Community detection in graphs

Santo Fortunato

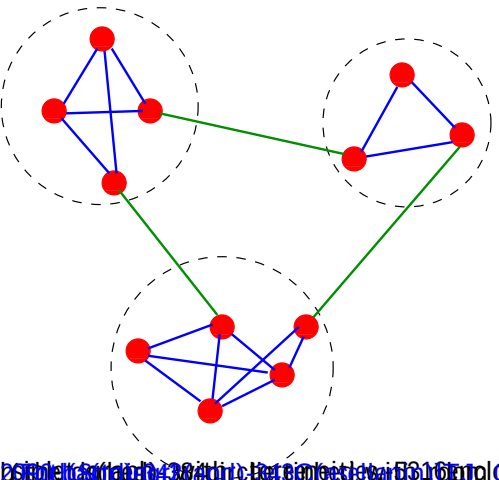
Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino, I-ITALY.

The modern science of networks has brought signi cant 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 di erent 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 e ort 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 de nition 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 signi cance of clustering and how methods should be tested and compared against each other, to the description of applications to real -479(systems)uprh2ks

I. INTRODUCTION

The origin of graph theory dates back to Euler's solution of the puzzle of Konigsberg's bridges in 1736 (Euler, 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 Barabasi, 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. Erdos and A. Renyi (Erdos and Renyi, 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 Auer-d [(c756(not)-747(ohly)-756gllobally)83e, alsoially in13(homoge)-1(neoue,)-325(with)-334(high)-763(coecen)28(rzatiosh)-763(of)]TJ 0 -11.457 Td [RG [(Au665(wit(in)-64 azatiosh Neb between these goupsI. Td [(c712(featuh)1(e)c712e)-37716(real)-730(neu-)]TJ 0 -11.457 Td [(w)28(orks)-342(is)-333(ia 002heir of t(sn)-332(reige)-1w- -355earlieor reigw(s)-3550see (



tional queries, like path searches (Agrawal and Jagadish, 1994; Wu *et al.*, 2004). *Ad hoc networks* (Perkins, 2001), i. e. self-con guring networks formed by communication nodes acting in the same region and rapidly changing (because the devices move, for instance), usually have no centrally maintained routing tables that specify how nodes have to communicate to other nodes. Grouping the nodes into clusters enables one to generate compact routing tables while the choice of the communication paths is still e cient (Steenstrup, 2001).

Community detection is important for other reasons, too. Identifying modules and their boundaries allows for a classi cation 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 di erent communities (alike to Csermely's \creative elements" (Csermely, 2008)). Such classi cation seems to be meaningful in social (Burt, 1976; Freeman, 1977; Granovetter, 1973) and metabolic networks (Guimera 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 [1]. Recent studies indicate that networks of communities have a di erent 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 o ers 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 rms, 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 rst and use them as building blocks to get larger structures, until the whole system is assembled. In this way it is also far more di cult 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 rst 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 identi ed by means of private interviews. Work groups were separated by removing the members working with people of di erent 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 nd 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 di erent groups is minimal. The mathematical formalization of this problem is called *graph partitioning* (Section IV.A). The rst 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 identi cation of edges lying between communities and their successive removal, a procedure that after some 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

[1] Coarse-graining a graph generally means mapping it onto a smaller graph having similar properties, which is easier to handle. For this purpose, the vertices of the original graph are not necessarily grouped in communities. Gfeller and De Los Rios have proposed coarse-graining schemes that keep the properties of dynamic processes acting on the graph, like random walks (Gfeller and De Los Rios, 2007) and synchronization (Gfeller and De Los Rios, 2008).

of the edges in processes where signals are transmitted across the graph following paths of minimal length. The

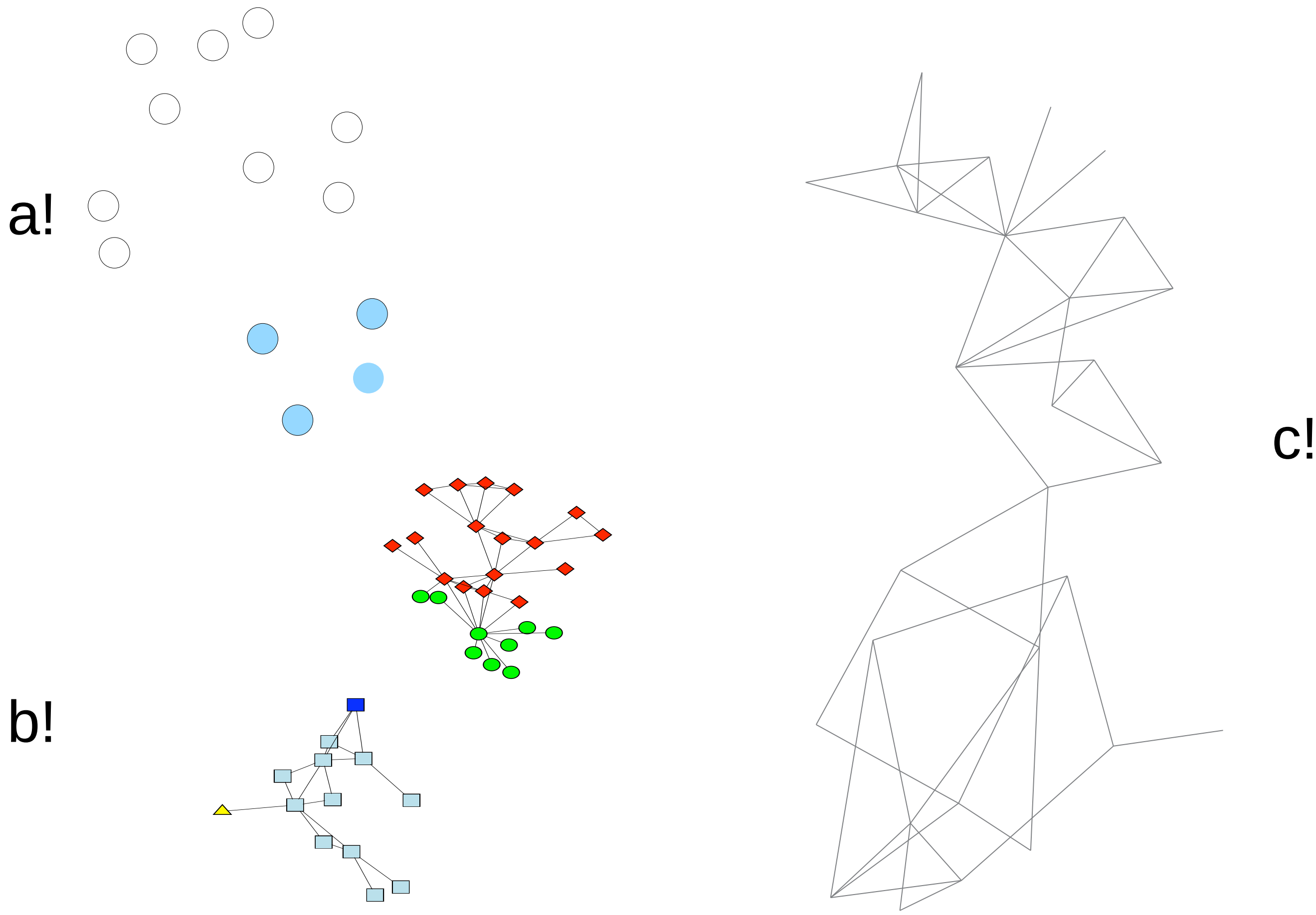
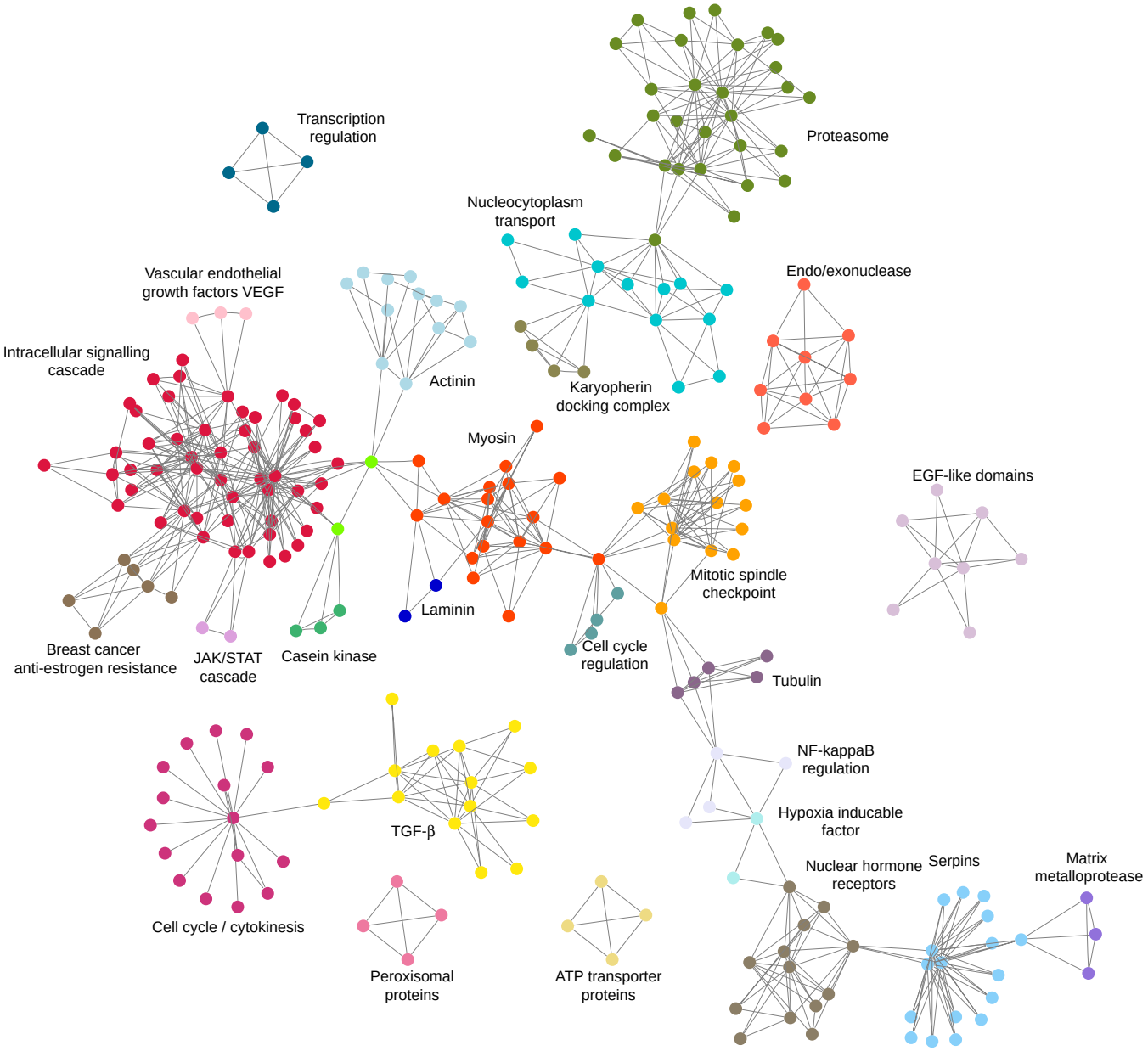


FIG. 2 Community structure in social networks. a) Zachary's karate club, a standard benchmark in community detection. The colors correspond to the best partition found by optimizing the modularity of Newman and Girvan (Section VI.A). Reprinted gure with permission from Ref. (Donetti and Muñoz, 2004). c 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 gure with permission from Ref. (Girvan and Newman, 2002). c 2002 by the National Academy of Science of the USA. c) Lusseau's network of bottlenose dolphins. The colors label the communities identi ed through the optimization of a modi ed 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 classi cation of the dolphins proposed by Lusseau. Reprinted gure with permission from Ref. (Arenas *et al.*, 2008b). c 2008 by IOP Publishing.

portant 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 (Fig. 459(W)ld(...)-339(...)T8(4ted)-4-11.457graphction, bT369s356(identimizi0(et)28(bmizi0(pag85(ca


Transcription regulation
Proteasome
Nucleocytoplasm transport
Vascular endothelial growth factors VEGF
Endo/exonuclease
Intracellular signalling cascade
Actinin
Karyopherin docking complex
Myosin
EGF-like domains
Mitotic spindle checkpoint
Laminin
Breast cancer anti-estrogen resistance
JAK/STAT cascade
Casein kinase
Cell cycle regulation
Tubulin
NF-kappaB regulation
Hypoxia inducable factor
TGF-β
Nuclear hormone receptors
Serpins
Matrix metalloprotease
Cell cycle / cytokinesis
Peroxisomal proteins
ATP transporter proteins

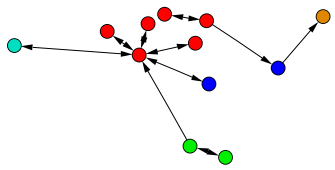


FIG. 4 Community structure in technological networks. Sample of the web graph consisting of the pages of a web site and their mutual hyperlinks, which are directed. Communities, indicated by the colors, were detected with the algorithm of Girvan and Newman (Section V.A), by neglecting the directedness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

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 de nition forbids multiple memberships of vertices. Classical examples are social networks, where an individual usually belongs to di erent circles at the same time, from that of work colleagues to family, sport associations, etc.. Traditional algorithms of community

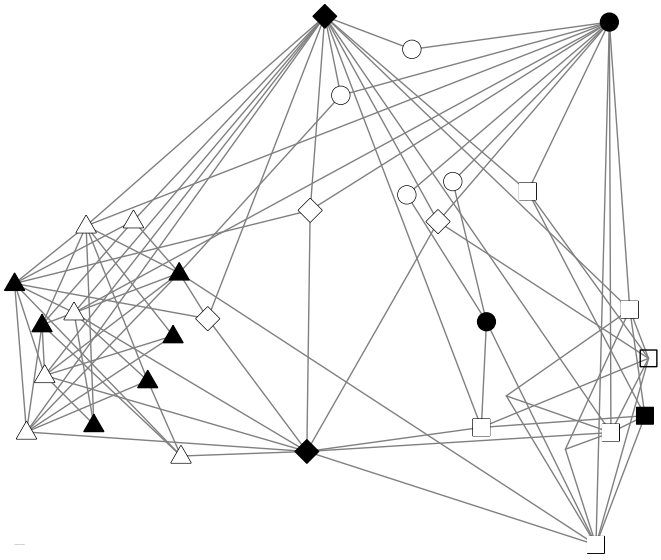


FIG. 6 Community structure in multipartite networks. This bipartite graph refers to the Southern Women Event Participation data set. Women are represented as open symbols with black labels, events as filled symbols with white labels. The illustrated vertex partition has been obtained by maximizing a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

long 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. 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[2]. 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 dis-
o58.717 3.615 Td [(29.64/F8 9.9626 Tf graphs)-316(are0 0 1 rg 0 0 1 F

A. Computational complexity

The massive amount of data on real networks currently available makes the issue of the e ciency 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

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

$$_{int}(C) = \frac{\#\text{ internal edges of } C}{n_c(n_c \quad 1)=2}: \qquad (1)$$

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

$$_{ext}(C) = \frac{\#\text{ inter-cluster edges of } C}{n_c(n \quad n_c)}: \qquad (2)$$

For C to be a community, we expect $_{int}(C)$ to be appreciably larger than the average link density (G) of G, which is given by the ratio between the number of edges of G and the maximum number of possible edges $n(n \quad 1)=2$. On the other hand, $_{ext}(C)$ has to be much smaller than (G). Searching for the best tradeo between a large

ternatives, the *n-clan* and the *n-club*. An *n*-clan is an *n*-clique whose diameter is not larger than *n*, i. e. a sub-

if it is di erent from a random graph. A random graph a la Erdos-Renyi (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. Therefore, one can de ne a *null model*, i. e. a graph which matches the original in some of its structural features, but which is otherwise a random graph. The null model is used as a term of comparison, to verify whether the graph at study displays community structure or not. The most popular null model is that proposed by Newman and Girvan and consists of a randomized version of the original graph, where edges are rewired at random, under the constraint that the expected degree of each vertex matches the degree of the vertex in the original graph (

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 in nite, but this can be avoided if one performs a weighted sum of the number of paths. For instance, paths of length l can be weighted by the factor ^{l}, with < 1. Another possibility, suggested by Estrada and Hatano (Estrada and Hatano, 2008, 2009), is to weigh paths of length l with the inverse factorial $1=l!$. In both cases, the contribution of long paths is strongly suppressed and the sum converges.

Another important class of measures of vertex similar-

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 gure with permission from Ref. (Lancichinetti *et al.*, 2009). c 2009 by IOP Publishing.

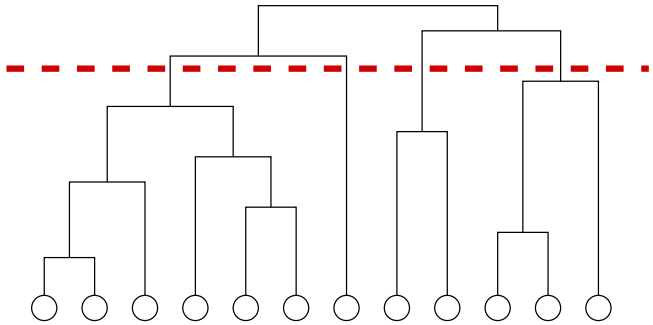


FIG. 8 A dendrogram, or hierarchical tree. Horizontal cuts correspond to partitions of the graph in communities.

does not make sense without a distance function[5], the other two are quite well de ned. 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 de nition 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 speci c 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 P of a graph

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

where C is a generic cluster of partition P. Eq. 11 states that the quality of a partition is given by the sum of the qualities of the individual clusters. The function $q(C)$

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 $P_{ij} = 2mp_i p_j = k_i k_j=2m$ of edges between i and j. So, the final expression of modularity reads

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

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

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

Here, n_c is the number of clusters, l_c the total number of edges joining vertices of module c and d_c the sum of the degrees of the vertices of c

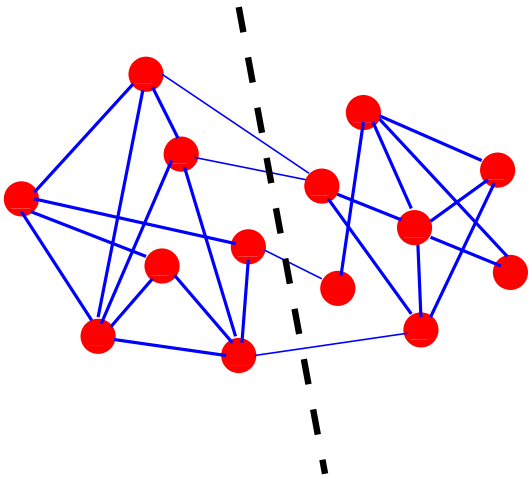


FIG. 9 Graph partitioning. The dashed line shows the solution of the minimum bisection problem for the graph illustrated, i. e. the partition in two groups of equal size with minimal number of edges running between the groups. Reprinted gure with permission from Ref. (Fortunato and Castellano, 2009).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

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.

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

Graph partitioning is a fundamental issue in parallel computing, circuit partitioning and layout, and in the design of many serial algorithms, including techniques to solve partial di erential 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 di erent boards need to be linked to each

where $_i$ is the Laplacian eigenvalue corresponding to eigenvector $\mathbf{v}_i$. It is worth remarking that the sum contains at most n 1 terms, as the Laplacian has at least one zero eigenvalue. Minimizing R equals to the minimization of the sum on the right-hand side of Eq. 19. This task is still very hard. However, if the second lowest eigenvector $_2$ is close enough to zero, a good approximation of the minimum can be attained by choosing s parallel to the corresponding eigenvector v

minimized via spectral clustering (Chan *et al.*, 1993; Hagen and Kahng, 1992) (Section IV.D).

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

cost function based on distances between points and/or from points to *centroid*s, i. e. suitably de ned 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 classi ed 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 de nes a reference point x_i, the centroid, and computes the maximum d_i

ee00.H(heref91(4(The)91(5(coss)]TJ -95.322 -11.457 Td 4(function)9372(is)937[(thg)937[otaall)9372cen)28(a-29

techniques, like *k*-means clustering (Section IV.C). One may wonder why it is necessary to cluster the points obtained through the eigenvectors, when one can directly cluster the initial set of objects, based on the similarity matrix. The reason is that the change of representation induced by the eigenvectors makes the cluster properties

non-zero elements of the eigenvectors corresponding to the connected components are proportional to the square root of the degree of the corresponding vertex. So, if degrees are very di erent from each other, and especially if there are vertices with very low degree, some eigenvector elements may be quite small. As we shall see below, in the context of the technique by Ng et al. (Ng *et al.*, 2001), a suitable normalization procedure is introduced to alleviate this problem.

graph vertices have the same or similar degrees, there is no substantial di erence between the unnormalized and the normalized Laplacians. If there are big inhomogeneities among the vertex degrees, instead, the choice of the Laplacian considerably a ects the results. In general, normalized Laplacians are more promising because the corresponding spectral clustering techniques implicitly impose a double optimization on the set of partitions, such that the intracluster edge density is high and, at

based on breadth- rst-search (Brandes, 2001; Newman and Girvan, 2004; Zhou *et al.*, 2006).

In the context of information spreading, one could imagine that signals ow 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

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[9]. Another fast version of the Girvan-Newman algorithm has been proposed by Rattigan et al. (Rattigan *et al.*, 2007). Here, a quick approximation of the edge betweenness values is carried out by using a *network structure index*, which consists of a set of vertex annotations combined with a distance measure (Rattigan *et al.*, 2006). Basically one divides the graph into regions and computes the distances of every vertex from each region. In this way Rattigan et al. showed that it is possible to lower the complexity of the algorithm to $O(m)$, by keeping a fair accuracy in the estimate of the edge betweenness values. This version of the Girvan-Newman algorithm gives good results on the benchmark graphs proposed by Brandes et al. (Brandes *et al.*, 2003) (see also Section XV.A), as well as on a collaboration network of actors and on a citation network.

Chen and Yuan have pointed out that counting all possible shortest paths in the calculation of the edge betweenness may lead to unbalanced partitions, with communities of very different size, and proposed to count only *non-redundant* paths, i. e. paths whose endpoints are all different from each other: the resulting betweenness yields better results than standard edge betweenness for mixed clusters on the benchmark graphs of Girvan and Newman (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

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

tion centrality of an edge requires the calculation of the distances between all pairs of vertices, which can be done with breadth- rst-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 nal complexity is

applying refinement strategies based on local search at various steps of the greedy agglomeration (Noack and Rotta, 2009). 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). Another good strategy consists in alternating greedy optimization with stochastic perturbations of the partitions (Mei *et al.*, 2009).

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. 35) coming from putting i in the community of its neighbor j and picks the community of the neighbor that yields the largest increase of Q, as long as it is positive. At the end of the sweep, one obtains the first level partition. In the second step communities are replaced by supervertices, and two supervertices are connected if there is at least an edge between vertices of the corresponding communities. In this case, the weight of the edge between the supervertices is the sum of the weights of the edges between the represented communities at the lower level. The two steps of the algorithm are then repeated, yielding new hierarchical levels and supergraphs (Fig. 12). We remark that modularity is always computed from the initial graph topology: operating on supergraphs enables one to consider the variations of modularity for partitions of the original graph after merging and/or splitting of groups of vertices. Therefore, at some iteration, modularity cannot increase 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 $\exp(-\beta F)$, where F is the decrease of the function and β 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 β is varied. Simulated annealing was first employed for modularity optimization by Guimera et al. (Guimera *et al.*, 2004). Its standard implementation (Guimera 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

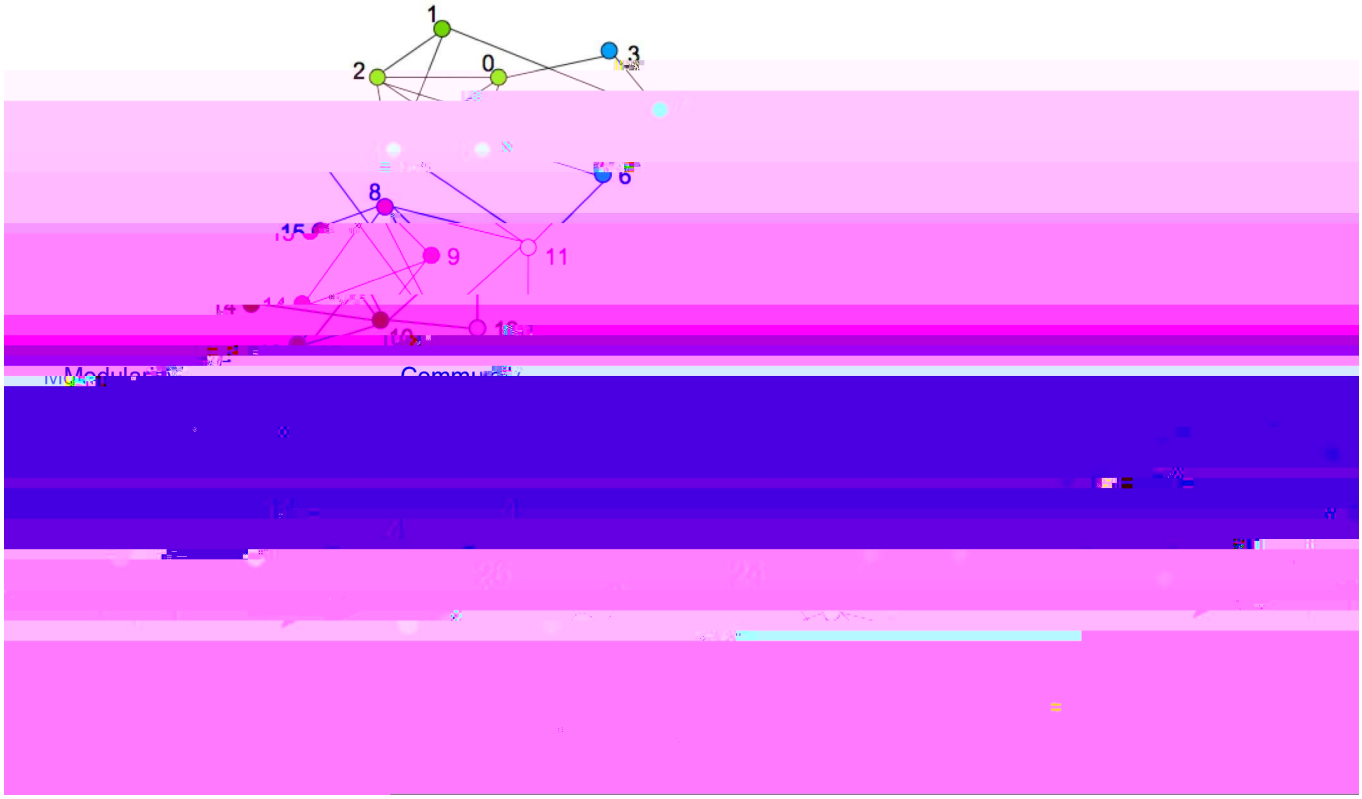
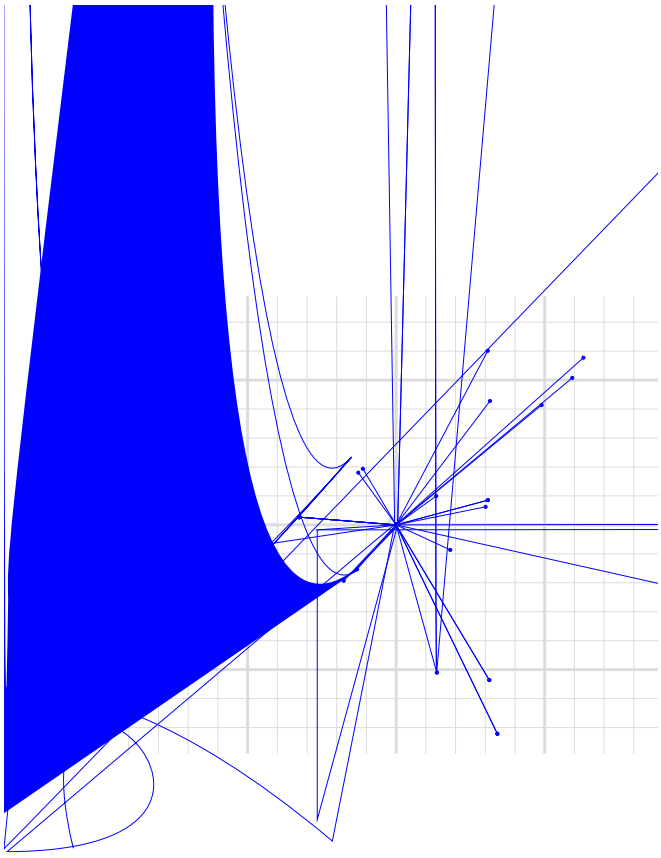


FIG. 12 Hierarchical optimization of modularity by Blondel et al. (Blondel *et al.*, 2008



partitions keeps growing. The advantage of *Kcut* is that one can play with low values for the (maximal) number of clusters ` at each iteration; if partitions are balanced, after *a* levels of recursions, the number of clusters of the partition is approximately $K = `^a$. Therefore the complexity of *Kcut* is $O[(n + m) \log K]$ for a nal partition in (at most) K clusters, which is much lower than the complexity of the algorithm by White and Smyth. Ruan and Zhang tested *Kcut* on arti cial graphs generated with the planted `-partition model (Section XV), and on real networks including Zachary's karate club (Zachary, 1977), the American college football network (Girvan and Newman, 2002) and two collaboration networks of Jazz musicians (Gleiser and Danon, 2003) and physicists (

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 tness 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 tness of the new pool of candidates is computed and the chromosomes with the highest tness have the greatest chances to survive in the next generation. After several iterations only solutions with large tness survive. In a work by Tasgin et al. (Tasgin *et al.*, 2007), partitions are the chromosomes and modularity is the tness 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 XV.A). Genetic algorithms were also adopted by Liu et al. (Liu *et al.*, 2007). Here the maximum modularity partition is obtained via successive bipartitions of the graph, where each bipartition is determined by applying a genetic algorithm to each subgraph (starting from the original graph itself), which is considered isolated from the rest of the graph. A bipartition is accepted only if it increases the total modularity of the graph.

In Section III.C.2 we have seen that the modularity maximum is obtained for the partition that minimizes the di erence between the cut size and the expected cut size of the partition (Eq. 17). In the complete weighted graph 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 G, and $-k_i k_j/2m$ if they are not, the di erence $|\mathrm{Cut}_P| - \mathrm{ExCut}_P$ is simply the cut size of partition P. So, maximizing modularity for G is equivalent to the problem of nding the partition with minimal cut size of the weighted graph G_w, i. e. to a graph partitioning problem. The problem can then be e ciently solved by using existing software for graph partitioning (Djidjev, 2007).

B. Modi cations of modularity

In the most recent literature on graph clustering several modi cations and extensions of modularity can be found. They are usually motivated by speci c 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. 14 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. 14 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

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^0, as well as B and B^0

based on the results of the optimization. Membership coefficients are also present in an extension of modularity to overlapping communities proposed by Shen et al. (Shen *et al.*, 2009). Here the membership coefficient of vertex v in community c is a sum over the edges of v belonging to c, where each edge has a weight proportional to the number of maximal cliques of c containing the edge.

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

$$S_{cov} = \frac{\sum_{c=1}^{n_c} l_c/m}{\sum_{c=1}^{n_c} (d_c/2m)^2}; \qquad (44)$$

where the notation is the same as in Eq. 15. 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_{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

edges in the graph, which in many graph realizations is not homogeneous even if the linking probability is constant, like in Erdos-Renyi graphs. The uctuations determine concentrations of links in some subsets of the graph, which then appear like communities. According to the de nition 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 signi cant community structure only if it is appreciably larger than the modularity maximum of random graphs of the same size and

are often possible as well. In this section we shall review recent spectral techniques proposed mostly by physicists explicitly for graph clustering.

Early works have shown that the eigenvectors of the *transfer matrix* 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

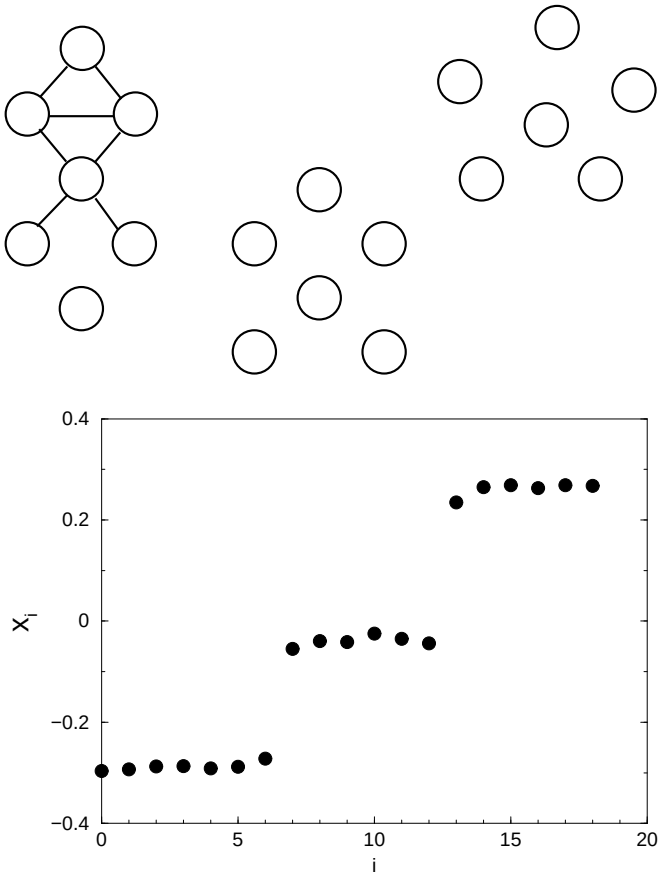


FIG. 18 Basic principle of the spectral algorithm by Capocci et al. (Capocci *et al.*, 2005). The bottom diagram shows the

state of the system may not be the one where all spins are aligned, but a state where di erent 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 et al. (Blatt *et al.*, 1996), Reichardt and Bornholdt proposed a method to detect communities that maps the graph onto a zero-temperature q-Potts model with nearest-neighbour interactions (Reichardt and Bornholdt, 2004). The Hamiltonian of the model, i. e. its energy, reads

$$H = J \sum_{i;j} A_{ij} \, (\ _i; \ _j) + \sum_{s=1}^{q} \frac{n_s(n_s \quad 1)}{2}; \qquad (54)$$

where A_{ij} is the element of the adjacency matrix, is Kronecker's function, n_s the number of spins in state s, J and are coupling parameters. The energy H is the sum of two competing terms: the rst is the classical ferromagnetic Potts model energy, and favors spin alignment; the second term instead peaks when the spins are homogeneously distributed. The ratio $=J$ expresses the relative importance of the two terms: by tuning $=J$ one can explore di erent levels of modularity of the system, from the whole graph seen as a single cluster to clusters consisting of individual vertices. If $=J$ is set to the value (G) of the average density of edges of the graph G, the energy of the system is smaller if spins align within subgraphs such that their internal edge density exceeds (G), whereas the external edge density is smaller than (G), i. e. if the subgraphs are clusters (Section III.B.1). The minimization of H is carried out via simulated annealing ((Kirkpatrick *et al.*, 1983) and Section VI.A.2), starting from a con guration 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 su ciently 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 H can be rewritten as

$$H = \sum_{i<j} (\ _i; \ _j)(\quad A_{ij}); \qquad (55)$$

which is the energy of an in nite-range Potts spin glass, as all pairs of spins are interacting (neighboring or not) and there may be both positive and negative couplings. 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 A with the weight matrix W in Eq. 54. Ispolatov et al. (Ispolatov *et al.*, 2006) have adopted a similar Hamiltonian as in Eq. 54, with a tunable antiferromagnetic term interpolating between the corresponding term of Eq. 54 and the entropy term (proportional to $n_s \log n_s$) of the free energy, whose minimization is equivalent to nding the states of the nite-temperature Potts model used by Blatt et al. (Blatt *et al.*, 1996). Eq. 55 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.

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 W, the Hamiltonian of the FRFIM on the graph is

$$H = \frac{1}{2} \sum_{i;j} W_{ij} \ _i \ _j \quad \sum_i B_i \ _i: \qquad (56)$$

In Eq. 56 $_i$ =

can become as low as $O(n)$, which enables the analysis of systems with millions of vertices. Tests on Barabasi-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 nd 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 de ne 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 de nes 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

dure 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 identity and adjacency matrix, respectively. The idea is that the vector $\mathbf{u_s}$ describes the in uence that vertex s exerts on the graph through signaling. Vertices of the same community are expected to have similar in uence on the graph and thus to correspond to vectors which are \close" in space. The vectors are nally 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 di erent communities. The signaling process is similar to di usion, but with the important di erence that here there is no ow 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 $O[T(\langle k\rangle i + 1)n^2]$, where $\langle k\rangle i$ is the average degree of the graph. Like in the previous algorithm by Latapy and Pons (Latapy and Pons, 2005), nding 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* $\mathbf{R}_t$, which, for a partition of the graph in c clusters, is de ned as

$$\mathbf{R}_t = \mathbf{H}^T(\quad \mathbf{M}^t \quad {}^T)\mathbf{H}: \tag{57}$$

Here, $\mathbf{H}$ is the n 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; the diagonal matrix whose elements are the stationary probabilities of the random walk, i. e. $_{ii} = k_i/2m$, k_i being the degree of vertex i; is the vector whose entries are the diagonal elements of . 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. de ned the *stability of the clustering*

$$r(t;\mathbf{H}) = \min_{0\ s\ t} \sum_{i=1} (R_s)_{ii} = \min_{0\ s\ t} \mathrm{trace}[R_s]: \tag{58}$$

The aim is then, for a given time t, nding 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). The cut size of the partition (Section IV.A) equals $[r(0) \quad r(1)]$, so it is also a one-step

in p steps (di usion ow). The second step, which has no physical counterpart, consists in raising each single entry of the matrix M to some power , where is now real-valued. This operation, called *in ation*, enhances the weights between pairs of vertices with large values of the di usion ow, 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 transfer 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 communities 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 nal partition is sensitive to the parameter used in the in ation step. Therefore several di erent 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

tions 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 *t* the actual graph topology. In this section we review those clustering techniques attempting to nd the best t of a model to the graph, where the model assumes that vertices have some sort of classi cation, based on their connectivity patterns. We mainly focus on methods adopting *Bayesian inference* (Winkler, 2003), in which the best t is obtained through the maximization of a likelihood (*generative models*), but we also discuss related techniques, based on

fraction of vertices in group r, and $_{ri}$

unable to provide. Another technique similar to that by Newman and Leicht has been designed by Ren et al. (Ren *et al.*, 2009). The model is based on the group fractions $f_i g$, de ned as above, and a set of probabilities $f_{r;i} g$, 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 Copic et al. to de ne an axiomatization of the problem of graph clustering and its related concepts (Copic *et al.*, 2005). The starting point is again the planted partition model (Section XV), with probabilities p_{in} and p_{out}. A novelty of the approach is the introduction of the *size matrix* S, whose element S_{ij} indicates the maximum strength of interaction between vertices i and j. For instance, in a graph with unweighted connections, all elements of S equal 1. In this case, the probability that the graph conceals a community structure coincides with the expression (63) by Hastings. Copic et al. used this probability as a quality function to de ne rankings between graph partitions (*likelihood rankings*). The authors show that the likelihood rankings satisfy a number of general properties, which should be satis ed by any reasonable ranking. They also propose an algorithm to nd 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 speci ed which pairs of vertices stay in the same community and which pairs instead stay in di erent 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 classi ed in the same group, and that j and k are classi ed in the same group, but that i and k are not classi ed as belonging to the same group. We believe that the work by Copic et al. is an important rst 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 Ref. (Hastings, 2006), they use the planted partition model to represent a graph with community structure; as in Ref. (Newman and Leicht, 2007), they maximize the classi cation likelihood using an expectation-maximization algorithm (Dempster *et al.*, 1977). The algorithm runs for a xed 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 Classi cation Likelihood introduced by Biernacki et al. (Biernacki *et al.*, 2000). The time complexity of the algorithm is $O(n^2)$.

B. Blockmodeling, model selection and information theory

Block modeling is a common approach in statistics and

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. 21). 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 XIII. 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,

$$H(X|Y) = \log \left[\prod_{i=1}^{q} n_i^{(n_i - 1)/2} \Big/ \prod_{i>j} l_{ij}^{n_i n_j} \right]; \quad (69)$$

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 because it does not correspond to a compression of information with respect to the original data set. One has to look for the ideal tradeo between a good compression and a small enough information $H(X|Y)$. The Minimum Description Length (MDL) principle (Grunwald *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

FIG. 21 Basic principle of the method by Rosvall and Bergstrom (Rosvall and Bergstrom, 2007). An encoder sends to a

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 de ned 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 de nition 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 di erent partitions starting from the same initial condition, with di erent 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 di erent 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

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 de ned as

proximates the graph at study, where the goodness of the approximation is expressed by the distance between the corresponding matrices. In this way the original problem of finding graph subsets becomes an optimization problem. Long et al. called this procedure *Community Learning by Graph Approximation* (CLGA). Sometimes the minimization of the matrix distance can be turned into the maximization of the trace of a matrix. Measures like cut size or ratio cut can be also formulated as the trace of matrices (see for instance Eq. 18). In fact, CLGA includes traditional graph partitioning as a special case (Section IV.A). Long et al. designed three algorithms for CLGA: two of them seek for divisions of the graph into overlapping or non-overlapping groups, respectively; in the third one an additional constraint is introduced to produce groups of comparable size. The complexity of these algorithms is $O(tn^2k)$, where t is the number of iterations until the optimization converges and k the number of groups. The latter has to be given as an input, which is a serious limit of CLGA.

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 solution would be too time consuming,

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.

Despite the signi cant improvements in computational complexity, it is still problematic to apply clustering algorithms to many large networks available today. Therefore Narasimhamurthy et al. (Narasimhamurthy *et al.*, 2008) proposed a two-step procedure: rst, the graph at study is decomposed in smaller pieces by a fast graph partitioning technique; then, a clustering method is applied to each of the smaller subgraphs obtained [Narasimhamurthy et al. used the Clique Percolation Method (Section XI.A)]. The initial decomposition of the graph is carried out through the multilevel method by Dhillon et al. (Dhillon *et al.*, 2007). It is crucial to verify that the initial partitioning does not split the communities of the graph among the various subgraphs of the decomposition. This can be done by comparing, on arti cial graphs, the nal clusters obtained with the two-step method with those detected by applying the chosen clustering technique to the entire graph.

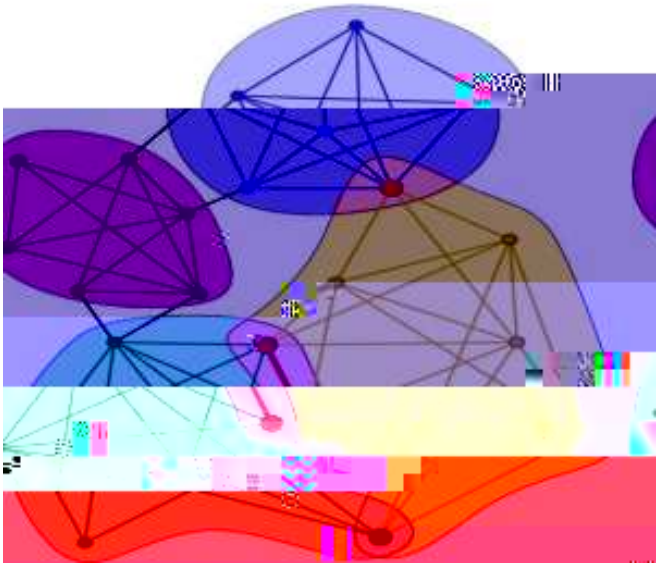


FIG. 23 Clique Percolation Method. The example shows communities spanned by adjacent 4-cliques. Overlapping vertices are shown by the bigger dots. Reprinted gure with permission from Ref. (Palla *et al.*, 2005). c 2005 by the Nature Publishing Group.

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 a 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[18]. Notice that a *k*-clique is di erent 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. 23. One could say that a *k*-clique community is identi ed by

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

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. Derenyi et al. (Derenyi *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 Erdős-Renyi 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 signi cance of the clusters found with the CPM, one can compare the detected cover[19] 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 Erdős-Renyi 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 uctuations. 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, de ned 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. de ned *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 nd 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 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 , 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 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

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

works. 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. A more fundamental issue is the fact that the method does not look for actual communities, consistent with the shared notion of dense subgraphs, but for subgraphs \containing" many cliques, which may be quite di erent objects than communities (for instance, they could be \chains" of cliques with low internal edge density). Another big problem is that on real networks there is a considerable fraction of vertices that are left out of the communities, like leaves. One could think of some post-processing procedure to include them in the communities, but for that it is necessary to introduce a new criterion, outside the framework that inspired the method. 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 de nition of directed k-cliques are rather arbitrary.

B. Other techniques

One of the rst methods to nd overlapping communities was designed by Baumes et al. (Baumes *et al.*, 2005b). A community is de ned as a subgraph which locally optimizes a given function W, typically some measure related to the edge density of the cluster[20]. Di erent overlapping subsets may all be locally optimal, so vertices can be shared between communities. Detecting the cluster structure of a graph amounts to nding the set of all locally optimal clusters. Two e cient 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 identi ed 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 re ne 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 di erent 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 every 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, $u_{ik} = {}_{k_i k}$, 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 di erent values of n_c. This scheme has been used in other techniques as well, like in the algorithm of Donetti and Muñoz (Donetti and Muñoz, 2004) (Section VII). The rst 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 right stochastic matrix W (Section A.2), derived from the adjacency matrix 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, 1974) (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}, de ned as

$$Q_{ov}^{zh} = \quad ^{=}$$

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

the contribution of the edges' weights to the sums in W_c and S_c by the (average) membership coe cients of the vertices of the edge. 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. $O(K^2n + Km)$, which is essentially linear in n if the graph is sparse and $K \quad n$.

Nepusz et al. proposed a di erent approach based on vertex similarity (Nepusz *et al.*, 2008). One starts from the membership matrix $\mathbf{U}$, de ned as in the previous method by Zhang et al. From $\mathbf{U}$ a matrix $\mathbf{S}$ is built, where $s_{ij} = \sum_{k=1}^{n_c} u_{ik}u_{jk}$, 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 $\mathbf{S}$, the best cover is obtained by choosing $\mathbf{U}$ such that $\mathbf{S}$ approximates as closely as possible $\mathbf{S}$. This amounts to minimize the function

$$D_G(\mathbf{U}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(s_{ij} \quad s_{ij})^2; \qquad (74)$$

74e1987i6 Tf 3.875 0 Td [(U)]TJ/74 es.457 Td [(vious)0TJ -222.4997j

gra,(p)-5 [(one)74(sets 3.875 0 Td [(U)]TJ699702es.457 Td [(vious)0TJ -222.4997)267 -1.494 Td [(ij)]TJ/F8 9.9626 T9.7.386 1.4 gd6cienbaseberina8(v)27ebmethod,291(lik(v)28(e)-11[(sec-)]TJ 0 -11.457 Td ulat(ned)-233(lin7(lgge.71196(The)-23optinimination42

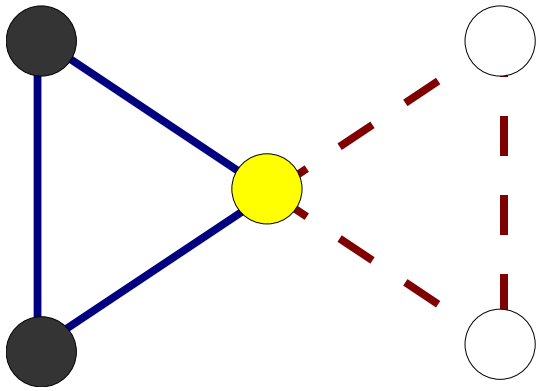


FIG. 24 Communities as sets of edges. In the gure, the graph has a natural division in two triangles, with the central vertex shared between them. If communities are identi ed by their internal edges, detecting the triangles and their overlapping vertex becomes easier than by using methods that group vertices. Reprinted gure with permission from Ref. (Evans and Lambiotte, 2009). c 2009 by the American Physical Society.

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]TJ 0 g cluf664 Tf 9.216 results.

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

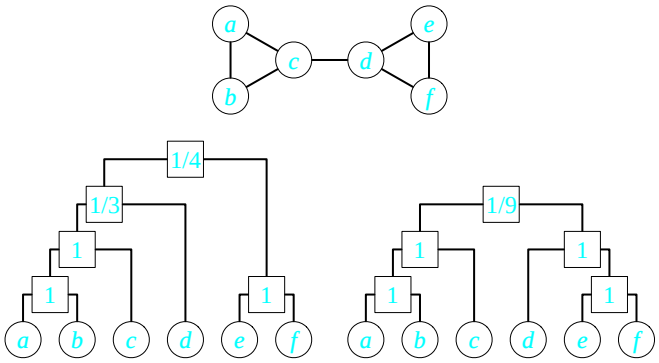
In general, multiresolution methods have a freely tunable parameter, that allows to set the characteristic size of the clusters to be detected. The general spin glass framework by Reichardt and Bornholdt ((Reichardt and Bornholdt, 2006a) and Section VI.B) is a typical example, where is the resolution parameter. The extension

plot of the number of clusters versus r (Fig. 25). 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. 25

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, 2008; Ronhovde and Nussinov, 2009)) 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 re ect the actual hi-


a
b
c
d
e
f
1/4
1/3
1
1
1
a
b
c
d
e
f
1/9
1
1
1
1
a
b
c
d
e
f

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 are conceptually similar to the *stable* communities we will see in Section XIV. 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 hierarchi-

FIG. 28 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 Ref. (Palla *et al.*, 2007). c 2007 by the Nature Publishing Group.

of the graph: the *stability index* (measuring the tendency of a vertex to interact with the same vertices over time), the *sociability index* (measuring the number of different interactions of a vertex, basically the number of *Join* and *Leave* events), the *popularity index* (measuring the number of vertices attracted by a cluster in a given time interval) and the *influence index* (measuring the influence a vertex has on the others, which is computed from the number of vertices that leave or join a cluster together with the vertex). Applications on a coauthorship network of computer scientists and on a network of subjects for clinical trials show that the behavioral measures above enable one to make reliable predictions about the time evolution of such graphs (including, e. g., the inference of missing links (Liben-Nowell and Kleinberg, 2003)).

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 (Grunwald *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 partition for each graph segment (high modularity[27] 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

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

well de ned. Fenn et al. (Fenn *et al.*, 2009) 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 is xed to the value that occurs in most stability plateaus of the system at di erent time steps. Motivated by the work of Guimera and Amaral (Guimera and Amaral, 2005) (Section XVI), 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), de ned in Eq. 98, and z^{b} the z-score of the site betweenness, de ned by replacing the internal degree with the site betweenness of Freeman (Freeman, 1977) in Eq. 98. We remind that the site betweenness is a measure of the number of shortest paths running through a vertex. The variable z^{b}

(for LiveJournal) with their degree of interconnectedness. Moreover, the probability of growth of LiveJournal communities is positively correlated to a combination of factors including the community size, the number of friends of community members which are not in the community and the ratio of these two numbers. A high density of triads within a community appears instead to hinder its growth.

XIV. 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 necessarily 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? 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 $0 < \quad < 1$. This choice also allows to account for the possible effects of uncertainties in the values of the edge weights, resulting from measurements/experiments carried out on a given system. After fixing (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 (usually 0:8) are called *external edges*. The stability of a partition is estimated through the *clustering entropy*

$$S = \frac{1}{m} \sum_{(i;j):A_{ij}=1} [p_{ij} \log_2 p_{ij} \quad (1 \quad p_{ij}) \log_2 (1 \quad p_{ij})]; \tag{84}$$

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. 97). Unstable vertices end up in components

FIG. 29 Application of the *C*-score by Lancichinetti et al. (Lancichinetti *et al.*, 2009) to identify modules within subgraphs. In (a) the subgraph consists of a compact cluster (generated with the LFR benchmark (Lancichinetti and Fortunato, 2009; LancichinettiIn

limit of the eld. 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 in nite, whereas it is not clear what a

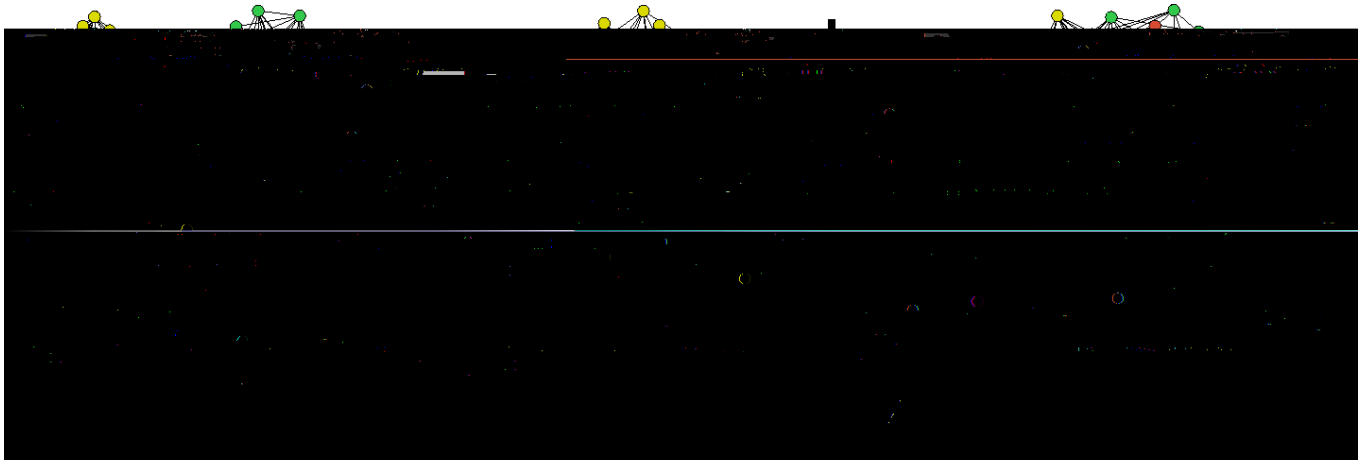
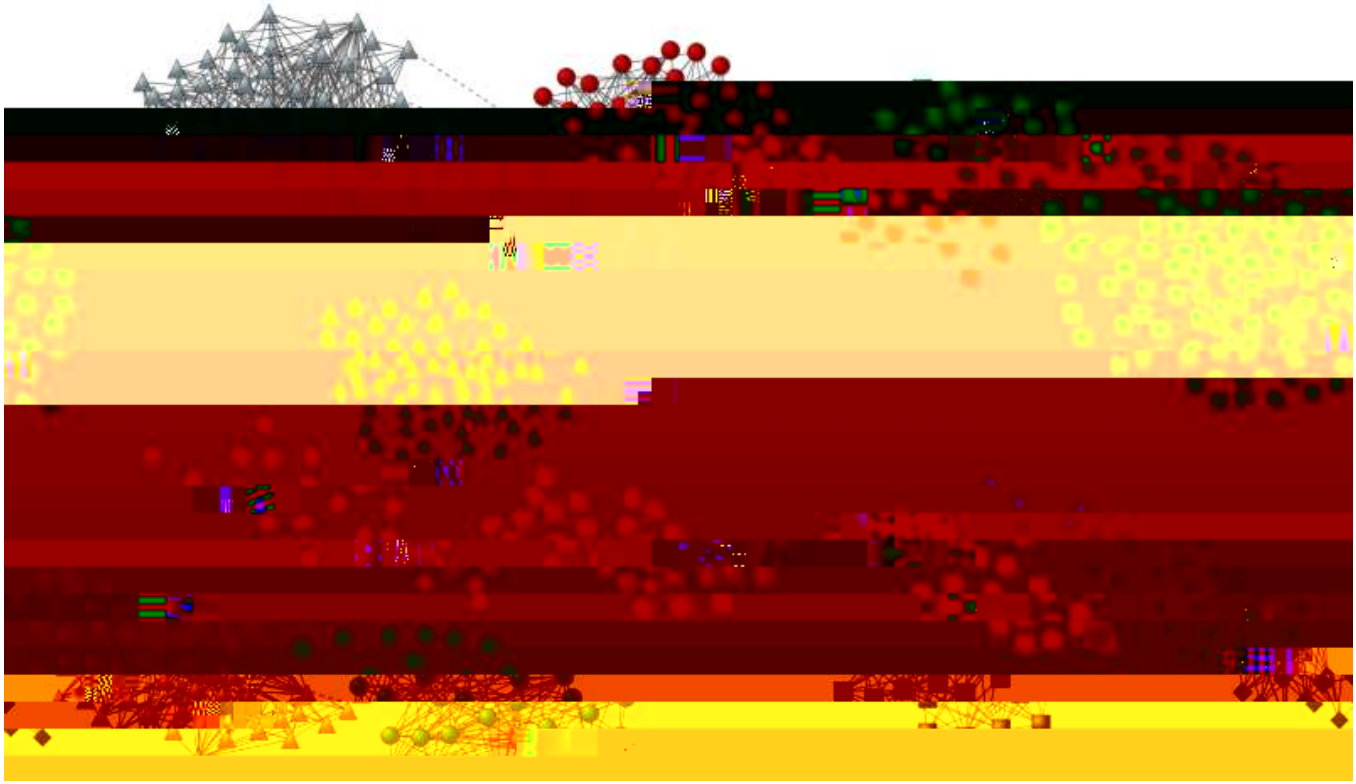


FIG. 30 Benchmark of Girvan and Newman. The three pictures correspond to z_{in} = 15 (a), z_{in} = 11 (b) and z_{in} = 8 (c). In (c) the four groups are hardly visible. Reprinted gure with permission from Ref. (Guimera and Amaral, 2005). c 2005 by the Nature Publishing Group.

a la Erdos-Renyi. 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 XVI. So, the planted `-partition model is not a good description of a real graph with community structure. However, the model can be modi ed to account for the heterogeneity of degrees and community sizes. A modi ed 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 di er 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. (LFR benchmark) (Lancichinetti *et al.*, 2008). They assume that the distributions of degree and community size are power laws, with exponents $_1$ and $_2$, respectively. Each vertex shares a fraction 1



comprises all vertices of the subgroups C_i^A and C_i^B, respectivelysp

lace, 1983) proposed the two indices

$$W_I = \frac{a_{11}}{\sum_k n_k^X (n_k^X \quad 1)=}$$

ison of proper partitions without overlap, even though the values are close.

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

$$V(X;Y) = H(X|Y) + H(Y|X); \qquad (94)$$

which has some desirable properties with respect to the normalized mutual information and other measures. In particular, it de nes 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 di ering only in a small portion of a graph depends on the di erences 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 di erent size cannot be compared with each other. For meaningful comparisons one could divide $V(X;Y)$ 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. de ned two distance measures for partitions (Gustafsson *et al.*, 2006). They are both based on the concept of *meet* of two partitions, which is de ned as

$$M = \{X_i \cap Y_j : \qquad (95)$$

The distance measures are m_{moved} and m_{div}. In both cases they are determined by summing the distances of X and Y from the meet M. For m_{moved} the distance of X (Y) from the meet is the minimum number of elements that must be moved between X and Y so that X (Y) and M coincide (Gus eld, 2002). For m_{div} the distance of X (Y) from the meet is the minimum number of divisions that must be done in X (Y) so that X (Y) and M coincide (Stanley, 1997). Such distance measures can easily be transformed in similarity measures, like

$$I_{moved} = 1 - m_{moved}/n; \qquad I_{div} = 1 - m_{div}/n: \qquad (96)$$

Identical partitions have zero mutual distance and similarity 1 based on Eqs. 96.

Finally an important problem is how to de ne the sim-

and/or overlapping communities (Lancichinetti and Fortunato, 2009) were carried out. Lancichinetti and Fortunato also tested the methods on random graphs, to check whether they are able to notice the absence of community structure. From the results of all tests, the Infomap method by Rosvall and Bergstrom (Rosvall and Bergstrom, 2008) appears to be the best, but also the algorithms by Blondel et al. (Blondel *et al.*, 2008) and by Ronhovde and Nussinov (Ronhovde and Nussinov, 2009) have a good performance. These three methods are also very fast, with a complexity which is essentially linear in the system size, so they can be applied to large systems. On the other hand, modularity-based methods (with the exception of the method by Blondel et al.) have a rather poor performance, which worsens for larger systems and smaller communities, due to the well known resolution limit of modularity (

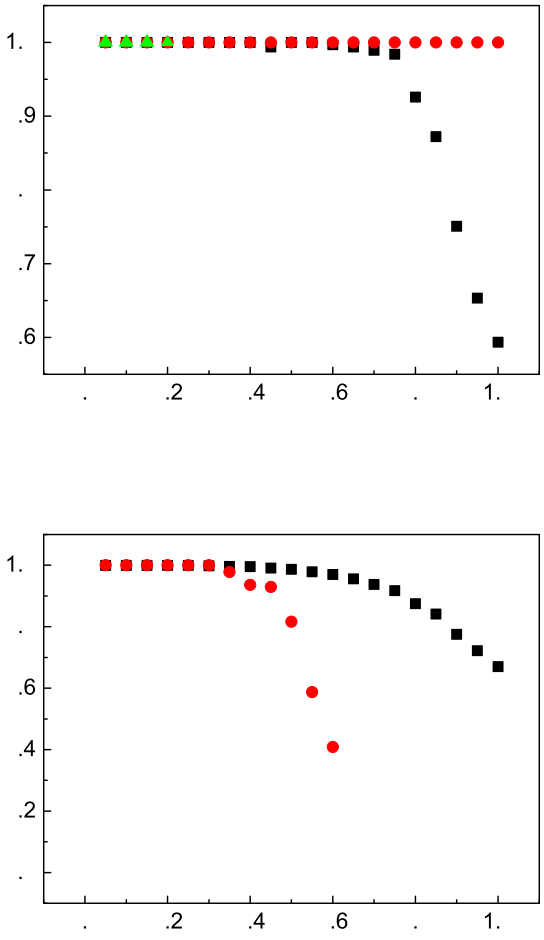


FIG. 34 Comparative evaluation of the performances of algorithms to nd communities in weighted graphs. Tests are carried out on a weighted version of the benchmark of Girvan and Newman. The two plots show how good the algorithms are in terms of the precision and accuracy with which they recover the planted 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. 97. 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 gures with permission from Ref. (Fan *et al.*, 2007). c 2007 by Elsevier.

scribed this benchmark in Section XV.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 nd overlapping communities, has a poor performance, as the overlapping vertices found by the algorithm are in general di erent 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 di erent, 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 identi ed, 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 con gurations in which the graphs have a cluster structure, that an algorithm should then be able to resolve, from con gurations 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 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.

XVI. 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

FIG. 35 Cumulative distribution of community sizes for the Amazon purchasing network. The partition is derived by greedy modularity optimization. Reprinted gure with permission from Ref. (Clauset *et al.*, 2004). c 2004 by the American Physical Society.

variety of clustering methods that one could employ, in many cases partitions derived from di erent 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.

One of the rst 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 nd 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. 35 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 if B was frequently purchased by bupd [(ok1ged [(if)]TJ/F11 9.962657.8531.555 0 Td [(A)]TJ/F8 9.9627.4721.555 0 n.c8ng)Rec62(smok1g
distriborin th7 Tu2(comm)29(ulativ)2248 42(distribut2re)-4472re pow28(o)[(if)]TJ/F11 9.962ifif

FIG. 36 Analysis of communities in large real networks by Leskovec et al. (Leskovec *et al.*, 2008). (Left) Typical shape of the network community pro le plot (NCPP), showing how the minimum conductance of subgraphs of size n varies with n. The plot indicates that the \best" communities have a size of about 100 vertices (minimum of the curve), whereas communities of larger sizes are not well-de ned. In the plot two other NCPPs are shown: the one labeled *Rewired network* corresponds to a randomized version of the network, where edges are randomly rewired by keeping the degree distribution; the one labeled *Bag of whiskers* gives the minimum conductance scores of clusters composed of disconnected pieces. (Right) Scheme of the core-periphery structure of large social and information networks derived by Leskovec et al. based on the results of their empirical analysis. Most of the vertices are in a central core, which does not have a clear community structure, whereas the best communities, which are rather small, are weakly connected to the core. Reprinted gure with permission from Ref. (Leskovec *et al.*, 2008).

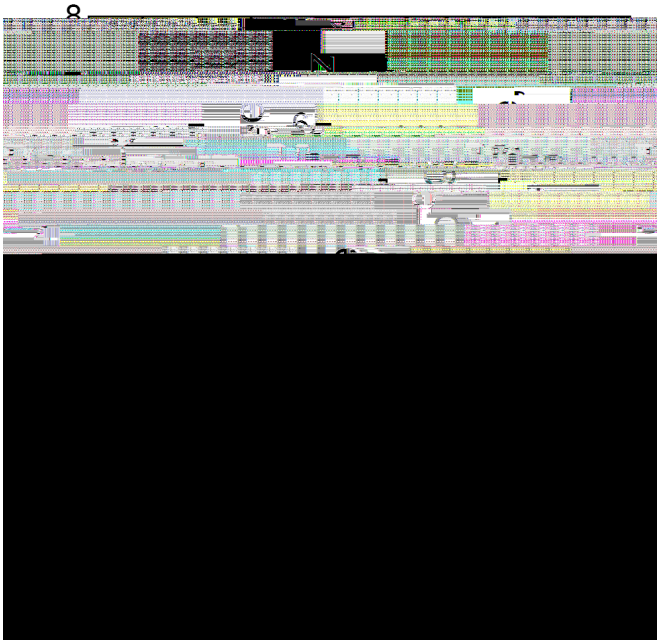


FIG. 37 Regions of the z P plane de ning the roles of vertices in the modular structure of a graph, according to the scheme of Guimera and Amaral (Guimera and Amaral, 2005; Guimera and Amaral, 2005). Reprinted gure with permission from Ref. (Guimera and Amaral, 2005). c 2005 by the Nature Publishing Group.

81(ha)6 Tf 13.618 10.Eaole 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 de ned as

$$z_i = \frac{i \quad s_i}{s_i}; \qquad (98)$$

random graphs and Barabasi-Albert (Barabasi and Albert, 1999) graphs (Section A.3), non-hubs are mostly kinless vertices. In addition, if there are hubs, like in Barabasi-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 (Guimera and Amaral, 2005).

If communities are overlapping, one can explore other

ters 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 modi ed de - nition 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 pro les. 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

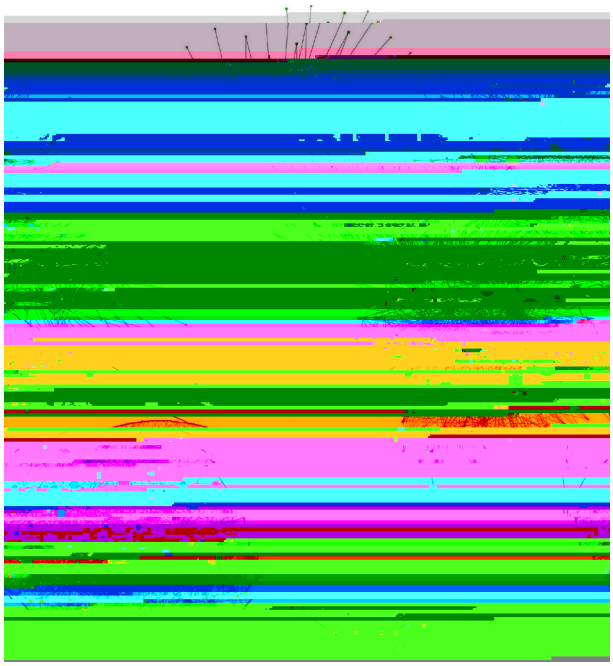


FIG. 38 Community structure of a social network of mobile phone communication in Belgium. Dots indicate subcommunities at the lower hierarchical level (with more than 100 people) and are colored in a red-green scale to represent the level of representation of the two main languages spoken in Belgium (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 gure with permission from Ref. (Blondel *et al.*, 2008). c 2008 by IOP Publishing and SISSA.

communications between users of a Belgian phone operator (Blondel *et al.*, 2008). The vertices of the graph are 2.6 millions and the edges are weighted by the cumulative

of modularity (Section VI.A.4): the results were further re ned through additional steps a la Kernighan-Lin (Section IV.A). One of the goals of the study was to infer relationships between the online and o ine lives of the students. By using demographic information on the students' populations, one nds that communities are organized by class year or by House (dormitory) a liation, depending on the university (Fig. 39). Yuta et al. (Yuta *et al.*, 2007) observed a gap in the community size distribution of a friendship network extracted from *mixi* (`mixi.jp`), the largest social networking site in Japan (as of December 2006). Communities were identi ed with the fast greedy modularity optimization by Clauset et al. (Clauset *et al.*, 2004). The gap occurs

FIG. 40 Map of science derived from a clustering analysis of a citation network comprising more than 6000 journals. Reprinted gure with permission from Ref. (Rosvall and Bergstrom, 2008). c 2008 by the National Academy of Science of the USA.

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, identi ed with a modi cation of Newman's spectral optimization of modularity (Section VI.A.4), are correlated with party a liation, but also with geography and committee memberships of the legislators.

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 satis ed. But this is by no means the only possibility. A community can be generically de ned 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

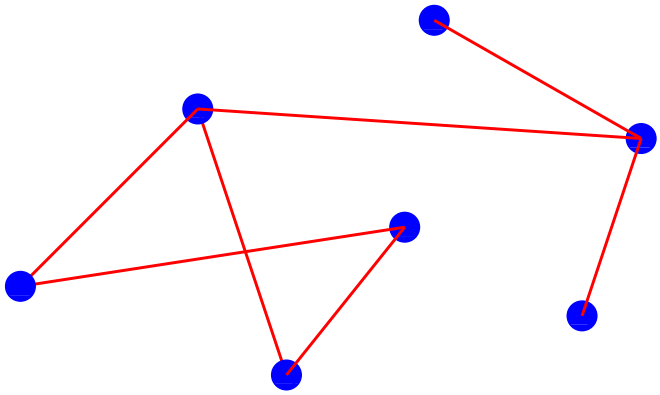


FIG. 41 A sample graph with seven vertices and seven edges.

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

two disjoint subsets V_1 and V_2, or *classes*, and every edge joins a vertex of

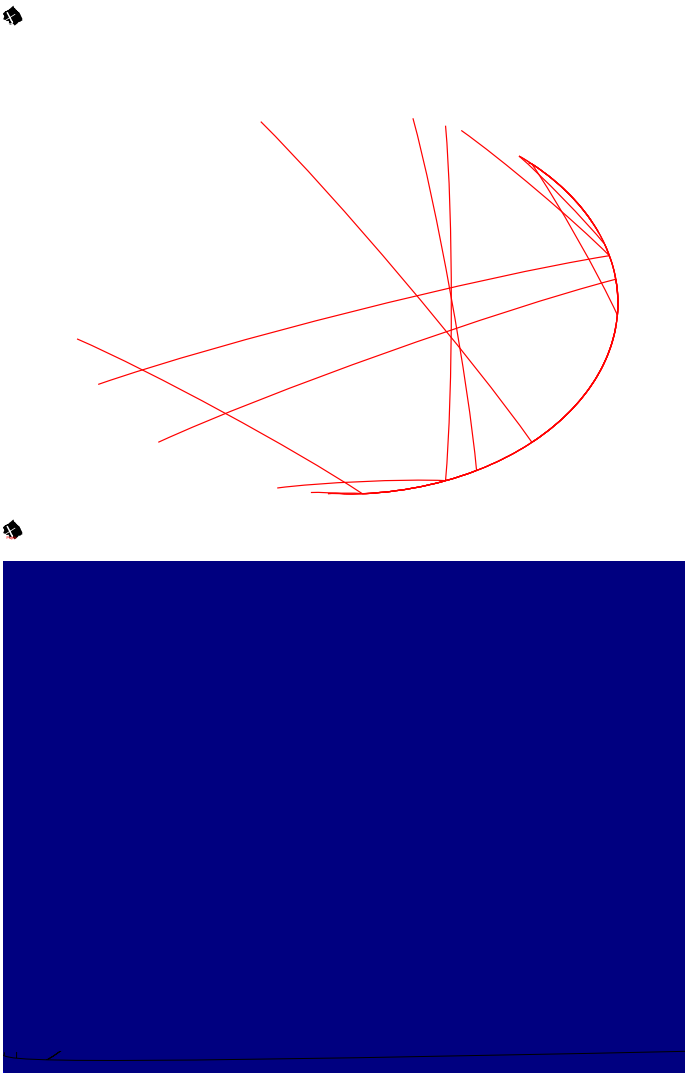


FIG. 42 Basic models of complex networks. (Top) Erdos-Renyi random graph with 100 vertices and a link probability $p = 0.02$. (Center) Small world graph a la Watts-Strogatz, with 100 vertices and a rewiring probability $p = 0.1$. (Bottom) Barabasi-Albert scale-free network, with 100 vertices

Boccaletti *et al.*, 2006; Mendes and Dorogovtsev, 2003;

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,

80(1), 016114.
Good, B. H., Y. de Montjoye, and A. Clauset, 2009, eprint arXiv:0910.0165.
Gori, M., and A. Pucci, 2007, in *IJCAI'07: Proceedings of the 20th international joint conference on Arti cal intelligence*

ference on Knowledge discovery and data mining (ACM, New York, NY, USA), pp. 611{617.

Kumpula, J. M., M. Kivela, K. Kaski, and J. Saramaki, 2008, Phys. Rev. E **78**(2), 026109.

Kumpula, J. M., J. Saramaki, K. Kaski, and J. Kertesz, 2007a, in *Noise and Stochastics in Complex Systems and Finance*, volume 6601 of *SPIE Conference Series*, p. 660116.

Kumpula, J. M., J. Saramaki, K. Kaski, and J. Kertesz, 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, Phys. Rev. E **80**(1), 016118.

Lancichinetti, A., and S. Fortunato, 2009, Phys. Rev. E **80**(1), 056117.

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.

Lancichinetti, A., F. Radicchi, and J. J. Ramasco, 2009, eprint arXiv:0907.3708.

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.

Leskovec, J., L. Backstrom, R. Kumar, and A. Tomkins, 2008, in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, New York, NY, USA), pp. 462{470.

Leskovec, J., J. Kleinberg, and C. Faloutsos, 2005, in *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (ACM, New York, NY, USA), pp. 177{187.

Leskovec, J., K. J. Lang, A. Dasgupta, and M. W. Mahoney, 2008, eprint arXiv:0810.1355.

Leung, I. X. Y., P. Hui, P. Lio, and J. Crowcroft, 2009, Phys. Rev. E **79**(6), 066107.

Lewis, A. C. F., N. S. Jones, M. A. Porter, and C. M. Deane, 2009, eprint arXiv:0904.0989.

Li, D., I. Leyva, J. A. Almendral, I. Sendina-Nadal, J. M. Buldu, S. Havlin, and S. Boccaletti, 2008a, Phys. Rev. Lett. **101**(16), 168701.

Li, Z., S. Zhang, R.-S. Wang, X.-S. Zhang, and L. Chen, 2008b, 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.

Lin, Y.-R., Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, 2008, in *WWW '08: Proceedings of the 17th international conference on the World Wide Web* (ACM, New York, NY, USA), pp. 685{694.

Liu, X., D. Li, S. Wang, and Z. Tao, 2007, in *ICCS '07: Proceedings of the 7th international conference on Computational Science, Part II* (Springer-Verlag, Berlin, Heidelberg), pp. 657{664.

Lloyd, S., 1982, IEEE Trans. Inf. Theory **28**(2), 129.

Long, B., X. Xu, Z. Zhang, and P. S. Yu, 2007, in *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining* (IEEE Computer Society, Washington, DC, USA), pp. 232{241.

Lovasz, L., 1993, *Combinatorial Problems and Exercises* (North-Holland, Amsterdam, The Netherlands).

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.

Luczak, T., 1992, in *Proceedings of the Symposium on Random Graphs, Poznan 1989* (John Wiley & Sons, New York, USA), pp. 165{182.

Lusseau, D., 2003, Proc. Royal Soc. London B **270**, S186.

von Luxburg, U., 2006, *A tutorial on spectral clustering*, Technical Report 149, Max Planck Institute for Biological Cybernetics.

Mackay, D. J. C., 2003, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, UK).

MacQueen, J. B., 1967, in *Proc. of the fth Berkeley Symposium on Mathematical Statistics and Probability*, edited

(Kluwer Academic Press, Norwell, USA).
Mitrovic, M., and B. Tadic, 2009, Phys. Rev. E 80(2), 026123.
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.
Mu , S., F. Rao, and A. Ca isch, 2005, Phys. Rev. E 72(5), 056107.
Mungan, M., and J. J. Ramasco, 2008, eprint arXiv:0809.1398.
Nadler, B., S. Lafon, R. R. Coifman, and I. G. Kevrekidis, 2006, Applied and Computational Harmonic Analysis 21(1), 113.
Narasimhamurthy, A., D. Greene, N. Hurley, and P. Cunningham, 2008, in *Proc. 19th Irish Conference on Arti cial Intelligence and Cognitive Science (AICS'08)*.
Nelson, D. L., C. L. McEvoy, and T. A. Schreiber, 1998, The university of south orida word association, rhyme, and word fragment norms.
Nepusz, T., A. Petroczi, L. Negyessy, and F. Bazso, 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.
Ng, A. Y., M. I. Jordan, and Y. Weiss, 2001, in *Advances in Neural Information Processing Systems*, edited by T. G. Dietterich, S. Becker, and Z. Ghahramani (MIT Press, Cam-

Ravasz, E., and A.-L. Barabasi, 2003, Phys. Rev. E **67**(2), 026112.

Ravasz, E., A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabasi, 2002, Science **297**(5586), 1551.

Reddy, K. P., M. Kitsuregawa, P. Sreekanth, and S. S. Rao, 2002, in *DNIS '02: Proceedings of the Second International Workshop on Databases in Networked Information Systems* (Springer-Verlag, London, UK), pp. 188{200.

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. **P06016**.

Reichardt, J., and S. Bornholdt, 2007, Phys. Rev. E **76**(1), 015102 (R).

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 L. Xiao, 2009, Phys. Rev. E **79**(3), 036111.

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, 2009, Phys. Rev. E **80**(3), 036111.

Rissanen, J., 1978, Automatica **14**, 465.

Rives, A. W., and T. Galitski, 2003, Proc. Natl. Acad. Sci. USA **100**(3), 1128.

Rodrigues, F. A., G. Travieso, and L. da F. Costa, 2007, Int. J. Mod. Phys. C **18**, 937.

Ronhovde, P., and Z. Nussinov, 2008, eprint arXiv:0803.2548.

Ronhovde, P., and Z. Nussinov, 2009, Phys. Rev. E **80**(1), 016109.

Rosvall, M., D. Axelsson, and C. T. Bergstrom, 2009, eprint arXiv:0906.1405.

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 Engineer-*