# Clustering algorithm for determining community structure in large networks

Josep M. Pujol,* Javier Béjar, and Jordi Delgado

*Software Department, Technical University of Catalonia, Jordi Girona 1-3 A0-S106, 08034, Barcelona, Spain*

We propose an algorithm to find the community structure in complex networks based on the combination of spectral analysis and modularity optimization. The clustering produced by our algorithm is as accurate as the best algorithms on the literature of modularity optimization; however, the main asset of the algorithm is its efficiency. The best match for our algorithm is Newman's fast algorithm, which is the reference algorithm for clustering in large networks due to its efficiency. When both algorithms are compared, our algorithm outperforms the fast algorithm both in efficiency and accuracy of the clustering, in terms of modularity. Thus, the results suggest that the proposed algorithm is a good choice to analyze the community structure of medium and large networks in the range of tens and hundreds of thousand vertices.

## I. INTRODUCTION

Clustering plays a key role in the analysis and exploration of data. In short, clustering is the method by which meaningful clusters, or groups, within collections of data are created. These clusters are intended to group individuals—or samples—who are similar to each other so that the hidden structure within the collection of data is revealed, resulting in a valuable acquisition of knowledge. Data mining and machine learning are disciplines that extensively work with clustering, in particular, with data sets composed by individuals and attributes. The aim is to identify groups of individuals that are similar based on their attributes. However, thanks to the recent collective effort on analyzing and compiling very large networks, there is a growing interest in methods based on the structure—topology—of the networks rather than on the individuals' attributes.

This method for clustering is possible thanks to the characterization of many systems as networks. Despite the very different nature of modeled systems (the Web [1], sexual relations [2], scientific collaboration [3,4], protein interactions [5], the Internet [6], phone calls [7]), they do exhibit a nontrivial pattern of interactions. One of the regularities found in complex networks [8,9] is the high cliquishness of the network [10], which leads to the fact that there are groups of vertices that are very interconnected among them with few interactions outside the groups. Therefore, there is an implicit community structure within complex networks.

Girvan and Newman [11] proposed an algorithm to extract the community structure from complex networks that has become one of the most used among the researchers in this community. From that important work, a branch of research on complex networks has turned into clustering algorithms able to discover the community structure in those networks. To evaluate the accuracy—or quality—of a community structure yielded by a clustering algorithm Newman and Girvan devised a quantitative measure called *modularity Q*. Although there are other quantitative measures [12], modularity is widely accepted in the physics community. $Q$ is defined in [13] as

$$Q = \sum_i (e_{ii} - a_i^2). \tag{1}$$

Modularity is the addition of the modularity of all the groups, $Q = \Sigma_i q_i$. Thus, for each group $i$ that contains $k$ vertices, the modularity is calculated as the fraction of edges that have both ends pointing at vertices in group $i$, $e_{ii}$. The fraction of *intragroup* edges is confronted with the fraction of edges of that group, $a_i$, which are edges whose end points belong to at least one of vertices in $i$. This successful measure has been adopted not only to bench-mark the accuracy of the clustering but also as the *fitness* value for clustering algorithms based on optimization. Finding the partition of groups that maximizes $Q$ is believed to be a *NP-hard* problem, which makes a brute force exploration impossible for networks bigger than dozens of vertices. However, several search heuristics can be applied to explore the huge space of states to find a good partition. Following this approach, many algorithms have investigated different exploration heuristics to find the community structure while maximizing $Q$. Newman proposed in [14] a hill-climbing heuristic to create the hierarchy following an agglomerative strategy. The baseline is that every single node is a cluster, then the pair of clusters whose union produces the biggest increment in $Q$ are merged into one. The process is repeated until only one cluster remains. By following the merging operations, the hierarchy that reveals the community structure is built. However, a *hill-climbing* heuristic cannot escape a suboptimal maximum. Therefore, other search heuristics were devised. For instance, Guimerà and Amaral [15] proposed a simulated annealing approach. Duch and Arenas [16] proposed an algorithm based on extremal optimization. Both algorithms were able to extract the community structure more accurately in terms of modularity, although they were not as efficient as Newman's fast algorithm [14]. Newman has very recently proposed another clustering algorithm [17] that outperforms the previous algorithms in both modularity and efficiency, although it is not as efficient as his previous fast algorithm [14]. Danon *et al.* [18] have also very recently presented a modification of Newman's fast algorithm that while maintaining its computation efficiency yields more accurate partitions in terms of modularity.

Modularity optimization methods are neither the first nor the only ones to work on clustering in complex networks. In

---

*Electronic address: jmpujol@lsi.upc.edu

[11], Girvan and Newman reviewed classical hierarchical clustering algorithms on networks, showing that some classical distance measures were not well suited to work with complex networks. Although the review done by Girvan and Newman in [11] was essentially correct, it overlooked two relevant areas that were already working in clustering of complex networks. Sociology was addressing clustering in social network analysis [19]. On the other hand, Computer Science was also working on clustering of a particular instance of complex networks: the Web. Gibson *et al.* [20] and Kumar *et al.* [21] addressed clustering based on the analysis of the links between Web pages. A common tool used to address clustering on the Web is spectral analysis. However, this technique is applicable to any kind of network, for instance, newsgroups [22] and protein networks [23]. Spectral analysis has also been used in many other areas besides clustering. For instance, in workload distribution between processors [24] and to find the relevant vertices of a network [25,26]. Obviously, not all clustering is limited to spectral analysis. Flake *et al.* [27] proposed an alternative approach based on minimum cut-trees over expanding networks that worked over the Web and could be applied to other kinds of networks. However, we find particularly interesting clustering based on random walks [28,29], which can be seen as a particular case of the spectral analysis. The underlying idea behind clustering using random walks is very intuitive: if a random walker starts in a given node, it will tend to visit more often vertices that belong to the same community of the initial node. Thus, provided that there is community structure, a random walker will spend most of the time stuck within the community from which it started.

Our algorithm is a combination of spectral analysis and modularity optimization in order to achieve a good compromise between efficiency and accuracy of the cluster. Spectral analysis is used to reduce the number of initial vertices of the network: by means of a set of random walkers we create an initial partition of the network in a number of groups much smaller than the initial number of vertices. Consequently, the number of merge operations required to build up the hierarchy is reduced. Asymptotically, our algorithm has a complexity $O(n^2)$, which is the same complexity of Newman's fast algorithm [14]. However, in terms of computational cost it is more efficient since the complexity can be decomposed as $O(ns) + O(s^2)$, where $n$ is the number of vertices and $s$ is the number of groups in the initial partition produced by the random walkers. Despite $s$ being smaller than $n$, it is not upper-bounded by a sublinear function of $n$, so that the complexity remains $O(n^2)$. Yet, it is clearly more efficient and allow us to analyze very large networks in reasonable time while maintaining high-quality clustering.

## II. ALGORITHM

The proposed algorithm, henceforth PBD (after the initials of the authors), consists of an agglomerative hierarchical clustering where the initial groups are those produced by an initial partition of the network. The first step of the algorithm consists of a process of $s$ random walkers traversing the network. The transition probability matrix $M$ is defined as

$$M = (A + I)D^{-1}, \tag{2}$$

where $I$ is the identity matrix and $D$ is a diagonal matrix of the form $D_{ii} = 1 + \Sigma_j A_{ij}$. Thus, $M_{ij}$ is the probability to go to node $j$ from node $i$. The process carried out by the random walkers is defined by

$$G^{t+1} = M'G^t, \tag{3}$$

where $G^t$ is the matrix that contains the probability distribution of each random walker, $G_{ij}^t$ is the probability that the random walker $j$ is at node $i$ at time $t$. Usually, the process is repeated iteratively until the stationary state is reached. However, we are interested in the transient state for all random walkers; consequently, the process is repeated until we obtain $G^T$, where $T$ is set to 3. Therefore, each random walker has done three jumps, which is the minimum number of hops to complete the shortest path to the origin point. Once the stochastic process is finished, each node $i$ is classified into the group $j$, which corresponds to the largest column at row $i$ in $G$. Through this process, the initial $n$ vertices are classified in approximately $s$ groups and all the vertices of the same groups share that they were visited the most by the same random walker. Consequently, this means that they have a high degree of neighbors in common, which implies a community. Although this method is far from perfect, it allows us to drastically reduce the initial number of groups. The final number of groups might not correspond exactly to $s$ since random walkers could preclude others. A random walker $i$ is precluded when all vertices by a random walker $i$ are also visited more often by other random walkers; consequently, the visited nodes are classified into others groups rather than the group started by $i$. Furthermore, since the Markov process is only iterated $T$ times, there is no guarantee that all vertices will be visited at least once, and in this case an extra group with a single node is created.

The partition of the network heavily depends on which vertices are seeds—origins—of the random walkers. This problem is very much related to classical clustering algorithms [30,31] such as *k-means* [32]. How many seeds are required and where to place them is an open question [33]. We propose a straightforward heuristic that selects which vertices will be the seeds for the random walkers, i.e., to define $G^0$. Let $R$ be the fraction of the most connected vertices chosen as seeds. If $k_i \geqslant z$, a random walker will start at node $i$, where the connectivity threshold $z$ is defined as the maximum connectivity that makes the partition composed of the most connected nodes larger or equal to $R$, $\Sigma_{j=z}^{j \leqslant \max(k)} p(k_j) \leqslant R$, where $p(k_j)$ is the fraction of vertices with connectivity $k_j$. The parameter $R$ allows us to decide approximately the number of seeds, although the initial number depends on the structure of the network. For $R = 1$, there would be too many seeds for the algorithm to be efficient. On the contrary, $R \sim 0$ would be very efficient, but the partition would be ill-constructed. In our experiment, we set $R$ to $\frac{1}{5}$ and obtain good results for a wide variety of networks, as shown in Tables I and II. Future work will look into different heuristics to choose the seeds, the quality-efficiency trade-off of our algorithm is really dependant on this process, and other heu-

TABLE I. Comparison between maximum modularity $Q$ and number of communities $g$ obtained by Newman's fast algorithm (N), Duch and Arenas extremal optimization algorithm (EO), and the PBD algorithm.

| Network | size ($n$) | $Q_N$ | $g_N$ | $Q_{EO}$ | $g_{EO}$ | $Q_{PBD}$ | $g_{PBD}$ |
|---|---|---|---|---|---|---|---|
| Zachary | 34 | 0.3807 | 3 | 0.4176 | 4 | 0.3937 | 4 |
| LSI | 139 | 0.6428 | 6 | 0.6572 | 7 | 0.6604 | 6 |
| *C. Elegans* | 453 | 0.40 | 10 | 0.4376 | 10 | 0.4164 | 7 |
| Directors Board | 598 | 0.8046 | 21 | 0.8113 | 27 | 0.8273 | 16 |
| Scientometrics | 2678 | 0.5555 | 24 | 0.6042 | 19 | 0.5629 | 10 |
| Erdös (2002) | 6927 | 0.6723 | 57 | 0.6520 | 88 | 0.6817 | 20 |
| Cond-Mat | 27519 | 0.6653 | 324 | 0.6790 | 647 | 0.7251 | 44 |
| Word-Net | 75606 | 0.7963 | 453 | N/A | N/A | 0.7885 | 47 |
| WWE ND | 325729 | 0.9273 | 2192 | N/A | N/A | 0.9272 | 83 |
| Actors ND | 498925 | 0.7243 | 2113 | N/A | N/A | 0.7297 | 14 |

ristics more elaborate might provide better results than our current straightforward selection rule.

The complexity of finding the seeds for the random walkers is $O(n)$. The connectivity distribution and the connectivity threshold $z$ can be computed in linear time respect to the number of vertices. In fact, the first approach we tried was to get the $nR$ most connected nodes, which would entail a sort operation with cost $O(n \log n)$. This option was discarded in favor of the connectivity threshold due to the extra cost, which our algorithm intends to minimize.

The stochastic process defined in Eq. (3) has the iterative multiplication of two matrices, $M$ and $G$, of dimension $n \times n$ and $n \times s$, respectively. However, thanks to the sparseness of both networks the cost can be reduced from $O(n^2 s)$ to $O(ms)$, where $m$ is the number of edges. For each random walker $j$, its probability distribution can be calculated in the worst-case scenario with cost $O(m)$. Thus, the final cost can be considered $O(ns)$ because the number of edges scales with $n$ in the limit of large $n$.

Once the initial partition is created, the algorithm builds an agglomerative hierarchical clustering. This method consists of creating a series of partitions of the data:

TABLE II. Comparison between CPU time $t$ (in seconds) between the Newman's fact algorithm and the PBD algorithm. It also includes the number of random walkers required to create the initial partition $s_{PBD}$.

| Network | Size ($n$) | $t_N$ | $t_{PBD}$ | $s_{PBD}$ |
|---|---|---|---|---|
| Zachary | 34 | 0.002 | 0.014 | 16 |
| LSI | 139 | 0.003 | 0.015 | 42 |
| C. Elegans | 453 | 0.026 | 0.064 | 118 |
| Directors Board | 598 | 0.038 | 0.031 | 125 |
| Scientometrics | 2678 | 1.6 | 0.320 | 619 |
| Erdös (2002) | 6927 | 3.14 | 2.6 | 2155 |
| Cond-Mat | 27 519 | 125.8 | 11.2 | 6224 |
| Work-Net | 75 606 | 490.6 | 204.1 | 38 701 |
| WWW ND | 325 729 | 10 932.1 | 1775.6 | 86 908 |
| Actors ND | 498 925 | 34 208.3 | 3326.3 | 118 897 |

$C_s, C_{s-1}, \ldots, C_1$, where first $C_s$ consist of $s$ single clusters (groups), and the last $C_1$, consists of a single group containing all the individuals. The method iteratively joins the two individuals or clusters (groups of individuals) that are most similar. Thus, after $s-1$ join operations, the clustering is complete and the result is a binary tree known as *dendrogram*, which reveals the underlying structure of the data.

Let us say that the initial partition yielded $s$ groups, despite the fact that it is an upper bound, since some groups might be empty because their random walkers were precluded by others. For each group $j$, the contribution to the total modularity; that is $q_j = e_{jj} - aj^2$ can be calculated in linear time $O(s)$. The group that contributes the least to the total modularity $Q$—let us say $j$ such that $j = \mathrm{argmin}_k(q_k)$—is selected to be joined to the group that maximizes the increment of modularity as defined in the following equation:

$$\Delta Q = (2e_{ij} + e_{ii} + e_{jj}) - (a_i + a_j)^2 - (e_{ii} - a_i^2) - (e_{jj} - a_j^2).$$

$$(4)$$

The increment in total modularity is the modularity of the merged group $(2e_{ij} + e_{ii} + e_{jj}) - (a_i + a_j)^2$ minus the contribution to the modularity of both groups, $q_i$ and $q_j$. Equation (4) can be reduced trivially to Eq. (2) of Newman's fast algorithm [14]

$$\Delta Q = 2e_{ij} - 2a_i a_j = 2(e_{ij} - a_i a_j). \quad (5)$$

In the event that two candidates, $i_1$ and $i_2$, have the same effect over the total modularity, the candidate group chosen will be the one that has the least modularity, $\min(q_{i_1}, q_{i_2})$. Thus, groups with low modularity are preferred in the merge operation.

The merge operation can be performed in the worst-case scenario in linear time with respect to the current number of groups, thus $O(s)$. Furthermore, the operation needs to be done $s-1$ times. Therefore, the complexity of building up the hierarchy is $O(s^2)$. The search heuristic proposed is extremely *greedy* since it only takes into consideration pairs of groups, provided that one groups is fixed. Conversely, Newman's fast algorithm calculates the gain of modularity for each possible pair of groups. Besides that, other algorithms
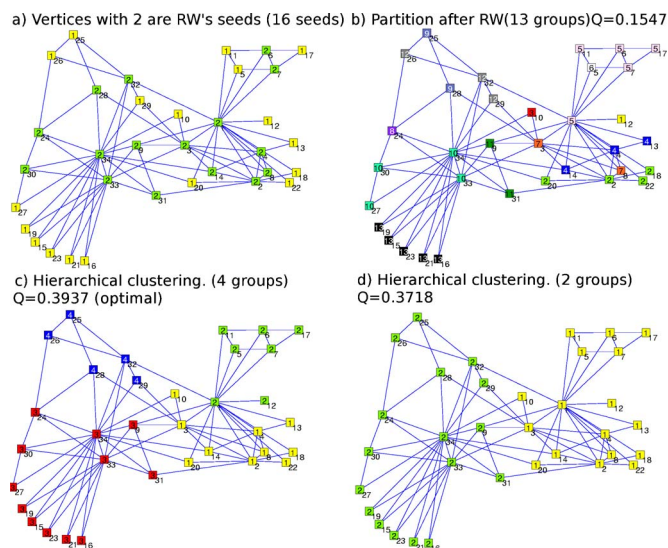
FIG. 1. (Color online) Working of the PDB algorithm in the Zachary network. (a) shows the initial vertices (labeled with 2), which are seeds for the random walkers vertices labeled. (b) shows the initial partition of the network into communities produced by the random walker stage. (c) and (d) show different partitions created by the modularity optimization process. The optimal partition, whose modularity is maximal, is shown in (c).

based on modularity optimization usually have even more expensive search heuristics that allow a better exploration at the expense of efficiency. Our proposal was designed to focus on efficiency, as it can be seen in the heuristic decisions made by the algorithm. However, as we will show in the experiments section, this focus on efficiency does not necessarily imply a loss of quality of the clustering.

### A. Parallelization

To reduce even further the execution time of the algorithm its parallelization could be easily implemented. The stochastic process as defined in Eq. (3) can be carried out in parallel by different computers or processors. To calculate the probability distribution of a given set of random walkers, only the transition probability matrix is required. Thus, the matrix $G$ of dimension $n \times s$ could be split column by column into a set of smaller matrices of dimension $n \times \varsigma$, where $\varsigma \ll s$. This would drastically reduce the cost of the stochastic process. Unfortunately, the modularity optimization step cannot be parallelized so easily. Thus, trivial parallelization would only affect the spectral analysis part of the algorithm. Although this part is the most expensive part in the algorithm $O(ns)$, the modurality optimization $O(s^2)$ would still be executed sequentially. Therefore, the asymptotic cost of the PBD algorithm would still be $O(n^2)$.

### B. Working example with Zachary's network

To illustrate our algorithm, we include an execution on the Zachary network [34], which is a well-known data set in the literature of community extraction. In Fig. 1, we can find the network at the different stages of the execution of the
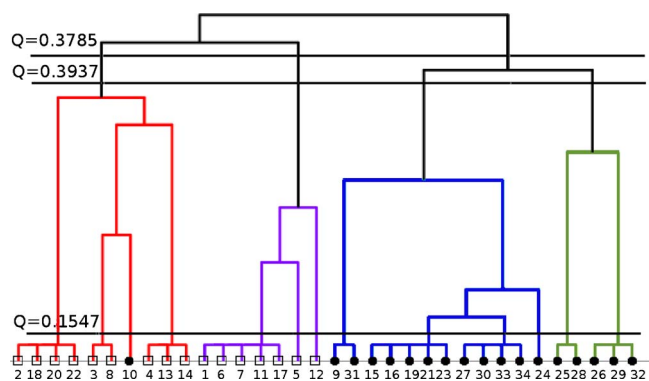


FIG. 2. (Color online) Community structure of the Zachary network produced by the PDB algorithm. Circles and squares over the individuals denote to whom they align to after the karate club broke up. Those who aligned with the instructor are represented by circles, and those who aligned with the administrator are represented by squares.

algorithm. Figure 1(a) shows which vertices, labeled 2, are seeds of the random walkers, 16 in total. Thus, $s$ is 16 compared to the original 34 vertices. In Table II, the relation between network size $n$ and the number of random walkers $s$ for a wide range of networks is shown. Figure 1(b) shows the initial partition of the networks produced by the random walker process, described in Eq. (3). The initial 34 vertices are grouped into 13 groups. This partition has a modularity $Q$ of 0.1547. From that point on the algorithm starts the modularity optimization stage governed by Eq. (4). At each step, two of the remaining groups are joined according to Eq. (4). Figures 1(c) and 1(d) show the network divided into four and two groups, respectively.

The community structure of Zachary's is better seen in Fig. 2. The maximum modularity is obtained by the partition in 4 communities, achieving $Q=0.3937$. However, the original empirical work on the Zachary Karate Club [34] found two communities: those aligned with the instructor and those aligned with the administrator. The division into two groups produced by the PBD algorithm produces a high modularity $Q=0.3718$. Also, the two original communities found empirically correspond to the communities found by the algorithm with the exception of node 10, which is misclassified.

It is evident that Fig. 2 is not a dendrogram over all the vertices of the network, but a dendrogram over the initial communities. As a consequence of the random walker process, the structure between vertices belonging to the same initial community remains unknown. However, this loss is negligible since the relevant high-level structure is not affected as can be seen in Fig. 2.

### III. EXPERIMENTS

In order to further analyze our algorithm, we chose a set of ten different networks of different sizes, ranging from 34 to 498 925 vertices. The networks modeled a wide spectrum of systems. There are social networks, such as the Zachary Karate Club [34] and the social network of the Software Department (LSI) at the Technical University of Catalonia

[35]; scientific collaboration networks, such as Cond-mat [4] and the Erdös collaboration network [36]; citation networks, such as Scientometrics [37]; and affiliation network among Spanish top director boards [38]; a network of relations between words, such as WordNet [39]; metabolic networks, such as the *C. Elegans* [40]; a portion of the Web from the Notre Dame University data set [1]; and the last type of network was the movie collaboration [41] network, again obtained from the Notre Dame University data set [42]. In all the networks we only worked with the biggest connex component, removing all multiple relations and self-reference edges.

### A. Comparing modularity

Table I summarizes the highest modularity achieved by Newman's fast algorithm ($Q_N$) [14] and our algorithm ($Q_{PBD}$). For eight out of the ten tested networks, the PBD algorithm produces a higher modularity, and the maximum difference in favor of the PBD algorithm is in the cond-mat network. Thus, in general, the PBD algorithm yields a slightly better modularity than the Newman's fast algorithm. However, as mentioned in the Introduction, there exist in the literature other algorithms based on modularity optimization that also outperform the Newman's fast algorithm [14]. In Table I, we also included the results obtained by the extremal optimization algorithm (EO) by Duch and Arenas [16]. In this case the maximum modularity obtained by EO outperforms in two of three cases the modularity obtained using PDB, and in all the available cases it outperforms the modularity obtained using the Newman's fast algorithm. However, the complexity of the algorithms that use elaborated search heuristics is superior to the complexity of both the Newman's Fast algorithm and the PDB algorithm. For instance, EO's complexity is $O(n^2 \log^2 n)$. Thus, we can conclude that PDB has a good balance between efficiency and quality.

### B. Comparing the number of communities

Clustering, though, does not only depend on obtaining the partition that maximizes modularity. The number of communities—or groups—contained in the optimal partition is also very important. In Table I, the number of communities is denoted by $g_N$, $g_{EO}$, $g_{PBD}$. We can clearly observe an order $g_{EO} > g_N > g_{PBD}$. Provided that maximum modularity does not greatly differ between partitions, the huge differences in the number of communities obtained by the different algorithms are indeed striking and requires further study.

Provided we assume that modularity $Q$ is a good measure for community structure, we must take for granted that two partitions with similar modularity are equally accurate. Thinking otherwise would lead us to the conclusion that modularity is not a good measure. Finding a *bogus* partition that yielded higher modularity than a *good* partition would mean that modularity is not representative of the structure. Consequently, it could not be used as the fitness variable for the optimization. However we do not believe that this is the case.

So, if we assume that modularity is a good measure, what happens when two partitions having very similar modularity
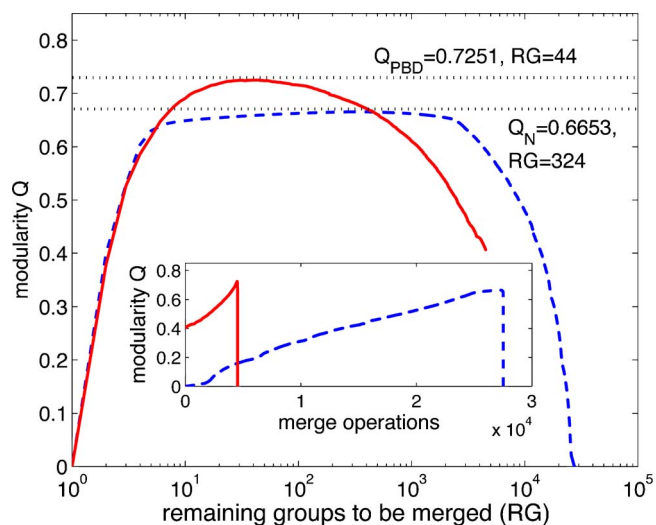


FIG. 3. (Color online) Modularity $Q$ over the execution of the algorithms for the cond-mat network. The PBD algorithm is shown as the solid line and the Newman's fast algorithm (N) as the dashed line. Note that the number of merge operations required for both algorithms is different (PBD optimization stage starts with the partition given by the spectral analysis stage). PBD algorithm requires approximately the 22% of the merge operations performed by the N algorithm. Both the figure and the inset contain the very same information, but it is depicted in a different fashion to better illustrate the evolution of modularity. Note that the main figure is plotted in a log scale, and the time goes from right to left. This is done in order to magnify the last steps of the algorithms, which is when maximum modularity is found. We would like to remark that the number of remaining groups decreases over time, while the number of merge operations increases over time.

have a very different number of groups? A first approach would be to think that the partition with smaller number of groups is more general than the partition with the larger number of groups. Thus, partitions with smaller number of groups would be, in principle, more interesting for several reasons. (i) They would provide a more general perspective on the underlying structure of the network, since they would be able to find a meaningful partition at a higher level of the structure. (ii) A small number of groups would simplify the analysis of the obtained results. And (iii), general or *high-order* partitions could always be reclustered to further analyze the structure of a particular group if a more detailed, fine-grained analysis was required. The opposite could not be done; otherwise, the algorithm would have detected the more general partition with higher modularity.

Table I shows that the PBD algorithm yields more general partitions while having similar or better modularity. This effect is specially acute in large networks. One might conclude from the results of the experiments that both EO and N algorithms undergo an unnecessary over-specialization. For instance, let us we take the optimal partition of the cond-mat network. The PBD provides the highest modularity and the smallest number of partitions, which is 647, 324, and 44 using EO, N, and PBD algorithms, respectively.

Figure 3 shows the evolution of $Q$ in the cond-mat network using N and PDB algorithm. Note that the modularity
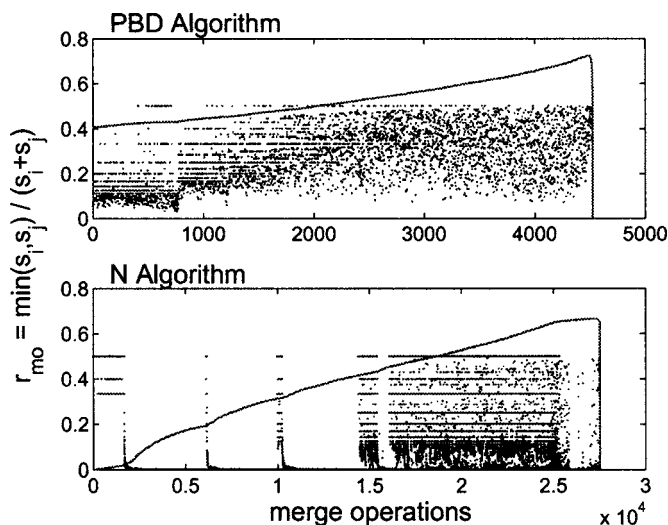
FIG. 4. Behavior of the merge operation of the PBD (above) and the N (below) algorithms. For each merge operation the normalized ratio of the size of groups to be merged is calculated as $r_{mo} = \frac{\min(s_i, s_j)}{s_i + s_j}$, where $s_i$ and $s_j$ are the size of group $i$ and $j$, respectively. The merge operation creates a new group $z = i \cup j$ of size $s_z = s_i + s_j$. The line corresponds to the evolution of modularity $Q$.

obtained by the N algorithm is very close to its maximum modularity when the number of remaining groups is in the range between 40 and 1000. Once the partition contains more than 1000 groups, it does not increase or decrease substantially until the number of groups in the partition reaches 40; at that point, modularity starts to decrease abruptly. This range of stable modularity contains a partition with 44 groups, which is the maximum modularity found by the PBD algorithm. This behavior is not observed in the evolution of modularity for the PBD algorithm, which keeps increasing modularity until it gets to the maximum and then it starts a quick descent. Being on a *plateau* where modularity is neither increasing or decreasing significatively might suggest that the search heuristic of the N algorithm could be stuck in a local suboptimal state.

A more detailed study of the internal behavior of the N and PBD algorithms is described in Fig. 4, where the evolution of the normalized size of groups to be merged is depicted. Each merge operation joins group $i$ and $j$ in a new group $z$. The size of the two groups chosen by the algorithm to be merged is expressed as $r_{mo} = \frac{\min(s_i, s_j)}{s_i + s_j}$. The ratio $r_{mo}$ has values in the range $\left(0 \ldots \frac{1}{2}\right]$, when $r_{mo}$ is close to zero means that there is one group that is clearly bigger than its counterpart. When $r_{mo} = \frac{1}{2}$ both groups have the same size. It is easy to see how the N algorithm tends to join groups of very different size, whereas the PBD algorithm tends to do the opposite, i.e., it prefers groups of similar size. Consequently, the N algorithm produces at the very beginning groups of extremely large size by joining a single vertice to a very large group, when this group cannot accept more vertices a new group is started. This behavior is clearly observed at operation 6000, 10 000, and 15 000, approximately, creating three massive groups of 13 000 vertices and leaving the rest of groups composed mostly by individual vertices. This

might be the reason why the search heuristic fails to increase modularity in the *plateau* of Fig. 3. Approximately from operation 16 000 to 26 000, this behavior is not so evident, where groups of similar same size are often merged. However, the behavior that merges very dissimilar groups appears again between operations 26 000 and 27 000, overlapping the *plateau* of modularity. At this point, big groups cannot be merged obtaining a gain of modularity; therefore, the only possible option left to be explored by the search heuristic is to merge *leftover* groups of very few vertices into the existing big groups. As a consequence, the search heuristic gets stuck in a situation where changes of modularity are minimum and there is no escape from the suboptimal partition. The bias of the N algorithm toward creating very large groups has also been very recently reported by Danon *et al.* [18]. We can see how this behavior is not present in the PBD algorithm. The greedy search heuristic used by the PBD algorithm forces the worst group to be the one going to be merged, and this results in groups having similar sizes. By doing so, the search heuristic has many possible combinations to explore, avoiding getting trapped too early in a local suboptimum as it happened in the N algorithm. That is the reason why the PBD algorithm is able to find in the cond-mat network a higher modularity partition, which also has less groups. As for the EO algorithm, we could not carry out the same analysis, but it is plausible that the resulting large number of groups can be attributed to its divisive clustering strategy.

In order to analyze with more detail the partitions yielded by the N and the PDB algorithms, the distance between the optimal partitions for the cond-mat network are calculated. Gustafsson *et al.* [43] reviewed different distance measures to compare partitions of the same network. Since we want to look into the hypothesis that the partition yielded by the PDB ($P_{PBD}$) is more general than the partition yielded by N algorithm ($P_N$), we chose the $m_{div}$ measure, which is the minimum number of divisions to be applied to partitions $A$ and $B$ to obtain the partition $C$ defined in [44] as

$$C = \bigcup_{i=1}^{|A|} \bigcup_{j=1}^{|B|} (a_i \cap b_j). \tag{6}$$

Partition $C$ is the union of all possible intersections between the groups in $A$ and $B$. We rename $C$ as $P_{N\text{-PBD}}$. The distance between partition $P_{PBD}$ and $P_{N\text{-PBD}}$ is 894, which is the number of divisions to be applied to $P_{PBD}$ in order to obtain $P_{N\text{-PBD}}$. The distance between $P_N$ and $P_{N\text{-PBD}}$ is 614. Thus, the total distance is 1508, which is the sum of both distances. In order to have a baseline comparison for the distance between $P_{PBD}$ and $P_N$, we created a random partition $P_{rand}$ with the same cardinality as $P_{PBD}$. The random partition was replicated 30 times and so was the measurement; the average distance between $P_N$ and $P_{rand}$ was 7166.7 with a standard deviation of 33.65.

### C. Comparing efficiency

After comparing the quality of the clustering produced by the PBD algorithm, we must turn our attention toward its performance. As we already mentioned, all design decisions
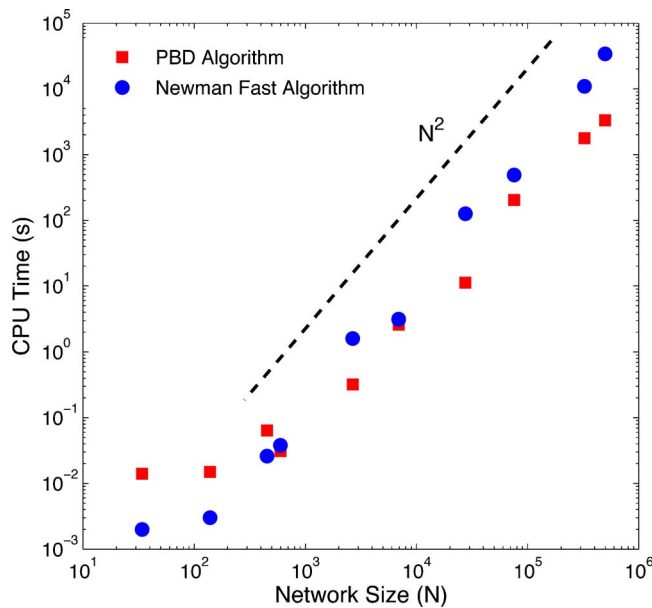
FIG. 5. (Color online) Comparison between the efficiency in CPU time (in seconds) for ten networks of different sizes. The solid circles show the results for the Newman's fast algorithm. The results of the PBD algorithm are represented by solid squares.

were biased toward improving the efficiency so that the algorithm could cope with medium and large networks, which other algorithms cannot handle in reasonable time. Table II summarizes the run time (CPU time) of our algorithm compared to the Newman's fast algorithm, which is the reference algorithm due to its efficiency. We are perfectly aware of the problems related to comparison of algorithms based on run time instead of only considering their complexity. In order allow a fair comparison between both algorithms, we implemented them from scratch optimizing them to the best of our abilities. Needless to say, the runs were executed on the same desktop computer (Pentium 4, 3 GHz) exclusively dedicated to the experiment. Table II shows that PBD is much faster than the Newman's fast algorithm for networks bigger than 1000 vertices. Conversely, the PBD algorithm is slower than the N algorithm for small networks. This is because the two sequential processes—spectral analysis and modularity optimization—that take place in the PBD algorithm. The difference in execution time heavily depends on the number of random walkers ($s_{PBD}$) required for the first stage of the PDB algorithm. The smaller the ratio between network size and the number of random walkers, the faster the PDB algorithm is.

Figure 5 graphically shows the relation between network size and the running time of both algorithms already seen in Table II. The PDB algorithm clearly outperforms the Newman's fast algorithm, often by one order of magnitude. However, the asymptotic quadratic behavior of both algorithms is evident, which lead us to think that our algorithm cannot scale to very large networks of millions of nodes. Resorting to parallelization would allow us to analyze networks of a few million nodes in an acceptable time; however, the PDB will undoubtedly become too slow for very large networks. Nonetheless, it allows to shift the network size threshold far
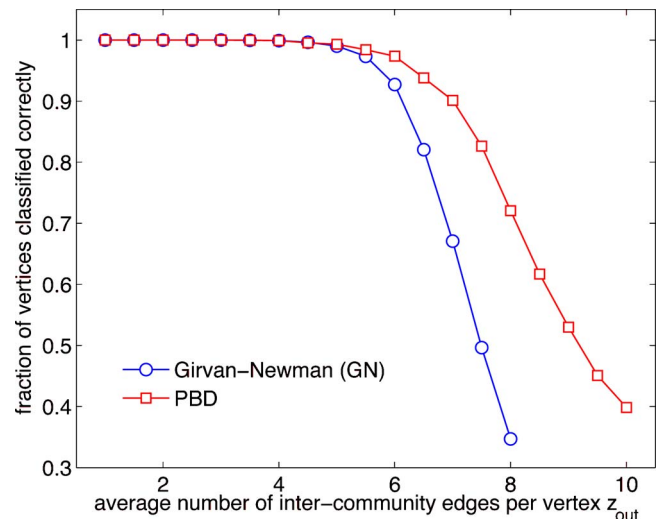


FIG. 6. (Color online) Fraction of vertices correctly classified using computer-generated networks. The circles show the result of the Girvan-Newman algorithm (GN). The squares show the results of the PBD algorithm. Each point is an average over 50 networks.

enough to be useful for medium and large networks. Reduction of approximately one order of magnitude allows one to shift from minutes to seconds or from hours to minutes. For instance, the clustering of the largest network we had access to was reduced from ~9 h to just one.

### Computer-generated networks

To conclude, we include experiments carried out with the computer-generated networks first proposed by Girvan and Newman [11], which have become a common testbed in the field. Those networks are constructed with 128 nodes divided into four groups of the same size. For each node, eight edges are deployed. With probability $P_{in}$ the edge is connected to a node that belongs to the same group chosen at random. Otherwise, the edge is connected to a node that does not belong to the same group. Thus, the average degree of a node is 16. Accordingly to the nomenclature of Girvan-Newman, we will use $z_{out}$ which is the number of *intercommunity* edges per node. It is important to note that for $z_{out}=12$, the network is totally random; that is, without community structure. Another important value for $z_{out}$ is 8, since it marks the boundary between having more *intercommunity* than *intracommunity* edges. The quality measurement is the fraction of vertices that are correctly classified, explained in more detail in [14].

In Fig. 6, we can see the results obtained by our algorithm compared to the results yield by the Girvan-Newman (GN) algorithm based on edge betweenness [11]. We decided to use the GN algorithm instead of the Newman's fast algorithm since it obtains slightly better results, and it is the reference algorithm for this particular experiment [11]. Although the results of GN outperform those obtained by the N algorithm, it is not a suitable option for medium and large networks due to its complexity; that is $O(n^3)$.

The GN algorithm correctly detects the communities until values of $z_{out}=6$ are reached. From this point on, the quality

of the communities decreases very quickly. On the other hand, the PBD algorithm detects the communities very well up to values of $z_{out}=7$; from that point on the performance starts to decay, although the pace is more steady than in the GN case. As already mentioned, other algorithms based on modularity optimization perform much better in the computer-generated networks example. For instance, Duch and Arenas extremal optimization algorithm [16] starts to decline at $z_{out}=8$. However, its increase on clustering quality is done at the expense of efficiency. Thus, it is unadvisable in the case of large networks.

## IV. CONCLUSIONS

In this paper, we have presented an algorithm to extract community structure from networks by using a combination of different existing methods. First, the algorithm uses spectral analysis via the multiple random walker process to reduce the dimensionality of the network by creating the initial partition of the network into communities. Then, a modularity optimization process with a extremely greedy search heuristic is applied to extract the underlying structure of the network.

Experiments show that our algorithm outperforms the Newman's fast algorithm both in clustering quality and efficiency. The Newman's fast algorithm is the reference algorithm in terms of efficiency, and while asymptotically both algorithms are $O(n^2)$, the PDB algorithm is always faster in computation time for medium and large networks, as it has been shown in the experiments.

The reason behind this is the reduction of dimensionality provided by the random walker process, such that the cost of the PBD can be expressed as $O(ns)$ where $s \ll n$. Furthermore, experiments also show that the PDB retrieves more *general* community structure than other algorithms. The number of existing communities in the partition with maximum modularity is notably smaller in the case of the PBD. This fact leads us to think that other algorithms tend to unnecessarily over-specialize their clustering.

In summary, the presented algorithm is an interesting choice when analyzing medium and large networks. The structure of large networks can be found in reasonable time, from seconds for a network of 27 000 vertices to less than one hour for a network of 500 000 vertices. The gain in efficiency does not come with a loss in the quality of the clustering as the maximum modularity obtained by the algorithm is comparable to the reference algorithms in the literature.

[1] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **401**, 130 (1999).

[2] F. Liljeros, C. Edling, L. Amaral, H. Stanley, and Y. Aberg, Nature (London) **411**, 907 (2001).

[3] B. A.-L., H. Jeong, E. Ravasz, Z. Néda, A. Schubert, and T. Vicsek, e-print cond-mat/0104162.

[4] M. Newman, Phys. Rev. E **64**, 016131 (2000).

[5] H. Jeong, S. Mason, Z. Oltvai, and A.-L. Barabási, Nature (London) **411**, 41 (2001).

[6] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **406**, 378 (2000).

[7] W. Aiello, F. Chung, and L. Lu, in *Proceedings of the 32nd ACM Symposium on the Theory of Computing*, edited by F. Yao (ACM, New York, 2000), p. 171.

[8] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).

[9] M. Newman, SIREV **45**, 167 (2003).

[10] D. Watts and S. Strogatz, Nature (London) **393**, 440 (1998).

[11] M. Girvan and M. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).

[12] R. Kannan, S. Vampala, and A. Vetta, in *Foundations of Computer Science* (IEEE Computer Society, Redondo Beach, CA 2000), p. 367.

[13] M. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).

[14] M. Newman, Phys. Rev. E **69**, 066113 (2004).

[15] R. Guimerà and L. Amaral, Nature (London) **433**, 895 (2005).

[16] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).

[17] M. Newman, e-print physics/0602124.

[18] L. Danon, A. Díaz-Guilera, and A. Arenas, e-print physics/0601144.

[19] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge University Press, Cambridge, England, 1994).

[20] D. Gibson, J. Kleinberg, and P. Raghavan, in *Proceedings of the ACM Symposium on Hypertext and Hypermedia* (ACM Press, New York, 1998), p. 225.

[21] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, in *Proceedings of the 8th World Wide Web Conference* (Elsevier Science, Essex, UK, 1999), p. 403.

[22] C. Borgs, J. Chayes, M. Mahdian, and A. Saberi, in *Proceedings of KDD* (ACM Press, New York, 2004), p. 783.

[23] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang *et al.*, Nucleic Acids Res. **91**, 2443 (2003).

[24] H. D. Simon, Comput. Sci. Eng. **2**, 135 (1991).

[25] L. Page, S. Brin, R. Motowani, and T. Winograd, Tech. Report No. 1997-0072, Stanford Digital Library Working Paper (1997).

[26] J. Kleinberg, Tech. Report No. RJ 10076, IBM (1997).

[27] G. Flake, R. Tarjan, and K. Tsioutsiouliklis, Internet Mathematics **1**, 385 (2003).

[28] D. Harel and Y. Koren, Lect. Notes Comput. Sci. **2245**, 18 (2001).

[29] U. Brandes, M. Gaertler, and D. Wagner, Lect. Notes Comput. Sci. **2832**, 568 (2003).

[30] A. Jain and R. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[31] A. Jain, M. Murty, and P. Flynn, ACM Comput. Surv. **31** (1999).

[32] E. Forgy, Biometrics **21**, 768 (1965).

[33] P. Bradley and U. Fayyad, in *Proceedings of the 15th Intl. Conf. on Machine Learning* (Morgan Kaufmann, New York, 1998), p. 91.

[34] W. W. Zachary, J. Anthropol. Res. **33** (1977).

[35] R. Sangüesa and J. Pujol, *Knowledge Management and Organizational Memories* (Kluwer, Dordrecht, 2002).

[36] Erdös Number Project. The network contains scientists with Erdös number less than or equal to 2 up to year 2002. http://www.oakland.edu/enp/thedata.html

[37] Network from Garfield's collection of citation networks. Available at http://www.garfield.library.upenn.edu/histcomp/index.html

[38] Data provided by Fabrizio Ferraro, from the project *Small Worlds of Corporate Networks* at IESE Business School, University of Navarra

[39] Network obtained from the Pajek Network dataset. Available at http://vlado.fmf.uni-lj.si/pub/networks/data/

[40] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A.-L. Barabási, Nature (London) **407**, 651 (2000).

[41] This network is a bipartite network (two-mode). Consequently, vertices have different roles and thus the structure extracted by the algorithms might be biased. We did not transform it to a one-mode network to maintain the maximum of vertices. Thus, the results on this network are only relevant when speaking about the algorithm's efficiency.

[42] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).

[43] M. Gustafsson, M. Hörnquist, and A. Lombardi, Physica A (to be published).

[44] D. Gusfield, Inf. Process. Lett. **3**, 159 (2002).