SCIENTIFIC REP⚙RTS

OPEN

A Comparative Analysis of Community Detection Algorithms on Artificial Networks

Zhao Yang, René Algesheimer & Claudio J. Tessone

Many community detection algorithms have been developed to uncover the mesoscopic properties of complex networks. However how good an algorithm is, in terms of accuracy and computing time, remains still open. Testing algorithms on real-world network has certain restrictions which made their insights potentially biased: the networks are usually small, and the underlying communities are not defined objectively. In this study, we employ the Lancichinetti-Fortunato-Radicchi benchmark graph to test eight state-of-the-art algorithms. We quantify the accuracy using complementary measures and algorithms' computing time. Based on simple network properties and the aforementioned results, we provide guidelines that help to choose the most adequate community detection algorithm for a given network. Moreover, these rules allow uncovering limitations in the use of specific algorithms given macroscopic network properties. Our contribution is threefold: firstly, we provide actual techniques to determine which is the most suited algorithm in most circumstances based on observable properties of the network under consideration. Secondly, we use the mixing parameter as an easily measurable indicator of finding the ranges of reliability of the different algorithms. Finally, we study the dependency with network size focusing on both the algorithm's predicting power and the effective computing time.

Relationships between constituents of complex systems (be it in nature, society, or technological applications) can be represented in terms of networks. In this portrayal, the elements composing the system are described as nodes and their interactions as links. At the global level, the topology of these interactions – far from being trivial – is in itself of complex nature[1,2]. Importantly, these networks further display some level of organisation at an intermediate scale. At this *mesoscopic* level, it is possible to identify groups of nodes that are heavily connected among themselves, but sparsely connected to the rest of the network. These interconnected groups are often characterised as *communities*, or in other contexts *modules*, and occur in a wide variety of networked systems[3,4].

Detecting communities has grown into a fundamental, and highly relevant problem in network science with multiple applications. First, it allows to unveil the existence of a non-trivial internal network organisation at coarse grain level. This allows further to infer special relationships between the nodes that may not be easily accessible from direct empirical tests[5]. Second, it helps to better understand the properties of dynamic processes taking place in a network. As paradigmatic examples, spreading processes of epidemics and innovation are considerably affected by the community structure of the graph[6].

Taking into account its importance, it is not surprising that many community detection methods have been developed, using tools and techniques from variegated disciplines such as statistical physics, biology, applied mathematics, computer science, and sociology. All these methods aim at improving the identification of meaningful communities, while keeping as low as possible the computational complexity of the underlying algorithm. Clearly, these algorithms are based on slightly different definitions of community, and therefore the results are not always directly comparable. Further, in most real-world applications, a *ground truth* – i.e. a *unique* identification of nodes to communities – is simply non-existent, which makes it even more difficult to assess the reliability of the community detection procedures. To address these shortcomings and test the algorithms' reliability, different benchmarks have been developed.

Essentially, testing a community detection algorithm implies analysing computer-generated or real-world networks with a well defined community structure (a known ground truth) in order to obtain the community decomposition. One of the most used techniques is the GN benchmark (for Girvan & Newman[3]), which is a

URPP Social Networks, University of Zürich, Andreasstrasse 15, CH-8050 Zürich, Switzerland. Correspondence and requests for materials should be addressed to Z.Y. (email: zhao.yang@business.uzh.ch)

Parameter	Value
Number of nodes N	$233 \sim 31948$
Maximum degree	$0.1N$
Maximum community size	$0.1N$
Average degree	20
Degree distribution exponent	-2
Community size distribution exponent	-1
Mixing coefficient μ	$[0.03, 0.75]$

Table 1. Parameters of LFR benchmark graphs. To deal with possible discrepancies in the network properties, we have randomly generated 100 network for every set of parameters. Due to the slow computing speed, Spinglass and Edge betweenness algorithms have been tested only on small networks with $N \leq 1000$.

special case of the planted l–partition model[7] with a prior specification of the number of nodes (128) and equally sized communities (4). When the expected number of links joining a node to others in different groups is smaller than 8, the four groups are strongly defined communities. In these conditions, a well functioning detection algorithm should be able to identify the communities in reasonable time. Different community detection algorithms can be compared based on their performances on the GN benchmark, which has already been done by Danon *et al.*[8]. However, there are several drawbacks to the GN benchmark: All nodes have the same expected degree, communities are separated in the same way, and the network is of an unrealistic small size.

It is a well established fact that most real complex networks are characterised by largely heterogeneous degree distributions[1,2,9] and heterogeneous community sizes[10–12]. For this reason, the GN benchmark cannot be considered as a good proxy for a real network. By consequence, in a newer stream of research[5,13], the authors proposed an alternative benchmark, which is usually referred to as LFR (for Lancichinetti, Fortunato & Radicchi). This method introduces power-law distributions of degree and community size to the graphs to generalise the GN benchmark. The performances of most existing community detection algorithms are good on the GN benchmark. In contrast, the LFR benchmark presents a harder test for algorithms and makes it easier to unveil their limitations. It has been shown that the *mixing parameter*, which is defined as

$$\mu = \frac{\sum_i k_i^{ext}}{\sum_i k_i^{tot}} \tag{1}$$

is the most influential parameter in the LFR benchmark graphs[14]. Here k_i^{ext} and k_i^{tot} stand for the external degree of node i, i.e. the number of edges connecting it to others that belong to different communities, and the total degree of said node. Although it would be possible to define a mixing parameter for each node, it is assumed that μ is a global property and is the same for every node in the LFR benchmark. The reason here is to be consistent with the standard hypotheses of the planted l-partition model[15].

According to the definition of community in a strong sense, each node should have more connections within the community than with the rest of the graph[16]. Therefore, for $\mu > 1/2$ communities in the strong sense disappear. However, it is worth to mention that Lancichinetti and Fortunato[15] found a weaker condition for community detection which can be applied to any version of the planted l-partition model: $\mu < (N - n_c^{max})/N$, where N is the total number of nodes, and n_c^{max} is the size of the largest community. In our study, although we stick to the strong definition of communities, we have also taken the general condition of μ into consideration (see Table 1).

In the following, we briefly review studies comparing community detection algorithms in chronological order[5,8,13–15,17,18] to highlight the research interests shift. In one of the early studies in comparing community detection algorithms, Danon *et al.* had tested ten algorithms on the GN benchmark7[8] and collected estimates of how time complexity scales with network observables. However, the authors were not able to compare the actual computational effort as a result of the small sizes of graphs. Later on, Lancichinetti *et al.* had employed the LFR benchmark to measure the accuracy of two algorithms on undirected unweighted networks without overlapping communities[5] and two algorithms on directed weighted networks with overlapping communities[13]. Concurrently, the authors tested twelve different algorithms on the GN and LFR benchmarks, and random graphs. For the tests on the LFR benchmark, the authors had considered various parameters, including undirected unweighted graphs with non-overlapping communities, directed unweighted graphs with non-overlapping communities, undirected weighted graphs with non-overlapping communities, and undirected unweighted graphs with overlapping communities[15]. Orman and Labatut later tested five community detection algorithms on the LFR benchmark[14]. They measured the accuracy of algorithms and studied the properties of the LFR benchmark graphs. Later, Peel applied two algorithms on both weighted and unweighted networks with 100 nodes and examined the performance of algorithms developed for weighted networks against those for unweighted ones for different parts of the problem space[17]. Recently, Hric *et al.* compared the accuracy of eleven different algorithms on both the LFR benchmark and a collection of real world graphs with sizes vary from 34 to 5189809 nodes[18]. Overall, as an extension of the GN benchmark, the LFR has drawn a lot of attention: Early, researchers employed small artificial and/or real world networks as benchmarks (e.g. the GN benchmark and the Zachary's karate club network); while nowadays people shifted towards the use of large stylised large artificial or real world networks with some kind of ground truth obtained from metadata information (e.g. the LFR benchmark and the DBLP collaboration network[19]). However, as of today, a detailed study of the dependency with the network size is missing as most of the existing

studies include a few, selected, set of values of the number of nodes and the mixing parameter, and do not consider the real computing time needed to perform the analysis.

In this paper, we evaluate eight different state-of-the-art community detection algorithms available in the "igraph" package[20], which is a widely used collection of network analysis tools in R, Python, C and C++, on the LFR benchmark for undirected, unweighted graphs with non-overlapping communities. Details of the algorithms can be found in the methods section. Our contribution is threefold: First and foremost, we provide actual techniques to determine which is the most suited algorithm in most circumstances based on observable properties of the network under consideration. Secondly, we use the mixing parameter as an easily measurable indicator of finding the ranges of reliability of the different algorithms. Finally, we systematically study the dependency with network size focusing on both the algorithm's predicting power and the effective computing time.

Results

In this section, we compare the results of community detection algorithms in terms of accuracy and computing time. The former is defined as a measure of similarity between the modular structure generated by the LFR benchmark $\mathcal{P}$ (see Methods Section) and the partition identified by the respective community detection algorithms $\overline{\mathcal{P}}$. The latter is the real computing time needed to perform the community detection. This section is organised as follows: First, by employing the LFR generative model, we unveil the relationship between the mixing parameter and the accuracy of the community detection algorithms. Accuracy is measured in two different, complementary ways: The normalised mutual information[8], and the ratio between the number of detected communities and the number of communities given by the LFR generating model. Then, we measure the computing time of community detection algorithms and show the relationship between the mixing parameter and the computing time. We then present the mixing parameter as computed from the communities detected by the different algorithms as a function of the input mixing parameter. Last, we present the comparisons of community detection algorithms in terms of accuracy and computing time as a function of network sizes.

The role of the network mixing parameter on accuracy and computing time.   First, we study the accuracy of the community detection algorithms as a function of the mixing parameter μ. To measure the accuracy we have employed the *normalised mutual information*, i.e., NMI. This is a measure borrowed from information theory which has been regularly used in papers comparing community detection algorithms[13].

Defining a *confusion matrix* **N**, where the rows correspond to the 'real' communities, and the columns correspond to the 'found' communities. The element of **N**, N_{ij}, is the number of nodes in the real community i that appear in the j-th detected community. The *normalised mutual information* is then[8]

$$I(\mathcal{P}, \overline{\mathcal{P}}) = \frac{-2\sum_{i=1}^{C}\sum_{j=1}^{\overline{C}} N_{ij} \, \log(N_{ij}N/N_{io}N_{oj})}{\sum_{i=1}^{C} N_{io} \, \log(N_{io}/N) + \sum_{j=1}^{\overline{C}} N_{oj} \, \log(N_{oj}/N)} \tag{2}$$

where the number of communities given by the LFR model is denoted by C and the number of communities detected by the algorithm is denoted by $\overline{C}$. The sum over the i-th row of **N** is denoted N_{io} and the sum over the j-th column is denoted N_{oj}. If the estimated communities are identical to the real ones, $I(\mathcal{P}, \overline{\mathcal{P}})$ equals to 1. If the partition found by the algorithm is totally independent from the real partition, $I(\mathcal{P}, \overline{\mathcal{P}})$ vanishes.

As pointed out in ref. 21, the mutual information can be normalised in different ways. These different normalisation methods are sensitive to different partition properties and have different theoretical properties[21–23]. To get a better overview of the accuracy, we have calculated the NMI by using all these five different definitions (cf. SI). We conclude that in the current study different normalisation procedures provide qualitatively similar behaviours. Just for the sake of brevity, and consistently with Danon *et al.*[8], we report in this section only I_{sum} (i.e. normalisation by the arithmetic mean). The results of the other NMIs are shown in the "Supplementary Information".

The results are shown in Fig. 1. Each panel presents the accuracy of a given community detection algorithm and is subdivided into two plots: The lower axis depict the average value of NMI and the upper ones contain the standard deviation of the measures when repeated over 100 different network realisations. Most of the algorithms can uncover well the communities when the mixing parameter μ is small, as it is apparent from the large values of I in the limit $\mu \to 0$. The accuracy of algorithms decreases, then, with increasing values of both network size and μ. Different algorithms behave differently: the accuracy of Fastgreedy algorithm decreases monotonically, in a smooth fashion and has a very small standard deviation along all the range (Panel (a), Fig. 1). Whereas that of Leading eigenvector algorithm falls rapidly even with small value of μ (Panel (c), Fig. 1). All the other algorithms display abrupt changes of behaviour: their performances remain relatively stable before a turning point where the NMI drops very fast as a function of μ. The changes of behaviour are usually around $\mu = 1/2$, which corresponds to the strong definition of community[16]. Interestingly, Label propagation and Edge betweenness algorithms have turning points smaller than said value; while Infomap, Multilevel, Walktrap, and Spinglass algorithms have turning points greater than $\mu = 1/2$. We have also noticed that for the Infomap algorithm the normalised mutual information has a point of discontinuous behaviour at around $\mu \cong 0.55$. On the other hand, for Label propagation, I vanishes around $\mu \cong 0.5$ falling in a continuous fashion. This supports the conjecture that Infomap displays a first order phase transition as a function of the mixing parameter, while Label propagation algorithm may have a second order one. Nonetheless, we have not performed an exhaustive analysis on the matter to systematically analyse the existence (or not) of critical points. Further studies concerning the properties of these points are definitely needed.

Network size also plays the role here that a larger network size will lead to loss of accuracy at a lower value of μ. For small enough networks ($N \leq 1000$), Infomap, Multilevel, Walktrap, and Spinglass outperform the other algorithms with higher values of I and very small standard deviations, which shows the repeatability of
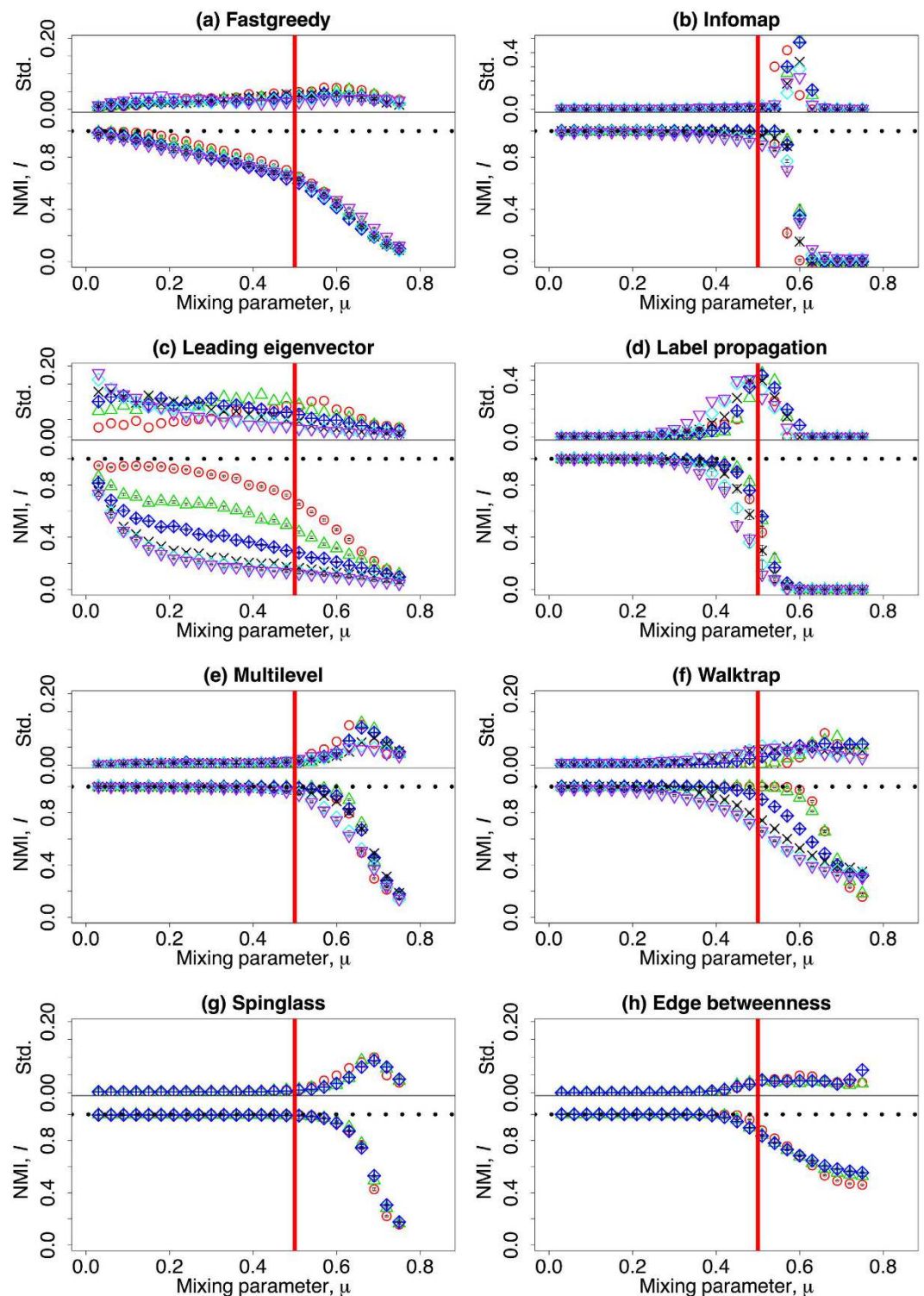
Figure 1. (Lower row) The mean value of normalised mutual information depending on the mixing parameter μ. (upper row) The standard deviation of the NMI as a function of μ. Different colours refer to different number of nodes: red ($N = 233$), green ($N = 482$), blue ($N = 1000$), black ($N = 3583$), cyan ($N = 8916$), and purple ($N = 22186$). Please notice that the vertical axis on the subfigures might have different scale ranges. The vertical red line corresponds to the strong definition of community, i.e. $\mu = 0.5$. The horizontal black dotted line corresponds to the theoretical maximum, $I = 1$. The other parameters are described in Table 1.

the partitions detected. Besides, the turning point for accuracy is after $\mu = 1/2$. For larger networks ($N > 1000$), Infomap, Multilevel and Walktrap algorithms have relatively better accuracies and smaller standard deviations.

Label propagation algorithm has much larger standard deviations such that its outputs are not stable. Due to the long computing time, Spinglass and Edge betweenness algorithms are too slow to be applied on large networks.

Second, we study how well the community detection algorithms reproduce the number of communities. To do so, we compute the ratio $\overline{C}/C$ as a function of the mixing parameter. $\overline{C}$ is the average number of detected communities delivered by the different algorithms when repeated over 100 different network realisations. C is the average real number of communities provided by the LFR benchmark on the same 100 networks. If $\overline{C}/C = 1$, the community detection algorithms are able to estimate correctly the number of communities. It is important to remark that this parameter has to be analysed together with the normalised mutual information because the distribution of community sizes is very heterogeneous. With respect to the networks generated by the LFR model, for small network sizes the real number of communities is stable for all values of μ, while for larger network sizes ($N > 1000$), C grows up to $\mu \gtrsim 0.2$ and then it saturates.

The results for the ratio $\overline{C}/C$ as a function of the mixing parameter are shown in Fig. 2 on a *log-linear* scale for all the panels. The Fastgreedy algorithm constantly underestimates the number of communities, and the results worsen with increasing network size and μ (Panel (a), Fig. 2). For $\mu \lesssim 0.55$, the Infomap algorithm delivers the correct number of communities of small networks ($N \lesssim 1000$), and overestimates it for larger ones. For $\mu \gtrsim 0.55$, this algorithm fails to detect any community at all for small networks and all nodes are partitioned into a single community (Panel (b), Fig. 2). The leading eigenvector algorithm slightly overestimates the number of communities of small networks and the prediction worsens with increasing μ. Moreover, it underestimates the number of communities in large networks and even the behaviour do not change monotonically with μ (Panel (c), Fig. 2). The Label propagation algorithm is able to deliver the correct number of communities with small values of μ regardless of the network size. However, in the range $0.3 \lesssim \mu \lesssim 0.6$, it underestimates the number of communities and the prediction worsens with increasing network size and μ. For $\mu \gtrsim 0.6$, this algorithm fails to detect any community and all nodes are placed into the same community (Panel (d), Fig. 2). It is apparent that the Mutilevel algorithm constantly underestimates the number of communities and such behaviour worsens with increasing network size and μ (Panel (e), Fig. 2). In Fig. 2, Panel (f), for $\mu \lesssim 0.4$, the Walktrap algorithm delivers the correct number of communities regardless of network sizes, although the change of behaviour at which the prediction is correct depends on system size. For $\mu \gtrsim 0.4$, this algorithm behaves differently depending on network size: it slightly underestimates the number of communities of small networks and significantly overestimates it for large ones. For $\mu \lesssim 0.6$, the Spinglass algorithm constantly overestimates the number of communities, and its prediction worsens with network size. When $\mu \gtrsim 0.6$, it fails and tends to put nodes into a few giant communities (Panel (g), Fig. 2). The Edge betweenness algorithm is able to deliver the correct number of communities for $\mu \lesssim 0.4$ regardless of network size. It overestimates C for $\mu \gtrsim 0.4$ and the accuracy of the prediction worsens with increasing network size (Panel (h), Fig. 2). Overall, for $\mu \lesssim 1/2$, Infomap, Leading eigenvector, Multilevel, Spinglass, and Edge betweenness algorithms are able to deliver a reasonable estimator of the number of communities for small networks, while the number of communities obtained by Label propagation and Walktrap algorithms are relatively close to the real value regardless of network size. For $\mu \gtrsim 1/2$, all the algorithms are much worse at detecting the correct number of communities, and among all the algorithms, Multilevel, Walktrap, and Spinglass algorithms have better outputs when the network sizes are small.

Third, we turn to the real computing time of the algorithms. This measure is usually represented in theoretical estimations as a function of the number of nodes and edges. However, the real computing time may be also affected by the structure of the network. Given the number of nodes and a fixed average degree, we illustrate the computing time as a function of the mixing parameter. The results are shown in Fig. 3 on *log-linear* scale. Each panel presents the computing time of a given community detection algorithm and it is subdivided in two plots: the lower one depicts the average computing time, while the upper sub-panel contains the standard deviation of the computing time when repeated over 100 different network realisations. Some algorithms barely depend on the mixing parameter. This is not the case for Multilevel, Spinglass, and Edge betweenness algorithms (Panel (e,g,h), Fig. 3). There is a slight dependency for Infomap algorithm that cannot be disregarded (Panel (b), Fig. 3). The decrease of computing time for Infomap, Leading eigenvector, and Label propagation algorithms (Panel (b–d), Fig. 3) are accompanied with the significant worsening of NMI and $\overline{C}/C$ in Figs 1 and 2. Among all the algorithms, Label propagation and Multilevel algorithms are much faster than the others (Panel (d,e), Fig. 3), while Spinglass and Edge betweenness are the slowest ones (Panel (g,h), Fig. 3).

The observed mixing parameter. Unlike the number of nodes in a network, the exact value of the mixing parameter of a graph is unobservable if ground truth is unavailable for the community assignment of nodes. In this section, we study the mixing parameter delivered by the community detection algorithms $\overline{\mu}$ as a function of the mixing parameter μ (see Eq. 1). The results of the different algorithms are shown in the different panels of Fig. 4. Each panel is subdivided in two plots: the lower has the average computed value of $\overline{\mu}$, while the upper sub-panel contains the standard deviation of the measures when repeated over 100 different network realisations. All algorithms have a linear (identity) relationship between $\overline{\mu}$ and μ except for the Leading eigenvector algorithm, which overshoots the results (Panel (c), Fig. 4). Most of the algorithms display a turning point where the estimation of $\overline{\mu}$ breaks down. For the Fastgreedy, Multilevel, Walktrap, Spinglass, and Edge betweenness algorithms, $\overline{\mu}$ changes in a smooth fashion (Panel (a,e–h), Fig. 4). For the Infomap and Label propagation algorithms, the estimated mixing parameter $\overline{\mu}$ has a steep change at around $\mu \cong 0.55$ and $\mu \cong 0.5$, separately (Panel (b,d), Fig. 4).

Overall, the mixing parameter obtained by the algorithms $\overline{\mu}$ fits well with the real mixing parameter at small value of μ, but it differs from the real value with increasing μ. For certain algorithms, the estimation fails completely for larger values of μ (Infomap, Label propagation), and for the others it is either overestimated (Edge
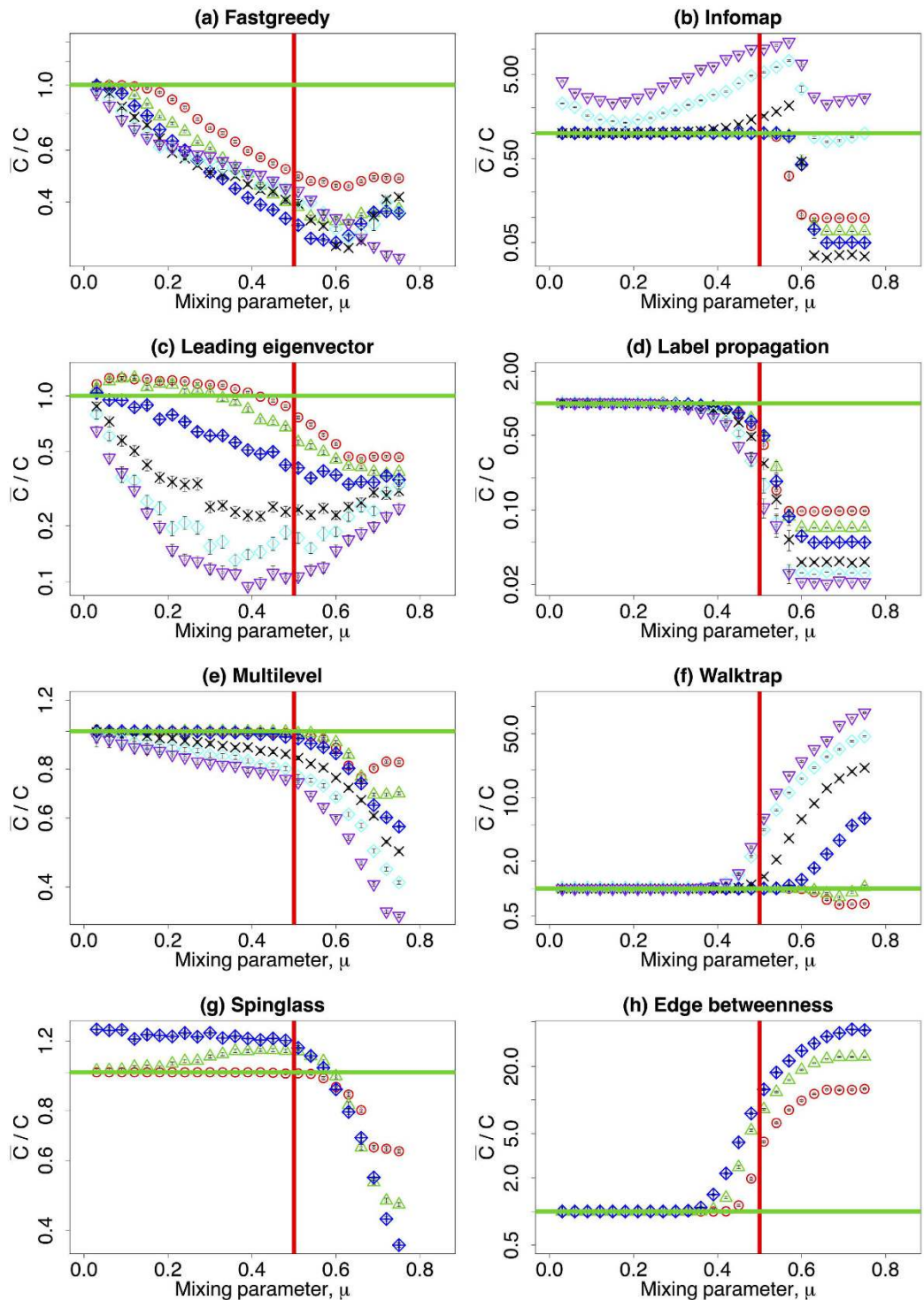
Figure 2. The mean value of the estimated number of communities delivered by different algorithms over the real number of communities given by the LFR benchmark, i.e., $\overline{C}/C$, dependent on the mixing parameter μ on a *log-linear* scale. Different colours refer to different number of nodes: red ($N=233$), green ($N=482$), blue ($N=1000$), black ($N=3583$), cyan ($N=8916$), and purple ($N=22186$). Please notice that the vertical axis might have different scale ranges. The vertical red line corresponds to the strong definition of community where $\mu=0.5$ and the horizontal green line represents the case that $\overline{C}=C$. The other parameters are described in Table 1.

betweenness) or slightly underestimated (Fastgreedy, Walktrap, Spinglass). Remarkably, in the Multilevel algorithm, the estimation is very accurate for values as large as $\mu=0.75$ for all network sizes analysed.

The role of network size. So far we have only discussed the role of the mixing parameter μ to the accuracy and the computing time of community detection algorithms. Now, as an important ingredient, we consider the
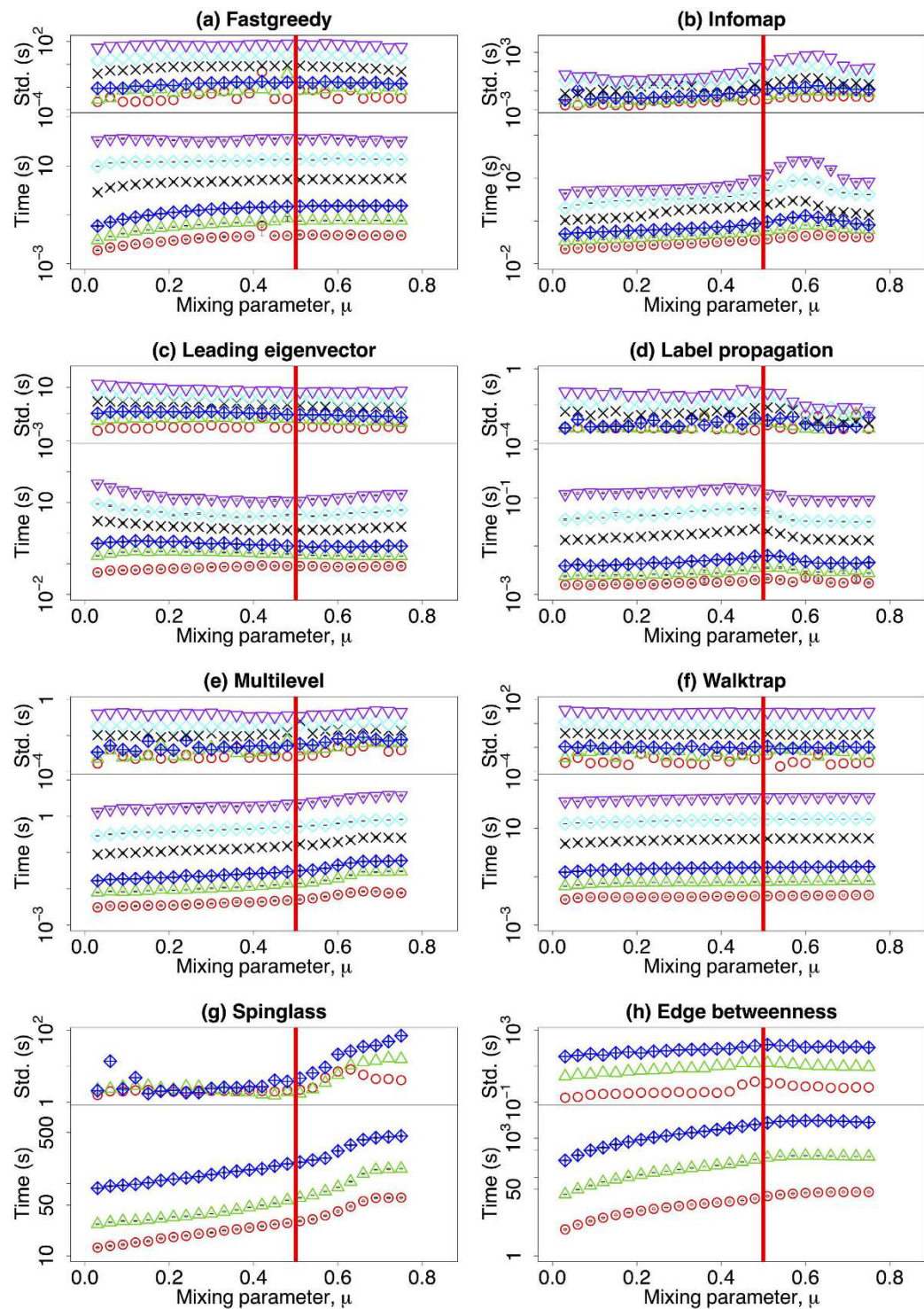
Figure 3. (Lower row) The mean value of the computing time of the community detection algorithms (in seconds) dependent on the mixing parameter μ on a *log-linear* scale. (upper row) The standard deviation of the measures on a *log-linear* scale. Different colours refer to different number of nodes: red ($N = 233$), green ($N = 482$), blue ($N = 1000$), black ($N = 3583$), cyan ($N = 8916$), and purple ($N = 22186$). Please notice that the vertical axis might have different scale ranges. The vertical red line corresponds to the strong definition of community where $\mu = 0.5$. The other parameters are described in Table 1.

effect of network size. In our definition of the benchmark graphs, with a fixed average degree, network size can be represented as the number of nodes in the network. The results are shown in Fig. 5 on a *linear-log* scale. Each of them presents the accuracy of a given community detection algorithms and is subdivided in two plots: one for the
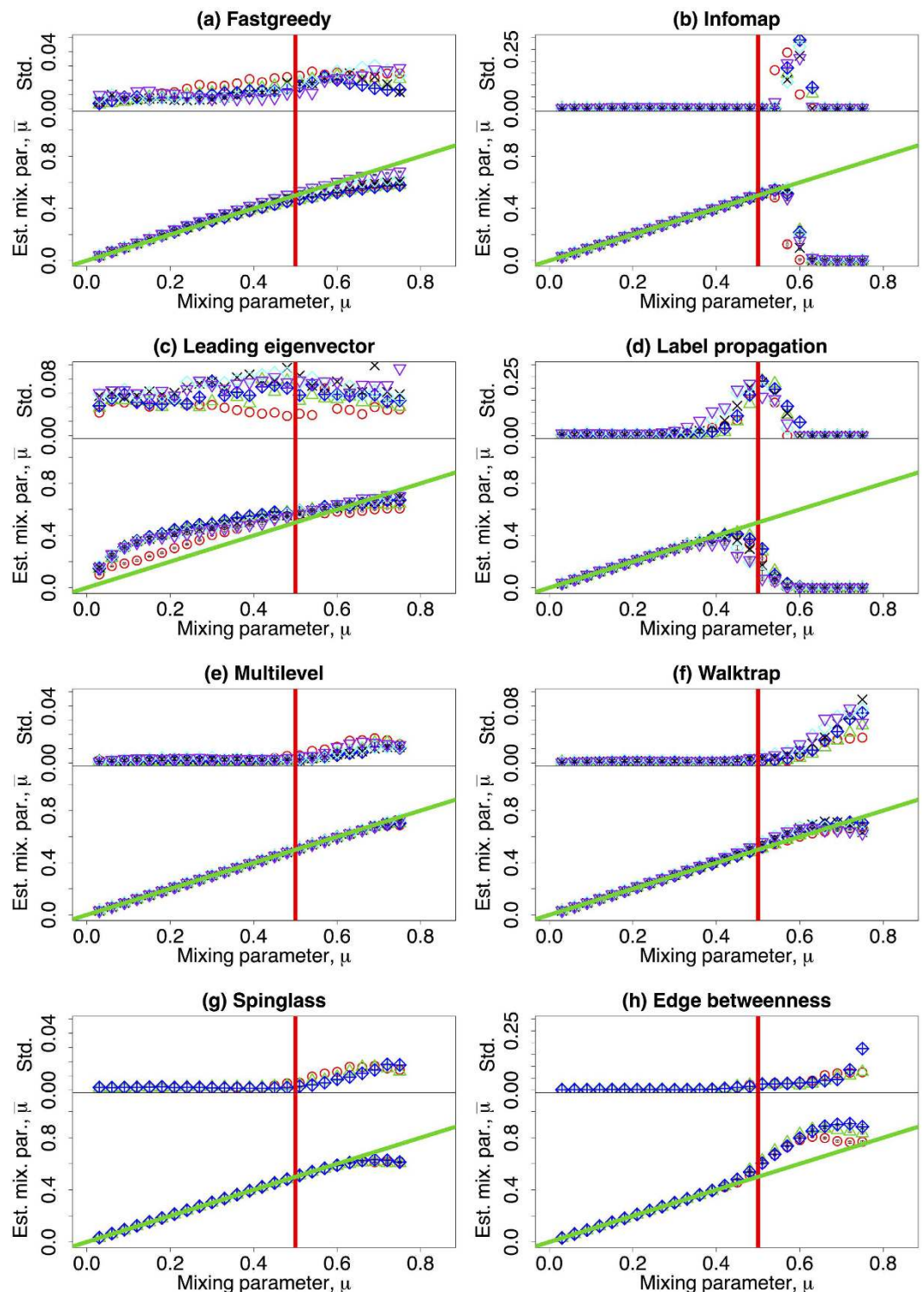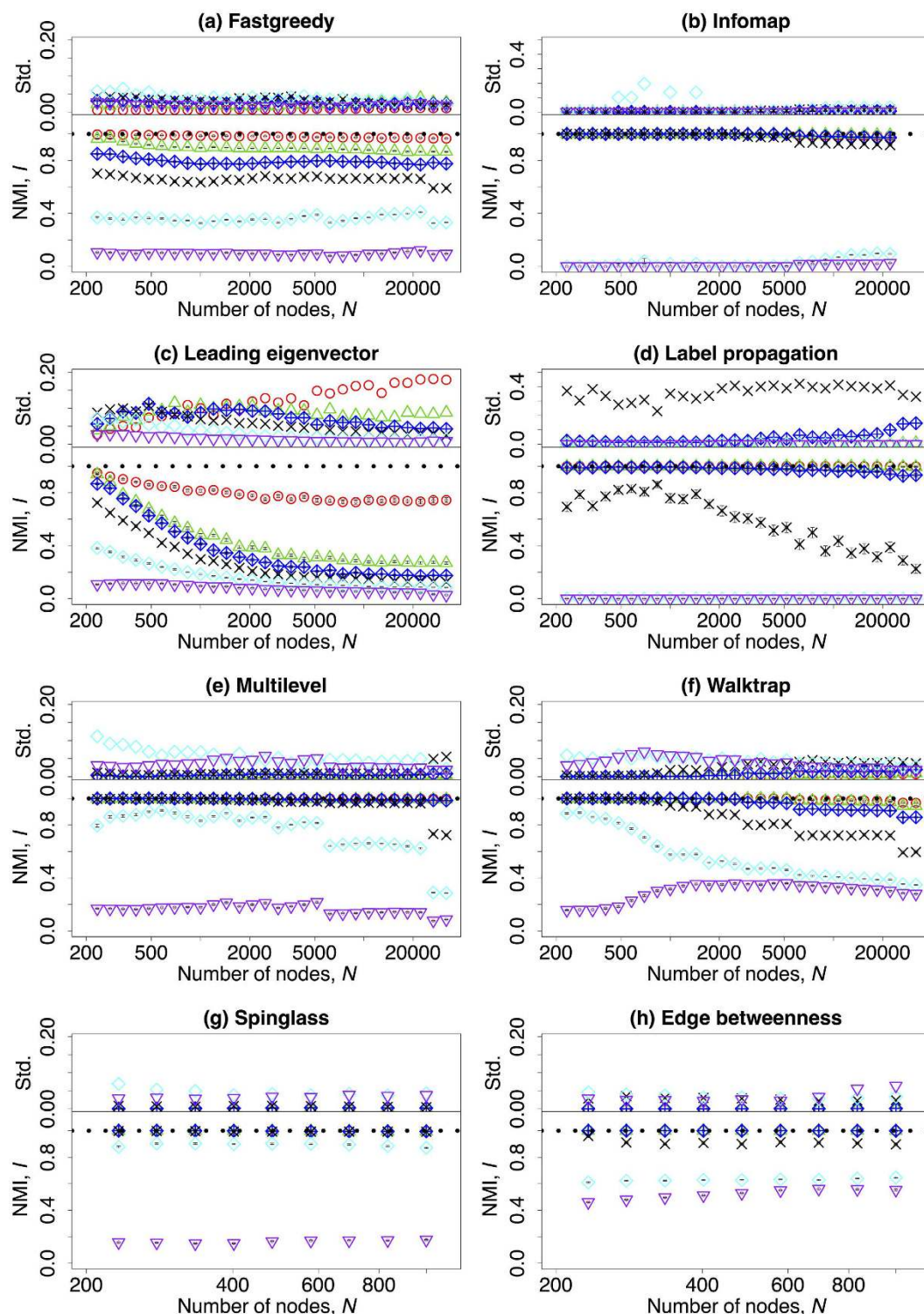
Figure 4. (Lower row) The mean value of the mixing parameter estimated by the community detection algorithms $\overline{\mu}$ dependent on the mixing parameter μ. (upper row) The standard deviation of $\overline{\mu}$ dependent on μ. Different colours refer to different number of nodes: red ($N = 233$), green ($N = 482$), blue ($N = 1000$), black ($N = 3583$), cyan ($N = 8916$), and purple ($N = 22186$). Please notice that the vertical axis on the subfigures might have different scale ranges. The vertical red line corresponds to the strong definition of community where $\mu = 0.5$. The green line $y = x$ corresponds to the case which $\overline{\mu} = \mu$. The other parameters are described in Table 1.

computed value of NMI and the upped sub-panel contains the standard deviation of the measures when repeated over 100 different network realisations. Most of the algorithms can well uncover the communities when $\mu \lesssim 0.2$. In this case, the detecting abilities of Fastgreedy, Infomap, Label propagation, Multilevel, Walktrap, Spinglass and

Figure 5. (Lower row) The mean value of normalised mutual information dependent on the number of nodes N in the benchmark graphs on a *linear-log* scale. (upper row) The standard deviation of the normalised mutual information dependent on N on a *linear-log* scale. Different colours refer to different values of the mixing parameter: red ($\mu = 0.03$), green ($\mu = 0.18$), blue ($\mu = 0.33$), black ($\mu = 0.48$), cyan ($\mu = 0.63$), and purple ($\mu = 0.75$). Please notice that the vertical axis on the subfigures might have different scale ranges. The horizontal black dotted line corresponds to $I = 1$. Due to the computing speed, Spinglass and Edge betweenness algorithms have been tested only on networks with $N \leq 1000$, and Infomap algorithm has been tested on networks with $N \leq 22186$. The other parameters are described in Table 1.

Edge betweenness algorithms are independent of network size (Panel (a,b,d–h), Fig. 5). For Leading eigenvector, the accuracies decrease smoothly with network size (Panel (c), Fig. 5). For very large $\mu \gtrsim 0.75$, most of the algorithms fail to detect the community structure except for the Walktrap and Edge betweenness algorithms and the accuracy barely depends on network size. In the intermediate region of μ, NMI is usually decreasing with network size and μ.

Finally, we present the computing time as a function of the network size. The results are represented in Fig. 6 on a *log-log* scale. Each panel presents the computing time of a given community detection algorithms and is subdivided in two plots: one for the measured value of computing time in second and the upped sub-panel contains the standard deviation of the measures when repeated over different network realisations. In the *log-log* scale, there is a significant linear correlation between the computing time and the network size. To further compare the computing speed of every algorithm, we have fitted the curves according to the exponential function $T \propto N^\alpha$. The fitted α together with the corresponding adjusted R-squared values are listed in Table 2. Only algorithms with small α can be applied to large networks. Overall, Label propagation algorithm is the method that scales best on network size; at the same time, Leading eigenvector, and Multilevel algorithms also have reasonable computation speeds on large networks. Fastgreedy, Infomap, Walktrap, and Spinglass algorithms scale much worse than the previous ones, and Edge betweenness algorithm is only suitable for small networks (with an almost cubic relation between network size and computing time).

Discussion

Traditionally, the aim of community detection in graphs has been to identify the modules by only using the information encoded in the graph topology[4]. In this study we have performed a comparative analysis of the accuracy and computing time of eight different community detection algorithms available in the "igraph" package. Each algorithm has been tested on a set of LFR benchmark graphs[5,13]. The size of the benchmark graphs varies from approximately 200 to 32,000 nodes. With a fixed average degree, we have changed the structure of networks by using different values of the mixing parameter μ.

In this study, the limited network sizes considered here pose no challenge for modern day computers in terms of Random-Access Memory (RAM). Therefore, the memory consumption is not analysed here. However, it is worth mentioning that the maximal memory consumption could be crucial for larger scale networks: if one algorithm is implemented in a way that it needs more memory for the optimal calculation, then it can easily happen that the process slows down for large networks due to low available RAM, or it switches to a suboptimal implementation, which needs less memory. A previous study showed[24] that (theoretically) many community detection methods have minimum memory consumption needs that scale linearly with the size of the graph $\mathcal{O}(2m + 2n)$, where m is the number of edges and n is the number of nodes. In practice, many of them need at least $\mathcal{O}(2m + 3n)$ in case of unweighted undirected graphs and when the Yale sparse matrix format is used[24].

Our results indicate that by taking both accuracy and computing time into account, the Multilevel algorithm, which was proposed by Blondel *et al.*[25], outperforms all the other algorithms on the set of benchmarks we have examined (although the modularity-based methods are known to suffer from the resolution limit of modularity[26]). We can further apply the results in three aspects: First, since the computing time is not relevant for small networks, one should choose algorithms based their accuracies. Among all the algorithms, Infomap, Label propagation, Multilevel, Walktrap, Spinglass, and Edge betweenness algorithms are able to successfully uncover the structure of small networks when the mixing parameter μ is small. With increasing value of μ, Infomap, Label propagation, and Edge betweenness algorithms' accuracies drop for smaller values of μ than Multilevel, Walktrap, and Spinglass algorithms. Second, for large networks, one should first choose algorithms which are able to detect the organisation of nodes in a reasonable time. In this sense, Infomap, Label propagation, Multilevel, and Walktrap algorithms are the *a priori* choices. After that, by taking the accuracy into account, Multilevel is superior to the other algorithms as it displays a performance drop for a larger value of the mixing parameter μ. Importantly, the exact value of the mixing parameter of a graph is usually unobservable. To get a rough idea about the value of μ, one may employ either the Spinglass or the Multilevel algorithm. Limited by the computing time required, Spinglass algorithm cannot be applied on large networks.

Based on the previous results, and taking into account both factors, accuracy and computing time, it is possible to suggest under which situations to use each algorithm depending sorely on topological properties of the network under study. Our recommendations for the use of community detection algorithms are summarised in Fig. 7. In the first region, $\mu \lesssim 0.5$ and the network size is small, $N \lesssim 1000$. There, most of the communities detection algorithms tested give accurate results (and the computing time is affordable): Infomap, Label propagation, Multilevel, Walktrap, Spinglass, and Edge betweenness can all be used in a trustworthy fashion. A second region has a relatively larger value of μ ($0.5 \lesssim \mu \lesssim 0.6$), and equally small sizes of network $N \lesssim 1000$. There, it is possible to use Multilevel, Walktrap, and Spinglass algorithms. A third region encompasses again smaller values of mixing parameter ($\mu \lesssim 0.5$) but an intermediate number of nodes ($1000 \lesssim N \lesssim 6000$). In this region, the best choices are Infomap, label propagation, Multilevel, and Walktrap algorithms. With increasing number of nodes in the networks ($6000 \lesssim N \lesssim 32000$), Infomap and Multilevel algorithm are very likely to provide the wrong number of communities and therefore they are no longer suitable in the fourth region. The last region has the highest requirement for the community detection algorithms. None of the algorithms performs very well in this region but the Multilevel algorithm outperforms all the others.

Besides, we illustrate the suggestion for the adaptive use of the methods for community detection process in a simplified flow diagram (see Fig. 8). With any given network, one should first employ either Spinglass algorithm or Multilevel algorithm in order to obtain an estimate of the value of the mixing parameter μ. Notice that the former one can only be used for small networks ($N \lesssim 1000$) due to the prohibitive computing time for larger network sizes. Second, one can choose a suitable method according to the values of N and μ to conduct the
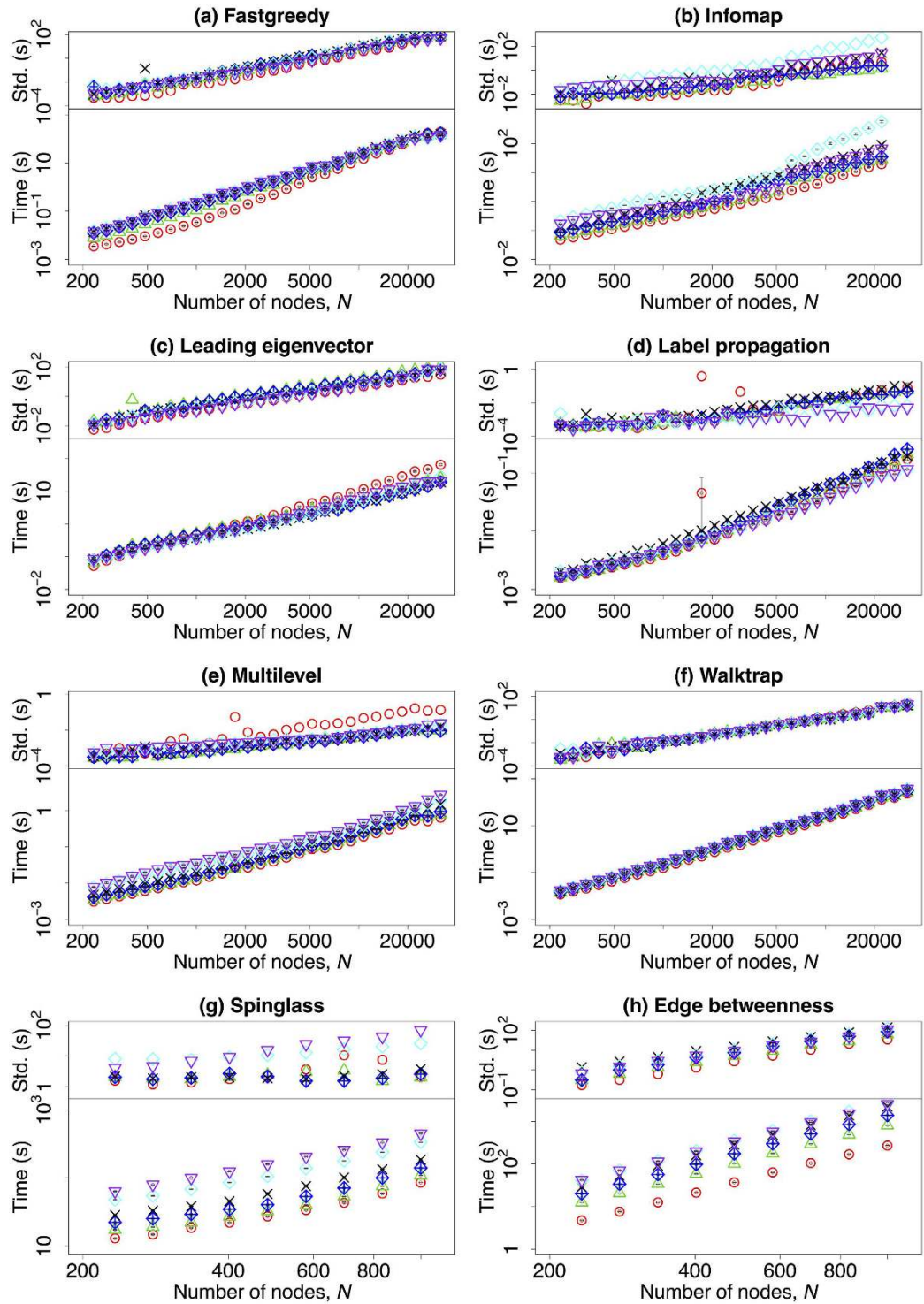
Figure 6. (Lower row) The mean value of the computing time of the community detection algorithms (in seconds) dependent on the number of nodes in the benchmark graphs on a *log-log* scale. (upper row) The standard deviation of the computing time on a *log-log* scale. Different colours refer to different values of the mixing parameter: red ($\mu = 0.03$), green ($\mu = 0.18$), blue ($\mu = 0.33$), black ($\mu = 0.48$), cyan ($\mu = 0.63$), and purple ($\mu = 0.75$). Please notice that the vertical axis might have different scale ranges. Due to the computing speed, Spinglass and Edge betweenness algorithms have been tested only on networks with $N \leq 1000$, and Infomap algorithm has been tested on networks with $N \leq 22186$. The other parameters are described in Table 1.

community detection such that both the accuracy and the computing time are acceptable. Third, as we have already shown, in certain situations, there might exist large standard deviations of NMI, i.e., the community detection algorithms are not stable and therefore not reliable. Thus, the value of $\overline{\mu}$ must be recalculated to get an

	Fastgreedy	Infomap	Leading eigenvector	Label propagation
α	2.048 [0.006]	1.421 [0.009]	1.123 [0.005]	0.959 [0.005]
R^2	0.956	0.933	0.951	0.947
	Multilevel	**Walktrap**	**Spinglass**	**Edge betweenness**
α	1.126 [0.003]	2.04 [0.002]	1.282 [0.013] ()	2.915 [0.005]
R^2	0.957	0.962	0.867	0.884

Table 2. Indexes of the exponential function $T \propto N^\alpha$ with the corresponding adjusted R-squared values. The standard errors are listed in brackets. All the results are statistically significant at the significance level of 0.05. Spinglass and Edge betweenness algorithms have been tested only on small networks with $N \leq 1000$, there might be some biases in the indexes of these two methods.
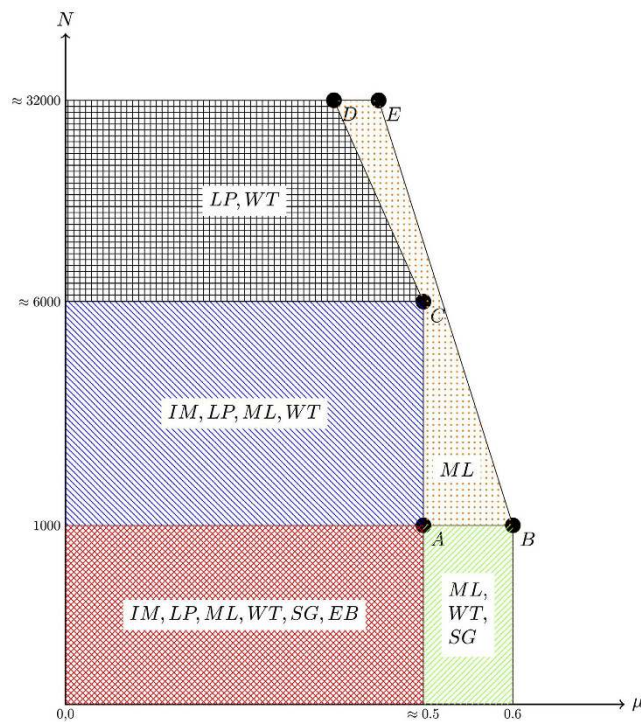

Figure 7. Recommendation for the choice of adaptable community detection algorithms. The x-axis is the mixing parameter μ and the y-axis is the number of nodes N. The y-axis is on a *log* scale for better visualisation. The coordinates of certain important points are: $A(0.48, 1000)$, $B(0.6, 1000)$, $C(0.48, 6192)$, $D(0.36, 31948)$, and $E(0.42, 31948)$. In different regions we would like to recommend different algorithms, which are represented by different abbreviations: *IM* is the Infomap algorithm, *LP* is the Label propagation algorithm, *ML* is the Multilevel algorithm, *WT* is the Walktrap algorithm, *SG* is the Spinglass algorithm, and *EB* represents the Edge betweenness algorithm.

idea of the repeatability of the results and confirm its validity. In some situations, one might need to repeat the detection processes several times or switch to another algorithm to ensure the validity of the community detection results.

Our suggestions have to be applied in conjunction with the concomitant research questions. As a pure application of the recommendations could bias the results. Once a researcher has decided to use a specific community detection algorithm, it is of crucial importance for her to keep in mind the limitations and the expected validity of the output of the community detection algorithm chosen. It is noteworthy that metadata would be helpful for evaluating network community detection methods and can be used to improve the analysis and understanding of network structure[19,27]. In real-world networks where metadata is available, researchers should also take into account the research question, the properties of the network, the interpretation and meaning of the communities while choosing the community detection algorithms. Different research questions together with the metadata might lead to different definitions of community, and further change the ground truth of the network.

Compared to previous works on benchmarking community detection algorithms, our study has many obvious advantages: First, we have considered networks which contain a wide spectrum of number of nodes and mixing parameters. Second, the algorithms we have tested are integrated in a cross-platform package which has been widely used in academic research in network science and related fields. Third, we have used the LFR benchmark
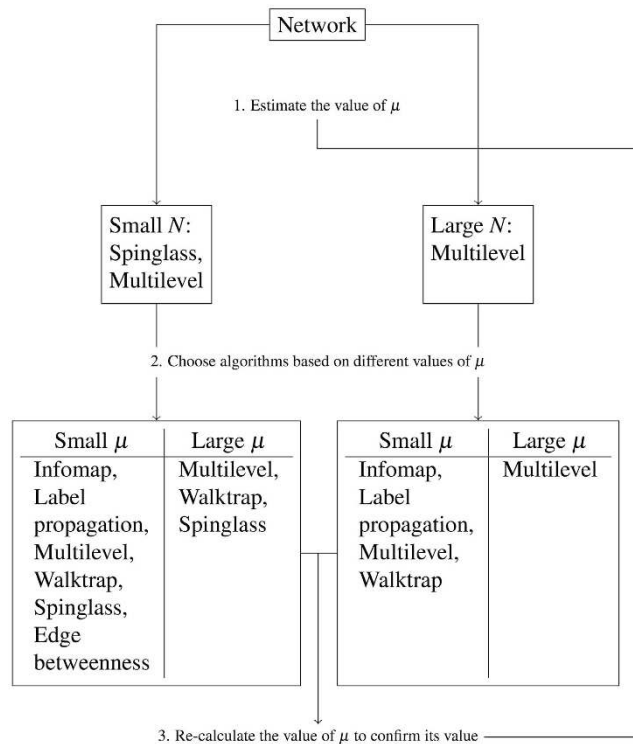
Figure 8. Suggestion for the community detection process. Small networks are those with number of nodes less than 1000, and small μ corresponds to $\mu \lesssim 0.5$. To be noticed that in the case that $N \geq 1000$ and $\mu \lesssim 0.5$, Infomap and Multilevel algorithms are no longer suitable choices if $N \geq 6000$.

graphs which have shown more realistic properties than the earlier computer-generated networks such as the GN benchmark.

There are also some limitations in our work: Although the LFR benchmark has generalised the previous GN benchmark by introducing power-law distributions of degree and community size, more realistic properties are still needed. We have mainly focused on testing the effects of the mixing parameter and the number of nodes. Other properties, such as the average degree, the degree distribution exponent, and the community distribution exponent may also play a role in the comparison of algorithms.

In the end, we stress that detecting the community structure of networks is an important issue in network science. For "igraph" package users, we have provided a guideline on choosing the suitable community detection methods. However, based on our results, existing community detection algorithms still need to be improved to better uncover the ground truth of networks.

Methods

In this section, we first describe in detail the procedure to obtain the benchmark networks used, then enumerate the community detection algorithms employed.

When comparing community detection algorithms, we can use either real or artificial network whose community structure is already known, which is usually termed as ground truth. Among the former, the celebrated Zachary's karate club[28] or the network of American college football teams[3] have been extensively used. Among the latter, the ones used more pervasively are the GN[3] and LFR[13] benchmarks. However, obtaining real networks to which a ground truth can be associated is not only difficult, but also costly in economic terms and time. Due to the complexity of data collection and costs, real world benchmarks usually consist of small-sized networks. Further, since it is not possible to control all the different features of a real network (e.g. average degree, degree distribution, community sizes, etc.), the algorithms can only be tested – if resorting in this kind of graphs – on very specific cases with a limited set of features. In addition, the communities of real world networks are not always defined objectively or, in the best case, they rarely have a unique community decomposition. On the other hand, artificially generated networks can overcome most of these limitations. Given an arbitrary set of meso- or macroscopic properties, it is possible to generate randomly an ensemble of networks that respect them, in what is usually called generative models. However, as one of the most popular generative models, GN benchmark suffers from the fact that it does not show a realistic topology of the real network[5,29] and it has very small network size. A recent strand of the literature on benchmark graphs tried to improve the quality of artificial networks by defining more realistic generative models: Lancichinetti *et al.* extended the GN benchmark by introducing power law degree and community size distributions[5]. Bagrow had employed the Barabási-Albert model[9] rather than the configuration model[30] to build up the benchmark graph[31]. Orman and Labatut proposed to use evolutionary preferential attachment model[32] for more realistic properties[33].

The first step to generate the LFR benchmark graph is to construct a network composed of N nodes, with average degree $\hat{k}$, maximum degree k_{max} and a power-law degree distribution with exponent α by using the configuration model. Once this step is finished, each node has a defined total degree. Then, given a power-law distribution of community sizes with exponent β, a set of community sizes is drawn (between arbitrarily chosen minimum and maximum values of community sizes that act as additional parameters). Nodes are then sequentially assigned to these communities. The mixing parameter μ, which represents the fraction of edges a node has with nodes belonging to other communities with respect to its total degree, is the most relevant value in terms of the community structure. To conclude the generative algorithm, edges are rewired in order to fit the mixing parameter, while preserving the degree sequence. This is achieved keeping fixed total degree of a node, the value of external degree is modified so that the ratio of external degree over the total degree is close to the defined mixing parameter. The LFR model was initially proposed to generate undirected unweighted networks with mutually exclusive communities, and was extended to generate weighted and/or directed networks, with or without overlapping communities. In this study, we focus on the undirected unweighted networks with non-overlapping communities since most of the existing community detection algorithms are designed for this type of networks. The parameter values used in our computer-generated graphs are indicated in Table 1.

In this paper, we have evaluated the most widely used, state-of-the-art community detection algorithms on the LFR benchmark graphs. In order to make the results comparable, and reproducible, we use the implementation of these algorithms shipped with the widely used "igraph" software package (Version 0.7.1)[20]. Here is the list of algorithms we have considered. For notation purposes when giving the computational complexity of the algorithms, the networks have N nodes and E edges.

Edge betweenness. This algorithm was introduced by Girvan & Newman[3]. To find which edges in a network exist most frequently between other pairs of nodes, the authors generalised Freeman's betweenness centrality[34] to edges betweenness. The edges connecting communities are then expected to have high edge betweenness. The underlying community structure of the network will be much clear after removing edges with high edge betweenness. For the removal of each edge, the calculation of edge betweenness is $\mathcal{O}(E\,N)$; therefore, this algorithm's time complexity is $\mathcal{O}(E^2 N)$[3].

Fastgreedy. This algorithm was proposed by Clauset et al.[12]. It is a greedy community analysis algorithm that optimises the modularity score. This method starts with a totally non-clustered initial assignment, where each node forms a singleton community, and then computes the expected improvement of modularity for each pair of communities, chooses a community pair that gives the maximum improvement of modularity and merges them into a new community. The above procedure is repeated until no community pairs merge leads to an increase in modularity. For sparse, hierarchical, networks the algorithm runs in $\mathcal{O}(N \log^2(N))$[12].

Infomap. This algorithm was proposed by Rosvall et al.[35,36]. It figures out communities by employing random walks to analyse the information flow through a network[17]. This algorithm starts with encoding the network into modules in a way that maximises the amount of information about the original network. Then it sends the signal to a decoder through a channel with limited capacity. The decoder tries to decode the message and to construct a set of possible candidates for the original graph. The smaller the number of candidates, the more information about the original network has been transferred. This algorithm runs in $\mathcal{O}(E)$[37].

Label propagation. This algorithm was introduced by Raghavan et al.[38]. It assumes that each node in the network is assigned to the same community as the majority of its neighbours. This algorithm starts with initialising a distinct label (community) for each node in the network. Then, the nodes in the network are listed in a random sequential order. Afterwards, through the sequence, each node takes the label of the majority of its neighbours. The above step will stop once each node has the same label as the majority of its neighbours. The computational complexity of label propagation algorithm is $\mathcal{O}(E)$[38].

Leading eigenvector. This algorithm was proposed by Newman[39]. The heart of this algorithm is the spectral optimisation of modularity by using the eigenvalues and eigenvectors of the modularity matrix. First, the leading eigenvector of the modularity matrix is calculated, and then the graph is split into two parts in a way that modularity improvement is maximised based on the leading eigenvector. After that, the modularity contribution is calculated at each step in the subdivision of a network. It stops once the value of the modularity contribution is not positive. Its computational complexity of each graph bipartition is $\mathcal{O}(N(E + N))$, or $\mathcal{O}(N^2)$ on a sparse graph[40].

Multilevel. This algorithm was introduced by Blondel et al.[25]. It is a different greedy approach for optimising the modularity with respect to the Fastgreedy method. This method first assigns a different community to each node of the network, then a node is moved to the community of one of its neighbours with which it achieves the highest positive contribution to modularity. The above step is repeated for all nodes until no further improvement can be achieved. Then each community is considered as a single node on its own and the second step is repeated until there is only a single node left or when the modularity can't be increased in a single step. The computational complexity of the Multilevel algorithm is $\mathcal{O}(N \log N)$[40].

Spinglass. This algorithm was first proposed by Reichardt & Bornholdt[41]. It is based on the Potts model[42]. The basic principle of the method is that edges should connect nodes of the same spin state (community, in the

current context), whereas nodes of different states (belonging to different communities) should be disconnected. Therefore, the aim of this algorithm is to find the ground state of a spin glass model with a Potts Hamiltonian. Simulated annealing[43] has been used to minimise the system's free energy[44]. In a sparse graph, the computational complexity of this algorithm is approximately $\mathcal{O}(N^{3.2})$[45].

Walktrap.   This algorithm was proposed by Pon & Latapy[46]. It is a hierarchical clustering algorithm. The basic idea of this method is that short distance random walks tend to stay in the same community. Starting from a totally non-clustered partition, the distances between all adjacent nodes are computed. Then, two adjacent communities are chosen, they are merged into a new one and the distances between communities are updated. This step is repeated $(N-1)$ times, thus the computational complexity of this algorithm is $\mathcal{O}(E N^2)$. For sparse networks the computational complexity is $\mathcal{O}(N^2 \log(N))$[40].

We have employed virtual machines to implement all the computation. For each network size and for each algorithm, a virtual machine is created using a pre-defined installation that guarantees the same execution environment conditions. The installation is tuned to guarantee that each virtual machine makes use of an entire physical node, and, at the same time, that all physical nodes where the virtual machines will be hosted have the very same hardware specifications. The workload distribution and collection for the results are commanded by a master-slave approach.

References

1. Newman, M. E. The structure and function of complex networks. *SIAM Review* **45,** 167–256 (2003).
2. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. & Hwang, D.-U. Complex networks: Structure and dynamics. *Physics Reports* **424,** 175–308 (2006).
3. Girvan, M. & Newman, M. E. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* **99,** 7821–7826 (2002).
4. Fortunato, S. Community detection in graphs. *Physics Reports* **486,** 75–174 (2010).
5. Lancichinetti, A., Fortunato, S. & Radicchi, F. Benchmark graphs for testing community detection algorithms. *Physical Review E* **78,** 046110 (2008).
6. Lancichinetti, A., Kivelä, M., Saramäki, J. & Fortunato, S. Characterizing the community structure of complex networks. *PloS ONE* **5,** e11976 (2010).
7. Condon, A. & Karp, R. M. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms* **18,** 116–140 (2001).
8. Danon, L., Díaz-Guilera, A., Duch, J. & Arenas, A. Comparing community structure identification. *Journal of Statistical Mechanics*: *Theory and Experiment* **2005,** P09008 (2005).
9. Barabási, A.-L. & Albert, R. Emergence of scaling in random networks. *Science* **286,** 509–512 (1999).
10. Palla, G., Derényi, I., Farkas, I. & Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435,** 814–818 (2005).
11. Guimerà, R., Danon, L., Díaz-Guilera, A., Giralt, F. & Arenas, A. Self-similar community structure in a network of human interactions. *Physical Review E* **68,** 065103 (2003).
12. Clauset, A., Newman, M. E. & Moore, C. Finding community structure in very large networks. *Physical Review E* **70,** 066111 (2004).
13. Lancichinetti, A. & Fortunato, S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E* **80,** 016118 (2009).
14. Orman, G. K. & Labatut, V. A comparison of community detection algorithms on artificial networks. In *Discovery Science* 242–256 (Springer, 2009).
15. Lancichinetti, A. & Fortunato, S. Community detection algorithms: a comparative analysis. *Physical Review E* **80,** 056117 (2009).
16. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. & Parisi, D. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences* **101,** 2658–2663 (2004).
17. Peel, L. Estimating network parameters for selecting community detection algorithms. In *13th Conference on Information Fusion* 1–8 (IEEE, 2010).
18. Hric, D., Darst, R. K. & Fortunato, S. Community detection in networks: Structural communities versus ground truth. *Physical Review E* **90,** 062805 (2014).
19. Yang, J. & Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* **42,** 181–213 (2015).
20. Csardi, G. & Nepusz, T. The igraph software package for complex network research. *InterJournal, Complex Systems* 1695, URL http://igraph.org (2006).
21. Vinh, N. X., Epps, J. & Bailey, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research* **11,** 2837–2854 (2010).
22. Romano, S., Bailey, J., Nguyen, V. & Verspoor, K. Standardized mutual information for clustering comparisons: one step further in adjustment for chance. In *Proceedings of the 31st International Conference on Machine Learning* (ICML-14), 1143–1151 (2014).
23. Zhang, P. Evaluating accuracy of community detection using the relative normalized mutual information. *Journal of Statistical Mechanics*: *Theory and Experiment* **2015,** P11006 (2015).
24. Papadopoulos, S., Kompatsiaris, Y., Vakali, A. & Spyridonos, P. Community detection in social media. *Data Mining and Knowledge Discovery* **24,** 515–554 (2012).
25. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*: *Theory and Experiment* **2008,** P10008 (2008).
26. Fortunato, S. & Barthélemy, M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* **104,** 36–41 (2007).
27. Newman, M. & Clauset, A. Structure and inference in annotated networks. *arXiv preprint arXiv:1507.04001* (2015).
28. Zachary, W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* 452–473 (1977).
29. Danon, L., Díaz-Guilera, A. & Arenas, A. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics*: *Theory and Experiment* **2006,** P11010 (2006).
30. Molloy, M. & Reed, B. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms* **6,** 161–180 (1995).
31. Bagrow, J. P. Evaluating local community methods in networks. *Journal of Statistical Mechanics*: *Theory and Experiment* **2008,** P05001 (2008).
32. Poncela, J., Gómez-Gardeñes, J., Floria, L. M., Sánchez, A. & Moreno, Y. Complex cooperative networks from evolutionary preferential attachment. *PLoS One* **3,** e2449 (2008).

33. Orman, G. K. & Labatut, V. The effect of network realism on community detection algorithms. In *International Conference on Advances in Social Networks Analysis and Mining* 301–305 (IEEE, 2010).
34. Freeman, L. C. Centrality in social networks conceptual clarification. *Social Networks* **1,** 215–239 (1979).
35. Rosvall, M. & Bergstrom, C. T. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences* **104,** 7327–7331 (2007).
36. Rosvall, M., Axelsson, D. & Bergstrom, C. T. The map equation. *The European Physical Journal Special Topics* **178,** 13–23 (2010).
37. Mukherjee, A., Choudhury, M., Peruani, F., Ganguly, N. & Mitra, B. *Dynamics On and Of Complex Networks, Volume 2: Applications to Time-Varying Dynamical Systems* (Springer Science & Business Media, 2013).
38. Raghavan, U. N., Albert, R. & Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* **76,** 036106 (2007).
39. Newman, M. E. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* **74,** 036104 (2006).
40. Xie, J. & Szymanski, B. K. Community detection using a neighborhood strength driven label propagation algorithm. In *Network Science Workshop* 188–195 (IEEE, 2011).
41. Reichardt, J. & Bornholdt, S. Statistical mechanics of community detection. *Physical Review E* **74,** 016110 (2006).
42. Wu, F.-Y. The Potts model. *Reviews of Modern Physics* **54,** 235 (1982).
43. Kirkpatrick, S. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics* **34,** 975–986 (1984).
44. Traag, V. & Bruggeman, J. Community detection in networks with positive and negative links. *Physical Review E* **80,** 036115 (2009).
45. Dahlin, J. & Svenson, P. Ensemble approaches for improving community detection methods. *arXiv:1309.0242* [*physics.soc-ph*] (2013).
46. Pons, P. & Latapy, M. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, 284–293 (Springer, 2005).

Acknowledgements

Author Contributions

Z.Y., R.A. and C.J.T. designed the analysis. Z.Y. and C.J.T. devised the methodology. Z.Y. analysed the data. Z.Y., R.A. and C.J.T. wrote the manuscript.

Additional Information

Supplementary information accompanies this paper at http://www.nature.com/srep

Competing financial interests: The authors declare no competing financial interests.

How to cite this article: Yang, Z. *et al.* A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Sci. Rep.* **6**, 30750; doi: 10.1038/srep30750 (2016).

SCIENTIFIC REP🞮RTS

OPEN

Corrigendum: A Comparative Analysis of Community Detection Algorithms on Artificial Networks

Zhao Yang, René Algesheimer & Claudio J. Tessone

This Article contained errors.

Figure 1f incorrectly showed curves for the values N = 233, 279, 335, 402, 482, and 579.

The correct Figure 1 appears below.

The Results section contained a typographical error under the subheading "The role of network size".

"Most of the algorithms can well uncover the communities when $\mu \gtrless 0.2$".

now reads:

"Most of the algorithms can well uncover the communities when $\mu \lessgtr 0.2$".

The Supplementary Information file originally published with this Article contained errors in the f panels of Supplementary Figures 1, 2, 3, 4, and 5, where these panels incorrectly showed curves for the values N = 233, 279, 335, 402, 482, and 579.

These errors have now been corrected in the PDF and HTML versions of this Article.

Figure 1.