Physica A 392 (2013) 2278–2293

Contents lists available at SciVerse ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Obtaining communities with a fitness growth process

Mariano G. Beiró [a,b,*], Jorge R. Busch [a,b], Sebastian P. Grynberg [a], J. Ignacio Alvarez-Hamelin [a,b]

[a] Facultad de Ingeniería, Universidad de Buenos Aires, Paseo Colón 850, C1063ACV, Buenos Aires, Argentina
[b] INTECIN (CONICET–U.B.A.), Paseo Colón 850, C1063ACV, Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 31 August 2012
Received in revised form 29 November 2012
Available online 17 January 2013

Keywords:
Community detection
Social networks
Complex systems

ABSTRACT

The study of community structure became an important topic of research over the last years. But, while successfully applied in several areas, the concept lacks of a general and precise notion. Facts like the hierarchical structure and heterogeneity of complex networks make it difficult to unify the idea of community and its evaluation. The global functional known as modularity is probably the most used technique in this area. Nevertheless, its limits have been deeply studied. Local techniques as the one by Lancichinetti et al. (2009) [1] arose as an answer to the resolution limit and degeneracies that modularity has.

Here we propose a unique growth process for a fitness function based on the algorithm by Lancichinetti et al. (2009) [1]. The process is local and finds a community partition that covers the whole network, updating the scale parameter dynamically. We test the quality of our results by using a set of benchmarks of both heterogeneous and homogeneous graphs. We discuss alternative measures for evaluating the community structure and, in the light of them, infer possible explanations for the better performance of local methods compared to global ones in these cases.

1. Introduction

In the last years, community detection became one of the top research topics in the area of Complex Networks. Due in part to the explosion of social networking, and also to its application in diverse areas as ecology and computational biology, an interest arose in defining, detecting, evaluating and comparing community structures. For a thorough, yet not exhaustive, reference of its applications, see the survey by Fortunato [2].

The early research by Newman started with use of *betweenness* to divide the network into modules [3], and the definition of *modularity* to evaluate communities [4]. Then he proposed using the modularity as a functional to be maximized [5]. Different optimization techniques were developed, of which we recall the algorithm by Guimerà et al. based on simulated annealing [6] for its good results, and the Louvain algorithm [7] for its fast convergence within large networks.

Later works questioned the global optimization methods based on modularity, for being prone to resolution limits (Fortunato [8]) and extreme degeneracies (Good et al. [9]). Local techniques were proposed, as the Clique Percolation Method (CPM) (Palla et al. [10]), and the algorithm in Lancichinetti et al. [1], based on a fitness function. Both of them find overlapping communities, and in the latter, the notion of *natural community* arose. The *natural community* of a vertex is a locally-computed set, and its size depends on a resolution parameter α.

It has also been observed that the resolution limits for modularity found in Fortunato et al. [8] are particularly common in heterogeneous graphs with heavy-tailed community sizes and vertex degree distributions (see Ref. [2], Section VI.C).

* Corresponding author at: Facultad de Ingeniería, Universidad de Buenos Aires, Paseo Colón 850, C1063ACV, Buenos Aires, Argentina. Tel.: +54 1143430891x258.
E-mail addresses: mbeiro@fi.uba.ar (M.G. Beiró), jbusch@fi.uba.ar (J.R. Busch), sebgryn@fi.uba.ar (S.P. Grynberg), ignacio.alvarez-hamelin@cnet.fi.uba.ar (J.I. Alvarez-Hamelin).

0378-4371/$ – see front matter
doi:10.1016/j.physa.2013.01.031

In these graphs, small communities will often be masked into larger ones by modularity maximization techniques when they are interconnected just by a few links.

In this work we detect communities based on a *fitness function* analogous to the one defined by Lancichinetti et al. in Ref. [1]. After analyzing the role of the resolution parameter α in these functions, we propose a *uniform fitness growth process* which scans the whole graph and whose resolution parameter is updated dynamically. Then, we extract a community partition from the output of this process. The details of our method are described in Sections 2 and 3, and the algorithmic complexity is discussed in Section 4.

In Section 5 we show the results. We use a benchmark developed in Ref. [11] to build a dataset of heterogeneous and homogeneous networks. We observe an important improvement using our fitness growth process when compared to the global modularity maximization techniques, which suggests that local methods may outperform global ones in these cases. In order to discuss this conjecture, we analyze the consequences of the resolution limits and give a possible explanation to the differences in performance between the two methods.

As a measure for comparing community structures, Danon et al. [12] proposed using the *normalized mutual information*. We shall use it in order to make comparisons with global methods and with community structures known *a priori*. We also apply the algorithm to real networks and show the results. Finally, we discuss the robustness (repeatability of the results) of our process.

2. Our method

Given a graph $G = (V, E)$, the work by Lancichinetti et al. [1] define a process based on a fitness function with a resolution parameter α such that, given a set $C \subset V$:

$$f(C) = \frac{k_{in}}{(k_{in} + k_{out})^{\alpha}}$$

where k_{in} is the number of edges that join vertices in C, and k_{out} is the number of edges that join some vertex in C to some vertex not in C. This notion of community is related to the one proposed by Radicchi et al. [13]. In fact, a choice of $\alpha = 1$ corresponds to the definition of weak community introduced in that paper.

The process starts with a community made up by the seed vertex v and proceeds by stages, where in each stage the steps are: (1) select a vertex whose addition increments the fitness function, and add it to the present community; (2) delete from the present community all the vertices whose deletion increments the fitness function. The algorithm stops when, being in stage 1, it finds no vertex to add. Step 2 is time consuming, and usually very few vertices are deleted, but it is necessary due to the local, vertex-by-vertex nature of the analysis. The authors called the final result of the algorithm the *natural community of* v. The resolution parameter α is related to the natural community size.

In order to obtain a covering by overlapping communities, they select a vertex at random, obtain its natural community, select a vertex not yet covered at random, obtain its natural community, and so on until they cover the whole graph.

In this process, the resolution parameter α of the fitness function is kept fixed. The authors perform an analysis in order to find the significant values of α.

Our contribution extends that work to define a *uniform growth process*. This process covers the whole graph by making a course throughout its communities. We modify the *fitness function* $f(C)$ and analyze the role of α in the termination criteria for the process. Then we propose an algorithm for increasing the fitness function monotonically while traversing the graph, dynamically updating the parameter. Finally, a *cutting technique* divides the sequence of vertices obtained by the process, in order to get a partition into communities.

2.1. Previous definitions

We shall deal with simple undirected graphs $G = (V, E)$, with $n = |V|$ vertices and m edges (here $|\cdot|$ denotes the cardinal of a set). To avoid unnecessary details, we assume that $E \subset V \times V$ is such that $(v, w) \in E$ implies that $(w, v) \in E$.

We set $\delta_E(v, w) = 1$ if $(v, w) \in E$, $\delta_E(v, w) = 0$ in the other case. We then have the following expression for the degree of a vertex v

$$\deg(v) = \sum_{w \in V} \delta_E(v, w).$$

Thus, $|E| = \sum_{w \in V} \deg(w) = 2m$. We shall use two measures, m_V and m_E, the first one on V and the second one on $V \times V$. Given $C \subset V$,

$$m_V(C) = \sum_{v \in C} \deg(v)/|E|$$

is the normalized sum of the degrees of the vertices in C. Given $D \subset V \times V$,

$$m_E(D) = \sum_{(v,w) \in D} \delta_E(v, w)/|E|.$$

Notice that when $C_1, C_2 \subset V$ are mutually disjoint, $m_E(C_1 \times C_2)$ is the normalized cut between C_1 and C_2, i.e., the number of pairs $(v, w) \in E$ such that $v \in C_1$ and $w \in C_2$. Notice also that m_V is the marginal measure of m_E, and that these measures are in fact probabilities.

Let $C \subset V$, and $v \in V$. We denote

$$ki_C(v) = \sum_{w \in C} \delta_E(v, w)$$

and

$$ko_C(v) = \sum_{w \notin C} \delta_E(v, w).$$

Thus $ki_C(v)$ is the number of vertices in C joined to v, and $ko_C(v)$ is the number of vertices not in C joined to v; it follows that $ki_C(v) + ko_C(v) = \deg(v)$.

We shall also use $ski(C) = \sum_{v \in C} ki_C(v)$, and $sko(C) = \sum_{v \in C} ko_C(v)$.

2.2. A growth process

Given a fitness function f on the subsets of V, we define a *growth process for* f as a sequence of subsets of V, such that each subset is obtained from the previous one, either by the incorporation or the elimination of a vertex. This sequence starts with a seed node and finishes by covering all the set V.

Given a seed node $v \in V$, we shall represent a growth process for f with seed v is a double sequence

$$D_{00}, D_{10}, \ldots, D_{1k_1}, \ldots, D_{a0}, \ldots, D_{ak_a}, \ldots, D_{b0}, \ldots, D_{bk_b}$$

of subsets of V, such that for each a such that $0 \leq a \leq b$, we have a subsequence $D_{a0}, \ldots, D_{ak_a}$.

- $D_{00} = \{v\}, k_0 = 0$.
- For $a \geq 0$, $D_{(a+1)0} = D_{ak_a}$ and $D_{(a+1)1}$ is obtained from $D_{(a+1)0}$ by adding to it one vertex such that $f(D_{(a+1)1}) > f(D_{(a+1)0})$.
- For $k \geq 1$, $D_{a(k+1)}$ is obtained from D_{ak} by elimination of a vertex (different from the seed vertex v), such that $f(D_{a(k+1)}) > f(D_{ak})$.

In addition, we assume that for each $a > 0$, there is no vertex $w \in D_{ak_a}$ such that its elimination induces an increase in f, and that there is no vertex out of D_{bk_b} whose addition induces an increase in f. Alternatively, we may describe the process by $v + s_1 w_1 + \cdots s_r w_r$, where the *signs* s_i (1 or -1) determine whether the vertex w_i is added or eliminated in this step, for example $v + w_1 + w_2 + w_3 + w_4 - w_5 + w_6$ means that in the first four steps we added w_1, w_2, w_3, w_4, in the fifth step we eliminated w_5 (which of course must be equal to some of the previously added vertices) and in the sixth step we added w_6.

2.3. Special cases

For $C \subset V$, consider $m_V(C), c_E(C) = m_E(C \times (V \setminus C))$, which we shall abbreviate m_V, c_E when there is no place for ambiguity. Recall that m_V is the normalized sum of the degrees of the vertices in C, and c_E is the normalized cut defined by C.

We shall deal with two parametric families of fitness functions, with a real parameter $t > 0$:

$$L_t = \frac{m_V - c_E}{m_V^{1/t}}$$

and

$$H_t = m_V(1 - m_V/2t) - c_E.$$

The first of these families is equivalent to the one used in Lancichinetti et al. [1], with $\alpha = 1/t$.

2.4. A differential analysis

In the following, we show these two facts:

- In both fitness functions, L_t and H_t, changing the resolution parameter t does not affect essentially the evolution of the growth process, but only defines the termination criteria. I.e., nodes that are candidates for addition (elimination) under some value of the resolution parameter, will remain candidates for addition (elimination) when the resolution is decreased.
- Both fitness functions, L_t and H_t, are essentially equivalent, in the sense that candidates for addition (elimination) for the L_t process are also candidates for addition (elimination) for the H_t process.

In order to prove this, let $C \subset V$, and $w \in V$. Suppose that we are to add w to C, if $w \notin C$, or to eliminate w from C, if $w \in C$, obtaining in either case a new set $C' = C \pm w$. Let us denote $\Delta m_V = m_V(C') - m_V(C)$, $\Delta c_E = c_E(C') - c_E(C)$, and

let $s, t > 0$ be two fixed values of the parameter. Then we have the following approximate expression for the difference quotient of L_t,

$$\frac{\Delta L_t}{\Delta m_V} \approx L'_t = \frac{1}{m_V^{1/t}}\left(1 - \frac{\Delta c_E}{\Delta m_V} - \frac{L_1}{t}\right).$$

For the difference quotient of H_t we obtain

$$\frac{\Delta H_t}{\Delta m_V} \approx H'_t = \left(1 - \frac{\Delta c_E}{\Delta m_V} - \frac{m_V}{t}\right).$$

Notice then the following relations

$$H'_t = H'_s + \frac{t-s}{ts} m_V \tag{1}$$

$$m_V^{1/t} L'_t = m_V^{1/s} L'_s + \frac{t-s}{ts} L_1 \tag{2}$$

$$H'_t = m_V^{1/t} L'_t + (L_1 - m_V)/t. \tag{3}$$

Eq. (1) shows us that if $t > s$ and $H'_s > 0$, then $H'_t > 0$, which means that if the vertex w is a candidate for addition (elimination) to C (from C) for the H_s process, it is also a candidate for addition (elimination) for the H_t process.

Eq. (2) shows us analogously that if $t > s$ and $L'_s > 0$, then $L'_t > 0$, which means that if the vertex w is a candidate for addition (elimination) to C (from C) for the L_s process, it is also a candidate for addition (elimination) for the L_t process.

This shows that the parameter t does not play an essential role during the growth process for H_t or L_t, but merely establishes the *termination criteria*.

Eq. (3) shows a delicate fact: If a vertex w is a candidate for addition (elimination) for the L_t process, and $m_V < L_1$ (this is usually true, notice that when $m_V > L_1$, $c_E > m_V(1 - m_V)$, which contradicts the notion of community, because the second term would be the mean of the first one if the vertices were to be selected randomly) then it is a candidate for addition (elimination) for the H_t process. Thus, both processes are essentially equivalent, their difference lying in the termination criteria. In exceptional cases, communities obtained with the H_t fitness functions are bigger than those obtained with the L_t fitness functions.

There are approximations involved, so that our previous comments are rough and qualitative: our experience testing both fitness functions confirms them.

2.5. Natural communities

The following is a formalization of the procedure described in Lancichinetti et al. [1] to obtain the natural community of a vertex v, generalized for any fitness function.

Algorithm 1: Natural communities

Input: A graph $G = (V, E)$, a fitness function f, a vertex $v \in V$
Output: A growth process $D_{00}, D_{10}, \ldots, D_{a0}, \ldots, D_{ak_a}, \ldots, D_{b0}, \ldots, D_{bk_b}$

```
1.1  begin
1.2      D_00 = {v}
1.3      m = 0
1.4      while there exists w out of D_m0 such that f(D_m0 + w) > f(D_m0) do
1.5          D_m1 = D_m0 + w
1.6          k = 1
1.7          while there exists w ∈ D_mk, w ≠ v : f(D_mk − w) > f(D_mk) do
1.8              D_m(k+1) = D_mk − w
1.9              k = k + 1;
1.10         end
1.11         D_(m+1)0 = D_mk
1.12         m = m + 1
1.13     end
1.14 end
```

The output of this "algorithm" is a growth process for f, $v + w_1 + w_2 \pm w_3 \pm \cdots \pm w_{r-1} + w_r$, such that there is no w not in D_{r0} with $f(D_{r0} + w) > f(D_{r0})$. Each D_{j0}, $0 \le j \le k$ satisfies that there is no $w \in D_{j0}$, $w \neq v$, such that $f(D_{j0} - w) > f(D_{j0})$. D_{r0} is a possible "natural community" with seed v.

Remark. Notice that the preceding prescription is not complete, because both the w that we choose to add, as well as the w that we choose to eliminate, depend upon a criterion that we do not fix.

2.6. Uniform growth processes

In the previous subsection we have described a method to obtain a natural community with seed v and fitness function f. Applying this with $f = H_t$ and fixed t, for different values of t we obtain different communities. Although it is not strictly true that "the bigger the t, the bigger the community", we have noticed in our differential analysis that this is essentially the case. Thus, it is reasonable to wonder whether it is possible to obtain all these communities with a unique process, starting with the smallest ones and proceeding with the biggest ones. The answer is affirmative, as we shall see now.

Let us assume that we have our parametric family of fitness functions $H_t : t > 0$. We shall call $\partial(C)$ to the *boundary* of C, that is the set of nodes not in C that have one or more connections to nodes in C. Given C and $w \in V$ such that $ki_C(w) > 0$, there always exists $t_c = t_c(C, w) > 0$ such that $H_{t_c}(C \pm w) = H_{t_c}(C)$. Indeed, we have:

$$\begin{aligned} H_t(C \pm w) &= (m_V + \Delta m_V)(1 - (m_V + \Delta m_V)/2t) - (c_E + \Delta c_E) \\ &= m_V(1 - m_V/2t) - c_E - \frac{\Delta m_V}{t}(m_V + \Delta m_V/2) + \Delta m_V - \Delta c_E \\ &= H_t(C) - \frac{\Delta m_V}{t}(m_V + \Delta m_V/2) + \Delta m_V - \Delta c_E \end{aligned}$$

and it follows that

$$t_c = \frac{\Delta m_V(m_V + \Delta m_V/2)}{\Delta m_V - \Delta c_E}$$

satisfies our exigencies. We also see that

$$\Delta H_t = -\frac{\Delta m_V}{t}(m_V + \Delta m_V/2) + \Delta m_V - \Delta c_E$$

and it follows that $\Delta H_t > 0$ when $t > t_c$ and $w \in \partial(C)$, and that $\Delta H_t > 0$ when $t < t_c$ and $w \in C$.

Let $v + \sum_{i=1}^{M} s_i w_i$ be an algebraic expression with the previously introduced meaning, where we naturally assume that each time we eliminate a vertex, that vertex had previously been added. Let $D_0 = v$ and for $r > 0$, $D_r = v + \sum_{i=1}^{r} s_i w_i$. We assume that for each r, $0 \le r < M$, $ki_{D_r}(w_{r+1}) > 0$. We shall consider values $0 = t_0, t_1, \ldots, t_r$ associated to this expression, $t_r = max\{t_{r-1}, t_c(D_{r-1}, w_r)\}$ when $s_r = 1$, $t_r = t_{r-1} < t_c(D_{r-1}, w_r)$ when $s_r = -1$. Thus, $t_0, \ldots, t_r$ is a non-decreasing sequence, and $D_0, \ldots, D_r$ is a growth process for H_t if $t > t_r$. We call $D_0, \ldots, D_M$ a *uniform growth process* for H.

Algorithm 2: A growth process for H

Input: A graph $G = (V, E)$, a vertex $v \in V$
Output: A growth process for H: $D_{00}, D_{10}, \ldots, D_{a0}, \ldots, D_{ak_a}, \ldots, D_{b0}, \ldots, D_{bk_b}$

```
2.1  begin
2.2      D_00 = {v}
2.3      t_a = 0
2.4      m = 0
2.5      while there exists w in ∂(D_m0) do
2.6          let w_0 be such that t_c(D_m0, w_0) = min_{w∈∂(D_m0)}(t_c(D_m0, w))
2.7          t_a = max{t_a, t_c(D_m0, w_0)}
2.8          D_m1 = D_m0 + w_0
2.9          k = 1
2.10         while there exists w ∈ D_mk, w ≠ v : t_c(D_mk, w) > t_a do
2.11             D_m(k+1) = D_mk − w
2.12             k = k + 1;
2.13         end
2.14         D_(m+1)0 = D_mk
2.15         m = m + 1
2.16     end
2.17 end
```

The output of this "algorithm" is a uniform growth process for H, which ends by covering the whole graph. The successive truncations of the sequence thus obtained are natural communities for v at different resolutions. In the sequel we assume, with empirical evidence, that these natural communities are made up of small subcommunities, which are inserted one after another during the growth process. The following section explains how to detect these communities.

3. Extracting the communities

The previous section described the growth process, which outputs a sequence $D_M = v + \sum_{i=1}^{M} s_i w_i$. Some vertices of the graph may be inserted, removed and later reinserted during this process. We filter this sequence in order to generate

a new one which only keeps the last insertion of each vertex. In this way we obtain a subsequence $\mathcal{S}$ of the original one, such that each vertex appears once and only once throughout it. Now, as the growth process tends to choose the vertices by their strong linkage to the natural community built so far, we state that two consecutive vertices in the sequence either belong to the same community or they are border vertices. Considering that the first case is the most frequent, an algorithm is needed in order to cut that sequence $\mathcal{S}$ into a list of communities $\mathcal{C} = (C_1, C_2, \ldots, C_N)$. The cuts are made by observing the behavior of the function

$$S(w) = \frac{c_E(C(w))}{m_V(C(w))},$$

where $C(w)$ are the sublists of $\mathcal{S}$, from the start of the last *partial community* up to w. We shall use the term *partial community* to denote the evolving set of nodes since the last cut in the process and up to the current node. The criterion to decide the closure of the community and the start of a new partial community is *an increase in the function* $S(w)$.

In other words, the function considers the sequence of vertices added since the start of the last *partial* community, C_i, and computes the evolution of the quotient between the "cut of the *partial* community with its external world" (c_E) and the "mass of the *partial* community" or cumulated degree (m_V).

To understand the statistical behavior of this function, we shall consider a community $C = (v_1, v_2, \ldots, v_n)$ whose vertices have a mixing parameter μ. That is, they share a fraction μ of its edges with other communities and a fraction $1-\mu$ with their own community. We shall call C_i to the successive subsets built by the growth process within the community C. The control parameter for the subdivision is then

$$S_i = S(v_i) = \frac{m_E(C_i \times (V \setminus C_i))}{m_V(C_i)} = 1 - L_1(C_i).$$

Our statistical analysis will be based in the following relations:

$$m_E(C_i \times (V \setminus C)) = \mu m_V(C_i)$$
$$m_E(C_i \times C_i) = \lambda_i m_E(C_i \times C).$$

The first follows from the hypothesis that all the vertices in C share a similar μ. The second is just a definition of a parameter λ_i which belongs to the interval [0, 1].

From these equations it can be shown, by a straightforward calculation and using the additivity of m_E, that

$$S_i = \mu + (1-\mu)(1-\lambda_i)$$
$$(1-\mu)\lambda_i = L_1(C_i).$$

We assume that L_1 has a monotone increasing behavior throughout the community construction, and this implies a monotone decreasing behavior on S_i also, even without assuming a constant μ. We also observe that, for the last vertex in the community, $S = \mu$ (because $\lambda = 1$).

Now, let us see what happens when the community is finished and we incorporate a vertex v from another community C', which has a different mixing parameter μ'. We shall call C^+ to $C \cup \{v\}$, and we define ϵ by the relation

$$m_E(\{v\} \times C) = \epsilon\, m_E(\{v\} \times (V \setminus C')) = \epsilon \mu' m_V(\{v\})$$

which represents the proportion of external connections from $v \in C'$ going to vertices in C.

The new value of our control parameter is then

$$S^+ = \frac{m_E(C^+ \times (V \setminus C^+))}{m_V(C^+)}$$

and it can be shown that

$$S^+ = \mu + \frac{(1 - 2\epsilon\mu' - \mu) m_V(\{v\})}{m_V(C^+)}.$$

If the mixing parameters are not very high or ϵ is small, which is expected, this new value S^+ will break the decreasing behavior of S inducing the closure of C and the start of a new partial community C' with v' as its first node v'_1.

We can now resume the behavior of $S(w)$ in the following way:

- The function starts from $S(w) = 1$ when the first node of the community is incorporated ($w = v_1$).
- The function $S(w)$ decreases from 1 up to μ throughout the community.
- The function $S(w)$ will increase when the community is finished and the process tries to incorporate an external node w'.
- Under that condition, a new community is started with that external node and $S(w')$ is set to 1.

Fig. 1 shows the behavior of $S(w)$ for the growth process on four different networks: dolphins [14], football [3], an instance of the LFR benchmark [11] (see Section 5) and a collaboration network [3]. We remark the decreasing trend of the function inside each community, reaching minimum values between 0.2 and 0.5 in general, which points out the average μ (and consequently, the cohesion) of each discovered community.

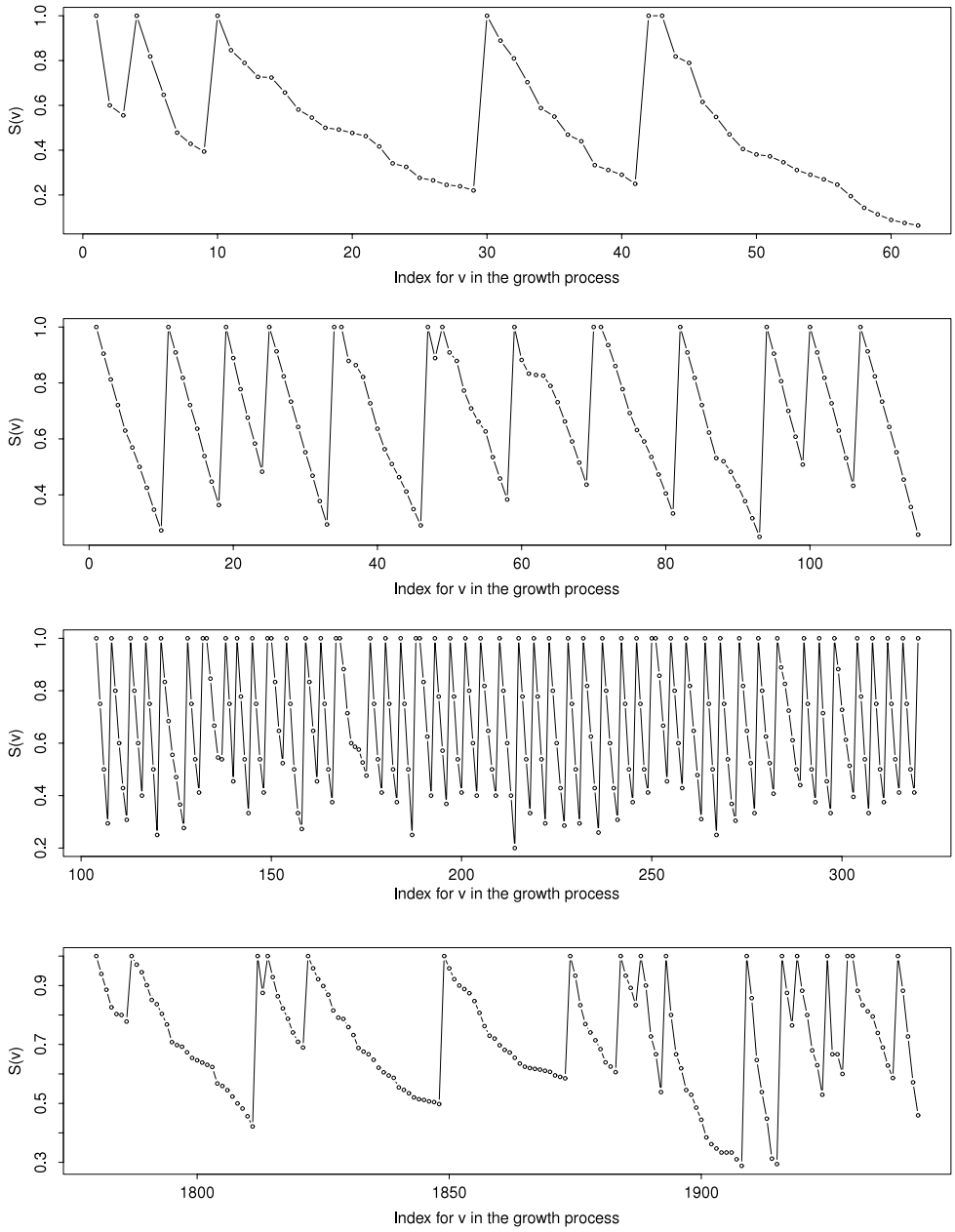

Fig. 1. The cuts in the growth process in different networks: dolphins [14], football [3], an instance of the LFR benchmark [11] (see Section 5) and a collaboration network [3]. In the first two ones we show the complete growth-process; the others are an extract of the full process.

4. Algorithmic complexity

In this section we shall prove that the proposed method for detecting community structure has a computational complexity of $O(n \cdot k_{\max} + m \cdot \log(n))$. We shall analyze the growth process in Section 2 (described in Algorithm 2) and the extraction of communities described in Section 3. We will use the notation $\mathcal{N}(v)$ for the neighborhood of v (the set of vertices which have an edge with v), and $k_{\max}$ for the maximum degree, i.e., $k_{\max} = \max\{k(v),\, v \in V\}$.

We begin by analyzing the growth process at a certain step r, when the natural community is $D_r = v + \sum_{i=1}^{r} s_i w_i$. The instruction 2.6 in Algorithm 2 points out that we must incorporate the node w in the boundary of D_r with the minimum $t_c(D_r, w)$. We observe, from the expression of t_c, that

$$t_c(D_r, w) = \frac{\Delta m_V}{\Delta m_V - \Delta c_E} \cdot (m_V + \Delta m_V/2).$$

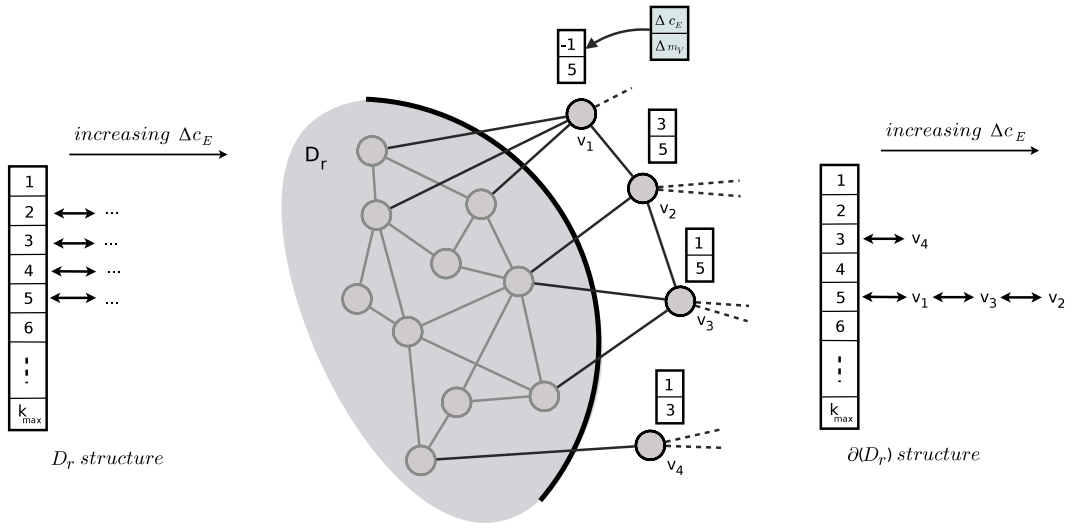

Fig. 2. Structure kept during the process for the natural community D_r and its boundary, $\partial(D_r)$. In each structure, the nodes are grouped by degree (represented by the columns with values $1, 2, \ldots, k_{\max}$). Nodes with the same degree are kept in a logical structure ordered by increasing Δc_E, like a tree or map (represented by the two-way arrows). The values at the head of these structures are the only ones that need to be considered at each step. In this example, v_1 and v_4 are considered for addition, and v_1 is chosen because it minimizes t_c. Using these structures, the complexity of the growth process was reduced to $O\,(n \cdot k_{\max} + m \cdot \log(n))$.

Given the subset of all the nodes in the boundary having the same degree as w, the one with the minimum t_c is the one that minimizes $\frac{\Delta m_V}{\Delta m_V - \Delta c_E}$. For a given degree, minimizing this expression is the same as minimizing Δc_E. So, if we group the nodes in the boundary into lists according to their degree, and we order each list by increasing value of Δc_E, then we can assure that the node in the boundary which minimizes t_c must be at the head of one of these lists. Then we propose to keep an updated structure with the boundary $\partial(D_r)$ (see Fig. 2). We shall need an analogous structure for the natural community D_r for the eliminations; this structure is also shown in the same figure. In this way we reduce the complexity from analyzing all the boundary into analyzing $k_{\max}$ nodes at most.

We shall call $l_{\max}$ to the maximum length of one list, and these lists will be implemented with a direct access, ordered structure, as a map or tree. The operations of insertion preserving order have complexity $O(\log(l_{\max}))$, while the access has complexity $O(1)$. We are now ready to analyze the complexity of the r-th step.

1. Looking for the node w with the minimum $t_c(D_r, w)$ implies finding the minimum between the heads of each of the lists. This has a complexity $O(k_{\max})$.
2. Updating the structures involves:
   (a) Removing w from its list in the $\partial(D_r)$ structure. Complexity $O(1)$.
   (b) Updating Δc_E for w to $(-\Delta c_E)$. Complexity $O(1)$.
   (c) Inserting w into the $k(w)$-list in the D_r structure. Complexity $O(\log(l_{\max}))$.
   (d) Updating Δc_E for the neighbors of w, i.e., $v \in \mathcal{N}(w)$:
      i. If $v \notin D_r$, update Δc_E to $\Delta c_E - 2/(2m)$. Complexity $O(1)$.
      ii. If $v \in D_r$, update Δc_E to $\Delta c_E + 2/(2m)$. Complexity $O(1)$.
   (e) Reinserting (or inserting) the neighbors of w in the lists:
      i. If $v \in D_r$, reinsert it into the $k(v)$-list of the structure for D_r, ordered by its new value of Δc_E. Complexity $O(\log(l_{\max}))$.
      ii. If $v \notin D_r$, $v \notin \partial D_r$, insert it into the $k(v)$-list of the structure for ∂D_r, ordered by its new value of Δc_E. Complexity $O(\log(l_{\max}))$.
      iii. If $v \notin D_r$, $v \in \partial D_r$, reinsert it into the $k(v)$-list of the structure for ∂D_r, ordered by its new value of Δc_E. Complexity $O(\log(l_{\max}))$.

Putting all together, the complexity of this step is $O(k_{\max} + |\mathcal{N}(w)| \cdot \log(l_{\max}))$.

Now, the steps during the growth process may consist not only of incorporations, but also of eliminations. The elimination condition is resumed in the instruction 2.10 in Algorithm 2.

The logic of eliminations is exactly the same: i.e., it consists on analyzing the heads of the lists *in the structure for* D_r, looking for a value of t_c bigger than the actual t_a. In that case, the node is removed from D_r and its neighbors are updated in an analogous way as with the incorporations, and with a similar complexity.

During all our experiments, we verified that the eliminations are scarce, and we shall assume that they are at most, of the same order as the incorporations. We can assume that the process consists only of incorporations in order to calculate the complexity. Under this hypothesis, each node is incorporated only once in the process, and the complexity of the growth process can be expressed as:

$$O\left(\sum_{w\in V}(k_{\max}+\mathcal{N}(w)\cdot\log(l_{\max}))\right).$$

The sum over all the neighbors of $\mathcal{N}(w)$ can be translated into the fact that *every edge in the network is considered only once*. Regarding $l_{\max}$, we cannot make any assumption. In heavy-tailed degree distributions, the amount of nodes with a certain small degree value, may be of $O(n)$, so we shall bound $l_{\max}$ with n. Thus, we have a complexity of

$$O\left(n\cdot k_{\max}+m\cdot\log(n)\right).$$

In the initialization of the process, the Δc_E and Δm_V of the nodes are both set to the node degrees. This step does not change the complexity.

Finally, as a result of the growth process in Algorithm 2, we get a sequence C_M of vertex insertions interleaved with some eliminations. Obtaining the subsequence $\mathcal{S}$ where each vertex appears once and only once requires running through C_M twice, and has a complexity $O(n)$. The *cutting algorithm* of Section 3 runs over $\mathcal{S}$ only once, computing $S(w)$ for each node based on the values of c_E and m_V, which were already computed during the growth process. The complexity here is linear.

Thus, the complexity of our method is dominated by the growth process and is $O\left(n\cdot k_{\max}+m\cdot\log(n)\right)$.

5. Results and discussion

In this section we show the results of our local method applying it to (i) a benchmark of heterogeneous and homogeneous networks, (ii) real networks of different sizes and(iii) random networks. In Section 5.1.1 we use some statistical tools to evaluate the method and compare it to global ones. In Section 5.2 we show that the algorithm is robust for large networks with a well-defined community structure, and in Section 5.3 we use it to analyze a scientific collaboration network.

5.1. Benchmarking with a set of heterogeneous networks

We evaluated our algorithm with a benchmark proposed in Lancichinetti et al. [11]. We used their software to create sets of random graphs following a power law for the vertex degree distribution (exponent α) and for the community size distribution (exponent β). Each set of graphs contains instances with different values of the *mixing parameter* μ, ranging from 0.05 to 0.80 (the mixing parameter is the fraction of neighbors that each vertex has in communities other than its own). Our comparisons are similar to the ones performed in Refs. [15,12].

We built 4 sets:

- BENCH1: Heterogeneous networks; $\alpha=2.0$; $\beta=3.0$; $\langle k\rangle=10$; $k_{\max}=50$; 1, 000 nodes
- BENCH2: Homogeneous networks; $k=10$; 1, 000 nodes
- BENCH3: Heterogeneous networks; $\alpha=2.0$; $\beta=3.0$; $\langle k\rangle=10$; $k_{\max}=50$; 5, 000 nodes
- BENCH4: Homogeneous networks; $k=10$; 5, 000 nodes.

For each set we constructed 1,600 graphs with 100 instances for each values of the mixing parameter μ, which moves between 0.05 and 0.80 in steps of 0.05.

We have used this benchmark for different reasons:

- It simulates networks with heterogeneous distributions as those of real networks. These distributions provide greater challenges to the community discovery algorithms with respect to fixed-degree networks like the ones generated by the Girvan–Newman benchmark [3]. For example, heterogeneous networks are subject to resolution limit problems when global methods are applied.
- The parameters adjust tightly to the proposed values, and the distribution of μ follows a roughly bell-shaped curve around the desired value of μ.
- It has a low complexity, which makes it suitable to generate a big set of graphs.

Fig. 3 analyzes the results for the different benchmarks. We used the *normalized mutual information* (see Appendix A) to compare our partition with the one issued from the benchmark generation, for each network instance. The 1,600 instances of each benchmark are grouped by their μ value. We used the `boxplot` command of the *R* statistical software [16]. This command computes the quartiles for the distribution of the results for a certain μ, displaying: the median (second quartile); boxes representing the 3rd and the 1st quartiles; and whiskers that are placed at the extremes of data.

The results are successful for a wide range of values of μ. In general, the mutual information is larger than 0.90 for values of the mixing parameter up to 0.60. This behavior was also observed in other benchmarks with values of α and β ranging from 1.00 to 3.00, which we do not reproduce here. We also run some tests on large networks with 1,000,000 nodes and $\mu=0.45$ generated with the same benchmark, and obtained values of the mutual information around 0.99.

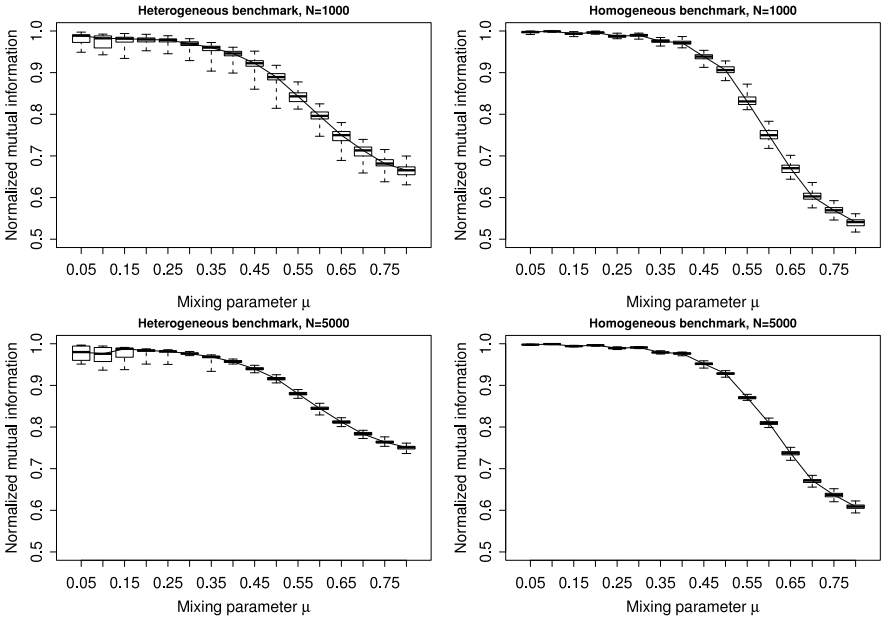

Fig. 3. Statistical analysis of the normalized mutual information between our partition and the communities known *a priori* as a function of the mixing parameter μ, for each of the four benchmark sets. Each box represents the 100 networks generated for each μ value.

5.1.1. A comparison with other methods

Fig. 4 compares the partitions found with our growth process based on the H fitness function against those obtained with two other methods:

- The Louvain algorithm [7], which is one of the most efficient modularity-based methods.
- Infomap [17]. This algorithm by Rosvall and Bergstrom is based on compression of the information about the graph structure. This is the best performing algorithm on the LFR benchmark [18].

The points represent median values of the normalized mutual information for the 100 different networks generated for each value of the mixing parameter μ, in the BENCH3 and BENCH4 sets. The reference partition is the one computed *a priori* by Lancichinetti's benchmark, from which the networks are generated. So when we mention the mutual information for the growth process we mean *the mutual information against the pre-computed communities*. The same holds for the mutual information for the Louvain and Infomap algorithms.

We observe that our growth process outperforms Louvain for the detection of communities in the benchmarks, and that the difference in performance increases for higher values of the mixing parameter μ. This behavior seems to be related to the resolution limit of modularity-based methods. On the other hand, the growth process is surpassed by Infomap, whose performance is known to be very good on the LFR benchmark.

In order to make a comparison regarding the resolution limit of modularity, we analyze how many pairs of communities found with our growth process would be usually joined by agglomerative modularity maximization techniques, because their union increases modularity. In Fig. 5 we show the results of this analysis for the BENCH1 set. The *y-axis* represents the percentage of 'joinable' pairs $(C_i, C_j), i \neq j$ in a given partition $\mathcal{C}$ (i.e., the amount of 'joinable' pairs in relation to $|\mathcal{C}| \cdot (|\mathcal{C}| - 1)/2$). The boxes represent the 100 network instances for a given value of the mixing parameter μ in the *x-axis*. The left plot corresponds to Lancichinetti's *a priori* partition, while the right plot is for the communities that we obtain. The linear behavior of the percentage as a function of μ explains why modularity-based techniques tend to fail when the values of μ are bigger. In fact, the Louvain algorithm cannot find these communities because they are merged until a local maximum is achieved, at least theoretically: this may be slightly affected by the precision parameter included in the software for stability reasons.

Fig. 6 shows a comparison with the original Lancichinetti's method for an instance of BENCH1 with $\mu = 0.20$. When we ran Lancichinetti's algorithm we chose a value of $\alpha = 0.70$, so that its resolution was consistent with that of the *a priori* communities. The three subfigures represent the community cover found by Lancichinetti's method with $\alpha = 0.70$ (top), the community partition found by our growth process (center) and the *a priori* community partition defined by the LFR benchmark (bottom). Each subfigure contains a binary matrix whose column axis holds the vertices of the graph and the row axis represents the communities found by each method. The matrix has a 1 (black dot) in position (i, j) when the i-th vertex in the order belongs to the j-th community in the order.

The vertices are ordered in the same way in the three subfigures, and this order is defined by the index of the vertices in our growth process. The communities are ordered on the row axis by computing the *center of mass* of its rows. The election of these orders naturally implies that the matrix for our growth process (center) displays a stair shape.

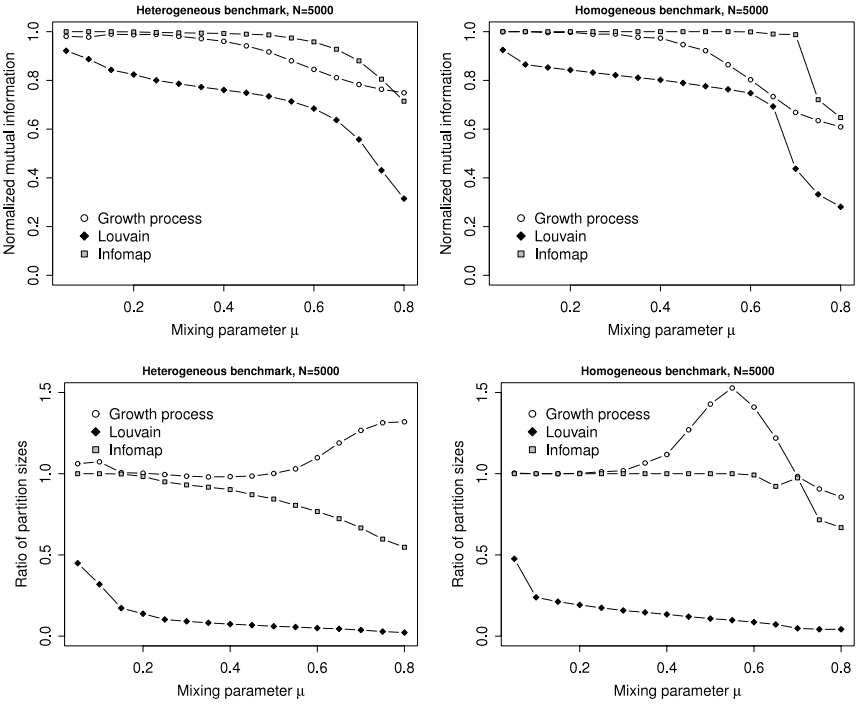

Fig. 4. Comparison between our growth process, Louvain's modularity-based method and Infomap for the BENCH3 *(left)* and BENCH4 *(right)* sets of benchmarks. We consider the communities generated *a priori* by Lancichinetti's benchmark and use them as reference partitions. In the *upper* subfigures we compare the normalized mutual information (against the reference partitions) for both methods. In the *lower* subfigures we compare the ratio between the size of the community partitions found by the methods and the size of the reference partitions. The points always represent median values for the 100 networks generated for each μ value.

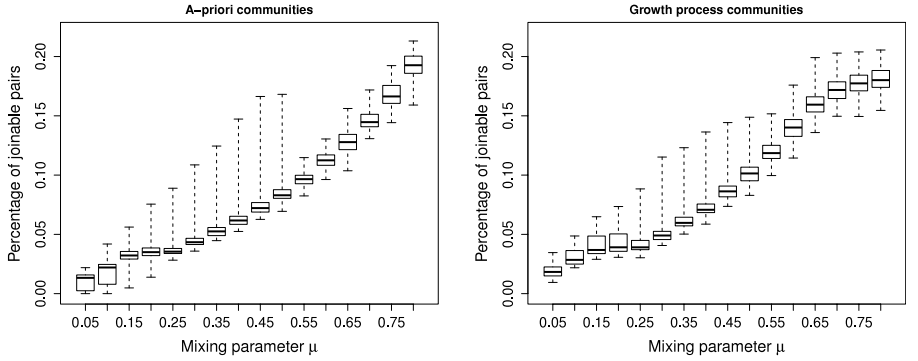

Fig. 5. Boxplots representing the percentage of community pairs (C_i, C_j), $i \neq j$ whose join increases modularity in the instances of the BENCH1 set. Each box represents the 100 instances generated for each μ value. (a) Lancichinetti's *a priori* communities. (b) Communities obtained by our fitness growth process. It is a remarkable fact that the original (*a priori*) communities are not local maxima for modularity, and that this trend grows when mixing parameter μ increases. In other words, the benchmark generates partitions for which modularity optimization techniques would tend to fail. We also point out that a similar plot for the partitions obtained by the Louvain algorithm would show a constant zero for the percentage of joinable pairs. This is a mandatory fact for any modularity maximization agglomerative technique which attains a local maximum.

The presence of black segments in the top and bottom subfigures implies that vertices which are consecutive in our growth process are part of the same community in Lancichinetti's cover or in the *a priori* partition, respectively. Notice also that in Lancichinetti's cover (top) many vertices belong to several of its communities due to the overlapping. In fact, there is a strong coincidence between our communities and those in Lancichinetti's cover, except for the fact that some of the communities in the cover are essentially the same, i.e., they have a high overlap.

The comparison with the *a priori* partition (bottom) confirms the fact that our growth process visits the communities one after the other, which is a relevant thesis in our work. The isolated vertices in the bottom figure represent misclassified vertices and are usually located at the borders of the intervals that in the growth process order represent communities.

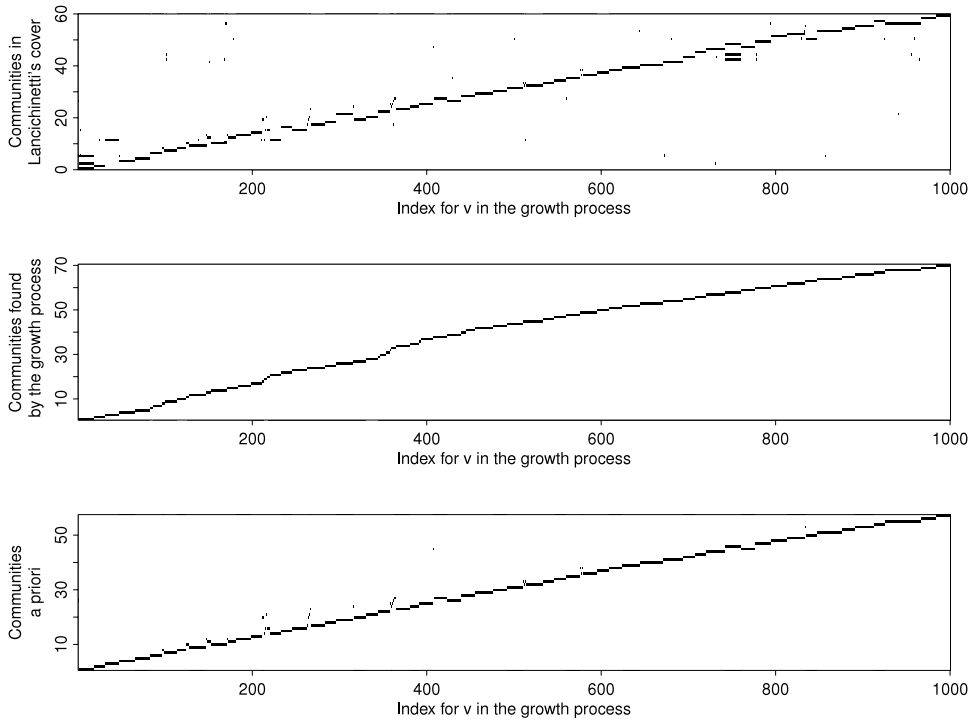

Fig. 6. Comparison between Lancichinetti's method and our growth process, for an instance of BENCH1 with $\mu = 0.2$. The three subfigures represent the community cover found by Lancichinetti's algorithm with $\alpha = 0.70$ *(top)*, the community partition found by our growth process *(center)* and the community set up *a priori* by the LFR benchmark *(bottom)*. Each of the subfigures contains a binary matrix. Each matrix has one column for each vertex and one row for each community in the respective community structure. The matrices have a 1 (black dot) in position (i, j) when the i-th vertex in the order belongs to the j-th community in the order. The vertices (columns) are always ordered according to their position in our growth process, and the communities (rows) are ordered by computing the center of mass of its rows in an increasing fashion. The election of these orders naturally implies that the subfigure for our growth process *(center)* displays a stair. From observing the existence of black segments in the top and bottom figures we see that: (i) Lancichinetti's process visits some communities several times; (ii) Throughout our growth process we find the different communities in Lancichinetti's cover; (iii) Our growth process visits the *a priori* communities one after the other.

Table 1
Summary of results for the analyzed networks. The columns represent: network size (number of vertices and edges), average number of communities found with the Fitness Growth Process and standard deviation, and the amount of modules discovered by Louvain's algorithm.

Network	n	m	$\langle\lvert\mathcal{C}_{FGP}\rvert\rangle$	$stdev(\lvert\mathcal{C}_{FGP}\rvert)$	$\lvert\mathcal{C}_{Louvain}\rvert$
karate	34	78	3.21	0.54	4
dolphins	62	159	6.31	2.09	5
football	115	613	12.57	0.82	10
e-mail	1,133	5,451	173.86	9.30	10
BENCH1	1,000	5,022	148.74	2.13	22
arXiv	9,377	24,107	1,573.52	22.59	62
CondMat	36,458	171,736	5,476.10	38.53	802
WWW	325,729	1,090,108	19,972.11	78.77	358
ER100	100	508	16.09	2.77	8
ER1k	1,000	5,111	173.59	7.80	16
ER10k	10,000	100,261	1,732.26	26.52	10

5.2. Robustness analysis

In order to study the robustness of our method in real networks, whose communities are generally unknown *a priori*, we propose to analyze the mutual information between different partitions starting from randomly chosen vertices, and observe the repeatability of the results. The studied networks include karate club [19], the bottlenose dolphins network [14], the American college football network [3], an e-mail interchange network [20], Erdös–Rényi random graphs ER* [21], an instance from the BENCH1 benchmark with $\mu = 0.30$ (see Section 5.1), a portion of arXiv [22], a collaboration network in

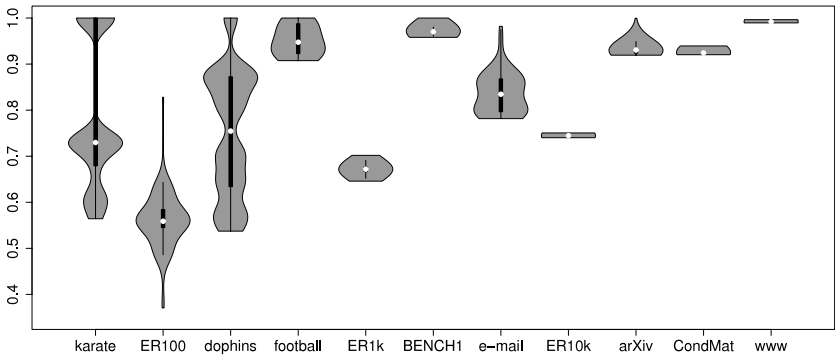

Fig. 7. Boxplots (with density) representing the results for different real networks and some Erdös–Rényi random graphs. The networks are spread over the *x*-axis. Each boxplot with density is composed by a white central point representing the median, a black segment whose extrema represent the first and third quartiles, and a vertical curve which is an approximation to the histogram function of the mutual information values. These values of mutual information compare the partitions obtained from different seed vertices in the network to a reference partition obtained from a fixed seed vertex.

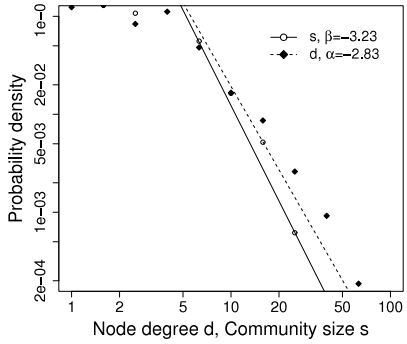

Fig. 8. Community size and node degree distribution for the collaboration network `CondMat`. The histograms were built with a log-binning procedure.

Condensed Matter `CondMat` [3], and a portion of the World Wide Web network `WWW` [23]. Table 1 shows the sizes of these networks.

Fig. 7 shows the boxplots, together with the density functions, of the mutual information for each network. In each of them we picked a random vertex as seed, run the algorithm, and took the resulting partition as the *reference partition*. Then we started the algorithm from other seed vertices, and measured the mutual information between these partitions and the reference partition. In small networks, we considered all the vertices, and just 1000 different vertices for `arXiv`, `CondMat` and the `WWW` network. The fact that we just consider one reference partition to compare with the others and do not make an all pairwise comparison is justified by the transitivity relationship that we develop in Appendix B.

The first observation of Fig. 7 is that the random graphs (`ER100`, `ER1k`, `ER10k`) show small values of mutual information when the robustness analysis is performed. This is an expected result, as it is in accordance with the fact that *ER* graphs do not have a community structure, as Ref. [24] point out.

The `karate` case is interesting because many instances have a mutual information of 0.74. In these, the amount of misclassified nodes is just 4, but as they represent a considerable fraction of the whole network (34 nodes) the mutual information has an important fall. Something similar happens in the `dolphins` network. The other networks present high values of mutual information with small dispersions (i.e., boxplots are quite narrow). This trend is even more noticeable in the large networks. In fact, the `WWW` is an interesting case because all the mutual information values that we found lay around its median value of 0.989 with extremes at 0.989 ± 0.02, which means, by transitivity, that the different partitions found when starting the process from different vertices, are quite similar between them.

5.3. Application to a collaboration network

Finally, we applied our algorithm to a network of coauthorships from the Condensed Matter E-Print Archive (`CondMat`). We analyzed the giant component of the network, composed by 36,458 vertices and 171,736 edges. The result was a partition with 5,508 communities. Both the community size distribution and the degree distribution follow a power-law on the community size (see Fig. 8) which may be due to the self-similarity of the network [25].

While the biggest community in this network contains about 31% of the graph edges (53,880 internal connections), it has only 406 vertices (the 1.1%). Evidently, this community has a strong cohesion.

6. Conclusions

The work by Lancichinetti et al. [1] suggests the possibility of using different fitness functions for detecting local communities under a general procedure. In this work we have defined a fitness function H_t, depending on a real parameter t, and we have shown that it is essentially equivalent to the original one, which depends on a resolution parameter α. Then we proved an important fact: neither α nor t plays an important part in the vertex selection criteria, but only in the termination decision. This means, for example, that we can obtain a local community C_t for some t, and then build the local community for $t' > t$ by taking C_t and continuing the process until t'. So we proposed a unique fitness growth process which finds an ordering of the vertices such that the different communities lie one after the other. This sequence is the input of a cutting algorithm that extracts a community partition of the graph. The algorithm is freely available to the scientific community as an open-source software [26]. The algorithmic complexity was studied and we proposed efficient data structures that reduce it to $O\left(n \cdot k_{\max} + m \cdot log(n)\right)$. We remark that for networks which can be adjusted to a power-law distribution with exponent β, $k_{\max}$ is $O(\sqrt[\beta]{n})$ [27] and the complexity is then $O\left(n^{1+1/\beta} + m \cdot \log(n)\right)$.

We also exploited a benchmark of heterogeneous graphs to test our method. On one side, we tested the correctness of the results by comparing them against communities defined *a priori*. On the other side, we gave an explanation on why global methods tend to fail on some heterogeneous networks. These ideas were illustrated using the normalized mutual information.

Finally, we showed that the method is robust for many real networks. By analyzing random graphs, we pointed out that the behavior of the method may allow us to differentiate networks with a strong community structure from randomly connected ones.

As a future work we plan to study different ways of changing the vertex selection criteria of the growth processes, taking advantage of specific properties of the graphs of interest. We also intend to extend the results for detecting situations of overlapping communities.

Acknowledgments

This work was partially funded by an UBACyT 2010–2012 grant (20020090200119), an UBACyT 2012–2015 grant (20020110200181) and a PICT-Bicentenario 2010-01108. M.G. Beiró acknowledges a Peruilh fellowship.

Appendix A. Mutual information

For the purpose of comparing different community structures, we used the *normalized mutual information* [12]. In order to define it in terms of random variables, we consider the following process: we pick a vertex v at random from V with a uniform distribution, and define a random variable X_1 on V related to partition $\mathcal{C}_1$. This variable assigns to each vertex the subindex of the community it belongs to. The distribution of X_1 is

$$\mathbf{P}[X_1 = i] = p_i = \frac{|C_i|}{|V|},$$

where $i = 1, 2, \ldots, |\mathcal{C}_1|$. The entropy of $\mathcal{C}_1$ is defined by

$$H(\mathcal{C}_1) = -\sum_{i=1}^{|\mathcal{C}_1|} p_i \cdot \log\left(p_i\right).$$

If we introduce a second partition $\mathcal{C}_2$ with its related random variable X_2 on the same sample space V, the joint distribution of X_1, X_2 is

$$\mathbf{P}[X_1 = i, X_2 = j] = p_{ij} = \frac{|C_i \cap C_j|}{|V|},$$

where $i = 1, 2, \ldots, |\mathcal{C}_1|, j = 1, 2, \ldots, |\mathcal{C}_2|$. In these terms, the normalized mutual information is expressed by

$$NMI(\mathcal{C}_1, \mathcal{C}_2) = -2 \cdot \frac{\sum_{i=1}^{|\mathcal{C}_1|}\sum_{j=1}^{|\mathcal{C}_2|} p_{ij} \cdot \log\left(\frac{p_{ij}}{p_i \cdot p_j}\right)}{\sum_{i=1}^{|\mathcal{C}_1|} p_i \cdot \log\left(p_i\right) + \sum_{j=1}^{|\mathcal{C}_2|} p_j \cdot \log\left(p_j\right)},$$

where $\sum_{i=1}^{|\mathcal{C}_1|}\sum_{j=1}^{|\mathcal{C}_2|} p_{ij} \cdot \log\left(\frac{p_{ij}}{p_i \cdot p_j}\right) = MI(\mathcal{C}_1, \mathcal{C}_2)$ is the mutual information. The following equality holds:

$$MI(\mathcal{C}_1, \mathcal{C}_2) = H(\mathcal{C}_1) + H(\mathcal{C}_2) - H(\mathcal{C}_1, \mathcal{C}_2),$$

where $H(\mathcal{C}_1, \mathcal{C}_2)$ is the joint entropy. $NMI(\mathcal{C}_1, \mathcal{C}_2)$ falls between 0 and 1, and gives an idea of the similarity between partitions in terms of the information theory, i.e., in terms of the information about $\mathcal{C}_1$ that lies in $\mathcal{C}_2$, or vice versa.

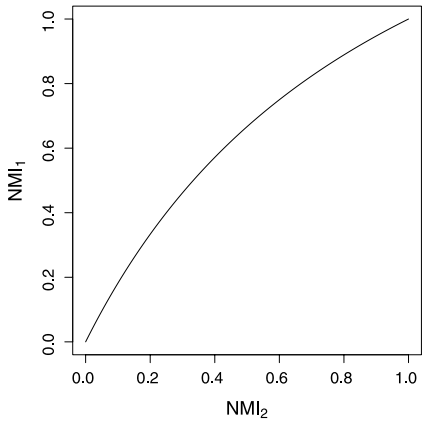

Fig. B.1. Functional relationship between two normalizations of the mutual information: NMI_1 and NMI_2.

The inherent idea is that a partition $\mathcal{C}$ of a graph gives us some information relative to the classification of vertices into groups. This amount of information is measured by its entropy, $H(\mathcal{C})$.

In fact, the denominator in $NMI(\mathcal{C}_1, \mathcal{C}_2)$ together with the -2 constant represents a normalization by the average entropy of the partitions, $\frac{H(\mathcal{C}_1)+H(\mathcal{C}_2)}{2}$. A normalized mutual information of 1 implies that the partitions are coincident.

Appendix B. Normalizations and triangular inequalities

Throughout this paper we used the normalization of the mutual information proposed in Appendix A, i.e. *NMI*. We remark that other normalizations also exist, like:

$$NMI_2(\mathcal{C}_1, \mathcal{C}_2) = \frac{MI(\mathcal{C}_1, \mathcal{C}_2)}{H(\mathcal{C}_1, \mathcal{C}_2)}$$

which has the advantage that $1 - NMI_2$ is a metric [28]. Although we consider it more correct to use this normalization, we shall hold to the first one for consistency with other works in the literature. Anyway, we were able to find a transitivity property for *NMI* too (we call it NMI_1 in the following equations). In fact, observing that:

$$\frac{2}{1 - NMI_1(\mathcal{C}_1, \mathcal{C}_2)} = \frac{H(\mathcal{C}_1, \mathcal{C}_2)}{H(\mathcal{C}_1) + H(\mathcal{C}_2) - H(\mathcal{C}_1, \mathcal{C}_2)}$$
$$\frac{1}{1 - NMI_2(\mathcal{C}_1, \mathcal{C}_2)} = \frac{H(\mathcal{C}_1) + H(\mathcal{C}_2)}{H(\mathcal{C}_1) + H(\mathcal{C}_2) - H(\mathcal{C}_1, \mathcal{C}_2)}$$

we can deduce a functional relationship between these two:

$$\frac{2}{1 - NMI_1(\mathcal{C}_1, \mathcal{C}_2)} - \frac{1}{1 - NMI_2(\mathcal{C}_1, \mathcal{C}_2)} = 1.$$

This relationship produces a hyperbole as in Fig. B.1. The good behavior of the function around (1, 1) assures that values of NMI_1 close to 1 imply values of NMI_2 close to 1 too. The transitivity of the metric implies that if $NMI_2(x, y) \geq 1 - \epsilon$ and $NMI_2(x, z) \geq 1 - \epsilon$, then $NMI_2(y, z) \geq 1 - 2\epsilon$. Then, by the functional relationship, $NMI_1(y, z)$ will be close to 1 too.

In other words, if $NMI(\mathcal{C}_R, \mathcal{C}_1)$ is high and $NMI(\mathcal{C}_R, \mathcal{C}_2)$ is high, then $NMI(\mathcal{C}_1, \mathcal{C}_2)$ is also high. This result is used in Section 5.2, where $\mathcal{C}_R$ is a reference partition used to analyze our algorithm's robustness.

References

[1] A. Lancichinetti, S. Fortunato, J. Kertsz, Detecting the overlapping and hierarchical community structure in complex networks, New Journal of Physics 11 (3) (2009) 033015.
[2] S. Fortunato, Community detection in graphs, Physics Reports 486 (3–5) (2010) 75–174.
[3] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences of the United States of America 99 (2002) 7821.
[4] M. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2) (2004) 026113.
[5] M. Newman, Modularity and community structure in networks, PNAS 103 (23) (2006) 8577–8582.
[6] R. Guimerà, L.A. Nunes Amaral, Functional cartography of complex metabolic networks, Nature 433 (7028) (2005) 895–900.
[7] V. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (1) (2008) 10008.
[8] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proceedings National Academy of Sciences 104 (1) (2007) 36–41.
[9] B.H. Good, Y.-A. de Montjoye, A. Clauset, Performance of modularity maximization in practical contexts, Physical Review E 81 (2010) 046106.

[10] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–818.
[11] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical Review E 78 (2008) 046110.
[12] L. Danon, J. Duch, A. Arenas, A. Díaz-guilera, Comparing community structure identification, Journal of Statistical Mechanics: Theory and Experiment 9008 (2005) 09008.
[13] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proceedings of the National Academy of Sciences 101 (9) (2004) 2658.
[14] D. Lusseau, M.E.J. Newman, Identifying the role that animals play in their social networks, Proceedings of the Royal Society B Biological Sciences 271 (Suppl. 6) (2004) S477–S481.
[15] A. Lancichinetti, S. Fortunato, Consensus clustering in complex networks, Scientific Reports 2 (2012).
[16] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2008, ISBN 3-900051-07-0.
[17] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proceedings of the National Academy of Sciences 105 (4) (2008) 1118–1123.
[18] A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis, Physical Review E 80 (2009) 056117.
[19] W.W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropological Research 33 (1977) 452–473.
[20] R. Guimerà, L. Danon, D.A. Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Physical Review E 68 (6) (2003) 065103.
[21] P. Erdös, A. Rényi, On random graphs I, Publicationes Mathematicae Debrecen 6 (1959) 290–297.
[22] Cornell KDD Cup, 2003. http://www.cs.cornell.edu/projects/kddcup/ (Online; accessed 23.07.2012).
[23] R. Albert, H. Jeong, A.-L. Barabasi, The diameter of the world wide web, Nature 401 (1999) 130–131.
[24] A. Lancichinetti, S. Fortunato, Limits of modularity maximization in community detection, Physical Review E 84 (2011) 066122.
[25] C. Song, S. Havlin, H.A. Makse, Self-similarity of complex networks, Nature 433 (7024) (2005) 392–395.
[26] CommUGP: Local community detection algorithm based on a uniform fitness growth process, 2012. http://code.google.com/p/commugp/.
[27] M.G. Beiró, J.I. Alvarez-Hamelin, J.R. Busch, A low complexity visualization tool that helps to perform complex systems analysis, New Journal of Physics 10 (12) (2008) 125003.
[28] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary? in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML' 09, ACM, New York, NY, USA, 2009, pp. 1073–1080.