that some clusters may not be taken into account, if their overlap with clusters of the other partition is not large enough. Therefore if we compute the similarity between two partitions $\mathcal{X}$ and $\mathcal{X}'$ and partition $\mathcal{Y}$, with $\mathcal{X}$ and $\mathcal{X}'$ differing from each other by a different subdivision of parts of the graph that are not used to compute the measure, one would obtain the same score.

The third class of similarity measures is based on reformulating the problem of comparing partitions as a problem of message decoding within the framework of information theory (Mackay, 2003). The idea is that, if two partitions are similar, one needs very little information to infer one partition given the other. This extra information can be used as a measure of dissimilarity. To evaluate the Shannon information content (Mackay, 2003) of a partition, one starts by considering the community assignements $\{x_i\}$ and $\{y_i\}$, where $x_i$ and $y_i$ indicate the cluster labels of vertex $i$ in partition $\mathcal{X}$ and $\mathcal{Y}$, respectively. One assumes that the labels $x$ and $y$ are values of two random variables $X$ and $Y$, with joint distribution $P(x,y) = P(X = x, Y = y) = n_{xy}/n$, which implies that $P(x) = P(X = x) = n_x^X/n$ and $P(y) = P(Y = y) = n_y^Y/n$. The *mutual information* $I(X,Y)$ of two random variables has been previously defined (Eq. 68), and can be applied as well to partitions $\mathcal{X}$ and $\mathcal{Y}$, since they are described by random variables. Actually $I(X,Y) = H(X) - H(X|Y)$, where $H(X) = -\sum_x P(x)\log P(x)$ is the Shannon entropy of $X$ and $H(X|Y) = -\sum_{x,y} P(x,y)\log P(x|y)$ is the conditional entropy of $X$ given $Y$. The mutual information is not ideal as a similarity measure: in fact, given a partition $\mathcal{X}$, all partitions derived from $\mathcal{X}$ by further partitioning (some of) its clusters would all have the same mutual information with $\mathcal{X}$, even though they could be very different from each other. In this case the mutual information would simply equal the entropy $H(X)$, because the conditional entropy would be systematically zero. To avoid that, Danon et al. adopted the *normalized mutual information* (Danon *et al.*, 2005)

$$I_{norm}(\mathcal{X},\mathcal{Y}) = \frac{2I(X,Y)}{H(X)+H(Y)}, \tag{89}$$

which is currently very often used in tests of graph clustering algorithms. The normalized mutual information equals 1 if the partitions are identical, whereas it has an expected value of 0 if the partitions are independent. The measure, defined for standard partitions, in which each vertex belongs to only one cluster, has been recently extended to the case of overlapping clusters by Lancichinetti et al. (Lancichinetti *et al.*, 2009). The extension is not straightforward as the community assignments of a partition are now specified by a vectorial random variable, since each vertex may belong to more clusters simultaneously.

Meilă (Meilă, 2007) introduced the *variation of information*

$$V(\mathcal{X},\mathcal{Y}) = H(X|Y) + H(Y|X), \tag{90}$$

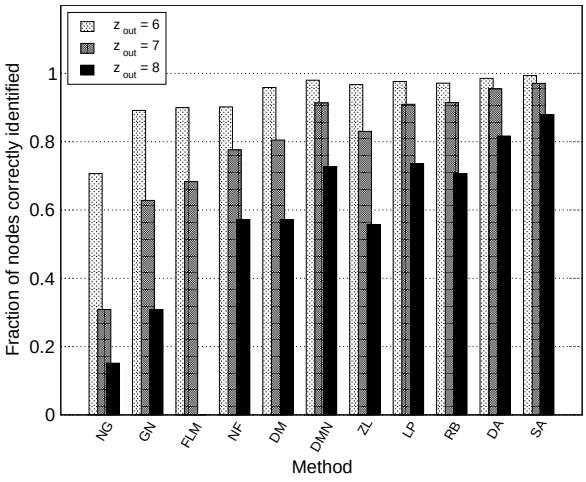


FIG. 28 Relative performances of the algorithms listed in Table I on the Girvan-Newman benchmark, for three values of the expected average external degree $z_{out}$. Reprinted figure with permission from (Danon *et al.*, 2005). ©2005 by IOP Publishing and SISSA.

which has some desirable properties with respect to the normalized mutual information and other measures. In particular, it defines a metric in the space of partitions as it has the properties of distance. It is also a local measure, i.e. the similarity of partitions differing only in a small portion of a graph depends on the differences of the clusters in that region, and not on the partition of the rest of the graph. The maximum value of the variation of information is $\log n$, so similarity values for partitions of graphs with different size cannot be compared with each other. For meaningful comparisons one could divide $V(\mathcal{X},\mathcal{Y})$ by $\log n$, as suggested by Karrer et al. (Karrer *et al.*, 2008).

A concept related to similarity is that of *distance*, which indicates basically how many operations need to be performed in order to transform a partition to another. Gustafsson et al. defined two distance measures for partitions (Gustafsson *et al.*, 2006). They are both based on the concept of *meet* of two partitions, which is defined as

$$\mathcal{M} = \bigcup_{i=1}^{n_A}\bigcup_{j=1}^{n_B}\left[X_i \bigcap Y_j\right]. \tag{91}$$

The distance measures are $m_{moved}$ and $m_{div}$. In both cases they are determined by summing the distances of $\mathcal{X}$ and $\mathcal{Y}$ from the meet $\mathcal{M}$. For $m_{moved}$ the distance of $\mathcal{X}$ ($\mathcal{Y}$) from the meet is the minimum number of elements that must be moved between $\mathcal{X}$ and $\mathcal{Y}$ so that $\mathcal{X}$ ($\mathcal{Y}$) and $\mathcal{M}$ coincide (Gusfield, 2002). For $m_{div}$ the distance of $\mathcal{X}$ ($\mathcal{Y}$) from the meet is the minimum number of divisions that must be done in $\mathcal{X}$ ($\mathcal{Y}$) so that $\mathcal{X}$ ($\mathcal{Y}$) and $\mathcal{M}$ coin-

cide (Stanley, 1997). Such distance measures can easily be transformed in similarity measures, like

$$I_{moved} = 1 - m_{moved}/n, \qquad I_{div} = 1 - m_{div}/n. \quad (92)$$

Identical partitions have zero mutual distance and similarity 1 based on Eqs. 92.

Finally an important problem is how to define the similarity between clusters. If two partitions $\mathcal{X}$ and $\mathcal{Y}$ of a graph are similar, each cluster of $\mathcal{X}$ will be very similar to one cluster of $\mathcal{Y}$, and viceversa, and it is important to identify the pairs of corresponding clusters. For instance, if one has information about the time evolution of a graph, one could monitor the dynamics of single clusters as well, by keeping track of each cluster at different time steps (Palla *et al.*, 2007). Given clusters $X_i$ and $Y_j$, their similarity can be defined through the *relative overlap* $s_{ij}$

$$s_{ij} = \frac{|X_i \bigcap Y_j|}{|X_i \bigcup Y_j|}. \quad (93)$$

In this way, looking for the cluster of $\mathcal{Y}$ corresponding to $X_i$ means finding the cluster $Y_j$ that maximizes $s_{ij}$. The index $s_{ij}$ can be used to define similarity measures for partitions as well (Fan *et al.*, 2007; Zhang *et al.*, 2006). An interesting discussion on the problem of comparing partitions, along with further definitions of similarity measures not discussed here, can be found in (Traud *et al.*, 2008).

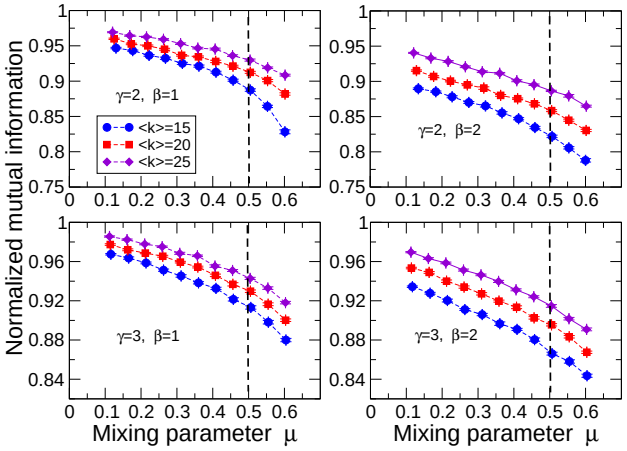


FIG. 29 Performance of modularity optimization, implemented with simulated annealing, on the benchmark graphs by Lancichinetti et al. (Lancichinetti *et al.*, 2008). The four panels refer to different choices for the exponents $\gamma$ and $\beta$ of the degree and community size distributions. All graphs have 5000 vertices. Each curve refers to a given value of the average degree. The similarity of the partition found through modularity optimization with the planted partition of the benchmark is not perfect even for very small values of the mixing parameter $\mu$. This is due to the resolution limit of modularity optimization (Section VI.C), which induces the method to artificially merge small clusters. Reprinted figure with permission from (Lancichinetti *et al.*, 2008). ©2008 by the American Physical Society.

### C. Comparing algorithms

The first and so far only systematic comparative analysis of graph clustering techniques has been carried out by Danon et al. (Danon *et al.*, 2005). They compared the performances of various algorithms on the benchmark graphs by Girvan and Newman (Section XIV.A). The algorithms examined are listed in Table I, along with their complexity. Fig. 28 shows the performance of all algorithms. Instead of showing the whole curves of the similarity versus $z_{out}$ (Section XIV.A), which would display a fuzzy picture with many strongly overlapping curves, difficult to appreciate, Danon et al. considered three values for $z_{out}$ (6, 7 and 8), and represented the result for each algorithm as a group of three columns, indicating the average value of the similarity between the planted partition and the partition found by the method for each of the three $z_{out}$-values. The similarity was measured in terms of the fraction of correctly classified vertices (Section XIV.A). The comparison shows that modularity optimization via simulated annealing (Section VI.A.2) yields the best results, although it is a rather slow procedure, that cannot be applied to graphs of size of the order of $10^5$ vertices or larger.

On the other hand, we have already pointed out that the benchmark by Girvan and Newman is not a good representation of real graphs with community structure, which are characterized by heterogeneous distribution of degree and community sizes. In this respect, the class of graphs designed by Lancichinetti et al. (Lancichinetti *et al.*, 2008) (Section XIV.A) poses a far more severe test to clustering techniques. Many methods, for instance, have problems to detect clusters of very different sizes, including many methods listed in Table I. In Section VI.C we have seen that modularity optimization has a resolution limit that makes small clusters (small with respect to the graph size) often undetectable. Tests on the benchmark by Lancichinetti et al. immediately disclose this problem (Fig. 29), which instead does not occur in the benchmark by Girvan and Newman. For this reason, we believe that in the future it is necessary to carry out a careful comparative analysis of community detection methods on the much more restrictive benchmark by Lancichinetti et al.

Fan et al. have evaluated the performance of some algorithms to detect communities on weighted graphs (Fan *et al.*, 2007). The algorithms are: modularity maximization, carried out with extremal optimization (WEO) (Section VI.A.3); the Girvan-Newman algorithm (WGN) (Section V.A); the Potts model algorithm by Reichardt and Bornholdt (Potts) (Section VIII.A). All these techniques have been originally introduced for unweighted graphs, but we have shown that they can easily be ex-

| Author | Ref. | Label | Order |
|---|---|---|---|
| Eckmann & Moses | (Eckmann and Moses, 2002) | EM | $O(m\langle k^2\rangle)$ |
| Zhou & Lipowsky | (Zhou and Lipowsky, 2004) | ZL | $O(n^3)$ |
| Latapy & Pons | (Latapy and Pons, 2005) | LP | $O(n^3)$ |
| Clauset, Newman & Moore | (Clauset *et al.*, 2004) | NF | $O(n\log^2 n)$ |
| Newman & Girvan | (Newman and Girvan, 2004) | NG | $O(nm^2)$ |
| Girvan & Newman | (Girvan and Newman, 2002) | GN | $O(n^2m)$ |
| Guimerà et al. | (Guimerà and Amaral, 2005; Guimerà *et al.*, 2004) | SA | parameter dependent |
| Duch & Arenas | (Duch and Arenas, 2005) | DA | $O(n^2\log n)$ |
| Fortunato, Latora & Marchiori | (Fortunato *et al.*, 2004) | FLM | $O(n^4)$ |
| Radicchi et al. | (Radicchi *et al.*, 2004) | RCCLP | $O(n^2)$ |
| Donetti & Muñoz | (Donetti and Muñoz, 2004, 2005) | DM/DMN | $O(n^3)$ |
| Bagrow & Bollt | (Bagrow and Bollt, 2005) | BB | $O(n^3)$ |
| Capocci et al. | (Capocci *et al.*, 2005) | CSCC | $O(n^2)$ |
| Wu & Huberman | (Wu and Huberman, 2004) | WH | $O(n+m)$ |
| Palla et al. | (Palla *et al.*, 2005) | PK | $O(\exp(n))$ |
| Reichardt & Bornholdt | (Reichardt and Bornholdt, 2004) | RB | parameter dependent |

TABLE I List of the algorithms used in the comparative analysis of Danon et al. (Danon *et al.*, 2005). The first column indicates the names of the algorithm designers, the second the original reference of the work, the third the symbol used to indicate the algorithm and the last the computational complexity of the technique. Adapted from (Danon *et al.*, 2005).

tended to weighted graphs. The algorithms were tested on the weighted version of the benchmark of Girvan and Newman, that we discussed in Section XIV.A. Edge weights have only two values: $w_{inter}$ for inter-cluster edges and $w_{intra}$ for intra-cluster edges. Such values are linked by the relation $w_{intra} + w_{inter} = 2$, so they are not independent. For testing one uses realizations of the benchmark with fixed topology (i.e. fixed $z_{out}$) and variable weights. In Fig. 30 the comparative performance of the three algorithms is illustrated. The topology of the benchmark graphs corresponds to $z_{out} = 8$, i.e. to graphs in which each vertex has approximately the same number of neighbors inside and outside its community. By varying $w_{inter}$ from 0 to 2 one goes smoothly from a situation in which the most of the weight is concentrated inside the clusters, to a situation in which instead the weight is concentrated between the clusters. From Fig. 30 we see that WEO and Potts are more reliable methods.

Sawardecker et al. have tested methods to detect overlapping communities (Sawardecker *et al.*, 2009). They considered three algorithms: modularity optimization, the Clique Percolation Method (CPM) (Section XI.A) and the modularity landscape surveying method by Sales-Pardo et al.(Sales-Pardo *et al.*, 2007) (Section XII.B). For testing, Sawardecker et al. defined a class of benchmark graphs in which the linking probability between vertices is an increasing function of the number of clusters the vertices belong to. We have described this benchmark in Section XIV.A. It turns out that the modularity landscape surveying method is able to identify overlaps between communities, as long as the fraction of overlapping vertices is small. Curiously, the CPM, designed to find overlapping communities, has a poor performance, as the overlapping vertices found by the algorithm are in general different from the overlapping vertices of the planted partition of the benchmark. The authors also remark that, if the overlap between two clusters is not too small, it may be hard (for any method) to recognize whether the clusters are overlapping or hierarchically organized, i.e. loosely connected clusters within a large cluster.

We close the section with some general remarks concerning testing. We have seen that a testing procedure requires two crucial ingredients: benchmark graphs with built-in community structure and clustering algorithms that try to recover it. Such two elements are not independent, however, as they are both based on the concept of community. If the underlying notions of community for the benchmark and the algorithm are very different, one can hardly expect that the algorithm will do a good job on the benchmark. Furthermore, there is a third element, i.e. the quality of a partition. All benchmarks start from a situation in which communities are clearly identified, i.e. connected components of the graph, and introduce some amount of noise, that eventually leads to a scenario where clusters are hardly or no longer detectable. It is then important to keep track of how the quality of the natural partition of the benchmark worsens as the amount of noise increases, in order to distinguish configurations in which the graphs have a cluster structure, that an algorithm should then be able to resolve, from configurations in which the noise prevails and the natural clusters are not meaningful. Moreover, quality functions are important to evaluate the performance of an algorithm on graphs whose community structure is

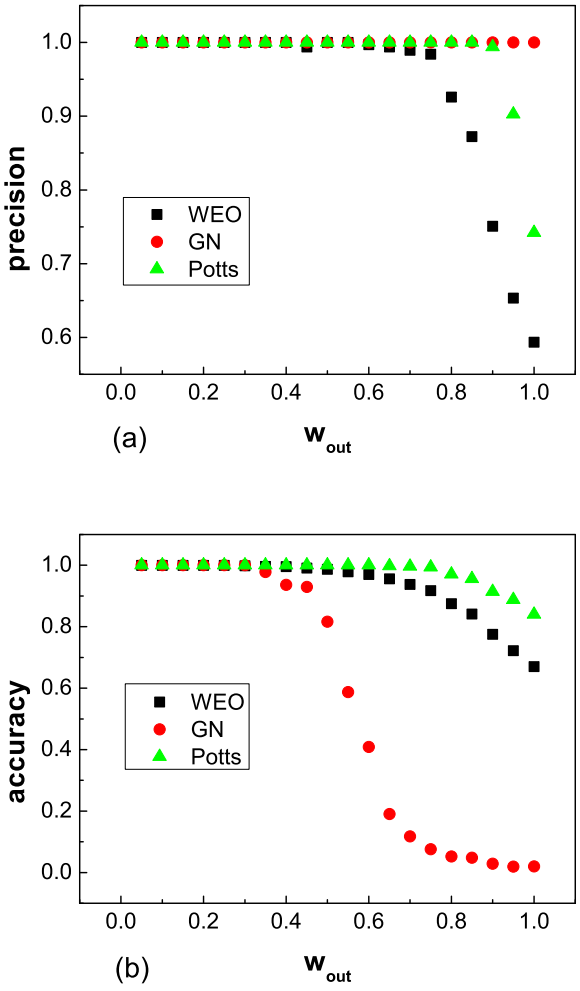


FIG. 30 Comparative evaluation of the performances of algorithms to find communities in weighted graphs. Tests are carried out on a weighted version of the benchmark of Girvan and Newman. The two plots show how good the algorithms are in terms of the precision and accuracy with which they recover the planted partition of the benchmark. Precision indicates how close the values of similarity between the planted and the model partition are after repeated experiments with the same set of parameters; accuracy indicates how close the similarity values are to the ideal result (1) after repeated experiments with the same set of parameters. The similarity measure adopted here is based on the relative overlap of clusters of Eq. 93. We see that the maximization of modularity with extremal optimization (WEO) and the Potts model algorithm (Potts) are both precise and accurate as long as the weight of the inter-cluster edges $w_{inter}$ remains lower than the weight of the intra-cluster edges ($w_{inter} < 1$). Reprinted figures with permission from (Fan *et al.*, 2007). ©2007 by Elsevier.

unknown. Quality functions are strongly related to the concept of community as well, as they are supposed to evaluate the goodness of the clusters, so they require a clear quantitative concept of what a cluster is. It is very important for any testing framework to check for the mutual dependencies between the benchmark, the quality function used to evaluate partitions, and the clustering algorithm to be tested. This issue has so far received very little attention (Delling *et al.*, 2007). Finally, empirical tests are also very important, as one ultimately wishes to apply clustering techniques to real graphs. Therefore, it is crucial to collect more data sets of graphs whose community structure is known or deducible from information on the vertices and their edges.

## XV. GENERAL PROPERTIES OF REAL CLUSTERS

What are the general properties of partitions and clusters of real graphs? In many papers on graph clustering applications to real systems are presented. In spite of the variety of clustering methods that one could employ, in many cases partitions derived from different techniques are rather similar to each other, so the general properties of clusters do not depend much on the particular algorithm used. The analysis of clusters and their properties delivers a *mesoscopic description* of the graph, where the communities, and not the vertices, are the elementary units of the topology. The term mesoscopic is used because the relevant scale here lies between the scale of the vertices and that of the full graph. Here we discuss separately results on *static communities*, i. e. clusters of individual system configurations, and results on *dynamic communities*, i. e. clusters of systems evolving in time.

### A. Static communities

One of the first issues addressed was whether the communities of a graph are usually about of the same size or whether the community sizes have some special distribution. Most clustering techniques consistently find skewed distributions of community sizes, with a tail described with good approximation by a power law (at least, a sizeable portion of the curve) with exponents in the range between 1 and 3 (Clauset *et al.*, 2004; Danon *et al.*, 2007; Newman, 2004a; Palla *et al.*, 2005; Radicchi *et al.*, 2004). So, there seems to be no characteristic size for a community: small communities usually coexist with large ones. As an example, Fig. 31 shows the cumulative distribution of community sizes for a recommendation network of the online vendor Amazon.com. Vertices are products and there is a connection between item $A$ and $B$ is $B$ was frequently purchased by buyers of $A$. Recall that the cumulative distribution is the integral of the probability distribution: if the cumulative distribution is a power law $s^{-\alpha}$, the probability distribution is also a power law with exponent $\alpha + 1$.

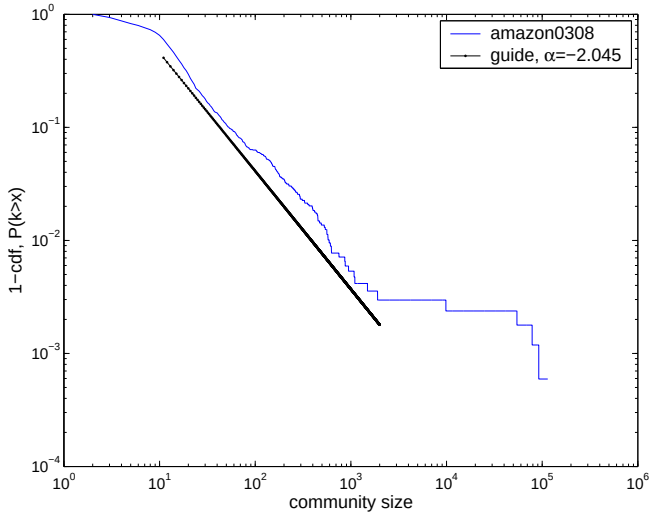


FIG. 31 Cumulative distribution of community sizes for the Amazon purchasing network. The partition is derived by greedy modularity optimization. Reprinted figure with permission from (Clauset *et al.*, 2004). ©2004 by the Americal Physical Society.

If communities are overlapping, one can explore other statistical properties, like the distributions of the overlaps and of the vertex memberships. The overlap is defined as the number of vertices shared by each pair of overlapping clusters; the membership of a vertex is the number of communities including the vertex. Both distributions turn out to be skewed, so there seem to be no characteristic values for the overlap and the membership. Moreover, one could derive a network, where the communities are the vertices and pairs of vertices are connected if their corresponding communities overlap (Palla *et al.*, 2005). Such networks seem to have some special properties. For instance, the degree distribution is a particular function, with an initial exponential decay followed by a slower power law decay[19]. We stress that the above results have been obtained with the Clique Percolation Method by Palla et al. (Section XI.A) and it is not clear whether other techniques would confirm them or not. In a recent analysis it has been shown that the degree distribution of the network of communities can be reproduced by assuming that the graph grows according to a simple preferential attachment mechanism, where communities with large degree have an enhanced chance to interact/overlap with new communities (Pollner *et al.*, 2006).

If the community structure of a graph is known, it is possible to classify vertices according to their roles within their community, which may allow to infer individual properties of the vertices. A promising classification has been proposed by Guimerà and Amaral (Guimerà and Amaral, 2005; Guimerà and Amaral, 2005). The role of a vertex depends on the values of two indices, the *within-module degree* and the *participation ratio* (though other variables may be chosen, in principle). The within-module degree $z_i$ of vertex $i$ is defined as

$$z_i = \frac{\kappa_i - \bar{\kappa}_{s_i}}{\sigma_{\kappa_{s_i}}}, \tag{94}$$

where $\kappa_i$ is the internal degree of $i$ in its cluster $s_i$, $\bar{\kappa}_{s_i}$ and $\sigma_{\kappa_{s_i}}$ the average and standard deviation of the internal degrees for all vertices of cluster $s_i$. The within-module degree is then defined as the $z$-score of the internal degree $\kappa_i$. Large values of $z$ indicate that the vertex has many more neighbors within its community than most other vertices of the community. Vertices with $z \geq 2.5$ are classified as *hubs*, if $z < 2.5$ they are *non-hubs*. The participation ratio $P_i$ of vertex $i$ is defined as

$$P_i = 1 - \sum_{s=1}^{n_c} \left( \frac{\kappa_{is}}{k_i} \right)^2 . \tag{95}$$

Here $\kappa_{is}$ is the internal degree of $i$ in cluster $s$, $k_i$ the degree of $i$. Values of $P$ close to 1 indicate that the neighbors of the vertex are uniformly distributed among all clusters; if all neighbors are within the cluster of the vertex, instead, $P = 0$. Based on the values of the pair $(z, P)$, Guimerà and Amaral distinguished seven roles for the vertices. Non-hub vertices can be *ultra-peripheral* ($P \approx 0$), *peripheral* ($P < 0.625$), *connectors* ($0.625 < P < 0.8$) and *kinless vertices* ($P > 0.8$). Hub vertices are classified in *provincial hubs* ($P <\sim 0.3$), *connector hubs* ($0.3 < P < 0.75$) and *kinless hubs* ($P > 0.75$). The regions of the $z - P$ plane corresponding to the seven roles are highlighted in Fig. 32. We stress that the actual boundaries of the regions can be chosen rather arbitrarily. On graphs without community structure, like Erdös-Rényi (Erdös and Rényi, 1959) random graphs and Barabási-Albert (Barabási and Albert, 1999) graphs (Section A.3), non-hubs are mostly kinless vertices. In addition, if there are hubs, like in Barabási-Albert graphs, they are kinless hubs. Kinless hubs (non-hubs) vertices have less than half (one third) of their neighbors inside any cluster, so they are not clearly associated to a cluster. On real graphs, the topological roles can be correlated to functions of vertices: in metabolic networks, for instance, connector hubs, which share most edges with vertices of other clusters than their own, are often metabolites which are more conserved across species than other metabolites, i.e. they have an evolutionary advantage (Guimerà and Amaral, 2005).

### B. Dynamic communities

The analysis of dynamic communities is still in its infancy. Studies in this direction have been mostly hin-

[19] This holds for the networks considered by Palla et al. (Palla *et al.*, 2005) like, e. g., the word association network (Section II) and a coauthorship network of physicists. There is no *a priori* reason to believe that this result is general.

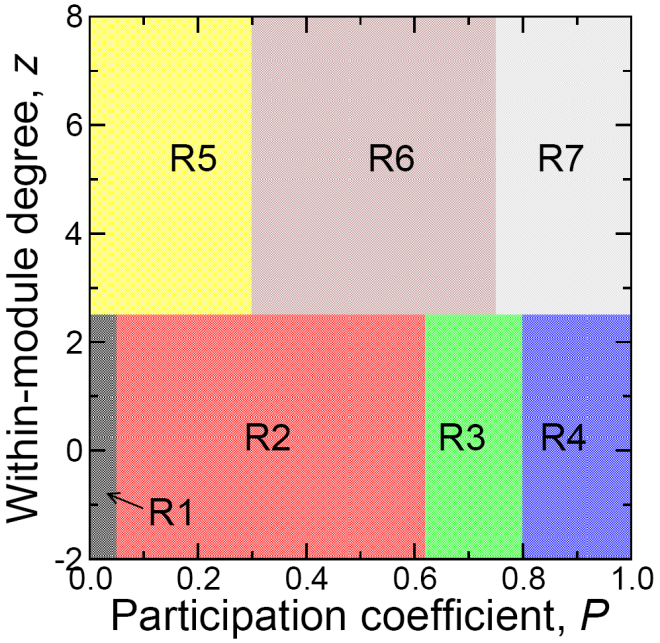


FIG. 32 Regions of the $z - P$ plane defining the roles of vertices in the modular structure of a graph, according to the scheme of Guimerà and Amaral (Guimerà and Amaral, 2005; Guimerà and Amaral, 2005). Reprinted figure with permission from (Guimerà and Amaral, 2005). ©2005 by the Nature Publishing Group.

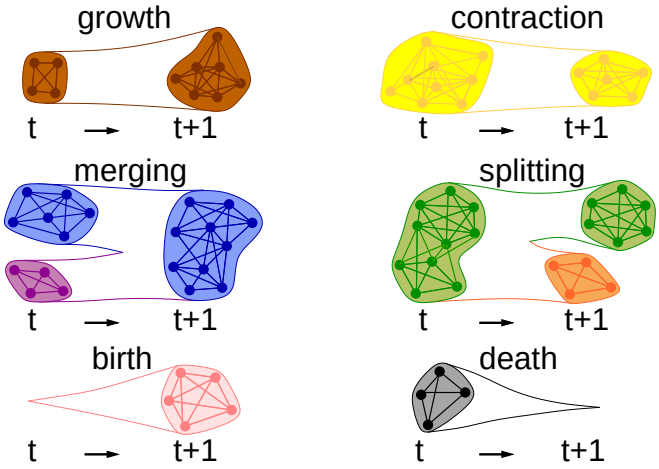


FIG. 33 Possible scenarios in the evolution of communities. Reprinted figure with permission from (Palla *et al.*, 2007). ©2007 by the Nature Publishing Group.

dered by the fact that the problem of graph clustering is already controversial on single graph realizations, so it is understandable that most efforts still concentrate on the "static" version of the problem. Another difficulty is represented by the dearth of time-stamped data on real graphs. Recently, several data sets have become available, enabling to monitor the evolution in time of real systems. So it has become possible to investigate how communities form, evolve and die. The main phenomena occurring in the lifetime of a community are (Fig. 33): birth, growth, contraction, merger with other communities, split, death.

The first study was carried out by Hopcroft et al. (Hopcroft *et al.*, 2004), who analyzed several snapshots of the citation graph induced by the NEC CiteSeer Database (Giles *et al.*, 1998). The snapshots cover the period from 1990 to 2001. Communities are detected by means of (agglomerative) hierarchical clustering (Section IV.B), where the similarity between vertices is the *cosine similarity* of the vectors describing the corresponding papers, a well known measure used in information retrieval (Baeza-Yates and Ribeiro-Neto, 1999). In each snapshot Hopcroft et al. identified the *natural communities*, defined as those communities of the hierarchical tree that are only slightly affected by minor perturbations of the graph, where the perturbation consists in removing a small fraction of the vertices (and their edges). Such natural communities remind us of the *stable* communities we have seen in Section XIII. Hopcroft et al. found the best matching natural communities across different snapshots, and in this way they could follow the history of communities. In particular they could see the emergence of new communities, corresponding to new research topics. The main drawback of the method comes from the use of hierarchical clustering, which is unable to sort out meaningful communities out of the hierarchical tree, which includes many different partitions of the graph.

More recently, Palla et al. performed the first systematic analysis of dynamic communities (Palla *et al.*, 2007). They studied two social systems: 1) a graph of phone calls between customers of a mobile phone company in a year's time; 2) a collaboration network between scientists, describing the coauthorship of papers in condensed matter physics from the electronic e-print archive (cond-mat) maintained by Cornell University Library, spanning a period of 142 months. The first problem is identifying the image of a community $\mathcal{C}(t+1)$ at time $t+1$ among the communities of the graph at time $t$. A simple criterion, used in other works, is to measure the relative overlap (Eq. 93) of $\mathcal{C}(t+1)$ with all communities at time $t$, and pick the community which has the largest overlap with $\mathcal{C}(t+1)$. This is intuitive, but in many cases it may miss the actual evolution of the community. For instance, if $\mathcal{C}(t)$ at time $t+1$ grows considerably and overlaps with another community $\mathcal{B}(t+1)$ (which at the previous time step was disjoint from $\mathcal{C}(t)$), the relative overlap between $\mathcal{C}(t+1)$ and $\mathcal{B}(t)$ may be larger than the relative overlap between $\mathcal{C}(t+1)$ and $\mathcal{C}(t)$. It is not clear whether there is a general prescription to avoid this problem. Palla et al. solved it by exploiting the features of the Clique Percolation Method (CPM) (Section XI.A), that they used to detect communities. The idea is to analyze the graph $\mathcal{G}(t, t+1)$, obtained by merging the two snapshots $\mathcal{G}(t)$ and $\mathcal{G}(t+1)$ of the evolving graph, at times $t$ and $t+1$ (i. e., by putting together all their vertices and edges). Any

CPM community of $\mathcal{G}(t)$ and $\mathcal{G}(t+1)$ does not get lost, as it is included within one of the CPM communities of $\mathcal{G}(t, t+1)$. For each CPM community $\mathcal{V}_k$ of $\mathcal{G}(t, t+1)$, one finds the CPM communities $\{\mathcal{C}_k^t\}$ and $\{\mathcal{C}_k^{t+1}\}$ (of $\mathcal{G}(t)$ and $\mathcal{G}(t+1)$, respectively) which are contained in $\mathcal{V}_k$. The image of any community in $\{\mathcal{C}_k^{t+1}\}$ at time $t$ is the community of $\{\mathcal{C}_k^t\}$ that has the largest relative overlap with it.

The age $\tau$ of a community is the time since its birth. It turns out that the age of a community is positively correlated with its size $s(\tau)$, i. e. that older communities are also larger (on average). The time evolution of a community $\mathcal{C}$ can be described by means of the relative overlap $C(t)$ between states of the community separated by a time $t$:

$$C(t) = \frac{|\mathcal{C}(t_0) \bigcap \mathcal{C}(t_0 + t)|}{|\mathcal{C}(t_0) \bigcup \mathcal{C}(t_0 + t)|}. \qquad (96)$$

One finds that, in both data sets, $C(t)$ decays faster for larger communities, so the composition of large communities is rather variable in time, whether small communities are essentially static. Another important question is whether it is possible to predict the evolution of communities from information on their structure or on their vertices. In Fig. 34a the probability $p_l$ that a vertex will leave the community in the next step of the evolution is plotted as a function of the relative external strength of the vertex, indicating how much of the vertex strength lies on edges connecting it to vertices outside its community. The plot indicates that there is a clear positive correlation: vertices which are only loosely connected to vertices of their community have a higher chance (on average) to leave the community than vertices which are more "committed" towards the other community members. The same principle holds at the community level too. Fig. 34b shows that the probability $p_d$ that a community will disintegrate in the next time step is positively correlated with the relative external strength of the community. Finally, Palla et al. have found that the probability for two communities to merge increases with the community sizes much more than what one expects from the size distribution, which is consistent with the faster dynamics observed for large communities. Palla et al. analyzed two different real systems, a network of mobile phone communications and a coauthorship network, to be able to infer general properties of community evolution. However, communities were only found with the CPM, so their results need to be cross-checked by employing other clustering techniques.

Dynamic communities can be as well detected with methods of information compression, such as some of those we have seen in Section IX.B. Sun et al. (Sun *et al.*, 2007) applied the Minimum Description Length (MDL) principle (Grünwald *et al.*, 2005; Rissanen, 1978) to find the minimum encoding cost for the description of a time sequence of graphs and their partitions in communities. The method is quite similar to that successively developed by Rosvall and Bergstrom (Rosvall and Bergstrom, 2007), which is however defined only for static graphs (Section IX.B). Here one considers bipartite graphs evolving in time. The time sequence of graphs can be separated in segments, each containing some number of consecutive snapshots of the system. The graphs of each segment are supposed to have the same modular structure (i. e. they represent the same phase in the history of the system), so they are characterized by the same partition of the two vertex classes. For each graph segment it is possible to define an encoding cost, which combines the encoding cost of the partition of the graphs of the segment with the entropy of compression of the segment in the subgraph segments induced by the partition. The total encoding cost $C$ of the graph series is given by the sum of the encoding costs of its segments. Minimizing $C$ enables one to find not only the most modular

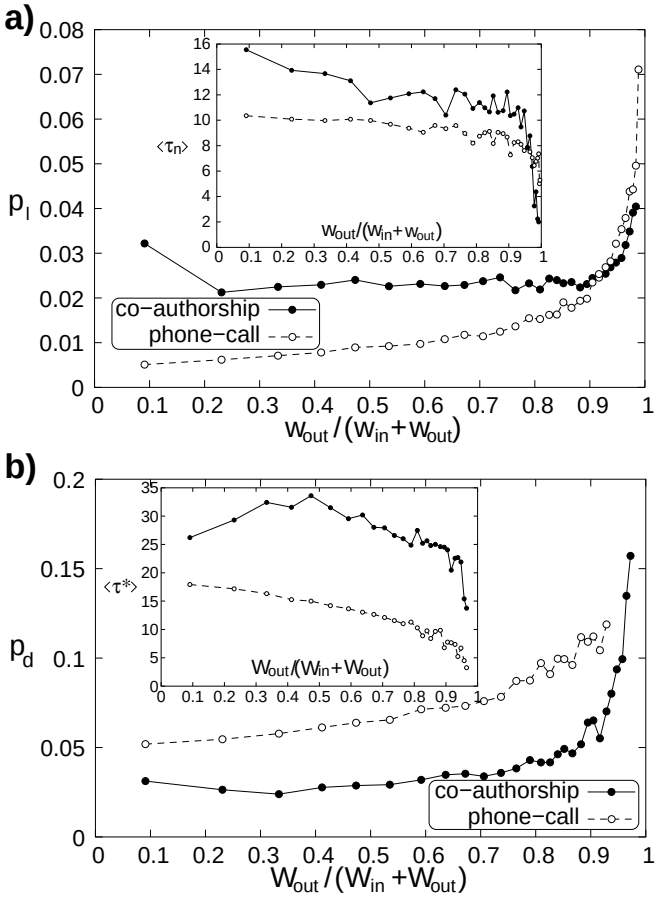


FIG. 34 Relation between structural features and evolution of a community. a) Relation between the probability that a vertex will abandon the community in the next time step and its relative external strength. b) Relation between the probability of disintegration of a community in the next time step and its relative external strength. Reprinted figure with permission from (Palla *et al.*, 2007). ©2007 by the Nature Publishing Group.

partition for each graph segment (high modularity[20] corresponds to low encoding costs for a partition), but also the most compact subdivision of the snapshots into segments, such that graphs in the same segment are strongly correlated with each other. The latter feature allows to identify *change points* in the time history of the system, i. e. short periods in which the dynamics produces big changes in the graph structure (corresponding to, e.g., extreme events). The minimization of $C$ is **NP**-hard, so the authors propose an approximation method called *GraphScope*, which consists of two steps: first, one looks for the best partition of each graph segment; second, one looks for the best division in segments. In both cases the "best" result corresponds to the minimal encoding cost. The best partition within a graph segment is found by local search. GraphScope has the big advantage not to require any input, like the number and sizes of the clusters. It is also suitable to operate in a streaming environment, in which new graph configurations are added in time, following the evolution of the system: the computational complexity required to process a snapshot (on average) is stable over time. Tests on real evolving data sets show that GraphScope is able to find meaningful communities and change points.

Since keeping track of communities in different time steps is not a trivial problem, as we have seen above, it is perhaps easier to adopt a vertex-centric perspective, in which one monitors the community of a given vertex at different times. For any method, given a vertex $i$ and a time $t$, the community to which $i$ belongs at time $t$ is well defined. Fenn et al. (Fenn *et al.*, 2008) used the multiresolution method by Reichardt et al. (Reichardt and Bornholdt, 2006a) (Section VI.B) and investigated a fully connected graph with time-dependent weights, representing the correlations of time series of hourly exchange rate returns. The resolution parameter $\gamma$ is fixed to the value that occurs in most stability plateaus of the system at different time steps. Motivated by the work of Guimerà and Amaral (Guimerà and Amaral, 2005) (Section XV.A), Fenn et al. identify the role of individual vertices in their community through the pair $(z^{in}, z^b)$, where $z^{in}$ is the $z$-score of the internal strength (weighted degree, Section A.1), defined in Eq. 94, and $z^b$ the $z$-score of the site betweenness, defined by replacing the internal degree with the site betweenness of Freeman (Freeman, 1977) in Eq. 94. We remind that the site betweenness is a measure of the number of shortest paths running through a vertex. The variable $z^b$ expresses the importance of a vertex in processes of information diffusion with respect to the other members of its community. Another important issue regards the *persistence* of communities in time, i. e. how stable they are during the evolution. As a measure of persistence, Fenn et al. introduced a vertex-centric version of the relative overlap of Eq. 96

$$a_i^t(\tau) = \frac{|\mathcal{C}_i(t) \bigcap \mathcal{C}_i(t+\tau)|}{|\mathcal{C}_i(t) \bigcup \mathcal{C}_i(t+\tau)|}, \tag{97}$$

where $i$ is the vertex and $\mathcal{C}_i(t)$, $\mathcal{C}_i(t+\tau)$ the communities of $i$ at times $t$, $t+\tau$, respectively. The decay of $a_i^t(\tau)$ depends on the type of vertex. In particular, if the vertex is strongly connected to its community ($z^{in}$ large), $a_i^t(\tau)$ decays quite slowly, meaning that it tends to stay attached to a stable core of vertices.

[20] We stress that here by modularity we mean the feature of a graph having community structure, not the modularity of Newman and Girvan.

## XVI. APPLICATIONS ON REAL-WORLD NETWORKS

The ultimate goal of clustering algorithms is trying to infer properties of and relationships between vertices, that are not available from direct observation/measurement. If the scientific community agrees on a set of reliable techniques, one could then proceed with careful investigations of systems in various domains. So far, most works in the literature on graph clustering focused on the development of new algorithms, and applications were limited to those few benchmark graphs that one typically uses for testing (Section XIV.A). Still, there are also applications aiming at understanding real systems. Some results have been actually mentioned in the previous sections. This section is supposed to give a flavor of what can be done by using clustering algorithms. Therefore, the list of works presented here is by no means exhaustive. Most studies focus on biological and social networks. We mention a few applications to other types of networks as well.

### A. Biological networks

The recent abundance of genomic data has allowed us to explore the cell at an unprecedented depth. A wealth of information is available on interactions involving proteins and genes, metabolic processes, etc. In order to study cellular systems, the graph representation is regularly used. Protein-protein interaction networks (PIN), gene regulatory networks (GRN) and metabolic networks (MN) are meanwhile standard objects of investigation in biology and bioinformatics (Junker and Schreiber, 2008).

Biological networks are characterized by a remarkable modular organization, reflecting functional associations between their components. For instance, proteins tend to be associated in two types of cellular modules: *protein complexes* and *functional modules*. A protein complex is a group of proteins that mutually interact at the same time and space, forming a sort of physical object. Examples are transcription factor complexes, protein transport and export complexes, etc. Functional modules instead are groups of proteins taking place in the same cellular process, even if the interactions may happen at different times and places. Examples are the CDK/cyclin

module, responsible for cell-cycle progression, the yeast pheromone response pathway, etc. Identifying cellular modules is fundamental to uncover the organization and dynamics of cell functions. However, the information on cell units (e. g. proteins, genes) and their interactions is often incomplete, or even incorrect, due to noise in the data produced by the experiments. Therefore, inferring modules from the topology of cellular networks enables one to restrict the set of possible scenarios and can be a safe guide for future experiments.

Rives and Galitski (Rives and Galitski, 2003) studied the modular organization of a subset of the PIN of the yeast (*Saccharomyces cerevisiae*), consisting of the (signaling) proteins involved in the processes leading the microorganism to a filamentous form. The clusters were detected with a hierarchical clustering technique. Proteins mostly interacting with members of their own cluster are often essential proteins; edges between modules are important points of communication. Spirin and Mirny (Spirin and Mirny, 2003) identified protein complexes and functional modules in yeast with different techniques: clique detection, superparamagnetic clustering (Blatt *et al.*, 1996) and optimization of cluster edge density. They estimated the statistical significance of the clusters by computing the $p$-values of seeing those clusters in random graphs with the same expected degree sequence as the original network. From the known functional annotations of yeast genes one can see that the modules usually group proteins with the same or consistent biological functions. Indeed, in many cases, the modules exactly coincide with known protein complexes. The results appear robust if noise is introduced in the system, to simulate the noise present in the experimental data. Functional modules in yeast were also found by Chen and Yuan (Chen and Yuan, 2006), who applied the algorithm by Girvan and Newman with a modified definition of edge betweenness (Section V.A). The standard Girvan-Newman algorithm has proved to be reliable to detect functional modules in PINs (Dunn *et al.*, 2005). The novelty of the work by Chen and Yuan is its focus on weighted PINs, where the weights come from information derived through microarray expression profiles. Weights add information about the system and should lead to a more reliable modular structure. By knocking out genes in the same structural cluster similar phenotypes appeared, suggesting that the genes have similar biological roles. Moreover, the clusters often contained known protein complexes, either entirely or to a large extent. Finally, Chen and Yuan were able to make predictions of the unknown function of some genes, based on the structural module they belong to: gene function prediction is the most promising outcome deriving from the application of clustering techniques to PINs. Farutin et al. (Farutin *et al.*, 2006) have adopted a local concept of community, and derived a hierarchical decomposition of PINs, in that the modules identified at some level become the vertices of a network at the higher level. Communities are overlapping, to account for the fact that proteins (and whole modules) may have diverse biological functions. High level structures detected in a human PIN correspond to general biological concepts like signal transduction, regulation of gene expression, intercellular communication. Sen et al. (Sen *et al.*, 2006) identified protein clusters for yeast from the eigenvectors of the Laplacian matrix (Section A.2), computed via Singular Value Decomposition.

Metabolic networks have also been extensively investigated. We have already discussed the "functional cartography" designed by Guimerà and Amaral (Guimerà and Amaral, 2005; Guimerà and Amaral, 2005), which applies to general types of networks, not necessarily metabolic. A hierarchical decomposition of metabolic networks has been derived by Holme et al. (Holme *et al.*, 2003), by using a hierarchical clustering technique inspired by the algorithm by Girvan and Newman (Section V.A). Here, vertices are removed based on their betweenness values, which are obtained by dividing the standard site betweenness scores (Freeman, 1977) by the indegree of the respective vertices. A picture of metabolic network emerges, in which there are core clusters centered at hub-substances, surrounded by outer shells of less connected substances, and a few other clusters at intermediate scales. In general, clusters at different scales seem to be meaningful, so the whole hierarchy should be taken into account.

Wilkinson and Huberman (Wilkinson and Huberman, 2004) analyzed a network of gene co-occurrence to find groups of related genes. The network is built by connecting pairs of genes that are mentioned together in the abstract of articles of the Medline database (`http://medline.cos.com/`). Clusters were found with a modified version of the algorithm by Girvan and Newman, in which edge betweenness is computed by considering the shortest paths of a small subset of all vertex pairs, to gain computer time (Section V.A). As a result, genes belonging to the same cluster turn out to be functionally related to each other. Co-occurrence of terms is also used to extract associations between genes and diseases, to find out which genes are relevant for a specific disease. Communities of genes related to colon cancer can be helpful to identify the function of the genes.

### B. Social networks

Networks depicting social interactions between people have been studied for decades (Scott, 2000; Wasserman and Faust, 1994). Recently the modern Information and Communication Technology (ICT) has opened new interaction modes between individuals, like mobile phone communications and online interactions enabled by the Internet. Such new social exchanges can be accurately monitored for very large systems, including millions of individuals, whose study represents a huge opportunity for social science. Communities of social networks can be friendship circles, or groups of people sharing com-

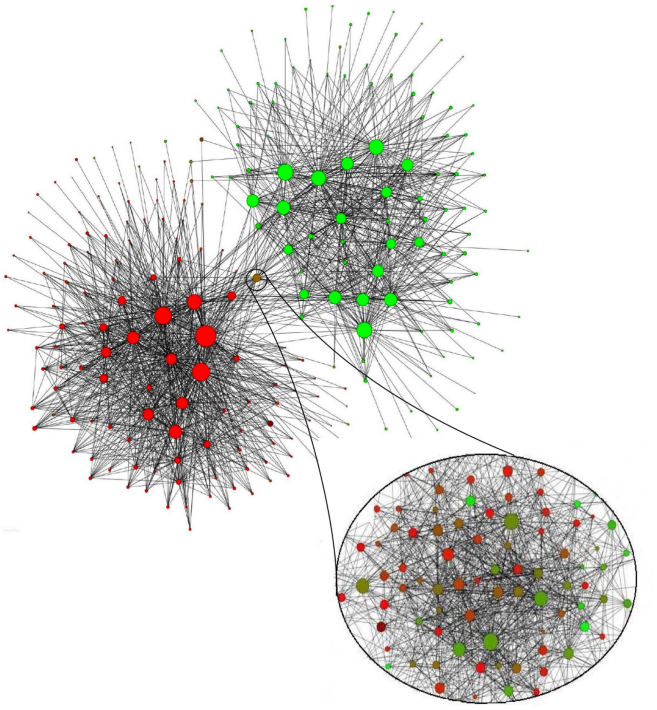

FIG. 35 Community structure of a social network of mobile phone communication in Belgium. Dots indicate subcommunities at the lower hierarchical level (with more than 100 people) and are colored in a red-green scale to represent the level of representation of the two main languages spoken in Belgium (red for French and green for Dutch). Communities of the two larger groups are linguistically homogeneous, with more than 85% of people speaking the same language. Only one community (zoomed), which lies at the border between the two main aggregations, has a more balanced distribution of languages. Reprinted figure with permission from (Blondel *et al.*, 2008). ©2008 by IOP Publishing and SISSA.

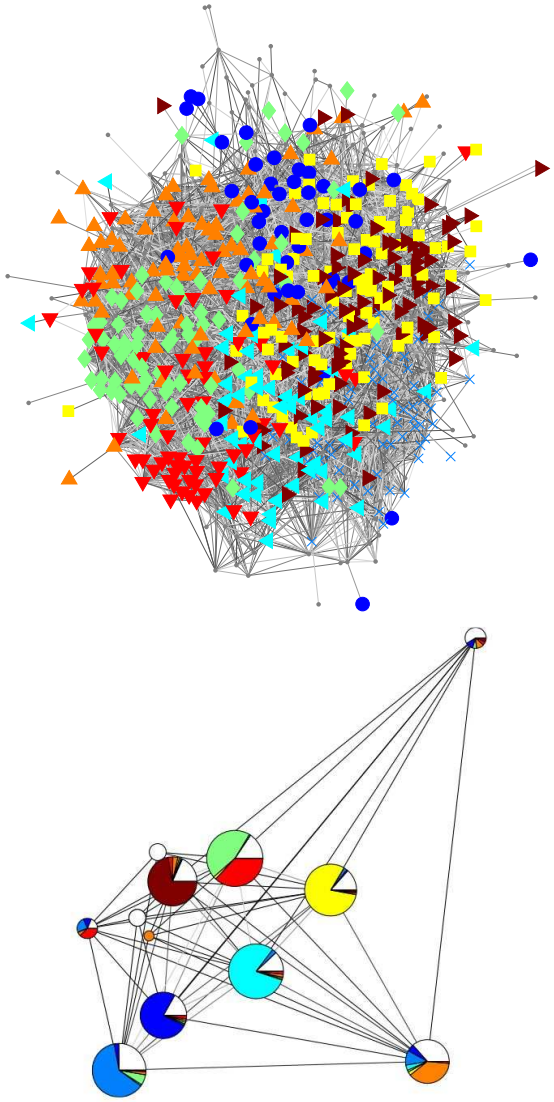

FIG. 36 Communities in social networking sites. (Top) Visualization of a network of friendships between students at Caltech, constructed from Facebook data (September 2005). The colors/shapes indicate the dormitories (Houses) of the students. (Bottom) Topological communities of the network, which are quite homogeneous with respect to House affiliation. Reprinted figures with permission from Refs. (Porter *et al.*, 2009) and (Traud *et al.*, 2008).

mon interests.

Blondel et al. have analyzed a network of mobile phone communications between users of a Belgian phone operator (Blondel *et al.*, 2008). The vertices of the graph are 2.6 million and the edges are weighted by the cumulative duration of phone calls between users in the observation time frame. The clustering analysis, performed with a fast hierarchical modularity optimization technique developed by the authors (discussed in Section VI.A.1), delivers six hierarchical levels. The highest level consists of 261 groups with more than 100 vertices, which are clearly arranged in two main groups, linguistically homogeneous, reflecting the linguistic split of Belgian population (Fig. 35). Tyler et al. (Tyler *et al.*, 2003) studied a network of e-mail exchanges between people working at HP Labs. They applied the same modified version of Girvan-Newman algorithm that two of the authors have used to find communities of related genes (Wilkinson and Huberman, 2004) (Section XVI.A). The method enables one to measure the degree of membership of each vertex in a community and allows for overlaps between communities. The detected clusters matched quite closely the organization of the Labs in departments and project groups, as confirmed by interviews conducted with researchers.

Social networking sites, like *Myspace* (`www.myspace.com`), *Friendster* (`www.friendster.com`), *Facebook* (`www.facebook.com`), etc. have become extremely popular in the last years. They are online platforms that allow people to communicate with friends, send e-mails, solicit opinions on specific issues, spread ideas and/or fads, etc. Traud et al. (Traud *et al.*, 2008)

used anonymous Facebook data to create networks of friendships between students of different American universities, where vertices/students are connected if they are friends on Facebook. Communities were detected by applying a variant of Newman's spectral optimization of modularity (Section VI.A.4): the results were further refined through additional steps á la Kernighan-Lin (Section IV.A). One of the goals of the study was to infer relationships between the online and offline lives of the students. By using demographic information on the students' populations, one finds that communities are organized by class year or by House (dormitory) affiliation, depending on the university (Fig. 36). Yuta et al. (Yuta *et al.*, 2007) observed a gap in the community size distribution of a friendship network extracted from the largest social networking site in Japan (as of December 2006), *mixi* (`mixi.jp`). Communities were identified with the fast greedy modularity optimization by Clauset et al. (Clauset *et al.*, 2004). The gap occurs in the intermediate range of sizes between 20 and 400, where but a few communities are observed. Yuta et al. introduced a model where people form new friendships both by "closing" ties with people who are friends of friends, and by setting new links with individuals having similar interests. In this way most groups turn out to be either small or large, and medium size groups are rare.

Collaboration networks, in which individuals are linked if they are (were) involved in a common activity, have been often studied because they embed an implicit objective concept of acquaintance, that is not easy to capture in direct social experiments/interviews. For instance, somebody may consider another individual a friend, while the latter may disagree. A collaboration instead is a proof of a social relationship between individuals. The analysis of the structure of scientific collaboration networks (Newman, 2001) has exerted a big influence on the development of the modern network science. Scientific collaboration is associated to coauthorship: two scientists are linked if they have coauthored at least one paper together. Information about coauthorships can be extracted from different databases of research papers. Communities indicate groups of people with common research interests, i. e. topical or disciplinary groups. In the seminal paper by Girvan and Newman (Girvan and Newman, 2002), the authors applied their method on a collaboration network of scientists working at the Santa Fe Institute, and were able to discriminate between research divisions (Fig. 2b). The community structure of scientific collaboration networks has been investigated by many authors (Danon *et al.*, 2006; Donetti and Muñoz, 2004; Duch and Arenas, 2005; Farkas *et al.*, 2007; Gregory, 2007; Lehmann and Hansen, 2007; Nepusz *et al.*, 2008; Newman, 2004b, 2006a; Noack and Rotta, 2008; Palla *et al.*, 2007, 2005; Pujol *et al.*, 2006; Radicchi *et al.*, 2004; Reichardt and Bornholdt, 2006a; Richardson *et al.*, 2008; S.-W. Son *et al.*, 2006; Shen *et al.*, 2009; Vragović and Louis, 2006; White and Smyth, 2005; Zhou, 2003b). Other types of collaboration networks have been studied too. Gleiser and Danon (Gleiser and Danon, 2003) considered a collaboration network of jazz musicians. Vertices are either musicians, connected if they played in the same band, or bands, connected if they have a musician in common. By applying the algorithm of Girvan and Newman they found that communities reflect both racial segregation (with two main groups comprising only black or white players) and geographical separation, due to the different recording locations.

### C. Other networks

Citation networks (de Solla Price, 1965) have been regularly used to understand the citation patterns of authors and to disclose relationships between disciplines. Rosvall and Bergstrom (Rosvall and Bergstrom, 2008) used a citation network of over 6000 scientific journals to derive a map of science. They used a clustering technique based on compressing the information on random walks taking place on the graph (Section IX.B). A random walk follows the flow of citations from one field to another, and the fields emerge naturally from the clustering analysis (Fig. 37). The structure of science resembles the letter **U**, with the social sciences and engineering at the terminals, joined through a chain including medicine, molecular biology, chemistry and physics.

Reichardt and Bornholdt (Reichardt and Bornholdt, 2007) performed a clustering analysis on a network built from bidding data taken from the German version of Ebay (`www.ebay.de`), the most popular online auction site. The vertices are bidders and two vertices are connected if the corresponding bidders have expressed interest for the same item. Clusters were detected with the multiresolution modularity optimization developed by the authors themselves (Reichardt and Bornholdt, 2006a) (Section VI.B). In spite of the variety of items that it is possible to purchase through Ebay, about 85% of bidders were classified into a few major clusters, reflecting bidders' broad categories of interests. Ebay data were also examined by Jin et al. (Jin *et al.*, 2007), who considered bidding networks where the vertices are the individual auctions and edges are placed between auctions having at least one common bidder. Communities, detected with greedy modularity optimization (Newman, 2004b) (Section VI.A.1), allow to identify substitute goods, i. e. products that have value for the same bidder, so that they can be purchased together or alternatively.

Legislative networks enable one to deduce associations between politicians through their parliamentary activity, which may be related or not to party affiliation. Porter and coworkers have carried out numerous studies on the subject (Porter *et al.*, 2007, 2005; Zhang *et al.*, 2008), by using data on the Congress of the United States. In Refs. (Porter *et al.*, 2007, 2005), they examined the community structure of networks of committees in the US House of Representatives. Committees sharing common members are connected by edges, which are weighted by

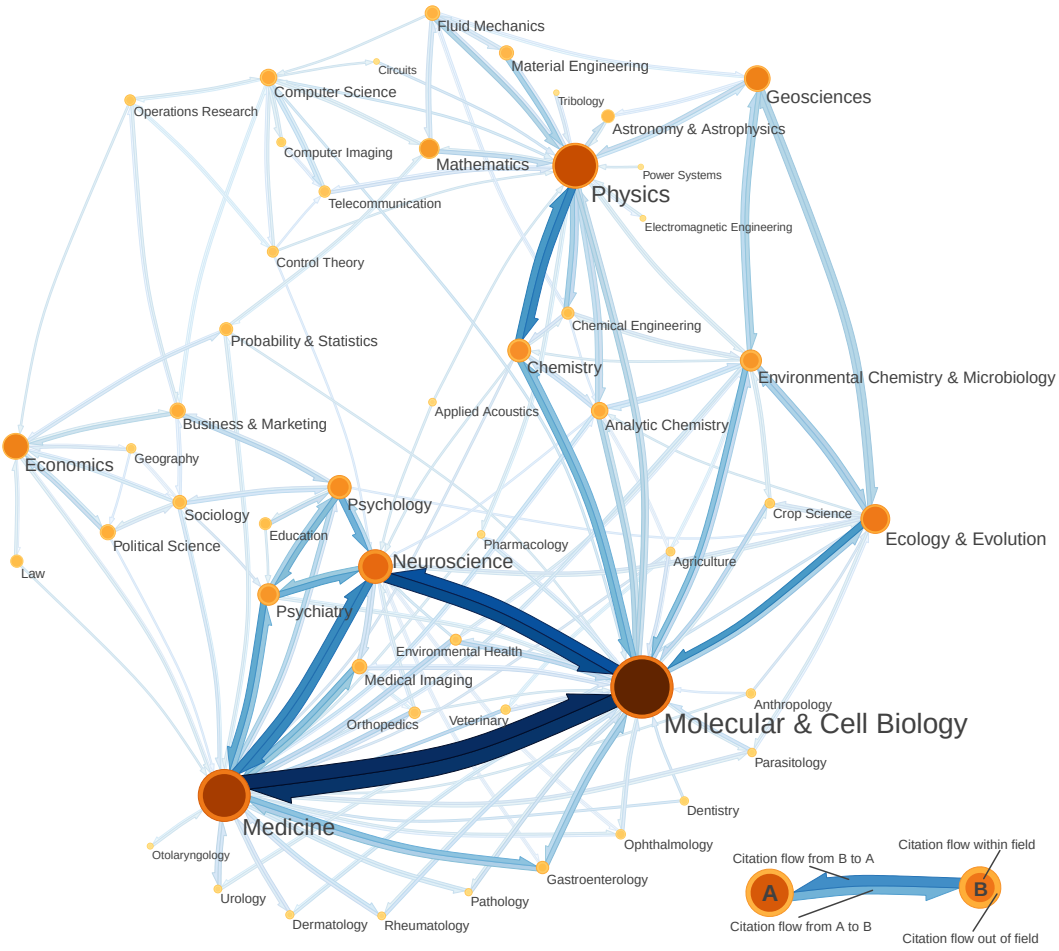


FIG. 37 Map of science derived from a clustering analysis of a citation network comprising more than 6000 journals. Reprinted figure with permission from (Rosvall and Bergstrom, 2008). ©2008 by the National Academy of Science of the USA.

dividing the number of common members by the number one would expect to have if committee memberships were randomly assigned. Hierarchical clustering (Section IV.B) reveals close connections between some of the committees. In another work (Zhang *et al.*, 2008), Zhang et al. analyzed networks of legislation cosponsorship, in which vertices are legislators and two legislators are linked if they support at least one common bill. Communities, identified with a modification of Newman's spectral optimization of modularity (Section VI.A.4), are correlated with party affiliation, but also with geography and committee memberships of the legislators.

Networks of correlations between time series of stock returns have received a growing attention in the past few years (Mantegna, 1999). In early studies, scholars found clusters of correlated stocks by computing the *maximum spanning tree* of the network (Bonanno *et al.*, 2003, 2000; Onnela *et al.*, 2003, 2002) (Section A.1), and realized that such clusters match quite well the economic sectors of the stocks. More recently, the community structure of the networks has been investigated by means of proper clustering algorithms. Farkas et al. (Farkas *et al.*, 2007) have applied the weighted version of the Clique Percolation Method (Section XI.A) and found that the presence of two strong (i. e. carrying high correlation) edges in triangles is usually accompanied by the presence of a strong third edge. Heimo et al. (Heimo *et al.*, 2008) used the weighted version of the multiresolution method by Reichardt and Bornholdt (Reichardt and Bornholdt, 2006a) (Section VI.B). Clusters correspond to relevant business sectors, as indicated by Forbes classification; moreover, smaller clusters at lower hierarchical levels seem to correspond to (economically) meaningful substructures of the main clusters.

## XVII. OUTLOOK

Despite the remote origins and the great popularity of the last years, research on graph clustering has not yet given a satisfactory solution of the problem and leaves us with a number of important open issues. From our exposition it appears that the field has grown in a rather chaotic way, without a precise direction or guidelines. In

some cases, interesting new ideas and tools have been presented, in others existing methods have been improved, becoming more accurate and/or faster.

What the field lacks the most is a theoretical framework that defines precisely what clustering algorithms are supposed to do. Everybody has his/her own idea of what a community is, and most ideas are consistent with each other, but, as long as there is still disagreement, it remains impossible to decide which algorithm does the best job and there will be no control on the creation of new methods. Therefore, we believe that the first and foremost task that the scientific community working on graph clustering has to solve in the future is defining a set of reliable benchmark graphs, against which algorithms should be tested (Section XIV.A). Defining a benchmark goes far beyond the issue of testing. It means designing practical examples of graphs with communities, and, in order to do that, one has to agree on the fundamental concepts of community and partition. Clustering algorithms have to be devised consistently with such definitions, in order to give the best performance on the set of designated benchmarks, which represent a sort of ground truth. The explosion in the number of algorithms we have witnessed in recent times is due precisely to the present lack of reliable mechanisms of control of their quality and comparison of their performances. If the community agrees on a benchmark, the future development of the field will be more coherent and the progress brought by new methods can be evaluated in an unbiased manner. The planted $\ell$-partition model (Condon and Karp, 2001) is the easiest recipe one can think of when it comes to defining clusters, and is the criterion underlying well-known benchmarks, like that by Girvan and Newman. We believe that the new benchmarks have to be defined along the same lines. The benchmark graphs recently introduced by Lancichinetti et al. (Lancichinetti and Fortunato, 2009; Lancichinetti *et al.*, 2008) and by Sawardecker et al. (Sawardecker *et al.*, 2009) are an important step in this direction.

Defining a benchmark implies specifying the "natural" partition of a graph, the one that any algorithm should find. This issue in turn involves the concept of quality of a partition, that has characterized large part of the development of the field, in particular after the introduction of Newman-Girvan modularity (Section III.C.2). Estimating the quality of a partition allows to discriminate among the large number of partitions of a graph. In some cases this is not difficult. For instance, in the benchmark by Girvan and Newman there is a single meaningful partition, and it is hard to argue with that. But most graphs of the real world have a hierarchical structure, with communities including smaller communities and so on. Hence there are several meaningful partitions, corresponding to different hierarchical levels, and discriminating among them is hard, as they may be all relevant, in a sense. If we consider the human body, we cannot say that the organization in tissues of the cells is more important than the organization in organs. We have seen that there are recent methods dealing with the problem of finding meaningful hierarchical levels (Section XII). Such methods rank the hierarchical partitions based on some criterion and one can assess their relevance through the ranking. One may wonder whether it makes sense sorting out levels, which means introducing a kind of threshold on the quality index chosen to rank partitions (to distinguish "good" from "bad" partitions), or whether it is more appropriate to keep the information given by the whole set of hierarchical partitions. The work by Clauset et al. on hierarchical random graphs (Clauset *et al.*, 2007; Clauset *et al.*, 2008), discussed in Section XII.B, indirectly raises this issue. There it was shown that the ensemble of model graphs, represented by dendrograms, encodes most of the information on the structure of the graph at study, like its degree distribution, transitivity and distribution of shortest path lengths. At the same time, by construction, the model reveals the whole hierarchy of communities, without any distinction between good and bad partitions. The information given by a dendrogram may become redundant and confusing when the graph is large, as then there is a big number of partitions. This is actually the reason why quality functions were originally introduced. However, in that case, one was dealing with artificial hierarchies, produced by techniques that systematically yield a dendrogram as a result of the analysis (like, e.g. hierarchical clustering), regardless of whether the graph actually has a hierarchical structure or not. Here instead we speak of real hierarchy, which is a fundamental element of real graphs and, as such, it must be considered in any serious approach to graph clustering. Any good clustering method must be able to tell whether a graph has community structure or not, and, in the first case, whether the community structure is hierarchical (i. e. with two or more levels) or flat (one level). We expect that the concept of hierarchy will become a key ingredient of future clustering techniques. In particular, assessing the consistence of the concepts of partitions' quality and hierarchical structure is a major challenge.

A precise definition of null models, i. e. of graphs without community structure, is also missing. This aspect is extremely important, though, as defining communities also implies deciding whether or not they exist in a specific graph. At the moment, it is generally accepted that random graphs have no communities. The null model of modularity (Section III.C.2), by far the most popular, comprises all graphs with the same expected degree sequence of the original graph and random rewiring of edges. This class of graphs is characterized, by construction, by the fact that any vertex can be linked to any other, as long as the constraint on the degree sequence is satisfied. But this is by no means the only possibility. A community can be generically defined as a subgraph whose vertices have a higher probability to be connected to the other vertices of the subgraph than to external vertices. The planted $\ell$-partition model (Condon and Karp, 2001) is based on this principle, as we have seen.

However, this does not mean that the linking probabilities of a vertex with respect to the other vertices in its community or in different communities be constant (or simply proportional to their degrees, as in the configuration model (Łuczak, 1992; Molloy and Reed, 1995)). In fact, in large networks it is reasonable to assume that the probability that a vertex is linked to most vertices is zero, as the vertex "ignores" their existence. This does not exclude that the probability that the vertex gets connected to the "known" vertices is the same (or proportional to their degrees), in which case the graph would still be random and have no communities. We believe that we are still far from a precise definition and a complete classification of null models. This represents an important research line for the future of the field, for three main reasons: 1) to better disentangle "true" communities from byproducts of random fluctuations; 2) to pose a stringent test to existing and future clustering algorithms, whose reliability would be questionable if they found "false positives" in null model graphs; 3) to handle "hybrid" scenarios, where a graph displays community structure on some portions of it, while the rest is essentially random and has no communities.

In the previous chapters we have seen a great number of clustering techniques. Which one(s) shall we use? At the moment the scientific community is unable to tell. Modularity optimization is probably the most popular method, but the results of the analysis of large graphs are likely to be unreliable (Section VI.C). Nevertheless, people have become accustomed to use it, and there have been several attempts to improve the measure. A newcomer, who wishes to find clusters in a given network and is not familiar with clustering techniques, would not know, off-hand, which method to use, and he/she would hardly find indications about good methods in any single paper on graph clustering, except perhaps on the method presented in the paper. So, people keep using algorithms because they have heard of them, or because they know that other people are using them, or because of the reputation of the scientists who designed them. Waiting for future reliable benchmarks, that may give an objective assessment of the quality of the algorithms, there are at the moment hardly solid reasons to prefer an algorithm to another. However, we want to stress that there is no such thing as the perfect method, so it is pointless to look for it. Among the other things, if one tries to look for a very general method, that should give good results on any type of graph, one is inevitably forced to make very general assumptions on the structure of the graph and on the properties of communities. In this way one neglects a lot of specific features of the system, that may lead to a more accurate detection of the clusters. Informing a method with features characterizing some types of graphs makes it far more reliable to detect the community structure of those graphs than a general method, even if its applicability may be limited. Therefore in the future we envision the development of domain-specific clustering techniques. The challenge here is to identify the peculiar features of classes of graphs, which are bound to become crucial ingredients in the design of suitable algorithms. Some of the methods available today are actually based on assumptions that hold only for some specific categories of graphs. The Clique Percolation Method by Palla et al. (Palla *et al.*, 2005), for instance, may work well for graphs characterized by a large number of cliques, like certain social networks, whereas it may give poor results otherwise.

Moving one step further, one should learn how to use specific information about a graph, whenever available, e. g. properties of vertices and/or partial information about their classification. For instance, it may be that one has some information on a subset of vertices, like demographic data on people of a social network, and such data may highlight relationships between people that are not obvious from the network of social interactions. In this case, using only the social network may be reductive and ideally one should exploit both the structural and the non-structural information in the search of clusters, as the latter should be consistent with both inputs. How to do this is an open problem.

Most algorithms in the literature deal with the "classical" case of a graph with undirected and unweighted edges. This is certainly the simplest case one could think of, and graph clustering is already a complex task on such types of graphs. We know that real networks may be directed, have weighted connections, be bipartite. Methods to deal with such systems have been developed, as we have seen, especially in the most recent literature, but they are mostly preliminary attempts and there is room for improvement. Another situation that may occur in real systems is the presence of edges with positive and negative weights, indicating attractive and repulsive interactions, respectively. This is the case, for instance, of correlation data (Mantegna, 1999). In this case, ideal partitions would have positively weighted intracluster edges and negatively weighted intercluster edges. We have discussed some studies in this direction (Gómez *et al.*, 2008; Kaplan and Forrest, 2008; Traag and Bruggeman, 2008), but we are just at the beginning of this endeavour. Instead, there are no algorithms yet which are capable to deal with graphs in which there are edges of several types, indicating different kinds of interactions between the vertices. Agents of social networks, for instance, may be joined by working relationships, friendship, family ties, etc. At the moment there does not seem to exist a better way of proceeding other than keeping edges of one type and forgetting the others, repeating the analysis for each type of edges and eventually comparing the results obtained. Finally, since most real networks are built through the results of experiments, which carry errors in their estimates, it would be useful to consider as well the case in which edges have not only associated weights, but also errors on their values.

Since the paper by Palla et al. (Palla *et al.*, 2005), overlapping communities have received a lot of attention (Section XI). However, there is still no consensus about

a quantitative definition of the concept of overlapping community, and most definitions depend on the method adopted. Intuitively, one would expect that clusters share vertices lying at their borders, and this idea has inspired most algorithms. However, clusters detected with the Clique Percolation Method (Section XI.A) often share central vertices of the clusters, which makes sense in specific instances, especially in social networks. So, it is still unclear how to characterize overlapping vertices. Moreover, the concept of overlapping clusters seems at odds with that of hierarchical structure. No dendrogram can be drawn if there are overlapping vertices, at least in the standard way. Due to the relevance of both features in real networks, it is necessary to adapt them to each other in a consistent way. Overlapping vertices pose problems as well when it comes to comparing the results of different methods on the same graph. Most similarity measures are defined only in the case of partitions, where each vertex is assigned to a single cluster (Section XIV.B). It is then necessary to extend such definitions to the case of overlapping communities, whenever possible.

Another issue that is getting increasingly more popular is the study of graphs evolving in time. This is now possible due to the availability of timestamped network data sets. Tracking the evolution of community structure in time is very important, to uncover how communities are generated and how they interact with each other. Scholars have just begun to study this problem (Fenn *et al.*, 2008; Hopcroft *et al.*, 2004; Palla *et al.*, 2007) (Section XV.B). Typically one analyzes snapshots at different times and checks what happened at time $t+1$ to the communities at time $t$. It would be probably better to use simultaneously the whole dynamic data set, and future work shall aim at defining proper ways to do that.

The computational complexity of graph clustering algorithms has improved by at least one power in the graph size (on average) in just a couple of years. Due to the large size of many systems one wishes to investigate, the ultimate goal would be to design techniques with linear or even sublinear complexity. Nowadays partitions in graphs with up to millions of vertices can be found. However, the results are not yet very reliable, as they are usually obtained by greedy optimizations, which yield rough approximations of the desired solution. In this respect the situation could improve by focusing on the development of efficient local methods, for two reasons: 1) they enable analyses of portions of the graph, independently of the rest; 2) they are often suitable for parallel implementations, which may speed up considerably the computation.

Finally, if there has been a tremendous effort in the design of clustering algorithms, basically nothing has been done to make sense of their results. What shall we do with communities? What can they tell us about a system? The hope is that they will enable one to disclose "hidden" relationships between vertices, due to features that are not known, because they are hard to measure, for instance. It is quite possible that the scientific community will converge sooner or later to a definition *a posteriori* of community. Already now, most algorithms yield similar results in practical applications. But what is the relationship between the vertex classification given by the algorithms and real classifications? This is the main question beneath the whole endeavor.


### Acknowledgments

I am indebted to these people for giving useful suggestions and advice to improve this manuscript at various stages: A. Arenas, A. Clauset, S. Gómez, R. Guimerà, R. Lambiotte, A. Lancichinetti, J.-P. Onnela, G. Palla, M. A. Porter, F. Radicchi, J. J. Ramasco, C. Wiggins.


## APPENDIX A: Elements of Graph Theory

### 1. Basic Definitions

A *graph* $\mathcal{G}$ is a pair of sets $(V, E)$, where $V$ is a set of *vertices* or *nodes* and $E$ is a subset of $V^2$, the set of unordered pairs of elements of $V$. The elements of $E$ are called *edges* or *links*, the two vertices that identify an edge are called *endpoints*. An edge is *adjacent* to each of its endpoints. If each edge is an ordered pair of vertices one has a *directed* graph (or *digraph*). In this case an ordered pair $(v, w)$ is an edge directed from $v$ to $w$, or an edge beginning at $v$ and ending at $w$. A graph is visualized as a set of points connected by lines, as shown in Fig. 38. In many real examples, graphs are *weighted*, i.e. a real number is associated to each of the edges. Graphs do not include *loops*, i.e. edges connecting a vertex to itself, nor multiple edges, i.e. several edges joining the same pair of vertices. Graphs with loops and multiple edges are called *multigraphs*. Generalizations of graphs admitting edges between any number of vertices (not necessarily two) are called *hypergraphs*.

A graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is a *subgraph* of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $V' \subset V$ and $E' \subset E$. If $\mathcal{G}'$ contains all edges of $\mathcal{G}$ that join vertices of $V'$ one says that the subgraph $\mathcal{G}'$ is induced or spanned by $V'$. A partition of the vertex set $V$ in two subsets $S$ and $V - S$ is called a *cut*; the *cut size* is the number of edges of $\mathcal{G}$ joining vertices of $S$ with vertices of $V - S$.

We indicate the number of vertices and edges of a graph with $n$ and $m$, respectively. The number of vertices is the *order* of the graph, the number of edges its *size*. The maximum size of a graph equals the total number of unordered pairs of vertices, $n(n-1)/2$. If $|V| = n$ and $|E| = m = n(n-1)/2$, the graph is a *clique* (or *complete graph*), and is indicated as $K_n$. Two vertices are *neighbors* (or *adjacent*) if they are connected by an edge. The set of neighbors of a vertex $v$ is called *neighborhood*, and we shall denote it with $\Gamma(v)$. The *degree* $k_v$ of a vertex $v$ is the number of its neighbors. The *degree sequence* is the list of the degrees of the graph vertices,

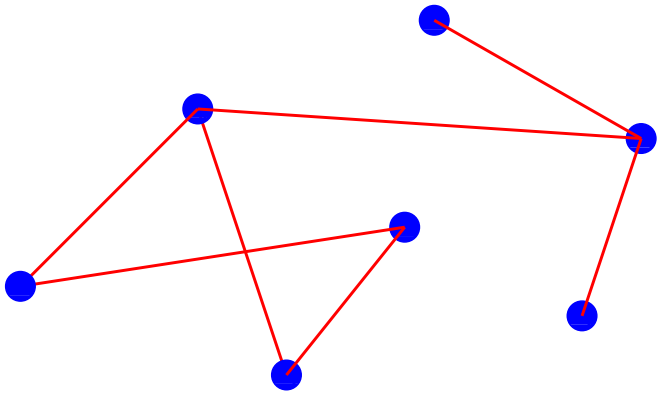

FIG. 38 A sample graph with seven vertices and seven edges.

$k_{v_1}, k_{v_2}, ..., k_{v-n}$. On directed graphs, one distinguishes two types of degree for a vertex $v$: the *indegree*, i.e. the number of edges beginning at $v$ and the *outdegree*, i.e. the number of edges ending at $v$. The analogue of degree on a weighted graph is the *strength*, i.e. the sum of the weights of the edges adjacent to the vertex. Another useful local property of graphs is *transitivity* or *clustering* (Watts and Strogatz, 1998), which indicates the level of cohesion between the neighbors of a vertex [21]. The clustering coefficient $c_v$ of vertex $v$ is the ratio between the number of edges joining pairs of neighbors of $v$ and the total number of possible edges, given by $k_v(k_v-1)/2$, $k_v$ being the degree of $v$. According to this definition, $c_v$ measures the probability that a pair of neighbors of $v$ are connected. Since all neighbors of $v$ are connected to $v$ by definition, edges connecting pairs of neighbors of $v$ form triangles with $v$. This is why the definition is often given in terms of number of triangles.

A *path* is a graph $\mathcal{P} = (V(\mathcal{P}), E(\mathcal{P}))$, with $V(\mathcal{P}) = \{x_0, x_1, ..., x_l\}$ and $E(\mathcal{P}) = \{x_0x_1, x_1x_2, ..., x_{l-1}x_l\}$. The vertices $x_0$ and $x_l$ are the *endvertices* of $\mathcal{P}$, whereas $l$ is its *length*. Given the notions of vertices, edges and paths, one can define the concept of *independence*. A set of vertices (or edges) of a graph are independent if no two elements of them are adjacent. Similarly, two paths are independent if they only share the endvertices. A *cycle* is a closed path whose vertices and edges are all distinct. Cycles of length $l$ are indicated with $C_l$. The smallest non-trivial cycle is the *triangle*, $C_3$.

Paths allow to define the concept of connectivity and distance in graphs. A graph is *connected* if, given any pair of vertices, there is at least one path going from one vertex to the other. In general, there may be multiple paths connecting two vertices, with different lengths. A *shortest path*, or *geodesic*, between two vertices of a graph, is a path of minimal length. Such minimal length is the *distance* between the two vertices. The *diameter* of a connected graph is the maximal distance between two vertices. If there is no path between two vertices, the graph is divided in at least two connected subgraphs. Each maximal connected subgraph of a graph is called *connected component*.

A graph without cycles is a *forest*. A connected forest is a *tree*. Trees are very important in graph theory and deserve some attention. In a tree, there can be only one path from a vertex to any other. In fact, if there were at least two paths between the same pair of vertices they would form a cycle, while the tree is an acyclic graph by definition. Further, the number of edges of a tree with $n$ vertices is $n-1$. If any edge of a tree is removed, it would get disconnected in two parts; if a new edge is added, there would be at least one cycle. This is why a tree is a minimally connected, maximally acyclic graph of a given order. Every connected graph contains a *spanning tree*, i.e. a tree sharing all vertices of the graph. On weighted graphs, one can define a *minimum* (*maximum*) *spanning tree*, i.e. a spanning tree such that the sum of the weights on the edges is minimal (maximal). Minimum and maximum spanning trees are often used in graph optimization problems, including clustering.

A graph $\mathcal{G}$ is *bipartite* if the vertex set $V$ is separated in two disjoint subsets $V_1$ and $V_2$, or *classes*, and every edge joins a vertex of $V_1$ with a vertex of $V_2$. The definition can be extended to that of *r-partition*, where the vertex classes are $r$ and no edge joins vertices within the same class. In this case one speaks of *multipartite* graphs.

### 2. Graph Matrices

The whole information about the topology of a graph of order $n$ is entailed in the *adjacency matrix* $\mathbf{A}$, which is an $n \times n$ matrix whose element $A_{ij}$ equals 1 if there is an edge joining vertices $i$ and $j$, otherwise it is zero. Due to the absence of loops the diagonal elements of the adjacency matrix are all zero. For an undirected graph $\mathbf{A}$ is a symmetric matrix. The sum of the elements of the $i$-th row or column yields the degree of node $i$. If the edges are weighted, one defines the *weight matrix* $\mathbf{W}$, whose element $W_{ij}$ expresses the weight of the edge between vertices $i$ and $j$.

The *spectrum* of a graph $\mathcal{G}$ is the set of eigenvalues of its adjacency matrix $\mathbf{A}$. Spectral properties of graph matrices play an important role in the study of graphs. For instance, the *stochastic matrices* rule the process of diffusion (random walk) on a graph. The *right stochastic matrix* $\mathbf{R}$ is obtained from $\mathbf{A}$ by dividing the elements of each row $i$ by the degree of vertex $i$. The *left stochastic matrix* $\mathbf{T}$, or *transfer matrix*, is the transpose of $\mathbf{R}$. The spectra of stochastic matrices allow to evaluate, for instance, the mixing time of the random walk, i.e. the

[21] The term clustering is commonly adopted to indicate community detection in some disciplines, like computer science, and we shall often use it in this context throughout the manuscript. We shall pay attention to disambiguate the occurrences in which clustering indicates instead the local property of a vertex neighborhood described here.

time it takes to reach the stationary distribution of the process. The latter is obtained by computing the eigenvector of the transfer matrix corresponding to the largest eigenvalue.

Another important matrix is the *Laplacian* $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix whose element $D_{ii}$ equals the degree of vertex $i$. The Laplacian is one of the most studied matrices and finds application in many different contexts, like graph connectivity (Bollobas, 1998), synchronization (Barahona and Pecora, 2002; Nishikawa *et al.*, 2003), diffusion (Chung, 1997) and graph partitioning (Pothen, 1997). By construction, the sum of the elements of each row of the Laplacian is zero. This implies that $\mathbf{L}$ always has at least one zero eigenvalue, corresponding to the eigenvector with all equal components, such as $(1, 1, ..., 1)$. Eigenvectors corresponding to different eigenvalues are all orthogonal to each other, so the sum of the elements of all eigenvectors but the trivial one(s) must be zero. In fact, the scalar product of the trivial eigenvector with equal components by any eigenvector yields just the sum of its elements. Interestingly, $\mathbf{L}$ has as many zero eigenvalues as there are connected components in the graph. So, the Laplacian of a connected graph has but one zero eigenvalue, all others being positive. The eigenvector corresponding to the second smallest eigenvalue is called *Fiedler vector* (Fiedler, 1973, 1975) and is regularly used in graph partitioning, as described in Section IV.A.

### 3. Model graphs

In this section we present the most popular models of graphs introduced to describe real systems, at least to some extent. Such graphs are useful null models in community detection, as they do not have community structure, so they can be used for negative tests of clustering algorithms.

The oldest model is that of *random graph*, proposed by Solomonoff and Rapoport (Solomonoff and Rapoport, 1951) and independently by Erdös and Rényi (Erdös and Rényi, 1959). There are two parameters: the number of vertices $n$ and the connection probability $p$. Each pair of vertices is connected with equal probability $p$ independently of the other pairs. The expected number of edges of the graph is $pn(n-1)/2$, and the expected mean degree $\langle k \rangle = p(n-1)$. The degree distribution of the vertices of a random graph is binomial, and in the limit $n \to \infty$, $p \to 0$ for fixed $\langle k \rangle$ it converges to a Poissonian. Therefore, the vertices have all about the same degree, close to $\langle k \rangle$ (Fig. 39, top). The most striking property of this class of graphs is the phase transition observed by varying $\langle k \rangle$ in the limit $n \to \infty$. For $\langle k \rangle < 1$, the graph is separated in connected components, each of them being microscopic, i.e. occupying but a vanishing portion of the system size. For $\langle k \rangle > 1$, instead, one of the components becomes macroscopic (*giant component*), i.e. it occupies a finite fraction of the graph vertices.

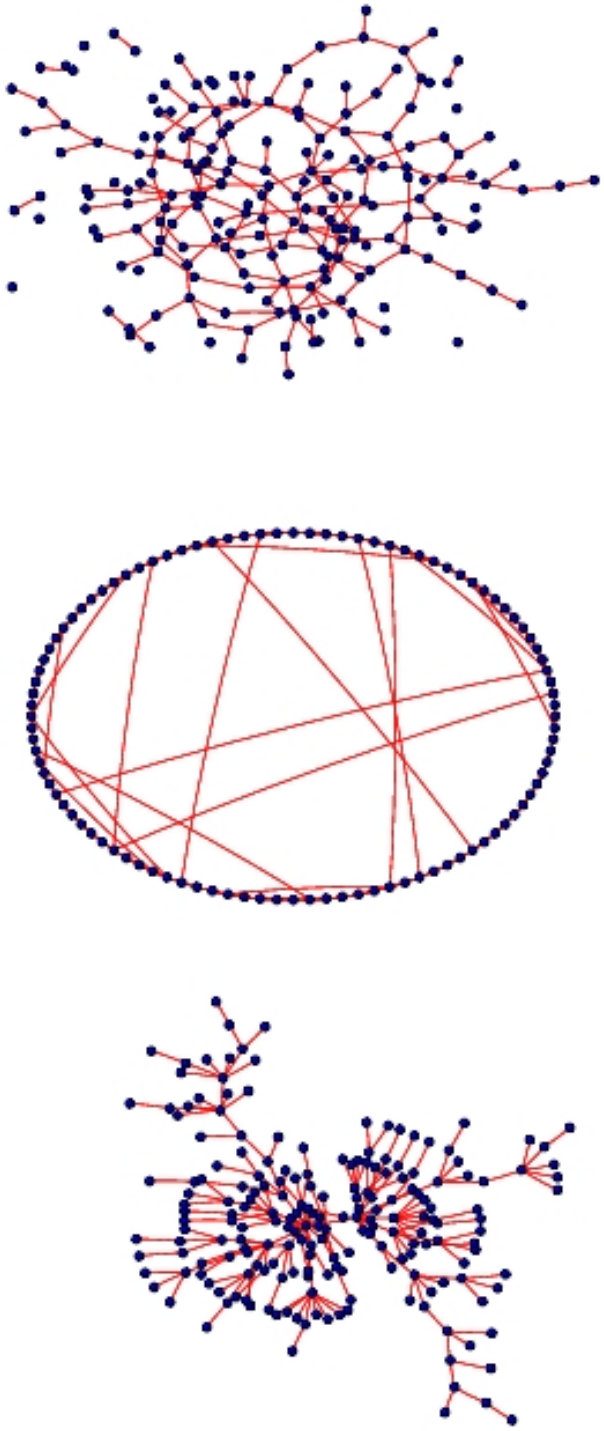

FIG. 39 Basic models of complex networks. (Top) Erdös-Rényi random graph with 100 vertices and a link probability $p = 0.02$. (Center) Small world graph á la Watts-Strogatz, with 100 vertices and a rewiring probability $p = 0.1$. (Bottom) Barabási-Albert scale-free network, with 100 vertices and an average degree of 2. Courtesy by J. J. Ramasco.

The diameter of a random graph with $n$ vertices is very small, growing only logarithmically with $n$. This property (*small-world effect*) is very common in many real graphs. The first evidence that social networks are characterized by paths of small length was provided by a series of famous experiments conducted by the phychologist Stanley Milgram (Milgram, 1967; Travers and Milgram, 1969). The expected clustering coefficient of a vertex of a random graph is $p$, as the probability for two vertices to be connected is the same whether they are neighbors of the same vertex or not. Real graphs, however, are characterized by far higher values of the clustering coefficient as compared to random graphs of the same size. Watts and Strogatz (Watts and Strogatz, 1998) showed that the small world property and high clustering coefficient can coexist in the same system. They designed a class of graphs which result from an interpolation between a regular lattice, which has high clustering coefficient, and a random graph, which has the small-world property. One starts from a ring lattice in which each vertex has degree $k$, and with a probability $p$ each edge is rewired to a different target vertex (Fig. 39, center). It turns out that low values of $p$ suffice to reduce considerably the length of shortest paths between vertices, because rewired edges act as shortcuts between initially remote regions of the graph. On the other hand, the clustering coefficient remains high, since few rewired edges do not perturb appreciably the local structure of the graph, which remains similar to the original ring lattice. For $p = 1$ all edges are rewired and the resulting structure is a random graph á la Erdös and Rényi.

The seminal paper of Watts and Strogatz triggered a huge interest towards the graph representation of real systems. One of the most important discoveries was that the distribution of the vertex degree of real graphs is very heterogeneous (Albert *et al.*, 1999), with many vertices having few neighbors coexisting with some vertices with many neighbors. In several cases the tail of this distribution can be described as a power law with good approximation[22], hence the expression *scale-free networks*. Such degree heterogeneity is responsible for a number of remarkable features of real networks, such as resilience to random failures/attacks (Albert *et al.*, 2000), and the absence of a threshold for percolation (Cohen *et al.*, 2000) and epidemic spreading (Pastor-Satorras and Vespignani, 2001). The most popular model of a graph with a power law degree distribution is the model by Barabási and Albert (Barabási and Albert, 1999). A version of the model for directed graphs had been proposed much earlier by de Solla Price (Price, 1976), building up on previous ideas developed by Simon (Simon, 1955). The graph is created with a dynamic procedure, where vertices are added one by one to an initial core. The probability for a new vertex to set an edge with a preexisting vertex is proportional to the degree of the latter. In this way, vertices with high degree have large probability of being selected as neighbors by new vertices; if this happens, their degree 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. 39 (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 (Albert and Barabási, 2002; Barrat *et al.*, 2008; Boccaletti *et al.*, 2006; Mendes and Dorogovtsev, 2003; Newman, 2003; Pastor-Satorras and Vespignani, 2004)).

[22] The power law is however not necessary to explain the properties of complex networks. It is enough that the tails of the degree distribution are "fat", i. e. spanning orders of magnitude in degree. They may or may not be accurately fitted by a power law.

## References


Adamcsek, B., G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek, 2006, Bioinformatics **22**(8), 1021.

Adomavicius, G., and A. Tuzhilin, 2005, IEEE Trans. Knowl. Data Eng. **17**(6), 734.

Agarwal, G., and D. Kempe, 2008, Eur. Phys. J. B **66**, 409.

Ahn, Y.-Y., J. P. Bagrow, and S. Lehmann, 2009, eprint arXiv:0903.3178.

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin, 1993, *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, USA).

Akaike, H., 1974, IEEE Trans. Autom. Control **19**(6), 716.

Alba, R. D., 1973, J. Math. Sociol. **3**, 113.

Albert, R., and A.-L. Barabási, 2002, Rev. Mod. Phys. **74**(1), 47.

Albert, R., H. Jeong, and A.-L. Barabási, 1999, Nature **401**, 130.

Albert, R., H. Jeong, and A.-L. Barabási, 2000, Nature **406**, 378.

Alves, N. A., 2007, Phys. Rev. E **76**(3), 036101.

Anthonisse, J. M., 1971, *The rush in a directed graph*, Technical Report, Stichting Mathematisch Centrum, 2e Boerhaavestraat 49 Amsterdam, The Netherlands.

Arenas, A., and A. Díaz-Guilera, 2007, Eur. Phys. J. Special Topics **143**, 19.

Arenas, A., A. Díaz-Guilera, and C. J. Pérez-Vicente, 2006, Phys. Rev. Lett. **96**(11), 114102.

Arenas, A., J. Duch, A. Fernández, and S. Gómez, 2007, New J. Phys. **9**, 176.

Arenas, A., A. Fernández, S. Fortunato, and S. Gómez, 2008a, J. Phys. A **41**(22), 224001.

Arenas, A., A. Fernández, and S. Gómez, 2008b, New J. Phys. **10**(5), 053039.

Asahiro, Y., R. Hassin, and K. Iwama, 2002, Discrete Appl. Math. **121**(1-3), 15.
Baeza-Yates, R., and B. Ribeiro-Neto, 1999, *Modern Information Retrieval* (Addison Wesley, Boston, USA).
Bagrow, J. P., 2008, J. Stat. Mech. **P05001**.
Bagrow, J. P., and E. M. Bollt, 2005, Phys. Rev. E **72**(4), 046108.
Balakrishnan, V. K., 1997, *Schaum's Outline of Graph Theory* (McGraw-Hill, New York, USA).
Bansal, N., A. Blum, and S. Chawla, 2004, Mach. Learn. **56**(1-3), 89.
Barabási, A.-L., and R. Albert, 1999, Science **286**, 509.
Barahona, M., and L. M. Pecora, 2002, Phys. Rev. Lett. **89**(5), 054101.
Barber, M. J., 2007, Phys. Rev. E **76**(6), 066102.
Barber, M. J., M. Faria, L. Streit, and O. Strogan, 2008, in *Stochastic and Quantum Dynamics of Biomolecular Systems*, edited by C. C. Bernido and M. V. Carpio-Bernido (American Institute of Physics, Melville, USA), volume 1021 of *American Institute of Physics Conference Series*, pp. 171–182.
Barnes, E. R., 1982, SIAM J. Alg. Discr. Meth. **3**, 541.
Barrat, A., M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, 2004, Proc. Natl. Acad. Sci. USA **101**(11), 3747.
Barrat, A., M. Barthelemy, and A. Vespignani, 2008, *Dynamical processes on complex networks* (Cambridge University Press, Cambridge, UK).
Batagelj, V., and M. Zaversnik, 2003, eprint cs.DS/0310049.
Baumes, J., M. Goldberg, and M. Magdon-ismail, 2005a, in *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 27–36.
Baumes, J., M. K. Goldberg, M. S. Krishnamoorthy, M. M. Ismail, and N. Preston, 2005b, in *IADIS AC*, edited by N. Guimaraes and P. T. Isaias (IADIS), pp. 97–104.
Beal, M. J., 2003, *Variational Algorithms for Approximate Bayesian Inference*, Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.
Berg, J., and M. Lässig, 2006, Proc. Natl. Acad. Sci. USA **103**(29), 10967.
Berry, J. W., B. Hendrickson, R. A. LaViolette, V. J. Leung, and C. A. Phillips, 2007, eprint arXiv:0710.3800.
Bezdek, J. C., 1981, *Pattern Recognition with Fuzzy Objective Function Algorithms* (Kluwer Academic Publishers, Norwell, USA).
Bianconi, G., 2008, Europhys. Lett. **81**, 28005.
Bianconi, G., A. C. C. Coolen, and C. J. Perez Vicente, 2008a, Phys. Rev. E **78**(1), 016114.
Bianconi, G., P. Pin, and M. Marsili, 2008b, eprint arXiv:0810.4412.
Biernacki, C., G. Celeux, and G. Govaert, 2000, IEEE Trans. Pattern Anal. Mach. Intell. **22**(7), 719.
Blatt, M., S. Wiseman, and E. Domany, 1996, Phys. Rev. Lett. **76**, 3251.
Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, 2008, Journal of Statistical Mechanics: Theory and Experiment **2008**(10), P10008 (12pp).
Boccaletti, S., M. Ivanchenko, V. Latora, A. Pluchino, and A. Rapisarda, 2007, Phys. Rev. E **75**(4), 045102.
Boccaletti, S., V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, 2006, Phys. Rep. **424**(4-5), 175.
Boettcher, S., and A. G. Percus, 2001, Phys. Rev. Lett. **86**, 5211.
Bollobas, B., 1998, *Modern Graph Theory* (Springer Verlag, New York, USA).
Bomze, I. M., M. Budinich, P. M. Pardalos, and M. Pelillo, 1999, in *Handbook of Combinatorial Optimization*, edited by D.-Z. Du and P. Pardalos (Kluwer Academic Publishers, Norwell, USA), pp. 1–74.
Bonanno, G., G. Caldarelli, F. Lillo, and R. N. Mantegna, 2003, Phys. Rev. E **68**(4), 046130.
Bonanno, G., N. Vandewalle, and R. N. Mantegna, 2000, Phys. Rev. E **62**(6), R7615.
Borgatti, S., M. Everett, and P. Shirey, 1990, Soc. Netw. **12**, 337.
Brandes, U., 2001, J. Math. Sociol. **25**, 163.
Brandes, U., D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikolski, and D. Wagner, 2006, URL http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/3255.
Brandes, U., M. Gaertler, and D. Wagner, 2003, in *Proceedings of ESA* (Springer-Verlag, Berlin, Germany), pp. 568–579.
Brin, S., and L. E. Page, 1998, Comput. Networks ISDN **30**, 107.
Bron, C., and J. Kerbosch, 1973, Commun. ACM **16**, 575.
Burnham, K. P., and D. R. Anderson, 2002, *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach* (Springer, New York, USA).
Burt, R. S., 1976, Soc. Forces **55**, 93.
C. Castellano, F. Cecconi, V. Loreto, D. Parisi, and F. Radicchi, 2004, Eur. Phys. J. B **38**(2), 311.
Capocci, A., V. D. P. Servedio, G. Caldarelli, and F. Colaiori, 2005, Physica A **352**, 669.
Chan, P. K., M. D. F. Schlag, and J. Y. Zien, 1993, in *Proceedings of the 30th International Conference on Design Automation* (ACM Press, New York, USA), pp. 749–754.
Chen, J., and B. Yuan, 2006, Bioinformatics **22**(18), 2283.
Chen, W. Y. C., A. W. M. Dress, and W. Q. Yu, 2008, Math. Comp. Sci. **1**(3), 441.
Chung, F. R. K., 1997, *Spectral Graph Theory* (American Mathematical Society, Providence, USA).
Clauset, A., 2005, Phys. Rev. E **72**(2), 026132.
Clauset, A., C. Moore, and M. E. J. Newman, 2007, in *Statistical Network Analysis: Models, Issues, and New Directions*, edited by E. M. Airoldi, D. M. Blei, S. E. Fienberg, A. Goldenberg, E. P. Xing, and A. X. Zheng (Springer, Berlin, Germany), volume 4503 of *Lect. Notes Comp. Sci.*, pp. 1–13.
Clauset, A., C. Moore, and M. E. J. Newman, 2008, Nature **453**(7191), 98.
Clauset, A., M. E. Newman, and C. Moore, 2004, Phys. Rev. E **70**(6), 066111.
Cohen, R., K. Erez, D. ben Avraham, and S. Havlin, 2000, Phys. Rev. Lett. **85**(21), 4626.
Coleman, J. S., 1964, *An Introduction to Mathematical Sociology* (Collier-Macmillan, London, UK).
Condon, A., and R. M. Karp, 2001, Random Struct. Algor. **18**, 116.
da Fontoura Costa, L., 2004, eprint arXiv:cond-mat/0405022.
Danon, L., A. Díaz-Guilera, and A. Arenas, 2006, J. Stat. Mech. **11**, 10.
Danon, L., A. Díaz-Guilera, J. Duch, and A. Arenas, 2005, J. Stat. Mech. **9**, 8.
Danon, L., J. Duch, A. Arenas, and A. Díaz-Guilera, 2007, in *Large Scale Structure and Dynamics of Complex Networks: From Information Technology to Finance and Natural Science*, edited by C. G. and V. A. (World Scientific, Singapore), pp. 93–114.

Davis, A., B. B. Gardner, and M. R. Gardner, 1941, *Deep South* (University of Chicago Press, Chicago, USA).
Delling, D., M. Gaertler, R. Grke, Z. Nikoloski, and D. Wagner, 2007, *How to Evaluate Clustering Techniques.*, Technical Report, Universität Karlsruhe, Germany.
Delvenne, J. C., S. N. Yaliraki, and M. Barahona, 2008, eprint arXiv/0812.1811.
Demmel, J., J. Dongarra, A. Ruhe, and H. van der Vorst, 2000, *Templates for the solution of algebraic eigenvalue problems: a practical guide* (Society for Industrial and Applied Mathematics, Philadelphia, USA).
Dempster, A. P., N. M. Laird, and D. B. Rdin, 1977, J. Roy. Stat. Soc. B **39**, 1.
Derényi, I., G. Palla, and T. Vicsek, 2005, Phys. Rev. Lett. **94**(16), 160202.
Djidjev, H., 2006, in *WAW*, edited by W. Aiello, A. Z. Broder, J. C. M. Janssen, and E. E. Milios (Springer-Verlag, Berlin, Germany), volume 4936 of *Lecture Notes in Computer Science*, pp. 117–128.
Donetti, L., and M. A. Muñoz, 2004, J. Stat. Mech. **P10012**.
Donetti, L., and M. A. Muñoz, 2005, in *Modeling Cooperative Behavior in the Social Sciences*, edited by P. Garrido, J. Maroo, and M. A. Muñoz, volume 779 of *American Institute of Physics Conference Series*, pp. 104–107.
van Dongen, S., 2000a, *Graph Clustering by Flow Simulation*, Ph.D. thesis, Dutch National Research Institute for Mathematics and Computer Science, University of Utrecht, Netherlands.
van Dongen, S., 2000b, *Performance criteria for graph clustering and Markov cluster experiments*, Technical Report, National Research Institute for Mathematicas and Computer Science in the Nederlands, Amsterdam, The Netherlands.
Doreian, P., V. Batagelj, and A. Ferligoj, 2005, *Generalized Blockmodeling* (Cambridge University Press, New York, USA).
Dorogovtsev, S. N., and J. F. F. Mendes, 2002, Adv. Phys. **51**, 1079.
Du, H., M. W. Feldman, S. Li, and X. Jin, 2007, Complexity **12**(3), 53.
Du, N., B. Wu, B. Wang, and Y. Wang, 2008, eprint arXiv:0804.3636.
Duch, J., and A. Arenas, 2005, Phys. Rev. E **72**(2), 027104.
Dunn, J. C., 1973, J. Cybernetics **3**, 32.
Dunn, R., F. Dudbridge, and C. M. Sanderson, 2005, BMC Bioinf. **6**, 39.
Earl, D. J., and M. W. Deem, 2005, Phys. Chem. Chem. Phys. **7**, 3910.
Eckmann, J.-P., and E. Moses, 2002, Proc. Natl. Acad. Sci. USA **99**, 5825.
Efron, B., and R. J. Tibshirani, 1993, *An Introduction to the Bootstrap* (Chapman & Hall, New York, USA).
Elias, P., A. Feinstein, and C. E. Shannon, 1956, IRE Trans. Inf. Theory **IT-2**, 117.
Erdös, P., and A. Rényi, 1959, Publ. Math. Debrecen **6**, 290.
Eriksen, K. A., I. Simonsen, S. Maslov, and K. Sneppen, 2003, Phys. Rev. Lett. **90**(14), 148701.
Euler, L., 1736, Commentarii Academiae Petropolitanae **8**, 128.
Evans, T. S., and R. Lambiotte, 2009, eprint arXiv:0903.2181.
Everett, M. G., and S. P. Borgatti, 1994, J. Math. Soc. **19**(1), 29.
Everett, M. G., and S. P. Borgatti, 1998, Connections **21**(1), 49.
Fan, Y., M. Li, P. Zhang, J. Wu, and Z. Di, 2007, Physica A **377**, 363.
Farkas, I., D. Ábel, G. Palla, and T. Vicsek, 2007, New J. Phys. **9**, 180.
Farutin, V., K. Robison, E. Lightcap, V. Dancik, A. Ruttenberg, S. Letovsky, and J. Pradines, 2006, Proteins **62**(3), 800.
Feige, U., D. Peleg, and G. Kortsarz, 2001, Algorithmica **29**(3), 410.
Fenn, D. J., M. A. Porter, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones, 2008, eprint arXiv:0811.3988.
Fiedler, M., 1973, Czechoslovak Math. J. **23**(98), 298.
Fiedler, M., 1975, Czechoslovak Math. J. **25**, 619.
Fienberg, S. E., and S. Wasserman, 1981, Sociol. Methodol. **12**, 156.
Flake, G. W., S. Lawrence, and C. L. Giles, 2000, in *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM Press, Boston, USA), pp. 150–160.
Flake, G. W., S. Lawrence, C. Lee Giles, and F. M. Coetzee, 2002, IEEE Computer **35**, 66.
F.Lorrain, and H. White, 1971, J. Math. Sociol. **1**, 49.
Ford, L. R., and D. R. Fulkerson, 1956, Canadian J. Math. **8**, 399.
Fortunato, S., 2007, in *Noise and Stochastics in Complex Systems and Finance*, volume 6601 of *SPIE Conference Series*, p. 660108.
Fortunato, S., and M. Barthélemy, 2007, Proc. Natl. Acad. Sci. USA **104**, 36.
Fortunato, S., and C. Castellano, 2009, in *Encyclopedia of Complexity and Systems Science*, edited by R. A. Meyers (Springer, Berlin, Germany), volume 1, eprint arXiv:0712.2716.
Fortunato, S., V. Latora, and M. Marchiori, 2004, Phys. Rev. E **70**(5), 056104.
Fowlkes, E. B., and C. L. Mallows, 1983, J. Am. Stat. Assoc. **78**, 553.
Freeman, L. C., 1977, Sociometry **40**, 35.
Freeman, L. C., 2004, *The Development of Social Network Analysis: A Study in the Sociology of Science* (BookSurge Publishing).
Fu, Y., and P. Anderson, 1986, J. Phys. A **19**, 1605.
G. Xu, S. Tsoka, and L.G. Papageorgiou, 2007, Eur. Phys. J. B **60**(2), 231.
Gaertler, M., R. Grke, and D. Wagner, 2007, in *AAIM*, edited by M.-Y. Kao and X.-Y. Li (Springer, Berlin, Germany), volume 4508 of *Lecture Notes in Computer Science*, pp. 11–26.
Gallager, R. G., 1963, *Low density parity check codes* (MIT Press, Cambridge, USA).
Gan, G., C. Ma, and J. Wu, 2007, *Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability)* (Society for Industrial and Applied Mathematics, Philadelphia, USA), ISBN 0898716233.
Garey, M. R., and D. S. Johnson, 1990, *Computers and Intractability : A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, USA).
Gfeller, D., J.-C. Chappelier, and P. de Los Rios, 2005, Phys. Rev. E **72**(5), 056135.
Giles, C. L., K. Bollacker, and S. Lawrence, 1998, in *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, edited by I. Witten, R. Akscyn, and F. M. Shipman III (ACM Press, Pittsburgh, PA), pp. 89–98.
Girvan, M., and M. E. Newman, 2002, Proc. Natl. Acad. Sci.

USA **99**(12), 7821.
Gleiser, P., and L. Danon, 2003, Adv. Complex Syst. **6**, 565.
Glover, F., 1986, Comput. Oper. Res. **13**(5), 533.
Goldberg, A. V., and R. E. Tarjan, 1988, Journal of the ACM **35**, 921.
Gómez, S., P. Jensen, and A. Arenas, 2008, eprint arXiv:0812.3030.
Granovetter, M., 1973, Am. J. Sociol. **78**, 1360.
Gregory, S., 2007, in *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)* (Springer-Verlag, Berlin, Germany), pp. 91–102.
Gregory, S., 2009, in *Complex Networks*, edited by S. Fortunato, R. Menezes, G. Mangioni, and V. Nicosia (Springer, Berlin, Germany), volume 207 of *Studies on Computational Intelligence*, pp. 47–62.
Grünwald, P. D., I. J. Myung, and M. A. Pitt, 2005, *Advances in Minimum Description Length: Theory and Applications* (MIT Press, Cambridge, USA).
Gudkov, V., V. Montealegre, S. Nussinov, and Z. Nussinov, 2008, Phys. Rev. E **78**(1), 016113.
Guimerà, R., and L. A. N. Amaral, 2005, J. Stat. Mech. **P02001**(02).
Guimerà, R., and L. A. N. Amaral, 2005, Nature **433**, 895.
Guimerà, R., L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, 2003, Phys. Rev. E **68**(6), 065103.
Guimerà, R., M. Sales-Pardo, and L. Amaral, 2007, Bioinformatics **23**(13), 1616.
Guimerà, R., M. Sales-Pardo, and L. A. Amaral, 2004, Phys. Rev. E **70**(2), 025101.
Guimerà, R., M. Sales-Pardo, and L. A. N. Amaral, 2007, Phys. Rev. E **76**(3), 036102.
Gusfield, D., 2002, Inform. Process. Lett. **82**(3), 159.
Gustafsson, M., M. Hörnquist, and A. Lombardi, 2006, Physica A **367**, 559.
Hagen, L., and A. B. Kahng, 1992, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **11**(9), 1074.
Handcock, M. S., A. E. Raftery, and J. M. Tantrum, 2007, J. Roy. Stat. Soc. A **170**(Working Paper no. 46), 1.
Hastie, T., R. Tibshirani, and J. H. Friedman, 2001, *The Elements of Statistical Learning* (Springer, Berlin, Germany), ISBN 0387952845.
Hastings, M. B., 2006, Phys. Rev. E **74**(3), 035102.
Heimo, T., J. M. Kumpula, K. Kaski, and J. Saramäki, 2008, J. Stat. Mech. **P08007**.
Hillier, F. S., and G. J. Lieberman, 2004, *MP Introduction to Operations Research* (McGraw-Hill, New York, USA).
Hofman, J. M., and C. H. Wiggins, 2008, Phys. Rev. Lett. **100**(25), 258701.
Holland, J. H., 1992, *Adaptation in natural and artificial systems* (MIT Press, Cambridge, USA), ISBN 0262581116.
Holland, P., K. B. Laskey, and S. Leinhardt, 1983, Soc. Netw. **5**, 109.
Holme, P., M. Huss, and H. Jeong, 2003, Bioinformatics **19**(4), 532.
Holzapfel, K., S. Kosub, M. G. Maa, and H. Täubig, 2003, in *CIAC*, edited by R. Petreschi, G. Persiano, and R. Silvestri (Springer), volume 2653 of *Lecture Notes in Computer Science*, pp. 201–212.
Homans, G. C., 1950, *The Human Groups* (Harcourt, Brace & Co., New York).
Hopcroft, J., O. Khan, B. Kulis, and B. Selman, 2004, Proc. Natl. Acad. Sci. USA **101**, 5249.
Hu, Y., H. Chen, P. Zhang, M. Li, Z. Di, and Y. Fan, 2008, Phys. Rev. E **78**(2), 026121.
Hu, Y., M. Li, P. Zhang, Y. Fan, and Z. Di, 2008, Phys. Rev. E **78**(1), 016115.
Huffman, D. A., 1952, Proc. IRE **40**(9), 1098.
Hughes, B. D., 1995, *Random Walks and Random Environments: Random Walks Vol 1* (Clarendon Press, Oxford, UK), ISBN 0198537883.
Itzkovitz, S., R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, 2005, Phys. Rev. E **71**(1), 016127.
Jin, R. K.-X., D. C. Parkes, and P. J. Wolfe, 2007, in *Proc. AAAI Workshop on Plan, Activity and Intent Recognition (PAIR)*, pp. 66–73.
Jonsson, P. F., T. Cavanna, D. Zicha, and P. A. Bates, 2006, BMC Bioinf. **7**, 2.
Jordan, M. I., Z. Ghahramani, T. Jaakkola, and L. K. Saul, 1999, Mach. Learn. **37**(2), 183.
Junker, B. H., and F. Schreiber, 2008, *Analysis of Biological Networks* (Wiley-Interscience, New York, USA).
Kaplan, T. D., and S. Forrest, 2008, eprint arXiv:0801.3290.
Karloff, H., 1991, *Linear Programming* (Birkhäuser Verlag, Basel, Switzerland).
Karrer, B., E. Levina, and M. E. J. Newman, 2008, Phys. Rev. E **77**(4), 046119.
Kernighan, B. W., and S. Lin, 1970, Bell System Tech. J. **49**, 291.
Kim, Y., S.-W. Son, and H. Jeong, 2009, eprint arXiv:0902.3728.
Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, 1983, Science **220**, 671.
Kleinberg, J., 2002, in *Advances in NIPS 15* (MIT Press, Boston, USA), pp. 446–453.
Koskinen, J. H., and T. A. B. Snijders, 2007, J. Stat. Plan. Infer. **137**(12), 3930.
Kottak, C. P., 2004, *Cultural Anthropology* (McGraw-Hill, New York, USA).
Krause, A. E., K. A. Frank, D. M. Mason, R. E. Ulanowicz, and W. W. Taylor, 2003, Nature **426**, 282.
Krawczyk, M. J., 2008, Phys. Rev. E **77**(6), 065701.
Krawczyk, M. J., and K. Kulakowski, 2007, eprint arXiv:0709.0923.
Kumpula, J. M., M. Kivelä, K. Kaski, and J. Saramäki, 2008, Phys. Rev. E **78**(2), 026109.
Kumpula, J. M., J. Saramäki, K. Kaski, and J. Kertész, 2007a, in *Noise and Stochastics in Complex Systems and Finance*, volume 6601 of *SPIE Conference Series*, p. 660116.
Kumpula, J. M., J. Saramäki, K. Kaski, and J. Kertész, 2007b, Eur. Phys. J. B **56**, 41.
Kuramoto, Y., 1984, *Chemical Oscillations, Waves and Turbulence* (Springer-Verlag, Berlin, Germany).
Lambiotte, R., J. . Delvenne, and M. Barahona, 2008, eprint arXiv:0812.1770.
Lancichinetti, A., and S. Fortunato, 2009, eprint arXiv:0904.3940.
Lancichinetti, A., S. Fortunato, and J. Kertesz, 2009, New J. Phys. **11**(3), 033015.
Lancichinetti, A., S. Fortunato, and F. Radicchi, 2008, Phys. Rev. E **78**(4), 046110.
Lanczos, C., 1950, J. Res. Natl. Bur. Stand. **45**, 255.
Latapy, M., and P. Pons, 2005, Lect. Notes Comp. Sci. **3733**, 284.
Latora, V., and M. Marchiori, 2001, Phys. Rev. Lett. **87**(19), 198701.
Lehmann, S., and L. K. Hansen, 2007, Eur. Phys. J. B **60**,

83.
Lehmann, S., M. Schwartz, and L. K. Hansen, 2008, Phys. Rev. E **78**(1), 016108.
Leicht, E. A., and M. E. J. Newman, 2008, Phys. Rev. Lett. **100**(11), 118703.
Li, Z., S. Zhang, R.-S. Wang, X.-S. Zhang, and L. Chen, 2008, Phys. Rev. E **77**(3), 036109.
Liben-Nowell, D., and J. Kleinberg, 2003, in *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management* (ACM, New York, NY, USA), pp. 556–559.
Lloyd, S., 1982, IEEE Trans. Inf. Theory **28**(2), 129.
Luccio, F., and M. Sami, 1969, IEEE Trans. Circuit Th. CT **16**, 184.
Luce, R. D., 1950, Psychometrika **15**(2), 169.
Luce, R. D., and A. D. Perry, 1949, Psychometrika **14**(2), 95.
Łuczak, T., 1992, in *Proceedings of the Symposium on Random Graphs, Poznań 1989* (John Wiley & Sons, New York, USA), pp. 165–182.
Lusseau, D., 2003, Proc. Royal Soc. London B **270**, S186.
Mackay, D. J. C., 2003, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, UK).
MacQueen, J. B., 1967, in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, edited by L. M. L. Cam and J. Neyman (University of California Press, Berkeley, USA), volume 1, pp. 281–297.
Mantegna, R. N., 1999, Eur. Phys. J. B **11**, 193.
Mantegna, R. N., and H. E. Stanley, 2000, *An introduction to econophysics: correlations and complexity in finance* (Cambridge University Press, New York, USA).
Massen, C. P., and J. P. Doye, 2005, Phys. Rev. E **71**(4), 046101.
Massen, C. P., and J. P. K. Doye, 2006, eprint cond-mat/0610077.
Matsuda, H., T. Ishihara, and A. Hashimoto, 1999, Theor. Comp. Sci. **210**, 305.
Matula, D. W., and F. Shahrokhi, 1990, Discrete Appl. Math. **27**(1-2), 113.
Medus, A., G. Acuña, and C. O. Dorso, 2005, Physica A **358**, 593.
Meilă, M., 2007, J. Multivar. Anal. **98**(5), 873.
Meilă, M., and D. Heckerman, 2001, Mach. Learn. **42**(1), 9.
Mendes, J. F. F., and S. N. Dorogovtsev, 2003, *Evolution of Networks: from biological nets to the Internet and WWW* (Oxford University Press, Oxford, UK).
Mézard, M., and G. Parisi, 2003, J. Stat. Phys. **111**, 1.
Mezard, M., G. Parisi, and M. Virasoro, 1987, *Spin glass theory and beyond* (World Scientific Publishing Company, Singapore).
Middleton, A. A., and D. S. Fisher, 2002, Phys. Rev. B **65**(13), 134411.
Milgram, S., 1967, Psychol. Today **2**, 60.
Milo, R., S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, 2002, Science **298**(5594), 824.
Mirkin, B., 1996, *Mathematical classification and clustering* (Kluwer Academic Press, Norwell, USA).
Mokken, R. J., 1979, Qual. Quant. **13**(2), 161.
Molloy, M., and B. Reed, 1995, Random Struct. Algor. **6**, 161.
Moody, J., and D. R. White, 2003, Am. Sociol. Rev. **68**(1), 103.
Muff, S., F. Rao, and A. Caflisch, 2005, Phys. Rev. E **72**(5), 056107.
Mungan, M., and J. J. Ramasco, 2008, eprint arXiv:0809.1398.
Nelson, D. L., C. L. McEvoy, and T. A. Schreiber, 1998, The university of south florida word association, rhyme, and word fragment norms.
Nepusz, T., A. Petróczi, L. Négyessy, and F. Bazsó, 2008, Phys. Rev. E **77**(1), 016107.
Newman, M. E. J., 2001, Proc. Nat. Acad. Sci. USA **98**(2), 404.
Newman, M. E. J., 2003, SIAM Rev. **45**(2), 167.
Newman, M. E. J., 2004, Phys. Rev. E **70**(5), 056131.
Newman, M. E. J., 2004a, Eur. Phys. J. B **38**, 321.
Newman, M. E. J., 2004b, Phys. Rev. E **69**(6), 066133.
Newman, M. E. J., 2005, Soc. Netw. **27**, 39.
Newman, M. E. J., 2006a, Phys. Rev. E **74**(3), 036104.
Newman, M. E. J., 2006b, Proc. Natl. Acad. Sci. USA **103**, 8577.
Newman, M. E. J., and T. Barkema, 1999, *Monte Carlo Methods in Statistical Physics* (Oxford University Press, Oxford, UK).
Newman, M. E. J., and M. Girvan, 2004, Phys. Rev. E **69**(2), 026113.
Newman, M. E. J., and E. A. Leicht, 2007, Proc. Natl. Acad. Sci. USA **104**, 9564.
Nicosia, V., G. Mangioni, V. Carchiolo, and M. Malgeri, 2009, J. Stat. Mech. **2009**(03), P03024.
Nishikawa, T., A. E. Motter, Y.-C. Lai, and F. C. Hoppensteadt, 2003, Phys. Rev. Lett. **91**(1), 014101.
Noack, A., 2009, Phys. Rev. E **79**(2), 026102.
Noack, A., and R. Rotta, 2008, eprint arXiv:0812.4073.
Noh, J. D., and H. Rieger, 2001, Phys. Rev. Lett. **87**(17), 176102.
Noh, J. D., and H. Rieger, 2002, Phys. Rev. E **66**(3), 036117.
Nowicki, K., and T. A. B. Snijders, 2001, J. Am. Stat. Assoc. **96**(455).
Ohkubo, J., and K. Tanaka, 2006, J. Phys. Soc. Jpn. **75**(11), 115001.
Onnela, J.-P., A. Chakraborti, K. Kaski, J. Kertész, and A. Kanto, 2003, Phys. Rev. E **68**(5), 056110.
Onnela, J.-P., A. Chakraborti, K. Kaski, and J. Kertiész, 2002, Eur. Phys. J. B **30**(3), 285.
Palla, G., A.-L. Barabási, and T. Vicsek, 2007, Nature **446**, 664.
Palla, G., I. Derényi, I. Farkas, and T. Vicsek, 2005, Nature **435**, 814.
Papadimitriou, C. M., 1994, *Computational complexity* (Addison-Wesley, Reading, USA).
Pastor-Satorras, R., and A. Vespignani, 2001, Phys. Rev. Lett. **86**(14), 3200.
Pastor-Satorras, R., and A. Vespignani, 2004, *Evolution and Structure of the Internet: A Statistical Physics Approach* (Cambridge University Press, New York, NY, USA).
Peeters, R., 2003, Discrete Appl. Math. **131**, 651.
Peterson, C., and J. R. Anderson, 1987, Compl. Syst. **1**, 995.
Pikovsky, A., M. G. Rosenblum, and J. Kurths, 2001, *Synchronization : A Universal Concept in Nonlinear Sciences* (Cambridge University Press, Cambridge, UK).
Pimm, S. L., 1979, Theor. Popul. Biol. **16**, 144.
Pinney, J. W., and D. R. Westhead, 2006, in *Interdisciplinary Statistics and Bioinformatics* (Leeds University Press, Leeds, UK), pp. 87–90.
Pluchino, A., V. Latora, and A. Rapisarda, 2005, Int. J. Mod. Phys. C **16**, 515.
Pollner, P., G. Palla, and T. Vicsek, 2006, Europhys. Lett. **73**, 478.

Pons, P., 2006, eprint arXiv:cs/0608050.
Porter, M. A., P. J. Mucha, M. E. J. Newman, and A. J. Friend, 2007, Physica A **386**, 414.
Porter, M. A., P. J. Mucha, M. E. J. Newman, and C. M. Warmbrand, 2005, Proc. Natl. Acad. Sci. USA **102**, 7057.
Porter, M. A., J.-P. Onnela, and P. J. Mucha, 2009, eprint arXiv:0902.3788.
Pothen, A., 1997, *Graph Partitioning Algorithms with Applications to Scientific Computing*, Technical Report, Norfolk, VA, USA.
Price, D. D., 1976, J. Am. Soc. Inform. Sci. **27**(5), 292.
Pujol, J. M., J. Béjar, and J. Delgado, 2006, Phys. Rev. E **74**(1), 016107.
Radicchi, F., C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, 2004, Proc. Natl. Acad. Sci. USA **101**, 2658.
Raghavan, U. N., R. Albert, and S. Kumara, 2007, Phys. Rev. E **76**(3), 036106.
Ramasco, J. J., and M. Mungan, 2008, Phys. Rev. E **77**(3), 036122.
Rand, W. M., 1971, J. Am. Stat. Assoc. **66**(336), 846.
Ravasz, E., and A.-L. Barabási, 2003, Phys. Rev. E **67**(2), 026112.
Ravasz, E., A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, 2002, Science **297**(5586), 1551.
Reichardt, J., and S. Bornholdt, 2004, Phys. Rev. Lett. **93**(21), 218701.
Reichardt, J., and S. Bornholdt, 2006a, Phys. Rev. E **74**(1), 016110.
Reichardt, J., and S. Bornholdt, 2006b, Physica D **224**, 20.
Reichardt, J., and S. Bornholdt, 2007, J. Stat. Mech. **2007**(06), P06016.
Reichardt, J., and S. Bornholdt, 2007, Phys. Rev. E **76**(1), 015102.
Reichardt, J., and M. Leone, 2008, Phys. Rev. Lett. **101**(7), 078701.
Reichardt, J., and D. R. White, 2007, Eur. Phys. J. B **60**, 217.
Ren, W., G. Yan, X. Liao, and Y. Cheng, 2007, eprint arXiv:0710.3422.
Rhodes, C. J., and E. M. J. Keefe, 2007, J. Oper. Res. Soc. **58**(12), 1605.
Rice, S. A., 1927, Am. Polit. Sci. Rev. **21**, 619.
Richardson, T., P. J. Mucha, and M. A. Porter, 2008, eprint arXiv:0812.2852.
Rissanen, J., 1978, Automatica **14**, 465.
Rives, A. W., and T. Galitski, 2003, Proc. Natl. Acad. Sci. USA **100**(3), 1128.
Ronhovde, P., and Z. Nussinov, 2008a, eprint arXiv:0803.2548.
Ronhovde, P., and Z. Nussinov, 2008b, eprint arXiv:0812.1072.
Rosvall, M., and C. T. Bergstrom, 2007, Proc. Natl. Acad. Sci. USA **104**, 7327.
Rosvall, M., and C. T. Bergstrom, 2008, eprint arXiv/0812.1242.
Rosvall, M., and C. T. Bergstrom, 2008, Proc. Natl. Acad. Sci. USA **105**, 1118.
Rowicka, M., and A. Kudlicki, 2004, in *Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, edited by R. Fischer, R. Preuss, and U. von Toussaint (American Institute of Physics, Melville, USA), volume 735, pp. 283–288.
Ruan, J., and W. Zhang, 2008, Phys. Rev. E **77**(1), 016104.
S.-W. Son, H. Jeong, and J.D. Noh, 2006, Eur. Phys. J. B **50**(3), 431.
Sales-Pardo, M., R. Guimerà, A. A. Moreira, and L. A. N. Amaral, 2007, Proc. Natl. Acad. Sci. USA **104**, 15224.
Sawardecker, E. N., M. Sales-Pardo, and L. A. N. Amaral, 2009, Eur. Phys. J. B **67**, 277.
Schaeffer, S. E., 2007, Comput. Sci. Rev. **1**(1), 27.
Schuetz, P., and A. Caflisch, 2008a, Phys. Rev. E **77**(4), 046112.
Schuetz, P., and A. Caflisch, 2008b, Phys. Rev. E **78**(2), 026112.
Schwarz, G., 1978, Ann. Stat. **6**(2), 461.
Scott, J., 2000, *Social Network Analysis: A Handbook* (SAGE Publications, London, UK).
Seidman, S. B., 1983, Soc. Netw. **5**, 269.
Seidman, S. B., and B. L. Foster, 1978, J. Math. Sociol. **6**, 139.
Sen, T. Z., A. Kloczkowski, and R. L. Jernigan, 2006, BMC Bioinf. **7**(1), 355.
Shen, H., X. Cheng, K. Cai, and M.-B. Hu, 2009, Physica A **388**, 1706.
Sherrington, D., and S. Kirkpatrick, 1975, Phys. Rev. Lett. **35**, 1792.
Šíma, J., and S. E. Schaeffer, 2006, in *Proceedings of the Thirty-second International Conference on Current Trends in Theory and Practice of Computer Science (Sofsem 06)*, edited by J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, and J. Štuller (Springer-Verlag, Berlin/Heidelberg, Germany), volume 3831 of *Lecture Notes in Computer Science*, pp. 530–537.
Simon, H., 1962, Proc. Am. Phil. Soc. **106**(6), 467.
Simon, H. A., 1955, Biometrika **42**, 425.
Simonsen, I., 2005, Physica A **357**(2), 317.
Simonsen, I., K. Astrup Eriksen, S. Maslov, and K. Sneppen, 2004, Physica A **336**, 163.
Slanina, F., and Y.-C. Zhang, 2005, Acta Phys. Pol. B **36**, 2797.
Snijders, T., and K. Nowicki, 1997, J. Classif. **14**, 75.
de Solla Price, D. J., 1965, Science **169**, 510.
Solomonoff, R., and A. Rapoport, 1951, Bull. Math. Biophys. **13**, 107.
Spirin, V., and L. A. Mirny, 2003, Proc. Natl. Acad. Sci. USA **100**(21), 12123.
Stanley, R. P., 1997, *Enumerative combinatorics, Vol. I* (Cambridge University Press, Cambridge, UK).
Suaris, P. R., and G. Kedem, 1988, IEEE Trans. Circuits Syst. **35**, 294.
Sun, J., C. Faloutsos, S. Papadimitriou, and P. S. Yu, 2007, in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, New York, USA), pp. 687–696.
Sun, Y., B. Danila, K. Josic, and K. E. Bassler, 2009, Europhys. Lett. **86**(2), 28004.
Tasgin, M., A. Herdagdelen, and H. Bingol, 2007, eprint arXiv:0711.0491.
Tibély, G., and J. Kertész, 2008, Physica A **387**, 4982.
Tishby, N., F. Pereira, and W. Bialek, 1999, in *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pp. 368–377.
Traag, V. A., and J. Bruggeman, 2008, eprint arXiv:0811.2329.
Traud, A. L., E. D. Kelsic, P. J. Mucha, and M. A. Porter, 2008, eprint arXiv:0809.0690.

Travers, J., and S. Milgram, 1969, Sociometry **32**, 425.
Tumminello, M., F. Lillo, and R. N. Mantegna, 2008, ArXiv e-prints eprint 0809.4615.
Tyler, J. R., D. M. Wilkinson, and B. A. Huberman, 2003, in *Communities and technologies* (Kluwer, B.V., Deventer, The Netherlands), pp. 81–96.
Vazquez, A., 2008, eprint arXiv:0805.2689.
Vazquez, A., 2008, Phys. Rev. E **77**(6), 066106.
Čopič, J., M. O. Jackson, and A. Kirman, 2005, URL http://www.hss.caltech.edu/~{}jernej/netcommunity.pdf.
Vragović, I., and E. Louis, 2006, Phys. Rev. E **74**(1), 016105.
Wakita, K., and T. Tsurumi, 2007, eprint arXiv:cs/0702048.
Wallace, C. S., and D. M. Boulton, 1968, The Computer Journal **11**(2), 185.
Wallace, D. L., 1983, J. Am. Stat. Assoc. **78**, 569.
Ward, J. H., 1963, J. Am. Stat. Assoc. **58**(301), 236.
Wasserman, S., and K. Faust, 1994, *Social network analysis* (Cambridge University Press, Cambridge, UK).
Watts, D., and S. Strogatz, 1998, Nature **393**, 440.
Watts, D. J., 2003, *Small Worlds : The Dynamics of Networks between Order and Randomness* (Princeton University Press, Princeton, USA).
Wei, Y.-C., and C.-K. Cheng, 1989, in *Proceedings of IEEE International Conference on Computer Aided Design* (Institute of Electrical and Electronics Engineers, New York), pp. 298–301.
Weiss, R. S., and E. Jacobson, 1955, Am. Sociol. Rev. **20**, 661.
White, D. R., and K. P. Reitz, 1983, Soc. Netw. **5**, 193.
White, S., and P. Smyth, 2005, in *Proceedings of SIAM International Conference on Data Mining*, pp. 76–84.
Wilkinson, D. M., and B. A. Huberman, 2004, Proc. Natl. Acad. Sci. U.S.A. **101**(1073), 5241.
Williams, R. J., and N. D. Martinez, 2000, Nature **404**, 180.
Winkler, R. L., 2003, *Introduction to Bayesian Inference and Decision* (Probabilistic Publishing, Gainesville, USA).
Wu, F., and B. A. Huberman, 2004, Eur. Phys. J. B **38**, 331.
Wu, F. Y., 1982, Rev. Mod. Phys. **54**(1), 235.
Yuta, K., N. Ono, and Y. Fujiwara, 2007, eprint arXiv:physics/0701168.
Zachary, W. W., 1977, J. Anthropol. Res. **33**, 452.
Zanghi, H., C. Ambroise, and V. Miele, 2008, Pattern Recogn. **41**(12), 3592.
Zarei, M., and K. A. Samani, 2009, Physica A **388**, 1721.
Zhang, A., 2009, *Protein Interaction Networks* (Cambridge University Press, Cambridge, UK).
Zhang, P., M. Li, J. Wu, Z. Di, and Y. Fan, 2006, Physica A **367**, 577.
Zhang, P., J. Wang, X. Li, Z. Di, and Y. Fan, 2007, eprint arXiv:0710.0117.
Zhang, S., R.-S. Wang, and X.-S. Zhang, 2007, Physica A **374**, 483.
Zhang, Y., A. J. Friend, A. L. Traud, M. A. Porter, J. H. Fowler, and P. J. Mucha, 2008, Physica A **387**(7), 1705.
Zhou, H., 2003a, Phys. Rev. E **67**(6), 061901.
Zhou, H., 2003b, Phys. Rev. E **67**(4), 041908.
Zhou, H., and R. Lipowsky, 2004, Lect. Notes Comp. Sci. **3038**, 1062.
Zhou, T., J.-G. Liu, and B.-H. Wang, 2006, Chin. Phys. Lett. **23**, 2327.
Ziv, E., M. Middendorf, and C. H. Wiggins, 2005, Phys. Rev. E **71**(4), 046117.