Community detection algorithms: A comparative analysis

Andrea Lancichinetti[1,2] and Santo Fortunato[1]
[1]*Complex Networks and Systems, Institute for Scientific Interchange (ISI), Viale S. Severo 65, 10133 Torino, Italy*
[2]*Physics Department, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy*
(Received 7 August 2009; published 30 November 2009)

Uncovering the community structure exhibited by real networks is a crucial step toward an understanding of complex systems that goes beyond the local organization of their constituents. Many algorithms have been proposed so far, but none of them has been subjected to strict tests to evaluate their performance. Most of the sporadic tests performed so far involved small networks with known community structure and/or artificial graphs with a simplified structure, which is very uncommon in real systems. Here we test several methods against a recently introduced class of benchmark graphs, with heterogeneous distributions of degree and community size. The methods are also tested against the benchmark by Girvan and Newman [Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002)] and on random graphs. As a result of our analysis, three recent algorithms introduced by Rosvall and Bergstrom [Proc. Natl. Acad. Sci. U.S.A. **104**, 7327 (2007); Proc. Natl. Acad. Sci. U.S.A. **105**, 1118 (2008)], Blondel *et al.* [J. Stat. Mech.: Theory Exp. (2008), P10008], and Ronhovde and Nussinov [Phys. Rev. E **80**, 016109 (2009)] have an excellent performance, with the additional advantage of low computational complexity, which enables one to analyze large systems.

DOI: 10.1103/PhysRevE.80.056117 PACS number(s): 89.75.Hc

I. INTRODUCTION

The modern science of networks is probably the most active field within the new interdisciplinary science of complex systems. Many complex systems can be represented as networks, where the elementary parts of a system and their mutual interactions are nodes and links, respectively [1,2]. Complex systems are usually organized in compartments, which have their own role and/or function. In the network representation, such compartments appear as sets of nodes with a high density of internal links, whereas links between compartments have a comparatively lower density. These subgraphs are called communities or modules and occur in a wide variety of networked systems [3,4].

Finding compartments may shed light on the organization of complex systems and on their function. Therefore detecting communities in networks has become a fundamental problem in network science. Many methods have been developed, using tools and techniques from disciplines such as physics, biology, applied mathematics, and computer and social sciences. However, it is still not clear which algorithms are reliable and shall be used in applications. The question of the reliability itself is tricky, as it requires shared definitions of community and partition which are, at present, still missing. This essentially means that, despite the huge literature on the topic, there is still no agreement among scholars on what a network with communities looks like. Nevertheless, there has been a silent acceptance of a simple network model, the *planted* ℓ-partition model [5], which is often used in the literature in various versions. In this model one "plants" a partition, consisting of a certain number of groups of nodes. Each node has a probability p_{in} of being connected to nodes of its group and a probability p_{out} of being connected to nodes of different groups. As long as $p_{in} \geq p_{out}$ the groups are communities, whereas when $p_{in} \leq p_{out}$ the network is essentially a random graph, without community structure. The most popular version of the planted ℓ-partition model was proposed by Girvan and Newman (GN benchmark) [3]. Here the graph consists of 128 nodes, each with expected degree 16, which are divided into four groups of 32. The GN benchmark is regularly used to test algorithms for community detection. Indeed, algorithms can be compared based on their performance on this benchmark. This has been done by Danon *et al.* [6]. However, the GN benchmark has two drawbacks: (1) all nodes have the same expected degree; (2) all communities have equal size. These features are unrealistic, as complex networks are known to be characterized by heterogeneous distributions of degree [1,2,7] and community sizes [8–12]. In recent papers [13,14], we have introduced a new class of benchmark graphs [Lancichinetti-Fortunato-Radicchi (LFR) benchmark] that generalize the GN benchmark by introducing power law distributions of degree and community size. The new graphs are a real generalization, in that the GN benchmark is recovered in the limit case in which the exponents of the distributions of degree and community sizes go to infinity. Most community detection algorithms perform very well on the GN benchmark due to the simplicity of its structure. The LFR benchmark, instead, presents a much harder test to algorithms and makes it easier to disclose their limits. Moreover, the LFR benchmark graphs can be built very quickly: the complexity of the construction algorithms is linear in the number of links of the graph, so one can perform tests on very large systems, provided the method at study is fast enough to analyze them.

For these reasons, we believe that a serious assessment of the goodness of community detection algorithms can be made by evaluating their performance on the LFR benchmark. In this paper we propose a comparative analysis of this kind. After explaining briefly the LFR benchmark and how to compare partitions quantitatively we will pass to the description of the algorithms that we examined. We will present the analysis of the algorithms' performance first on the GN benchmark and then on the LFR benchmark, in its various versions including weighted and directed graphs,

along with graphs with overlapping communities. Finally we will consider the issue of whether the algorithms are able to give a null result, i.e., how they handle networks without expected community structure, such as random graphs. Our analysis will reveal that there are, at present, algorithms which are fast and reliable in many situations. We will conclude with a summary of our results and their consequences.

II. LFR BENCHMARK

The LFR benchmark [13,14] is a special case of the planted ℓ-partition model, in which groups are of different sizes and nodes have different degrees. The node degrees are distributed according to a power law with exponent τ_1; the community sizes also obey a power law distribution, with exponent τ_2. In the following, N indicates the number of nodes of the network. In the construction of the benchmark graphs, each node receives its degree once and for all and keeps it fixed until the end. In this way, the two parameters p_{in} and p_{out} of the planted ℓ-partition model in this case are not independent. Once the value of p_{in} is set one obtains the value of p_{out} and vice versa. It is more practical to choose as independent parameter the *mixing parameter* μ, which expresses the ratio between the external degree of a node with respect to its community and the total degree of the node. Of course, in general one may take different values for the mixing parameter for different nodes, but we will assume, for simplicity, that μ is the same for all nodes, consistently with the standard hypotheses of the planted ℓ-partition model.

The benchmark graphs are built with a fast procedure, which requires a time proportional to the number of links of the graph. Here we briefly sketch the construction process for the case of undirected and unweighted graphs, without overlapping communities, of which the other cases are simple generalizations and/or refinements. First, one has to assign the sizes of the communities $\{s_\xi\}$. This is done by picking random numbers from a power law distribution with exponent τ_2. Obviously, the sum of the sizes of the communities has to match the number of nodes N of the graph (except in the case of overlapping communities, in which case the total "area" covered by the communities exceeds N). Then, one treats each community as an isolated graph and assigns to each node i of a community an internal degree $(1-\mu)k_i$, where k_i is the degree of node i, which is taken by a power law distribution with exponent τ_1. In this way, each node i has a number of stubs $(1-\mu)k_i$ that need to be attached to nodes of the same community. This is done according to the configuration model [15], i.e., by randomly attaching pairs of randomly selected stubs to each other until no more stubs are "free." Next, communities have to be linked to each other. For that, one adds to each node i a number of stubs μk_i [so that the degree of the node is finally $k_i=(1-\mu)k_i+\mu k_i$] and attaches them to each other again according to the configuration model. In this way, the final graph satisfies the conditions we imposed at the beginning on the distributions of degree and community size.

By construction, the groups are communities when $p_{in}>p_{out}$. This condition can be translated into a condition on the mixing parameter μ. Let us label k_i^{in} and k_i^{out} the internal and external degrees of node i with respect to its community (which we denote with c). By definition, k_i^{in} is the number of neighbors of i that belong to its community c and k_i^{out} is the number of neighbors of i that belong to the other communities. The number of available connections k_c^{out} (k_c^{in}) outside (inside) c is given by the sum of the degrees of the nodes outside (inside) the community. If the numbers of nodes inside and outside c are not too small, the sum of their degrees can be approximated by the product of the average degree $\langle k\rangle$ by the number of nodes. We indicate with n_c the number of nodes of the community c of node i, so we have that $k_c^{out}\sim(N-n_c)\langle k\rangle$ and $k_c^{in}\sim n_c\langle k\rangle$. By definition of the linking probabilities p_{in} and p_{out} we deduce that

$$p_{out}=\frac{k_i^{out}}{k_c^{out}}\sim\frac{k_i^{out}}{(N-n_c)\langle k\rangle} \tag{1}$$

and

$$p_{in}=\frac{k_i^{in}}{k_c^{in}}\sim\frac{k_i^{in}}{n_c\langle k\rangle}. \tag{2}$$

In this way, the condition for the existence of communities $p_{in}>p_{out}$ becomes

$$\frac{k_i^{in}}{n_c\langle k\rangle}>\frac{k_i^{out}}{(N-n_c)\langle k\rangle}, \tag{3}$$

from which we get

$$k_i^{in}>\frac{n_c k_i^{out}}{N-n_c}. \tag{4}$$

On the other hand, by definition we have that

$$\mu=\frac{k_i^{out}}{k_i^{in}+k_i^{out}}. \tag{5}$$

By comparing Eq. (5) with Eq. (4) we obtain the desired condition on μ,

$$\mu<\frac{N-n_c}{N}. \tag{6}$$

The condition expressed in Eq. (6) is general and applies to any version of the planted ℓ-partition model. When communities are different in size, the upper bound on μ depends on the specific community at hand. However, if n_c^{max} is the size of the largest community, we can safely assume that, whenever $\mu<(N-n_c^{max})/N$, all communities are well defined. In the GN benchmark, where $n_c=32$ and 128, the condition becomes $\mu<3/4$. This is interesting, as in most works using the GN benchmark, one usually assumes that communities are there as long as $\mu<1/2$, whereas they are not well defined for $\mu>1/2$. Instead, we see that communities are there, at least in principle, up until $\mu=3/4$. However, we stress that, even if communities are there, methods may be unable to detect them. The reason is that, due to fluctuations in the distribution of links in the graphs, already before the limit imposed by the planted partition model it may be impossible to detect the communities and the model graphs may look similar to random graphs. This issue of the actual

significance of communities and their detectability *a priori* is very important and has been recently discussed in the literature [16–18]. We notice that, on large networks, when $n_c \ll N$, the limit value of μ below which communities are defined approaches 1. In our tests with the LFR benchmark, we will often be in this regime.

III. COMPARING PARTITIONS

Testing an algorithm on any graph with built-in community structure also implies defining a quantitative criterion to estimate the goodness of the answer given by the algorithm as compared to the real answer that is expected. This can be done by using suitable similarity measures. For reviews of similarity measures see Refs. [19–21]. In the first tests of community detection algorithms, one used a measure called *fraction of correctly identified nodes*, introduced by Girvan and Newman [3]. However, it is not well defined in some cases (e.g., when a detected community is a merger of two or more "real" communities), so in the last years other measures have been used. In particular, measures borrowed from information theory have proved to be reliable.

To evaluate the Shannon information content [22] of a partition, one starts by considering the community assignments $\{x_i\}$ and $\{y_i\}$, where x_i and y_i indicate the cluster labels of vertex i in partitions $\mathcal{X}$ and $\mathcal{Y}$, respectively. One assumes that the labels x and y are values of two random variables X and Y, with joint distribution $P(x,y)=P(X=x,Y=y)=n_{xy}/n$, which implies that $P(x)=P(X=x)=n_x^X/n$ and $P(y)=P(Y=y)=n_y^Y/n$, where n_x^X, n_y^Y, and n_{xy} are the sizes of the clusters labeled by x, y, and of their overlap, respectively. The *mutual information* $I(X,Y)$ of two random variables is defined as

$$I(X,Y)=\sum_x \sum_y P(x,y)\log\frac{P(x,y)}{P(x)P(y)}. \tag{7}$$

The measure $I(X,Y)$ tells how much we learn about X if we know Y and vice versa. Actually $I(X,Y)=H(X)-H(X|Y)$, where $H(X)=-\Sigma_x P(x)\log P(x)$ is the Shannon entropy of X and $H(X|Y)=-\Sigma_{x,y} P(x,y)\log P(x|y)$ is the conditional entropy of X given Y. The mutual information is not ideal as a similarity measure: in fact, given a partition $\mathcal{X}$, all partitions derived from $\mathcal{X}$ by further partitioning (some of) its clusters would all have the same mutual information with $\mathcal{X}$, even though they could be very different from each other. In this case the mutual information would simply equal the entropy $H(X)$ because the conditional entropy would be systematically zero. To avoid that, Danon *et al.* adopted the *normalized mutual information* [6],

$$I_{norm}(X,Y)=\frac{2I(X,Y)}{H(X)+H(Y)}, \tag{8}$$

which equals 1 if the partitions are identical, whereas it has an expected value of 0 if the partitions are independent. The normalized mutual information is currently very often used in tests of community detection algorithms. We have recently proposed a definition of the measure to evaluate the similarity of *covers*, i.e., of divisions of the network in overlapping communities, which one needs for the tests of Sec. VI D. The details can be found in the Appendix in Ref. [12]. We stress that our definition is not a proper extension of the normalized mutual information, in the sense that it does not recover exactly the same value of the original measure for the comparison of proper partitions without overlap, even though the values are close. For consistency we used our definition in all tests, although in the tests involving benchmarks without overlapping communities the classic expression of Eq. (8) could be used. For this reason, we warn that in the plots showing the performance of the algorithms on the GN benchmark, the curves are not identical to those already seen in previous papers (for, e.g., modularity-based methods), where Eq. (8) was used, although they are rather close.

IV. ALGORITHMS

We have tested a wide spectrum of community detection methods. In some cases the software to implement the algorithms was publicly available; in other cases the original developers have let us use their own code, otherwise we have created the software on our own. We wanted to have a representative subset of algorithms, which exploit some of the most interesting ideas and techniques that have been developed over the last years. Obviously we could not by any means perform an analysis of all existing techniques, as their number is huge. Some of them were excluded *a priori*, if particularly slow, as our tests involve graphs with a few thousand nodes, which old methods are unable to handle. On the other hand, the code to create the LFR benchmark is freely available [23] and scholars are welcome to test their algorithms on it and compare their performance with that of the algorithms analyzed here. Here is the list of the algorithms we considered.

Algorithm of Girvan and Newman [3,24]. It is the first algorithm of the modern age of community detection in graphs. It is a hierarchical divisive algorithm in which links are iteratively removed based on the value of their betweenness, which expresses the number of shortest paths between pairs of nodes that pass through the link. In its most popular implementation, the procedure of link removal ends when the modularity of the resulting partition reaches a maximum. The modularity of Newman and Girvan is a well known quality function that estimates the goodness of a partition based on the comparison between the graph at hand and a null model, which is a class of random graphs with the same expected degree sequence of the original graph. The algorithm has a complexity $O(N^3)$ on a sparse graph. In the following we will refer to it as GN.

Fast greedy modularity optimization by Clauset et al. [11]. This method is essentially a fast implementation of a previous technique proposed by Newman [25]. Starting from a set of isolated nodes, the links of the original graph are iteratively added such to produce the largest possible increase of the modularity of Newman and Girvan at each step. The fast version of Clauset *et al.*, which uses more efficient data structures, has a complexity of $O(N\log^2 N)$ on sparse graphs.

Exhaustive modularity optimization via simulated annealing [26–29]. The goal is the same as in the previous algo-

rithm, but the precision of the final estimate of the maximum is far higher, due to the exhaustive optimization, at the expense of the computational speed. The latter cannot be expressed in closed form, as in the cases above, as it depends on the parameters used for the optimization. We will stick to the procedure used by Guimerà and Amaral [29].

Fast modularity optimization by Blondel et al. [30]. This is a multistep technique based on a local optimization of Newman-Girvan modularity in the neighborhood of each node. After a partition is identified in this way, communities are replaced by supernodes, yielding a smaller weighted network. The procedure is then iterated until modularity (which is always computed with respect to the original graph) does not increase any further. This method offers a fair compromise between the accuracy of the estimate of the modularity maximum, which is better than that delivered by greedy techniques like the one by Clauset *et al.* above, and computational complexity, which is essentially linear in the number of links of the graph.

Algorithm by Radicchi et al. [31]. This algorithm is in the spirit of that by Girvan and Newman above. In fact, it is a divisive hierarchical method, where links are iteratively removed based on the value of their edge clustering coefficient, which is defined as the ratio between the number of loops based on the link and the largest possible number of loops that can be based on the link. The edge clustering coefficient is a local measure, so its computation is not so heavy as that of edge betweenness, which yields a significant improvement in the complexity of the algorithm, which is $O(N^2)$ on a sparse graph. Another major difference from the GN algorithm is the stopping criterion of the procedure, which depends on the properties of the communities themselves and not on the values of a quality function such as modularity. Radicchi *et al.* considered two types of communities: *strong* communities are groups of nodes such that the internal degree of each node exceeds its external degree; *weak* communities are groups of nodes such that the total internal degree of the nodes of the group exceeds their total external degree.

Cfinder [8]. This is a local algorithm proposed by Palla *et al.* that looks for communities that may overlap, i.e., share nodes. It was the first paper in the physics literature on community detection to address this problem, which is important in many systems such as, e.g., social networks. Communities are defined as the largest possible subgraphs that can be explored by rolling k cliques across the network, where a k clique rolls by rotating about any of its component $(k-1)$ cliques (which are links when $k=3$). The complexity of this procedure can be high, as the computational time needed to find all k cliques of a graph is an exponentially growing function of the graph size [32], but in practical applications the method is rather fast, enabling one to analyze systems with up to 10^5 nodes.

Markov cluster algorithm [33]. This is an algorithm developed by van Dongen, which simulates a peculiar diffusion process on the graph. One starts from the right stochastic matrix (or diffusion matrix) of the graph, which is obtained from the adjacency matrix of the original graph by dividing the elements of each row by their sum. Then one computes an integer power of this matrix (usually the square), which yields the probability matrix of a random walk after a number of steps equal to the number of powers of the right stochastic matrix considered. This step is called expansion. Next, each element of the matrix is raised to some power α in order to enhance (artificially) the probability of the walker to be trapped within a community. This step is called inflation. The expansion and inflation steps are iterated until one obtains the adjacency matrix of a forest (i.e., a disconnected tree), whose components are the communities. This method, widely used in bioinformatics, is strongly dependent on the choice of the parameter α. Its complexity can be lowered to $O(Nk^2)$ if, after each inflation step, only the k largest elements of the resulting matrix are kept, whereas the others are set to zero. In the following we will refer to the method as MCL.

Structural algorithm by Rosvall and Bergstrom [34]. Here the problem of finding the best cluster structure of a graph is turned into the problem of optimally compressing the information on the structure of the graph, so that one can recover as closely as possible the original structure when the compressed information is decoded. This is achieved by computing the minimum of a function which expresses the best tradeoff between the minimal conditional information between the original and the compressed information (maximal faithfulness to the original information) and the maximal compression (least possible information to transmit). The optimization of the function is carried out via simulated annealing, which makes the algorithm quite slow, although one could always go for a faster and less accurate optimization. In the following we will refer to the method as Infomod.

Dynamic algorithm by Rosvall and Bergstrom [35]. This technique is based on the same principle as the previous one. The difference is that before one was compressing the information on the structure of the graph, here one wishes to compress the information of a dynamic process taking place on the graph, namely, a random walk. The optimal compression is achieved again by optimizing a quality function, which is the minimum description length [36,37] of the random walk. Such optimization can be carried out rather quickly with a combination of greedy search and simulated annealing. In the following we will refer to the method as Infomap.

Spectral algorithm by Donetti and Muñoz (*DM*) [38]. This is a method based on spectral properties of the graph. The idea is that eigenvector components corresponding to nodes in the same community should have similar values if communities are well identified. Donetti and Muñoz focused on the eigenvectors of the Laplacian matrix. They considered a limited number of eigenvectors, say g, and represented each node of the graph as a geometric point in an Euclidean g-dimensional space, whose coordinates are the eigenvector components corresponding to the node. The points are then grouped with traditional hierarchical clustering techniques. Of the resulting partitions, one picks the one that maximizes the modularity by Newman and Girvan. The method is rather quick when only a few eigenvectors are computed, which is usually the case, as this can be done via the Lanczos method [39]. In the following we will refer to the method as DM.

Expectation-maximization (*EM*) *algorithm by Newman and Leicht* [40]. Here Bayesian inference is used to deduce

the best fit of a given model to the data represented by the actual graph structure. The goodness of the fit is expressed by a likelihood that is maximized by means of the expectation-maximization technique [41]. This leads to a system of self-consistent equations, which can be solved by iteration starting from suitable initial conditions. The equations can be solved rather quickly and fairly large systems can be analyzed in this way (up until 10^6 nodes). A nice feature of the method is that it finds the most relevant group structure of the graph, whether the groups are communities or not (in graphs with multipartite structure the classes are rather anticommunities, as there are very few links inside the groups). A drawback of the method is the fact that one needs to feed the number of groups, which is usually not known *a priori*. In the following we will refer to the method as EM.

Potts model approach by Ronhovde and Nussinov (*RN*) [42]. This method is based on the minimization of the Hamiltonian of a Potts-like spin model, where the spin state represents the membership of the node in a given community. A resolution parameter enables one to span several community scales from very small to very large communities. The relevant scales are identified by checking for the stability of the partitions obtained for given values of the resolution parameter. This is done by computing the similarity of partitions obtained for the same resolution parameter but starting from different initial conditions. Peaks in the similarity spectrum correspond to stable and/or relevant partitions. The method is rather fast; its complexity is slightly superlinear in the number of links of the graph. In the following we will refer to the method as RN.

V. TESTS ON THE GN BENCHMARK

We begin by showing the performance of the algorithms on the GN benchmark. As we have explained in Sec. II, for the GN benchmark communities are well defined (in principle) up until a value $3/4=0.75$ for the mixing parameter. We will indicate the mixing parameter with the symbol μ_t to mean that we refer to topology. In Sec. VI C we will focus instead on the mixing parameter μ_w, which considers the weights of the links. In Fig. 1 we show the results of our analysis. Each point of every curve corresponds to an average over 100 realizations of the benchmark. For the algorithms by Radicchi *et al.* and by Newman and Leicht (EM), we have put two curves instead of one (likewise in Sec. VI A). In the first case, we showed the outcome of the method when one uses both possible stopping criteria, corresponding to a partition consisting of strong (black curve) and weak (red curve) communities, respectively. In the case of the EM method, we show the curves delivered by the iterative solution of the EM equations when one starts from a random partition (red curve) and from the planted partition of the benchmark (black curve). As one can see, results are different in these cases even if they are solutions of the same equation. This shows how sensitive the solution is to the choice of the initial condition. Moreover, the maximum likelihood achieved when one makes the "intelligent guess" of the real partition is higher compared to the maximum likelihood obtained starting from a random partition. This indicates that the greedy approach to the solution of the EM equations suggested by Newman and Leicht is not an efficient way to maximize the likelihood, as one may expect.

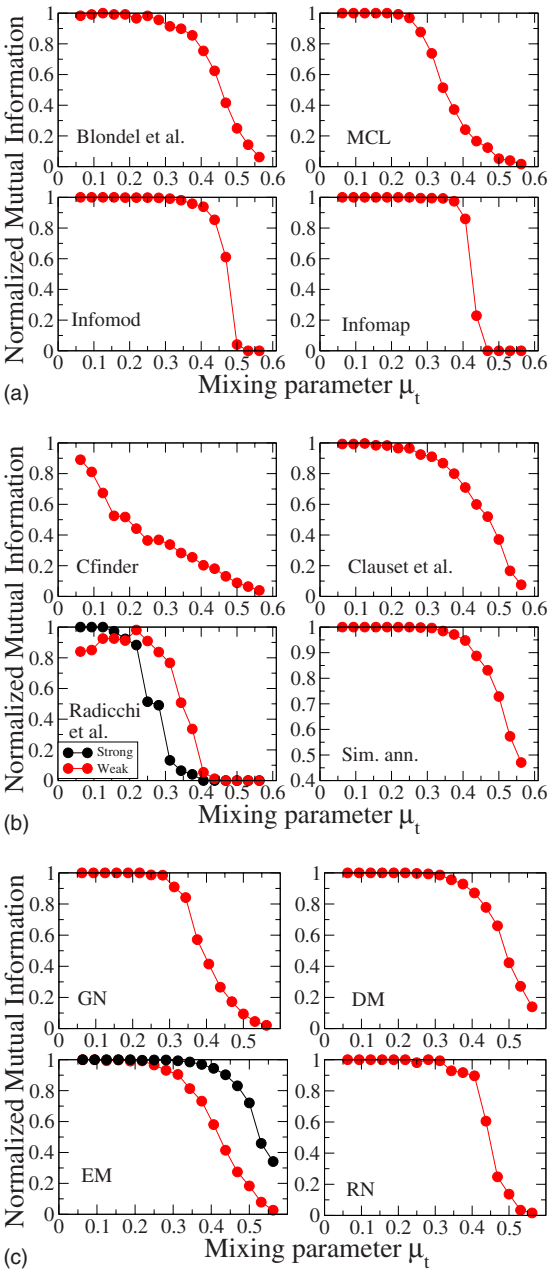

FIG. 1. (Color online) Tests of the algorithms on the GN benchmark. Each plot shows the normalized mutual information as a function of the mixing parameter μ_t between the planted partition of the benchmark and the one found by a given algorithm. The trend is generally the same: for small values of μ_t communities are well separated and most algorithms do a good job, so the normalized mutual information is 1 or close to 1. When μ_t increases, communities are more mixed and harder to detect, so the normalized mutual information is quite different from 1, indicating that the partition found by the algorithm is sensibly different from the planted partition of the benchmark.

Most methods perform rather well, although all of them start to fail much earlier than the expected threshold of 3/4. The Cfinder fails to detect the communities even when μ_t

~ 0 when they are very well identified. This is due to the fact that, even when μ_t is small, the probe clique that explores the system manages to pass from one group to the other and yields much larger groups, often spanning the whole graph. The method by Radicchi *et al.* does not have a remarkable performance either, as it also starts to fail for low values of μ_t, although it does better than the Cfinder. The MCL is better than the method by Radicchi *et al.* but is outperformed by modularity-based methods (simulated annealing, Clauset *et al.* and Blondel *et al.*), which generally do quite well on the GN benchmark, something that was already known from the literature. The DM and RN methods have a comparable performance as the exhaustive optimization of modularity via simulated annealing. The GN algorithm performs about as well as the MCL. Both methods by Rosvall and Bergstrom have a good performance. In fact, up until $\mu_t \sim 0.4$, they always guess the planted partition in four clusters.

VI. TESTS ON THE LFR BENCHMARK

In this section we will present the tests on the LFR benchmark. For a thorough analysis, we have considered various versions of the benchmark, in which links can have or not weights and/or direction. We have also examined the version which allows for community overlaps. In each test, we have averaged the value of the normalized mutual information over 100 realizations for each value of the mixing parameter.

A. Undirected and unweighted graphs

The plots of Fig. 2 illustrate the results of the analysis. The following input parameters are the same for all benchmark graphs used here, as well as in Secs. VI B–VI D: the average degree is 20, the maximum degree is 50, the exponent of the degree distribution is −2, and that of the community size distribution is −1. In each plot, except for the GN and the EM algorithms, we show four curves, corresponding to two different network sizes (1000 and 5000 nodes) and, for a given size, to two different ranges for the community sizes, indicated by the letters S and B: S (stays for "small") means that communities have between 10 and 50 nodes and B (stays for "big") means that communities have between 20 and 100 nodes. For the GN algorithm we show only the curves corresponding to the smaller network size, as it would have taken too long to accumulate enough statistics to present clean plots for networks of 5000 nodes due to the high computational complexity of the method. For the EM method we have plotted eight curves as for each set of benchmark graphs we have considered the two outcomes of the algorithm corresponding to the different choices of initial conditions we have mentioned in Sec. VI, namely, random (bottom curves) and planted (top curves) partitions. In this case, the difference in the performance of the algorithm in the two cases is remarkable. The fact that, by starting from the planted partition, the final likelihood is actually higher as compared with a random start, as we have seen in Sec. VI, confirms that the method has a great potential if only one could find a better way to estimate the maximum likelihood than the greedy approach currently adopted. Nevertheless we

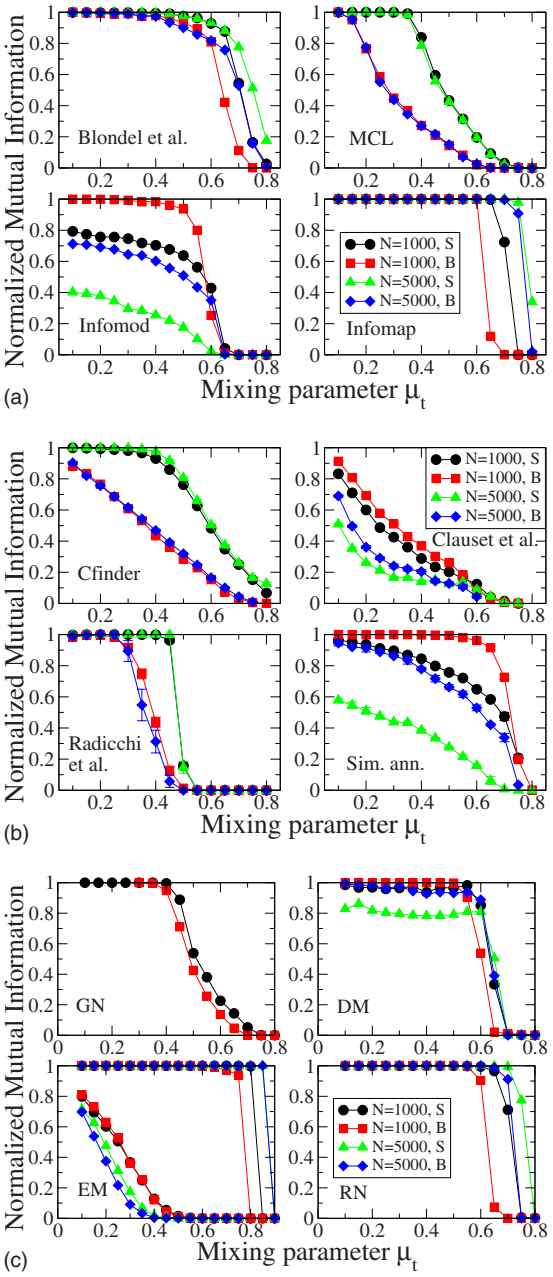

FIG. 2. (Color online) Tests of the algorithms on the LFR benchmark with undirected and unweighted links. Each plot shows the normalized mutual information as a function of the mixing parameter μ_t between the planted partition of the benchmark and the one found by a given algorithm. The general trend is qualitatively the same as in the analysis on the GN benchmark (Fig. 1).

remind that the EM also has the big drawback to require as input the number of groups to be found, which is usually unknown in applications.

As a general remark, we see that the LFR benchmark enables one to discriminate the performances of the algorithms much better than the GN benchmark, as expected. Modularity-based methods have a rather poor performance, which worsens for larger systems and smaller communities due to the well known resolution limit of the measure [43]. The only exception is represented by the algorithm by

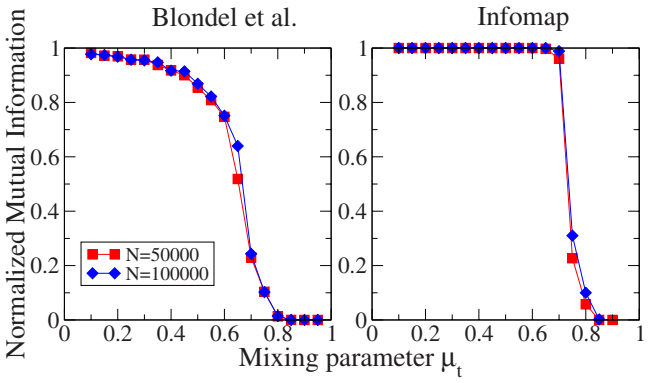

FIG. 3. (Color online) Tests of the algorithm by Blondel *et al.* and Infomap on large LFR benchmark graphs with undirected and unweighted links.

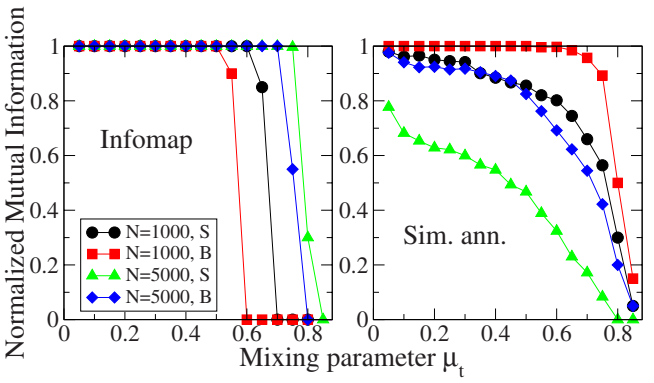

FIG. 4. (Color online) Tests of Infomap and of the exhaustive modularity optimization via simulated annealing on the LFR benchmark with directed and unweighted links.

Blondel *et al.*, whose performance is very good probably because the estimated modularity maximum is not a very good approximation of the real one, which is more likely found by simulated annealing. The Cfinder, the MCL, and the method by Radicchi *et al.* do not have impressive performances either and display a similar pattern, i.e., the performance is severely affected by the size of the communities (for larger communities it gets worse, whereas for small communities it is decent), whereas it looks rather insensitive to the size of the network. The DM has a fair performance, but it gets worse if the network size increases. The same trend is shown by Infomod, where the performance worsens considerably with the increase of the network size. Infomap and RN have the best performances, with the same pattern with respect to the size of the network and of the communities: up to values of $\mu_t \sim 1/2$ both methods are capable to derive the planted partition in the 100% of cases.

We conclude that Infomap, the RN method, and the method by Blondel *et al.* are the best performing algorithms on the LFR undirected and unweighted benchmark. Since Infomap and the method by Blondel *et al.* are also very fast, essentially linear in the network size, we wonder how good their performance is on much larger graphs than those considered in Fig. 2. For this reason we carried out another set of tests of these two algorithms on the LFR benchmark by considering graphs with 50 000 and 100 000 nodes. We have done so also because in the tests that can be found in the literature on community detection one typically uses very small graphs, and the performance can change considerably on large graphs. In Fig. 3 we show the performance of the two methods. Due to the large network size, we decided to pick a broad range of community sizes from 20 to 1000 nodes. In this way, the heterogeneity of the community sizes is manifest. The maximum degree here was fixed to 200. Remarkably, the performance of the method by Blondel *et al.* is worse than on the smaller graphs of Fig. 2, whereas that of Infomap is stable and does not seem to be affected.

B. Directed and unweighted graphs

Directedness is an essential features of many real networks. Ignoring direction, as one often does or is forced to do, may reduce considerably the information that one can extract from the network structure. In particular, neglecting link directedness when looking for communities may lead to partial or even misleading results. In the literature there has been no benchmark for directed graphs with communities for a long time. However, we have recently extended the LFR benchmark to directed networks [14], so we are in the position to evaluate the performance of community detection algorithms in this case. The presence of directed links is a serious obstacle toward a generalization of an algorithm for community detection. Therefore, very few algorithms currently available are able to handle directed graphs. In the set of methods we consider here, only five can be used as well for directed networks: Clauset *et al.*, simulated annealing for modularity, Cfinder, Infomap, and EM. For some of the other algorithms one may think of possible extensions which are, at present, still missing. The EM method, in its original definition in Ref. [40], has actually problems to deal with directed graphs [44]. We present here a comparison of the performances of two methods, exhaustive modularity optimization via simulated annealing and Infomap. The results are in Fig. 4. Here the topological mixing parameter μ_t refers to the indegree of the nodes which are distributed according to a power law as in the original undirected benchmark, while the outdegree is kept constant for all nodes, a choice made to avoid an unnecessary proliferation of input parameters. Again, we considered two different network sizes and ranges for the community size, which are the same as those in Fig. 2. The other input parameters for the benchmark are the same that we have given in Sec. VI A. As expected, modularity optimization shows the same limits that emerged in Fig. 2. On the other hand, the performance of Infomap is still very good.

C. Undirected and weighted graphs

In this section we focus on undirected graphs with weighted links.

Weights are also precious sources of information [45]. Just as in the case of link directedness above, neglecting weights may imply a significant limitation of the information on graph's properties, concealing features of real systems

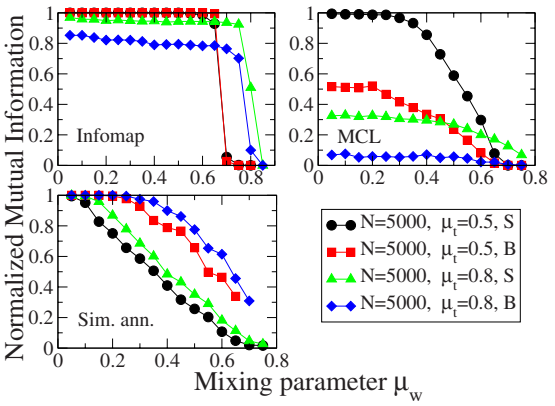

FIG. 5. (Color online) Tests of Infomap, MCL, and the exhaustive modularity optimization via simulated annealing on the LFR benchmark with undirected and weighted links.

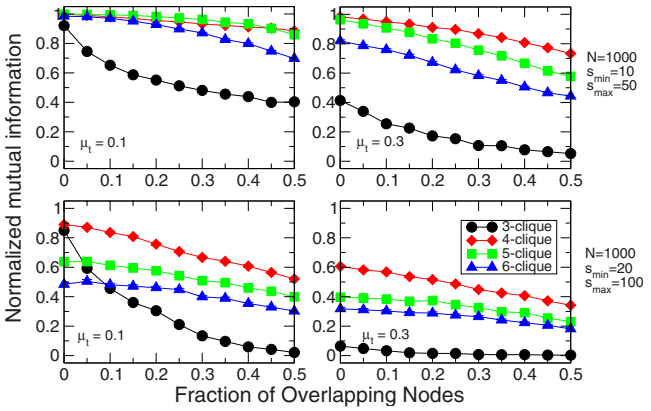

FIG. 6. (Color online) Tests of the Cfinder on the LFR benchmark with undirected and unweighted links and overlapping communities. The variable on the x axis is the fraction of overlapping nodes. The networks have 1000 nodes; the other parameters are $\tau_1=2$, $\tau_2=1$, $\langle k\rangle=20$, and $k_{max}=50$.

which may be very important and not deducible from the mere topology. Ideally, one should exploit the information from both topology and weights for a reliable analysis of a network. The LFR benchmark has been extended to weighted graphs as well [14]. Now there are two mixing parameters, one for topology, which is the same μ_t we have defined and used so far, and the other for the weights, μ_w, which is the weighted counterpart of μ_t, i.e., it expresses the fraction of the strength of the node that lies on links connecting the node to the nodes outside its community, with respect to the total strength of the node. We remind that the strength of the node is the sum of the weights of its links. Moreover, there is an additional parameter, i.e., the exponent of the distribution for the strength: we set it to 1.5 for all realizations. All other parameters are the same specified in Sec. VI A. Since we wish to show the results of the test on two-dimensional plots, as we have done so far, we need to keep fixed one of the two parameters and study the dependence on the other. Here we freeze the topological mixing parameter μ_t and study the dependence of the results on μ_w, so that we see how the performance of an algorithm varies when only the weights are redistributed but the topology is fixed. The results are in Fig. 5, where we consider only three methods: Infomap, MCL, and exhaustive modularity optimization via simulated annealing. The other methods have no weighted counterpart or the code for the weighted version was not available. In each plot we show four curves, corresponding to two choices for the topological mixing parameter μ_t and the two usual ranges of small (S) and big (B) communities that we have used so far. The network size is 5000 nodes in each case. The Infomap by Rosvall and Bergstrom has, once more, a remarkable performance, although it worsens if communities are topologically more mixed (higher μ_t) and larger in size (B). The MCL has a fair performance only in the case for $\mu_t=0.5$ and small communities, whereas in the other extreme of big topological mixture and big communities it fails for any value of μ_w. Modularity optimization seems to be more sensitive to the community size than to the other parameters.

D. Undirected and unweighted graphs with overlapping communities

The fact that communities in real systems often overlap has attracted a lot of attention in the last years, leading to the creation of new algorithms able to deal with this special circumstance, starting from the first work by Palla *et al.* [8]. Meanwhile, a few methods have been developed [12,40,46–51], but none of them has been thoroughly tested except on a bunch of specific networks taken from the real world. Indeed, there have been no suitable benchmark graphs with overlapping community structure until recently [14,52]. In particular, the LFR benchmark has been extended to the case of overlapping communities [14], and we use it here. Of our set of algorithms, only the Cfinder is able to find overlapping communities. In principle also the EM method assigns to each node the probability that it belongs to any community, but then one would need a criterion to define which, among such probability values, is significant and shall be taken or is not significant and shall be neglected. For this reason we report the results of tests carried out with the Cfinder only.

In Figs. 6 and 7 we show the results. The topological mixing parameter μ_t is fixed and one varies the fraction of overlapping nodes between communities. We have run the Cfinder for different types of k cliques (k indicates the number of nodes of the clique), with $k=3,4,5,6$. In general we notice that triangles ($k=3$) yield the worst performance, whereas four and five cliques give better results. In the two top diagrams community sizes range between 10 and 50 nodes, whereas in the bottom diagrams the range goes from 20 to 100 nodes. By comparing the diagrams in the top with those in the bottom we see that the algorithm performs better when communities are (on average) smaller. The networks used to produce Fig. 6 consist of 1000 nodes, whereas those of Fig. 7 consist of 5000 nodes. From the comparison of Fig. 6 with Fig. 7 we see that the algorithm performs better on networks of larger size.

VII. TESTS ON RANDOM GRAPHS

An important test of community detection algorithms, usually ignored in the literature, consists of applying them to

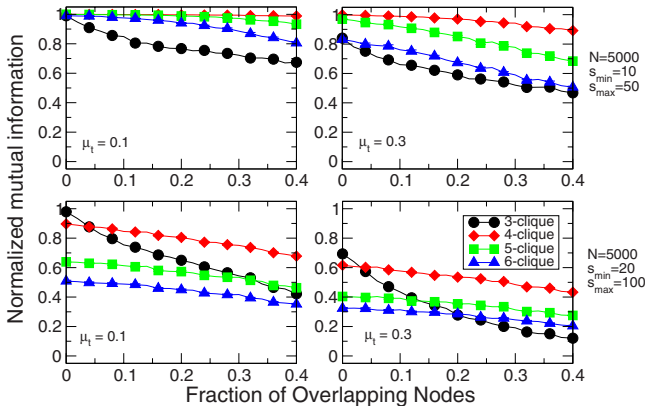

FIG. 7. (Color online) Tests of the Cfinder on the LFR benchmark with undirected and unweighted links and overlapping communities. The variable on the x axis is the fraction of overlapping nodes. The networks have 5000 nodes; the other parameters are the same used for the graphs of Fig. 6.

random graphs. In random graphs, by definition, the linking probabilities of the nodes are independent of each other. In this way one does not expect that there will be inhomogeneity in the density of links on the graphs, i.e., there should be no communities. For instance, in Erdös-Rényi random graphs [53] nodes are equivalent, as they have the same probability to get connected to each other. Therefore there are *a priori* no groups of nodes with special and/or stronger relationships between them. Things are not that simple though. It is certainly true that *on average* this is what happens. On the other hand, specific realizations of random graphs may display pseudocommunities, i.e., clusters produced by fluctuations in the link density. This is why, for instance, the maximum modularity of partitions in random graphs is not small [26,54–56]. However, a good method should distinguish between such pseudocommunities and meaningful modules. The comparison with random graphs is therefore mandatory. If the subgraphs found by a community detection technique on a network are not different from those found on a randomized version of the network, we should deduce that those subgraphs are not communities. We stress that this is also valid for small subgraphs, such as, e.g., triangles, even if they are very different objects than communities. Such small subgraphs can be found on random graphs as well, however finding many more subgraphs on a network than on a randomized version of it may indicate that those subgraphs are *motifs*, i.e., elementary building blocks of the system [57–59].

If one feeds a good community detection method with a random graph, the method should not find nontrivial partitions and ideally deliver only one community including all nodes of the graph or as many communities as there are nodes. This is what we want to check here. We considered two types of graphs: Erdös-Rényi random graphs [53], which have a binomial degree distribution, and random graphs with power law degree distributions (scale-free). The latter have been built via the configuration model [15], starting from a fixed degree sequence for the nodes obeying the predefinite power law distribution. The exponent of the distribution is −2; the maximum degree was fixed to 200. The size of all

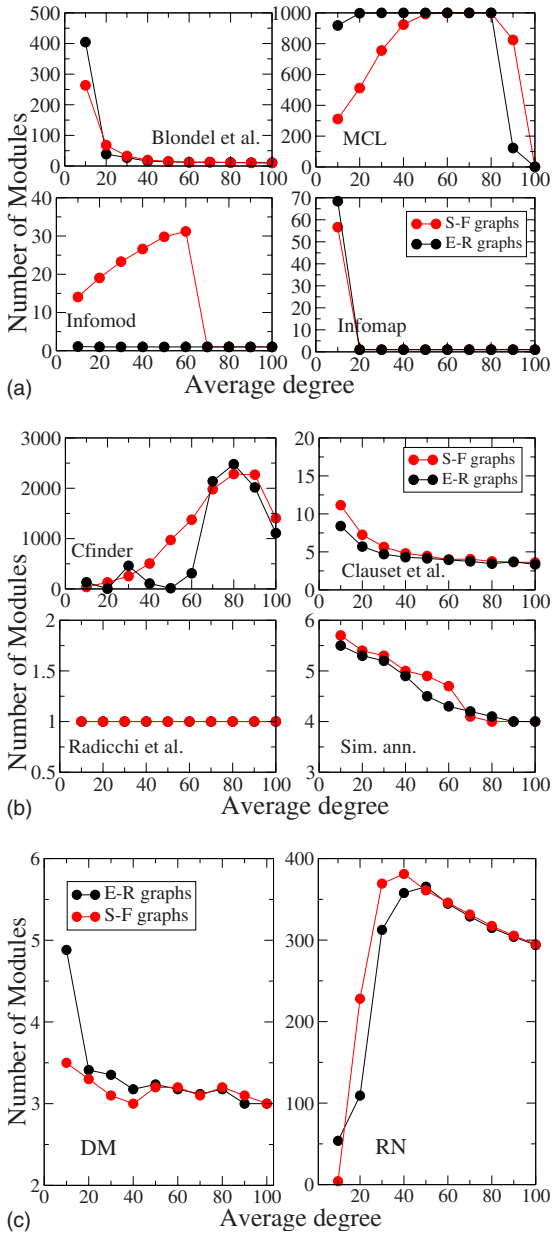

FIG. 8. (Color online) Tests of the algorithms on Erdös-Rényi and scale-free random graphs.

graphs, Erdös-Rényi and scale-free, is fixed to 1000 nodes. In Fig. 8 we show the number of modules found by various algorithms as a function of the average degree of the graph. Each point corresponds to an average over 100 graph realizations. We do not show the results of the EM method because the number of modules must be given by input and of the GN algorithm because it is too slow to be used for the analysis.

The best performance is that of the method by Radicchi *et al.*, which always finds a single cluster comprising all nodes. The MCL, instead, finds as many clusters as there are nodes, which is still reasonable. Here, however, the answer depends on the average degree $\langle k \rangle$ of the graph: if $\langle k \rangle$ is very low or very large the number of modules is smaller than 1000, i.e., the method finds small groups of nodes. This is particularly evident for scale-free graphs. Modularity-based methods,

such as Clauset *et al.*, the exhaustive optimization via simulated annealing, and the algorithm by Blondel *et al.*, are not so good, as they always find a few clusters, even in the limit of large $\langle k \rangle$: this is actually well known [55]. This is also the case for the DM method, which performs a sort of modularity optimization on the restricted set of partitions delivered by hierarchical clustering. Infomod finds just one community on Erdös-Rényi graphs for any value of $\langle k \rangle$, while it delivers nontrivial partitions on scale-free graphs up until $\langle k \rangle \sim 60$. The RN method finds nontrivial partitions for any value of $\langle k \rangle$. The Cfinder finds a single module for very low values of $\langle k \rangle$ and then a rapidly rising number of modules as $\langle k \rangle$ increases. Since the modules are strongly overlapping in this case, they may exceed the number of nodes, as we see from the plot. Instead, Infomap always finds a single module comprising all nodes except when $\langle k \rangle$ is low.

VIII. SUMMARY

We have carried out a comparative analysis of the performances of some algorithms for community detection on various graphs: the GN and LFR benchmarks and random graphs. Link direction, weights, and the possibility for communities to overlap have been taken into account in dedicated tests. We conclude that the Infomap method by Rosvall and Bergstrom [35] is the best performing on the set of benchmarks we have examined here. In particular, its results on the LFR benchmark graphs, which are much more difficult to examine than the GN benchmark graphs, as clearly shown by Figs. 1 and 2, are encouraging about the reliability of the method in applications to real graphs. Among the other things, the method can be applied to weighted and directed graphs as well, with excellent performances, so it has a large spectrum of potential applications. The algorithms by Blondel *et al.* [30] and by RN [42] also look very good from our analysis and could be used as well. In fact, for a study of the community structure in real graphs, one could think of using all three methods to be able to extract some algorithm-independent information. Furthermore, as we have seen in Sec. IV, these methods have a low computational complexity, so one could use them on graphs with millions of nodes and links. On the other hand, the algorithms are not able to account for overlapping communities, so they need to be properly refined to deal with this possibility, which is common in many real systems.

One may object that, despite the features planted in the LFR benchmark, i.e., the fat-tailed distributions of degree and community size, which are actually observed in real networks, our artificial graphs are still different from real systems. For instance, the clustering coefficient [60] of the LFR benchmark is very low due to the very small number of triangles, whereas real networks are characterized by many triangles and consequently a high clustering coefficient. On the one hand the GN benchmark also has very few triangles and low clustering coefficient (the LFR benchmark is just a generalization of the GN benchmark); nevertheless people have used it extensively for testing algorithms. On the other hand, nothing forbids to modify the building mechanism of the LFR benchmark so that it does include triangles. This is actually a potentially interesting improvement of the benchmark, which deserves some attention in the future.

Another important remark is in order. Our whole analysis has made use of graphs with a "flat" community structure, without hierarchy. Many real networks instead have a hierarchical community structure, with communities inside other communities. Good methods must be able to understand when a network has no communities, a flat or a hierarchical community structure. For an analysis of this kind we would need hierarchical benchmarks. There is actually a hierarchical version of the GN benchmark [61], not yet one of the LFR benchmark, which is sorely needed. Methods to find communities in multipartite graphs have yet to be tested as well.

From all of the above it is clear that this paper does not "kill" the issue of the actual efficiency and reliability of community detection methods. Our analysis represents a step, but it is clear that much more needs to be done along these lines. We are also aware that our subset of algorithms, while it includes the most popular methods used and discussed in the literature, is still a very limited sample of the endless variety of existing algorithms, whose number has literally exploded over the last few years. A thorough analysis of all existing methods would require years of work (during which many new methods would appear), and we do not believe it is feasible. On the other hand, we hope that the developers of algorithms not included in our analysis will try to perform themselves the tests on our benchmark graphs (the code to build the graphs can be freely downloaded [62]) and compare their results with the ones presented in this analysis for a fair evaluation. We are ready and willing to provide all necessary information and the sets of graphs used for this paper. More generally we hope that in the future scholars will submit new methods to the same type of strict tests before presenting them to the community.

ACKNOWLEDGMENTS

We thank L. Donetti, R. Lambiotte, F. Radicchi, P. Ronhovde, and M. Rosvall for kindly providing their code. We are also grateful to V. Latora, T. Nepusz, A. Pluchino, A. Rapisarda, M. Sales-Pardo, and C. Wiggins for useful suggestions. We gratefully acknowledge ICTeCollective, Grant No. 238597, of the European Commission.

[1] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, Phys. Rep. **424**, 175 (2006).
[3] M. Girvan and M. E. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[4] S. Fortunato, e-print arXiv:0906.0612.
[5] A. Condon and R. M. Karp, Random Struct. Algorithms **18**, 116 (2001).
[6] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech.: Theory Exp. (2005), P09008.
[7] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **401**, 130 (1999).
[8] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[9] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, Phys. Rev. E **68**, 065103(R) (2003).
[10] L. Danon, J. Duch, A. Arenas, and A. Díaz-Guilera, in *Large Scale Structure and Dynamics of Complex Networks: From Information Technology to Finance and Natural Science*, edited by G. Caldarelli and A. Vespignani (World Scientific, Singapore, 2007), pp. 93–114.
[11] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[12] A. Lancichinetti, S. Fortunato, and J. Kertesz, New J. Phys. **11**, 033015 (2009).
[13] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[14] A. Lancichinetti and S. Fortunato, Phys. Rev. E **80**, 016118 (2009).
[15] M. Molloy and B. Reed, Random Struct. Algorithms **6**, 161 (1995).
[16] G. Bianconi, P. Pin, and M. Marsili, Proc. Natl. Acad. Sci. U.S.A. **106**, 11433 (2009).
[17] J. Reichardt and M. Leone, Phys. Rev. Lett. **101**, 078701 (2008).
[18] A. Lancichinetti, F. Radicchi, and J. Ramasco, e-print arXiv:0907.3708.
[19] S. Fortunato and C. Castellano, in *Encyclopedia of Complexity and Systems Science*, edited by R. A. Meyers (Springer, Berlin, 2009), Vol. 1.
[20] M. Meilă, J. Multivariate Anal. **98**, 873 (2007).
[21] A. Traud, E. Kelsic, P. Mucha, and M. Porter, e-print arXiv:0809.0690.
[22] D. J. C. Mackay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, 2003).
[23] The software can be downloaded from http://santo.fortunato.googlepages.com/inthepress2
[24] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[25] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[26] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, Phys. Rev. E **70**, 025101(R) (2004).
[27] C. P. Massen and J. P. K. Doye, Phys. Rev. E **71**, 046101 (2005).
[28] A. Medus, G. Acuña, and C. O. Dorso, Physica A **358**, 593 (2005).
[29] R. Guimerà and L. A. N. Amaral, Nature (London) **433**, 895 (2005).
[30] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. (2008), P10008.
[31] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. **101**, 2658 (2004).
[32] C. Bron and J. Kerbosch, Commun. ACM **16**, 575 (1973).
[33] S. van Dongen, Ph.D. thesis, University of Utrecht, 2000.
[34] M. Rosvall and C. T. Bergstrom, Proc. Natl. Acad. Sci. U.S.A. **104**, 7327 (2007).
[35] M. Rosvall and C. T. Bergstrom, Proc. Natl. Acad. Sci. U.S.A. **105**, 1118 (2008).
[36] J. Rissanen, Automatica **14**, 465 (1978).
[37] P. D. Grünwald, I. J. Myung, and M. A. Pitt, *Advances in Minimum Description Length: Theory and Applications* (MIT Press, Cambridge, 2005).
[38] L. Donetti and M. A. Muñoz, J. Stat. Mech.: Theory Exp. (2004), P10012.
[39] C. Lanczos, J. Res. Natl. Bur. Stand. **45**, 255 (1950).
[40] M. E. J. Newman and E. A. Leicht, Proc. Natl. Acad. Sci. U.S.A. **104**, 9564 (2007).
[41] A. P. Dempster, N. M. Laird, and D. B. Rdin, J. R. Stat. Soc. Ser. B (Methodol.) **39**, 1 (1977).
[42] P. Ronhovde and Z. Nussinov, Phys. Rev. E **80**, 016109 (2009).
[43] S. Fortunato and M. Barthélemy, Proc. Natl. Acad. Sci. U.S.A. **104**, 36 (2007).
[44] J. J. Ramasco and M. Mungan, Phys. Rev. E **77**, 036122 (2008).
[45] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, Proc. Natl. Acad. Sci. U.S.A. **101**, 3747 (2004).
[46] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. M. Ismail, and N. Preston, in *IADIS Applied Computing 2005, Algarve, Portugal*, edited by N. Guimaraes and P. T. Isaias (unpublished), pp. 97–104.
[47] S. Zhang, R.-S. Wang, and X.-S. Zhang, Physica A **374**, 483 (2007).
[48] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó, Phys. Rev. E **77**, 016107 (2008).
[49] T. Evans and R. Lambiotte, Phys. Rev. E **80**, 016105 (2009).
[50] Y. Ahn, J. Bagrow, and S. Lehmann, e-print arXiv:0903.3178.
[51] S. Gregory, in *Complex Networks*, edited by S. Fortunato, R. Menezes, G. Mangioni, and V. Nicosia, Studies on Computational Intelligence Vol. 207 (Springer, Berlin, 2009), pp. 47–62.
[52] E. N. Sawardecker, M. Sales-Pardo, and L. A. N. Amaral, Eur. Phys. J. B **67**, 277 (2009).
[53] P. Erdös and A. Rényi, Publ. Math. (Debrecen) **6**, 290 (1959).
[54] J. Reichardt and S. Bornholdt, Phys. Rev. E **74**, 016110 (2006).
[55] J. Reichardt and S. Bornholdt, Physica D **224**, 20 (2006).
[56] J. Reichardt and S. Bornholdt, Phys. Rev. E **76**, 015102(R) (2007).
[57] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, Science **298**, 824 (2002).
[58] S. Itzkovitz, R. Milo, N. Kashtan, G. Ziv, and U. Alon, Phys. Rev. E **68**, 026127 (2003).
[59] S. Itzkovitz and U. Alon, Phys. Rev. E **71**, 026117 (2005).
[60] D. Watts and S. Strogatz, Nature (London) **393**, 440 (1998).
[61] A. Arenas, A. Díaz-Guilera, and C. J. Pérez-Vicente, Phys. Rev. Lett. **96**, 114102 (2006).
[62] http://santo.fortunato.googlepages.com/inthepress2