Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities

Andrea Lancichinetti and Santo Fortunato
Complex Networks Lagrange Laboratory (CNLL), Institute for Scientific Interchange (ISI), Viale S. Severo 65, 10133 Torino, Italy
(Received 24 April 2009; revised manuscript received 10 June 2009; published 31 July 2009)

Many complex networks display a mesoscopic structure with groups of nodes sharing many links with the other nodes in their group and comparatively few with nodes of different groups. This feature is known as community structure and encodes precious information about the organization and the function of the nodes. Many algorithms have been proposed but it is not yet clear how they should be tested. Recently we have proposed a general class of undirected and unweighted benchmark graphs, with heterogeneous distributions of node degree and community size. An increasing attention has been recently devoted to develop algorithms able to consider the direction and the weight of the links, which require suitable benchmark graphs for testing. In this paper we extend the basic ideas behind our previous benchmark to generate directed and weighted networks with built-in community structure. We also consider the possibility that nodes belong to more communities, a feature occurring in real systems, such as social networks. As a practical application, we show how modularity optimization performs on our benchmark.

DOI: 10.1103/PhysRevE.80.016118 PACS number(s): 89.75.Hc

I. INTRODUCTION

Complex systems are characterized by a division in subsystems, which in turn contain other subsystems in a hierarchical fashion. Herbert A. Simon, already in 1962, pointed out that such hierarchical organization plays a crucial role both in the generation and in the evolution of complex systems [1]. Many complex systems can be described as graphs, or networks, where the elementary parts of a system and their mutual interactions are nodes and links, respectively [2,3]. In a network, the subsystems appear as subgraphs with a high density of internal links, which are loosely connected to each other. These subgraphs are called communities and occur in a wide variety of networked systems [4,5]. Communities reveal how a network is internally organized, and indicate the presence of special relationships between the nodes that may not be easily accessible from direct empirical tests. Communities may be groups of related individuals in social networks [4,6], sets of Web pages dealing with the same topic [7], biochemical pathways in metabolic networks [8,9], etc.

For these reasons, 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, computer and social sciences. However, there is no agreement yet about a set of reliable algorithms that one can use in applications. The main reason is that current techniques have not been thoroughly tested. Usually, when a new method is presented, it is applied to a few simple benchmark graphs, artificial or from the real world, which have a known community structure. The most used benchmark is a class of graphs introduced by Girvan and Newman [4]. Each graph consists of 128 nodes, which are divided into four groups of 32: the probabilities of the existence of a link between a pair of nodes of the same group and of different groups are p_{in} and p_{out}, respectively. This benchmark is a special case of the *planted* ℓ-partition model [10]. However, it 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 [2,3,11] and community sizes [9,12–15]. In a recent paper [16], we have discussed a class of benchmark graphs, that generalize the benchmark by Girvan and Newman by introducing power-law distributions of degree and community size. Most community detection algorithms perform very well on the benchmark by Girvan and Newman, due to the simplicity of its structure. The benchmark, instead, poses a much harder test to algorithms, and makes it easier to disclose their limits.

Most research on community detection focuses on the simplest case of undirected and unweighted graphs, as the problem is already very hard. However, links of networks from the real world are often directed and carry weights, and both features are essential to understand their function [17,18]. Moreover, in real graphs communities are sometimes overlapping [9], i.e., they share vertices. This aspect, frequent in certain types of systems, such as social networks, has received some attention in the last years [15,19–21]. Finding communities in networks with directed and weighted edges and possibly overlapping communities is highly nontrivial. Many techniques working on undirected graphs, for instance, cannot be extended to include link direction. This implies the need of new approaches to the problem. In any case, once a method is designed, it is important to test it against reliable benchmarks. Since the benchmark of Ref. [16] is defined for undirected and unweighted graphs, we extend it here to the directed and weighted cases. For any type of benchmark, we will include the possibility to have overlapping communities. Sawardecker *et al.* have recently proposed a different benchmark with overlapping communities where the probability that two nodes are linked grows with the number of communities both nodes belong to [22].

Our algorithms to create the benchmark graphs have a computational complexity which grows linearly with the number of links and reduce considerably the fluctuations of

specific realizations of the graphs, so that they come as close as possible to the type of structure described by the input parameters. We use our benchmark to make some testing of modularity optimization [23], which is well defined in the case of directed and weighted networks [24].

In Sec. II we describe the algorithms to create the new benchmarks. Tests are presented in Sec. III. Conclusions are summarized in Sec. IV.

II. BENCHMARK

We start by presenting the algorithm to build the benchmark for undirected graphs with overlaps between communities. Then we extend it to the case of weighted and directed graphs.

A. Unweighted benchmark with overlapping nodes

The aim of this section is to describe the algorithm to generate undirected and unweighted benchmark graphs, where each node is allowed to have memberships in more communities. The algorithm consists of the following steps:

(1) We first assign the number ν_i of memberships of node i, i.e., the number of communities the node belongs to. Of course, if each node has only one membership, we recover the benchmark of Ref. [16]; in general we can assign the number of memberships according to a certain distribution. Next, we assign the degrees $\{k_i\}$ by drawing N random numbers from a power-law distribution [25] with exponent τ_1. We also introduce the *topological mixing parameter* μ_t: $k_i^{(in)}=(1-\mu_t)k_i$ is the internal degree of the node i, i.e., the number of neighbors of node i which have at least one membership in common with i. In this way, the internal degree is a fixed fraction of the total degree for all the nodes. Of course, it is straightforward to generalize the algorithm to implement a different rule (one can introduce a nonlinear functional dependence, individual mixing parameters, etc.).

(2) The community sizes $\{s_\xi\}$ are assigned by drawing random numbers from another power law with exponent τ_2. Naturally, the sum of the community sizes must equal the sum of the node memberships, i.e., $\Sigma_\xi s_\xi=\Sigma_i \nu_i$. Furthermore $s_{\max}=\max\{s_\xi\}\leq N$ and $\nu_{\max}=\max\{\nu_i\}\leq n_c$, where N is the number of nodes and n_c the number of communities. At this point, we have to decide which communities each node should be included into. This is equivalent to generating a bipartite network where the two classes are the n_c communities and the N nodes; each community ξ has s_ξ links, whereas each node has as many links as its memberships ν_i (Fig. 1). The network can be easily generated with the configuration model [26]. To build the graph, it is important to take into account the constraint

$$\sum_{i\to\xi} s_\xi \geq k_i^{(in)}, \quad \forall \quad i, \tag{1}$$

where the sum is relative to the communities including node i. This condition means that each node cannot have an internal degree larger than the highest possible number of nodes it can be connected to within the communities it stays in. We perform a rewiring process for the bipartite network until the constraint is satisfied. For some choices of the input parameters, it could happen that, after some iterations, the constraint is still unsatisfied. In this case one can change the sizes of the communities, by merging some of them, for instance. It turns out that this is not necessary in most situations and that, when it is, the perturbations introduced in the community size distributions are not too large. In general, it is convenient to start with a distribution of community sizes such that $s_{\min}\geq k_{\min}^{(in)}$ and $s_{\max}\geq k_{\max}^{(in)}$.

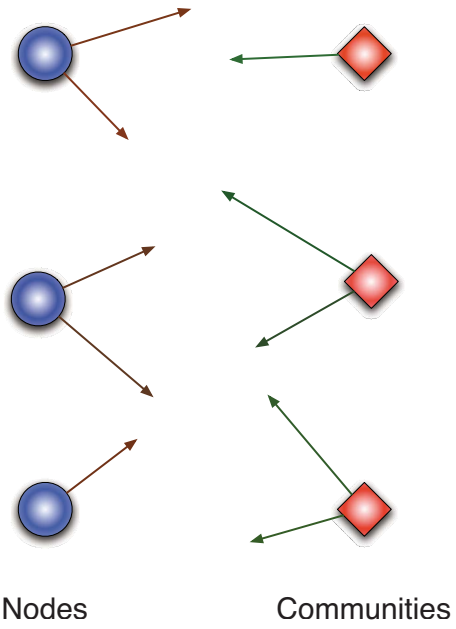

FIG. 1. (Color online) Schematic of the bipartite graph used to assign nodes to their communities. Each node has as many stubs as the number of communities it belongs to, whereas the number of stubs of each community matches the size of the community. The memberships are assigned by joining the stubs on the left with those on the right.

So far we assigned an internal degree to each node but it has not been specified how many links should be distributed among the communities of the node. Again, one can follow several recipes; we chose the simple equipartition $k_i(\xi)=k_i^{(in)}/\nu_i$, where $k_i(\xi)$ is the number of links which i shares in community ξ, provided that i holds membership in ξ. Some adjustments may be necessary to assure

$$(s_\xi)_{i\to\xi} \geq k_i(\xi) \ \forall \quad i, \tag{2}$$

which is the strong version of Eq. (1).

(3) Before generating the whole network, we start generating n_c subgraphs, one for each community. In fact, our definition of community ξ is nothing but a random subgraph of s_ξ nodes with degree sequence $\{k_i(\xi)\}$, which can be built via the configuration model, with a rewiring procedure to avoid multiple links. Note that Eq. (2) is necessary to generate the configuration model, but in general not sufficient. For one thing, we need $\Sigma_i k_i(\xi)$ to be even. This might cause a change in the degree sequence, which is generally not appreciable. Once each subgraph is built, we obtain a graph divided in components. Note that because of the overlapping nodes, some components may be connected to each other,

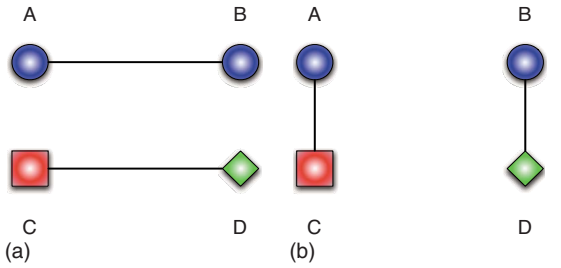

FIG. 2. (Color online) Scheme of the rewiring procedure necessary to build the graph $\mathcal{G}^{(ext)}$, which includes only links between nodes of different communities. (Top) If two nodes (A and B) with a common membership are neighbors, their link is rewired along with another link joining two other nodes C and D, where C does not have memberships in common with A, and D is a neighbor of C not connected to B. In the final configuration (bottom), the degrees of all nodes are preserved, and the number of links between nodes with common memberships has decreased by one (since A and B are no longer connected), or it has stayed the same (if B and D, which are now neighbors, have common memberships).

and in principle the whole graph might be connected. Furthermore, if two nodes belong simultaneously to the same two (or more) communities, the procedure may set more than one link between the nodes. A rewiring strategy similar to that described below suffices to avoid this problem.

(4) The last step of the algorithm consists in adding the links external to the communities. To do this, let us consider the degree sequence $\{k_i^{(ext)}\}$, where simply $k_i^{(ext)}=k_i-k_i^{(in)}=\mu_t k_i$. We want to insert randomly these links in our already built network without changing the internal degree sequences. In order to do so, we build a new network $\mathcal{G}^{(ext)}$ of N nodes with degree sequence $\{k_i^{(ext)}\}$, and we perform a rewiring process each time we encounter a link between two nodes which have at least one membership in common (Fig. 2), since we are supposed to join only nodes of different communities at this stage.

Let us assume that A and B are in the same community and that they are linked in $\mathcal{G}^{(ext)}$; we pick a node C which does not share any membership with A, and we look for a neighbor of C (call it D) which is not neighbor of B. Next, we replace the links A-B and C-D with the new links A-C and B-D. This rewiring procedure can decrease the number of internal links of $\mathcal{G}^{(ext)}$ or leave it unchanged (this happens only when B and D have one membership in common) but it cannot increase it. This means that after a few sweeps over all the nodes we reach a steady state where the number of internal links is very close to zero (if no node has $k_i \sim N$, the internal links of $\mathcal{G}^{(ext)}$ are just a few and one sweep is sufficient). Figure 3 shows how the number of internal links decreases during the rewiring procedure. Finally, we have to superimpose $\mathcal{G}^{(ext)}$ on the previous one.

In our previous work about benchmarking [16], we discussed the dispersion of the internal degree around the fixed value $k_i^{(in)}$. In this case, if the number of internal links of $\mathcal{G}^{(ext)}$ goes to zero, the only reason not to have a perfectly sharp function for the distribution of the mixing parameters of the nodes in specific realizations of the benchmark is a round-off problem, i.e., the problem of rounding integer numbers.

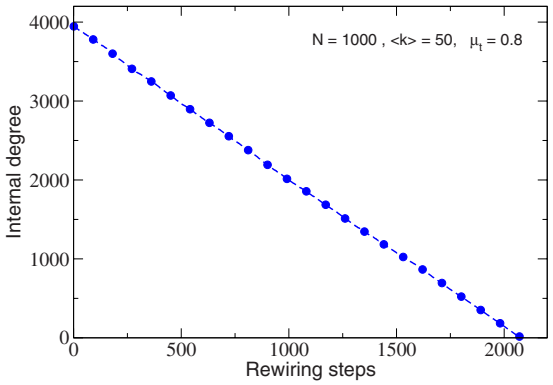

FIG. 3. (Color online) Number of internal links of $\mathcal{G}^{(ext)}$ as a function of the rewiring steps. The network has 1000 nodes, and an average degree $\langle k\rangle=50$. Since the mixing parameter is $\mu_t=0.8$ and there are ten equal-sized communities, at the beginning each node has an expected internal degree in $\mathcal{G}^{(ext)}$ $k_i^{(in)}=0.8\times 50\times 1/10=4$, so the total internal degree is around 4000. After each rewiring step, the internal degree either decreases by 2, or it does not change. In this case, less then 2100 rewiring steps were needed.

Other benchmarks, like that by Girvan and Newman, are based on a similar definition of communities, expressed in terms of different probabilities for internal and external links. One may wonder what is the connection between our benchmark and the others. It is not difficult to compute an approximation of how the probability of having a link between two nodes in the same community depends on the mixing parameter μ_t.

In the configuration model, the probability to have a connection between nodes i and j with k_i and k_j links, respectively, is approximately $p_{ij}=\frac{k_i k_j}{2m}$, provided that $k_i \ll 2m$ and $k_j \ll 2m$. If the approximation holds, our prescription to assign $k_i(\xi)$ allows us to compute the probability that i and j get a link in the community ξ,

$$p_{ij}(\xi) \simeq \frac{k_i(\xi)k_j(\xi)}{2m_\xi} = (1-\mu_t)^2 \frac{1}{\nu_i \nu_j}\frac{k_i k_j}{2m_\xi}, \tag{3}$$

where $2m_\xi=\Sigma_i k_i(\xi)$ is the number of internal links in the community (we recall that ν_i is the number of memberships of node i). If i and j share a number ν_{ij} of memberships and all the respective $p_{ij}(\xi)$ are small, the probability that they get a link somewhere can be approximated with the sum over all the common communities. The final result is

$$p_{ij} \simeq (1-\mu_t)^2 k_i k_j \frac{\nu_{ij}}{\nu_i \nu_j}\left\langle \frac{1}{2m_\xi}\right\rangle_\xi, \tag{4}$$

where $\langle \frac{1}{2m_\xi}\rangle_\xi=(1/\nu_{ij})\Sigma_\xi \frac{1}{2m_\xi}$, and ξ runs only over the common memberships of the nodes.

On the other hand, if i and j do not share any membership, the probability to have a link between them is

$$p_{ij} \simeq \frac{k_i^{(ext)}k_j^{(ext)}}{2m^{(ext)}} = \mu_t \frac{k_i k_j}{2m}, \tag{5}$$

where $2m^{(ext)}=\Sigma_i k_i^{(ext)}=\mu_t \Sigma_i k_i$ is the number of external links in the network. The equation holds only if the rewiring pro-

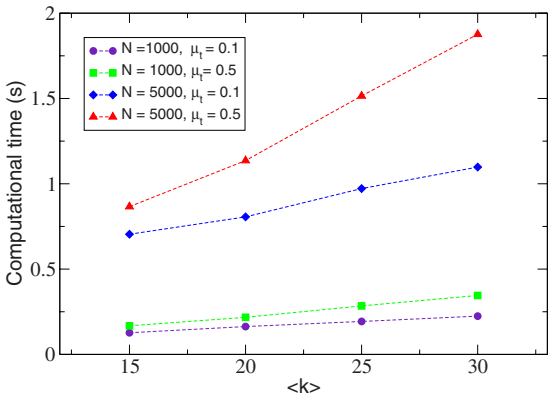

FIG. 4. (Color online) Computational time to build the unweighted benchmark as a function of the average degree. We show the results for networks of 1000 and 5000 nodes. μ_t was set equal to 0.1 and 0.5 (the latter requires more time for the rewiring process). Note that between the two upper lines and the lower ones there is a factor of about 5, as one would expect if complexity is linear in the number of links m.

cess does not affect too much the probabilities, i.e., if the communities are small compared to the size of the network.

These results are based on some assumptions which are likely to be not exactly, but only approximately valid. Anyway, carrying out the right calculation is far from trivial and surely beyond the scope of this paper.

We conclude this section with a remark about the complexity of the algorithm. The configuration model takes a time growing linearly with the number of links m of the network. If the rewiring procedure takes only a few iterations, like it happens in most instances, the complexity of the algorithm is $O(m)$ (Fig. 4).

B. Weighted networks

In order to build a weighted network, we first generate an unweighted network with a given topological mixing parameter μ_t and then we assign a positive real number to each link. To do this we need to specify two other parameters, β and μ_w. The parameter β is used to assign a strength s_i to each node, $s_i=k_i^{\beta}$; such power-law relation between the strength and the degree of a node is frequently observed in real weighted networks [18]. The parameter μ_w is used to assign the internal strength $s_i^{(in)}=(1-\mu_w)s_i$, which is defined as the sum of the weights of the links between node i and all its neighbors having at least one membership in common with i. The problem is equivalent to finding an assignment of m positive numbers $\{w_{ij}\}$ such to minimize the following function:

$$\mathrm{Var}(\{w_{ij}\}) = \sum_i (s_i - \rho_i)^2 + (s_i^{(in)} - \rho_i^{(in)})^2 + (s_i^{(ext)} - \rho_i^{(ext)})^2. \tag{6}$$

Here s_i and $s_i^{(in,ext)}$ indicate the strengths which we would like to assign, i.e., $s_i=k_i^{\beta}$, $s_i^{(in)}=(1-\mu_w)s_i$, $s_i^{(out)}=\mu_w s_i$; $\{\rho_i^*\}$ are the total, internal, and external strengths of node i defined through its link weights, i.e., $\rho_i=\Sigma_j w_{ij}$, $\rho_i^{(in)}=\Sigma_j w_{ij}\kappa(i,j)$, $\rho_i^{(out)}=\Sigma_j w_{ij}[1-\kappa(i,j)]$, where the function $\kappa(i,j)=1$ if nodes i and j share at least one membership, and $\kappa(i,j)=0$ otherwise.

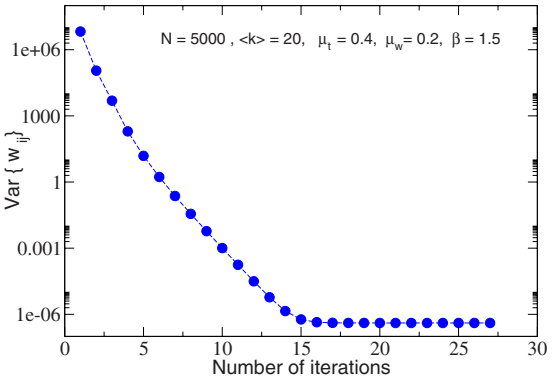

FIG. 5. (Color online) Value of Var($\{w_{ij}\}$) [Eq. (6)] after each update. Each point corresponds to one sweep over all the nodes.

We have to arrange things so that s_i and $s_i^{(in,ext)}$ are consistent with the $\{\rho_i^*\}$. For that we need a fast algorithm to minimize Var($\{w_{ij}\}$). We found that the greedy algorithm described below can do this job well enough for the cases of our interest.

(1) At the beginning $w_{ij}=0$, $\forall\ i,j$, so all the $\{\rho_i^*\}$ are zero.

(2) We take node i and increase the weight of each of its links by an amount $u_i=\frac{s_i-\rho_i}{k_i}$, where ρ_i indicates the sum of the links' weights resulting from the previous step, i.e., before we increment them. In this way, since initially $\{\rho_i^*\}=0$, the weights of the links of i after the first step take the (equal for all) value $\frac{s_i}{k_i}$, and $\rho_i=s_i$ by construction, condition that is maintained along the whole procedure. We update $\{\rho_i^*\}$ for the node i and its neighbors.

(3) Still for node i we increase all the weights w_{ij} by an amount $\frac{s_i^{(in)}-\rho_i^{(in)}}{k_i^{(in)}}$ if $\kappa(i,j)=1$ and by an amount $-\frac{s_i^{(in)}-\rho_i^{(in)}}{k_i^{(ext)}}$ if $\kappa(i,j)=0$. Again we update $\{\rho_i^*\}$ for the node i and its neighbors. These two steps assure to set the contribute of node i in Var($\{w_{ij}\}$) to zero.

(4) We repeat steps (2) and (3) for all the nodes. Two remarks are in order. First, we want each weight $w_{ij}>0$; so we update the weights only if this condition is fulfilled. Second, the contribute of the neighbors of node i in Var($\{w_{ij}\}$) will change and, of course, it can increase or decrease. For this reason, we need to iterate the procedure several times until a steady state is reached, or until we reach a certain value. With our procedure the value of Var($\{w_{ij}\}$) decreases at least exponentially with the number of iterations, consisting in sweeps over all network links (Fig. 5).

For the distribution of the weights w_{ij}, we expect the averages $\langle w_i^{(int)}\rangle=(1/k_i^{(in)})\Sigma_j w_{ij}\kappa(i,j)=s_i^{(in)}/k_i^{(in)}$ and $\langle w_i^{(ext)}\rangle=s_i^{(ext)}/k_i^{(ext)}$. Note that these expressions can be related to the mixing parameters in a simple way (Fig. 6),

$$\langle w_i^{(int)}\rangle = \frac{1-\mu_w}{1-\mu_t}k_i^{\beta-1} \quad \text{and} \quad \langle w_i^{(ext)}\rangle = \frac{\mu_w}{\mu_t}k_i^{\beta-1}. \tag{7}$$

Since Var($\{w_{ij}\}$) decreases exponentially, the number of iterations needed to reach convergence has a slow dependence

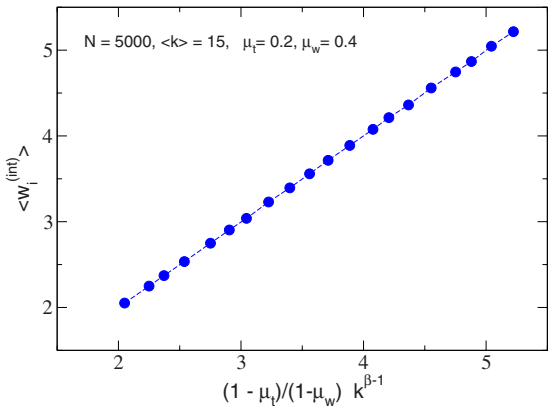

FIG. 6. (Color online) The average weight of an internal link for a node depends on its degree according to Eq. (7). The correlation plot in the figure, relative to a network of 5000 nodes, confirms the result.

on the size of the network so it does not contribute much to the total complexity, which remains $O(m)$ (Fig. 7).

C. Directed networks

It is quite straightforward to generalize the previous algorithms to generate directed networks. Now, we have an indegree sequence $\{y_i\}$ and an outdegree sequence $\{z_i\}$ but we can still go through all the steps of the construction of the benchmark for undirected networks with just some slight modifications. In the following, we list what to change in each point of the corresponding list in Sec. II A.

(1) We decided to sample the indegree sequence from a power law and the outdegree sequence from a δ distribution (with the obvious constraint $\Sigma_i y_i = \Sigma_i z_i$). We need to define the internal in- and outdegrees $y_i(\xi)$ and $z_i(\xi)$ with respect to every community ξ, which can be done by introducing two mixing parameters. For simplicity one can set them equal.

(2) It is necessary that Eq. (2) holds for both $\{y_i\}$ and $\{z_i\}$.

(3) We need to use the configuration model for directed networks, and the condition that $\Sigma_i k_i(\xi)$ should be even is replaced by $\Sigma_i y_i(\xi) = \Sigma_i z_i(\xi)$; because of this condition it might be necessary to change $y_i(\xi)$ and/or $z_i(\xi)$. We decided to modify only $z_i(\xi)$, whenever necessary.

(4) The rewiring procedure can be done by preserving both distributions of indegree and outdegree, for instance, by adopting the following scheme: before rewiring, A points to B and D to C; after rewiring, A points to C and D to B.

In order to generate directed and weighted networks, we use the following relation between the strength s_i of a node and its in- and outdegree: $s_i = (y_i + z_i)^{\beta}$. Given a node i, one considers all its neighbors, regardless of the link directions (note that i may have the same neighbor counted twice if the link runs in both directions). Otherwise, the procedure to insert weights is equivalent.

In directed networks, the directedness of the links may reflect some interesting structural information that is not present in the corresponding undirected version of the graph. For instance there could be flows, represented by many links with the same direction running from one subgraph to another: such subgraphs might correspond to important classifications of the nodes. Our directed benchmark is based on the balance between the numbers of internal and external links, and it does not seem suitable to generate graphs with flows. However, this is not true: flows can be generated by introducing proper constraints on the number of incoming and outgoing links of the communities.

Suppose we want to generate a network with two communities only, where the nodes of community 1 point to nodes of community 2 but not vice versa and there are few random connections among nodes in the same community. We could use our algorithm in this way: first we build separately the two subgraphs; then we set $y_i^{(ext)} \simeq 0$ for nodes in the community 1 and $z_i^{(ext)} \simeq 0$ for nodes in community 2 and build $\mathcal{G}^{(ext)}$. If there are more communities, one first builds as many subgraphs as necessary and then links them according to the desired flow patterns.

Methods based on mixture models [27,28] may detect this kind of structures. Methods based on a balance between internal and external links, like (directed) modularity optimization may have problems. For example (Fig. 8), consider a network with three communities A, B, C, with ten nodes in each community, each node with three in-links and three out-links on average; nodes in A point to two nodes in B, nodes in B point to two nodes in C, and nodes in C point to two nodes in A; each node points to one node in its own community. The modularity of this partition is $Q=0$, therefore the optimization would give a different partition, as the maximum modularity for a graph is usually positive.

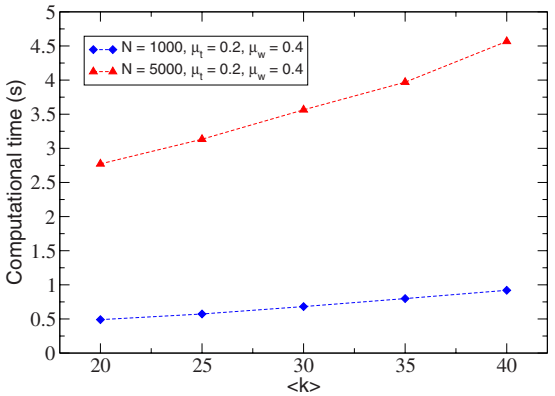

FIG. 7. (Color online) Computational time to build the weighted benchmark as a function of the average degree. We show the results for networks of 1000 and 5000 nodes. μ_t was set equal to 0.2 and μ_w to 0.4.

III. TESTS

Here we present some tests of community detection methods on our benchmark graphs. We focused on two techniques: modularity optimization, because it is one of very few methods that can be extended to the cases of directed and weighted graphs [24]; the clique percolation method (CPM) by Palla *et al.* [9], a popular method to find community structure with overlapping communities. The optimization of modularity was carried out by using simulated annealing [8].

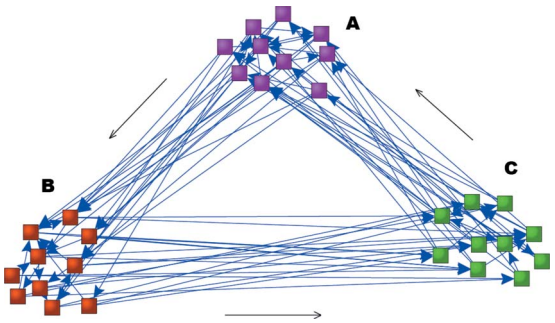

FIG. 8. (Color online) Example of directed graph with a flow running in a cycle between three groups of nodes. The directedness of the links enables to distinguish the three groups, and there are methods able to detect them. Standard community detection methods, instead, are likely to fail. For instance, the value of the directed modularity for the partitions in the three groups is zero, whereas the maximum modularity for the graph is positive and corresponds to a different partition.

To measure the similarity between the built-in modular structure of the benchmark and the one delivered by the algorithm we adopt the *normalized mutual information*, a measure borrowed from information theory [29]. We stress that other choices for the similarity measure are possible (for a survey, see [31]) and that we use the normalized mutual information for two main reasons: (1) it is regularly used in papers about community detection, so one has a clear idea of the performance of the algorithms by looking at the results, compared to similar plots; (2) it has been recently extended to the case of overlapping communities [15], whereas most other measures have no such extension.

Figure 9 shows the result for the directed (unweighted) benchmark graphs, without overlapping communities. The plot shows a very similar pattern as that observed in the undirected case [16].

For the weighted benchmark (still without overlapping communities) we can tune two parameters, μ_t and μ_w. Figure

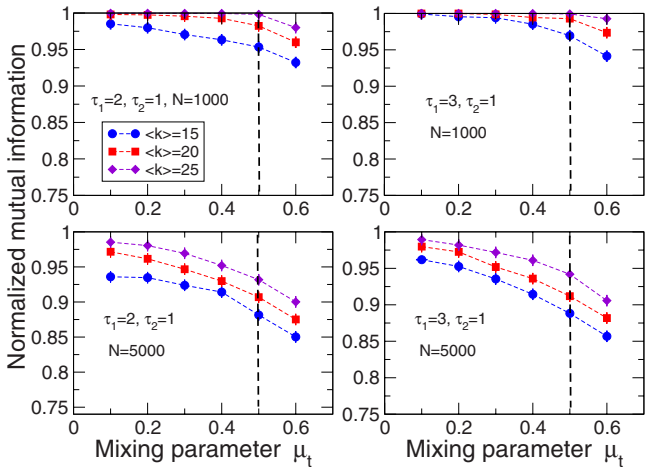

FIG. 9. (Color online) Test of modularity optimization on our benchmark for directed networks. The results get worse by increasing the number of nodes and/or decreasing the average degree, as we had found for the undirected case in Ref. [16]. Each point corresponds to an average over 100 graph realizations.

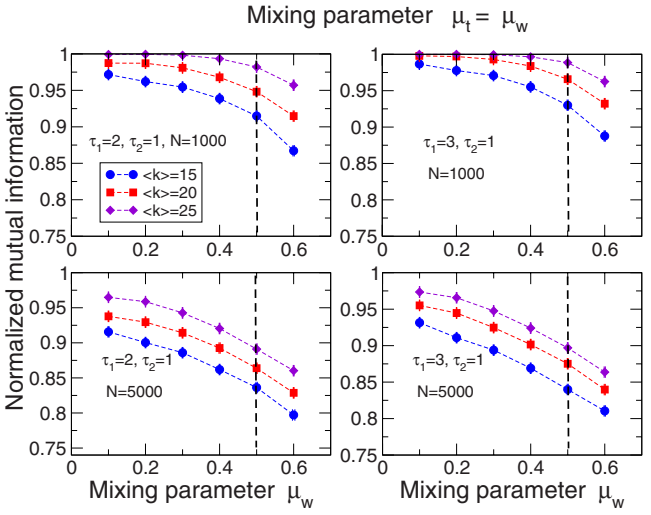

FIG. 10. (Color online) Test of modularity optimization on our benchmark for weighted undirected networks without overlaps between communities. The topological mixing parameter μ_t equals the strength mixing parameter μ_w. Each point corresponds to an average over 100 graph realizations.

10 refers to networks where we set $\mu_t=\mu_w$, while in Fig. 11 we set $\mu_t=0.5$. Since, for $\mu_w<0.5$, μ_t is smaller for the networks of Fig. 10 than for those in Fig. 11, we would expect to see better performances of modularity optimization in Fig. 10 in the range $0\leq\mu_w<0.5$. Instead, we get the opposite result. The reason is that the links between communities carry on average more weight when $\mu_t<\mu_w$ than when $\mu_t=\mu_w$, and this enhances the chance that mergers between small communities occur, leading to higher values of modularity [30]. Because of such mergers, the partition found by the method can be quite different from the planted partition of the benchmark.

In Figs. 12 and 13 we show the results of tests performed with the CPM on our benchmarks with overlapping commu-

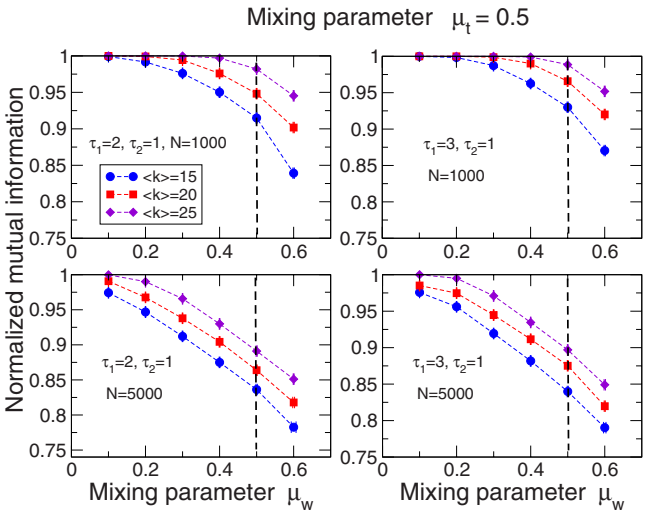

FIG. 11. (Color online) Test of modularity optimization on our benchmark for weighted undirected networks. The topological mixing parameter $\mu_t=0.5$. All other parameters are the same as in Fig. 10. Each point corresponds to an average over 100 graph realizations.

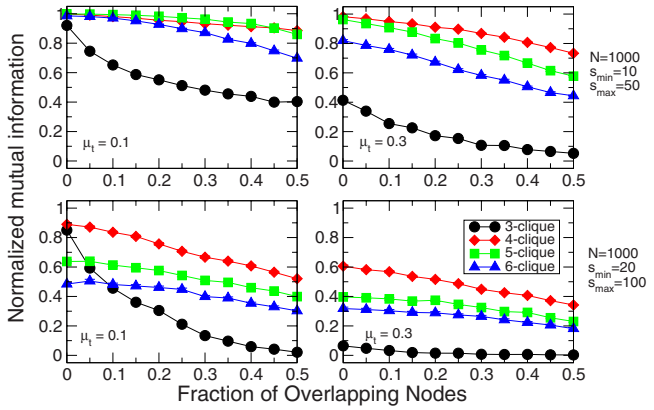

FIG. 12. (Color online) Test of the clique percolation method by Palla *et al.* [9] on our benchmark for undirected and unweighted networks with overlapping communities. The plot shows the variation of the normalized mutual information between the planted and the recovered partition, in its generalized form for overlapping communities [15], with 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 $k_{\max}=50$.

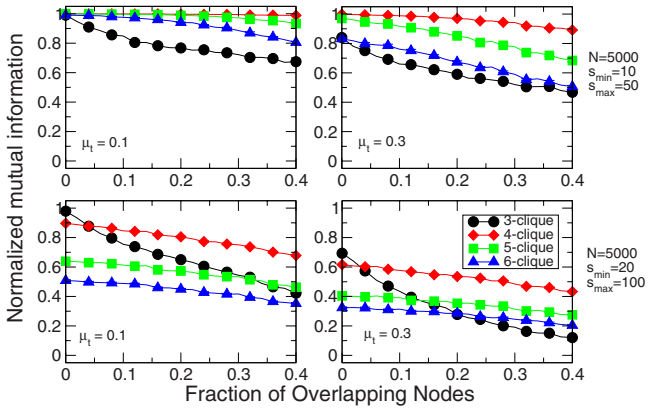

FIG. 13. (Color online) Test of the clique percolation method by Palla *et al.* [9] on our benchmark for undirected and unweighted networks with overlapping communities. The networks have 5000 nodes, the other parameters are the same used for the graphs of Fig. 12.

nities. In this case, the mixing parameter μ_t is fixed and one varies the fraction of overlapping nodes between communities. We have run the CPM 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 4 and 5 cliques give better results. In the two top diagrams community sizes range between $s_{\min}=10$ and $s_{\max}=50$, whereas in the bottom diagrams the range goes from $s_{\min}=20$ and $s_{\max}=100$. 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. 12 consist of 1000 nodes, whereas those of Fig. 13 consist of 5000 nodes. From the comparison of Fig. 12 with Fig. 13 we see that the algorithm performs better on networks of larger size.

IV. SUMMARY

In this paper we have discussed benchmark graphs to test community detection methods on directed and weighted networks. The graphs are suitable extensions of the benchmark we have recently introduced in Ref. [16], in that they account for the fat-tailed distributions of node degree and community size that are observed in real networks. Furthermore we have equipped all our benchmark graphs with the option of having overlapping communities, an important feature of community structure in real networks. With this work we have provided researchers working on the problem of detecting communities in graphs with a complete set of tools to make stringent objective tests of their algorithms, something which is sorely needed in this field. We have developed and carefully tested a software package for the generation of each class of benchmark graphs, all of which can be freely downloaded [32].

ACKNOWLEDGMENTS

We thank F. Radicchi and J. J. Ramasco for useful suggestions.

[1] H. A. Simon, Proc. Am. Philos. Soc. **106**, 467 (1962).
[2] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[3] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Phys. Rep. **424**, 175 (2006).
[4] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[5] S. Fortunato, e-print arXiv:0906.0612.
[6] D. Lusseau and M. E. J. Newman, Proc. Biol. Sci. **271**, S477 (2004).
[7] G. W. Flake, S. Lawrence, C. Lee Giles, and F. M. Coetzee, IEEE Comput. Graphics Appl. **35**, 66 (2002).
[8] R. Guimerà and L. A. Nunes Amaral, Nature (London) **433**, 895 (2005).
[9] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[10] A. Condon and R. M. Karp, Random Struct. Algorithms **18**, 116 (2001).
[11] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **401**, 130 (1999).
[12] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt and A. Arenas, Phys. Rev. E **68**, 065103(R) (2003).
[13] 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.
[14] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).

[15] A. Lancichinetti, S. Fortunato, and J. Kertész, New J. Phys. **11**, 033015 (2009).
[16] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[17] E. A. Leicht and M. E. J. Newman, Phys. Rev. Lett. **100**, 118703 (2008).
[18] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, Proc. Natl. Acad. Sci. U.S.A. **101**, 3747 (2004).
[19] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. M. Ismail, and N. Preston, in *IADIS AC*, edited by N. Guimaraes and P. T. Isaias (2005), p. 97.
[20] S. Zhang, R.-S. Wang, and X.-S. Zhang, Physica A **374**, 483 (2007).
[21] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó, Phys. Rev. E **77**, 016107 (2008).
[22] E. N. Sawardecker, M. Sales-Pardo, and L. A. N. Amaral, Eur. Phys. J. B **67**, 277 (2009).
[23] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[24] A. Arenas, J. Duch, A. Fernández, and S. Gómez, New J. Phys. **9**, 176 (2007).
[25] We stress that, since the exponent τ_1 can be arbitrarily chosen, in the limit of large τ_1 one converges to a δ function, so all nodes have the same degree. The same holds for the distribution of community sizes, which is characterized as well by a tunable power-law exponent τ_2. Therefore, our benchmark is a true generalization of the benchmark by Girvan and Newman, which is recovered in the limit where the exponents are infinitely large.
[26] M. Molloy and B. Reed, Random Struct. Algorithms **6**, 161 (1995).
[27] M. E. J. Newman and E. A. Leicht, Proc. Natl. Acad. Sci. U.S.A. **104**, 9564 (2007).
[28] J. J. Ramasco and M. Mungan, Phys. Rev. E **77**, 036122 (2008).
[29] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech.: Theory Exp. (2005) P09008.
[30] S. Fortunato and M. Barthélemy, Proc. Natl. Acad. Sci. U.S.A. **104**, 36 (2007).
[31] M. Meilă, J. Multivariate Anal. **98**, 873 (2007).
[32] The packages can be found in http://santo.fortunato.googlepages.com/inthepress2. In each package there are instruction files that enable to easily operate the software.