Applied Mathematics and Computation 217 (2010) 2007–2014

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

A fast and efficient algorithm to identify clusters in networks ☆

Francesc Comellas *,[1], Alicia Miralles

Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Spain

ARTICLE INFO

Keywords:
Graphs
Clusters
Networks
Complex systems

ABSTRACT

A characteristic feature of many relevant real life networks, like the WWW, Internet, transportation and communication networks, or even biological and social networks, is their clustering structure. We discuss in this paper a novel algorithm to identify cluster sets of densely interconnected nodes in a network. The algorithm is based on local information and therefore it is very fast with respect other proposed methods, while it keeps a similar performance in detecting the clusters.

1. Introduction

Many real life networks like the WWW, Internet, transportation and communication networks, or even biological and social networks have a strong clustering structure (they contain groups of vertices which are highly interconnected, having many mutual neighbors). Here we consider the notion of cluster in a general way. Therefore, depending on the context, it can be synonymous of community, class, module, etc. The problem of detecting clusters in a given network is an important issue in social studies, biological (epidemiology, ecological webs, metabolic), and computer science (WWW, Internet, distributed systems, cluster computing). Clusters are also interesting as they reflect hierarchical aspects and are related to classification issues for information retrieval. Clusters also play an important role when executing most communication algorithms and should be considered to improve their performance.

The construction of efficient and fast algorithms for the identification of the clustering structure in a generic network is a nontrivial task. The first problem is the nonexistence of a precise definition of cluster. Intuitively, a network can be said to have cluster structure if it consists of subsets of nodes, with many connections among the same subset, but few links between subsets, see, for example [1,2]. Algorithms to detect these subsets have appeared in the literature and they can be classified in two main groups (see the above two references for more details): hierarchical clustering methods (also known as agglomerative), which consist of generating a tree (dendrogram) from a complete graph with as many vertices as the original network and where each edge has a weight measuring how close the corresponding vertices are. Starting from the set of all vertices with no edges between them, edges are iteratively added between pairs of vertices in the order of

☆ Research supported by the Ministerio de Educación y Ciencia, Spain, and the European Regional Development Fund under project TEC2005-03575 and by the Catalan Research Council under project 2005SGR00256.

* Corresponding author.
E-mail addresses: comellas@ma4.upc.edu (F. Comellas), almirall@ma4.upc.edu (A. Miralles).
URL: http://www.ma4.upc.edu/comellas/ (F. Comellas).
[1] Avda. Canal Olímpic s/n, 08860, Castelldefels, Catalonia, Spain.

0096-3003/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.06.060

decreasing weight. From the tree one can then infer the different clusters. To obtain the weights, some algorithms consider the spectrum of the adjacency matrix of the graph representing the network. The other class of algorithms is called divisive. From the whole graph, by iteratively cutting the edges, one obtains a set of disconnected subgraphs identified as clusters. Of course, the selection of the edges to be cut, i.e. identifying those connecting clusters, is the crucial point of a divisive algorithm. Girvan and Newman (GN) [1] have recently provided a divisive algorithm based on the "edge betweenness": the betweenness of an edge is the number of shortest paths, between all pairs of vertices of a graph, that go through this edge. If a graph is made of dense loosely interconnected clusters clearly all shortest paths between vertices in different clusters will go through a few edges, joining the clusters, which will have a large betweenness value. The main step of the GN algorithm is the computation of the edge betweenness of all the edges and then the removal of those with the highest value. An iterative process allows the obtention of the clusters. This process, however, is computationally expensive as for a graph with n vertices and m edges the cost is $O(nm^2)$, making it impractical even for relatively small graphs.

More recently Newman [3] and Radicchi et al. [2] have provided faster, but less precise, methods. Newman's new method is based on what he calls "modularity" which measures the fraction of edges in a graph connecting different possible clusters and selecting these clusters by using a standard greedy optimization algorithm. The algorithm is $O((n+m)n)$ for the worst case and $O(n^2)$ on a sparse graph. On the other hand Radicchi et al. use the "edge clustering coefficient" defined as the number of triangles to which a given edge belongs, divided by the number of triangles that might potentially include it, given the degrees of the adjacent vertices. This algorithm is $O(m^2)$ in the worst case. A recent survey [4] compares these and other clustering algorithms.

In this paper we introduce a fast and efficient deterministic algorithm which uses local information based on each vertex. The algorithm is also agglomerative and has order $O(mn)$, providing similar results to the GN method and other faster methods.

2. Notation and definitions

We model networks as graphs. Given the graph $G=(V,E)$ with vertex set V and edge set E, we will denote by $\Gamma_k(v)$ the set of vertices at a distance k from a vertex v. Sometimes we will write, for short, $\Gamma(v)$ instead of $\Gamma_1(v)$. Thus, the *degree* of v is $\deg_G(v)=\delta(v):=|\Gamma(v)|$. In terms of the adjacency matrix of G, A_{ij}, we have $\delta(v_i)=\sum_{v_j\in V}A_{ij}=\sum_j A_{ij}$. We denote as Δ the maximum degree of a graph.

If we consider a subset of vertices $C\subset V$ and a vertex $v_i\in C$, we can split the degree of v_i into two contributions (with respect to the subset where it belongs): $\delta_C(v_i)=\delta_C^{in}(v_i)+\delta_C^{out}(v_i)$, where $\delta_C^{in}(v_i)=\sum_{v_j\in C}A_{ij}$ is the number of edges connecting this vertex v_i to other vertices of C and $\delta_C^{out}(v_i)=\sum_{v_j\notin C}A_{ij}$ is the number of edges towards vertices in the rest of the graph.

A cluster is roughly defined as a part of a network where internal connections are denser than external ones. In the implementation of our algorithm we consider a definition of cluster which is a normalized version of the definition of "cluster in the weak sense" as it appears in Radicchi et al. [2]. These authors define *cluster in a weak sense* as a subgraph C such that $\sum_{v_i\in C}\delta_C^{in}(v_i)>\sum_{v_i\in C}\delta_C^{out}(v_i)$. This is in contrast with the definition of *cluster in a strong sense* which for a subgraph C is $\delta_C^{in}(v_i)>\delta_C^{out}(v_i),\quad \forall v_i\in C$. This last definition has also been proposed in [5] for the identification of web clusters.

The problem of finding a partition of the vertex set which corresponds to an optimal clustering structure of the graph and in particular to the identification of this optimality has led to the introduction of modularity by Newman [3]. It has been shown that the problem is NP-hard [6]. Therefore it is of interest to provide algorithms that can produce near-optimal solutions quickly. On the other hand even the modularity method does not work for extremal artificial cases [7], and very often the experimentation with data coming from real networks provides best method to test the usefulness of a given method.

In our algorithm we will use the other definitions: If $C\subset V$, $\deg_C(v_i)$ is the number of edges between v_i and all other vertices in C. Then we have: $\deg_C(v_i)\leqslant\deg_G(v_i)$.The *average degree* of a graph $G(V,E)$ is $\mathrm{avgdeg}(G)=\sum_{v_i\in V}\deg_G(v_i)/|V|$. The *normalized average degree* of a set of vertices C is defined as

$$\mathrm{n\_avgdeg}(C)=\frac{\sum_{v_i\in C}\frac{\deg_C(v_i)}{\deg_G(v_i)}}{|C|}$$

and we have $0\leqslant \mathrm{n\_avgdeg}(C)\leqslant 1$. This parameter is a good measure to characterize if a set of vertices C is clustered. It takes into account, for each vertex in C, the distribution in and out of the set of its edges and also the total number of edges among all vertices in C. Thus, it is a clear improvement over the other cluster characterizations mentioned above. The parameter, in

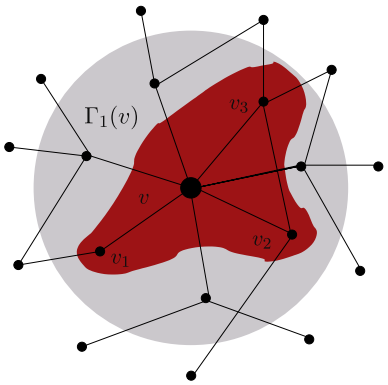

Fig. 1. Formation of a pre-cluster from v and some of its neighbors $\{v_1, v_2, v_3\}$. Pre-cluster $C_{pre} = \{v, v_1, v_2, v_3\}$ is constituted with elements of $\Gamma_1(v)$ (the set of vertices adjacent to v) such that the number of edges between C_{pre} and $\Gamma_1(v)$ is minimum.

fact, computes the edge density of C: a value n_avgdeg(C) = 2/3, for instance, means that on an average, each vertex in C has two-thirds of its edges inside C.

3. A neighborhood based clustering algorithm

The new clustering algorithm which we propose here is a deterministic constructive algorithm. The main aim of our algorithm is to sort out the set of vertices of the graph into clusters that maximize their normalized average degree. For the formation of these clusters we perform a dynamic process which involves the creation of pre-clusters that the algorithm joins to other pre-clusters under some conditions. (To facilitate the reading of this paper, we use in many occasions the word cluster instead of pre-cluster. However the clusters are not totally formed until the algorithm finishes.)

We form a pre-cluster using only local information. The pre-cluster is initialized from a vertex v which has at least half of its neighbors not belonging to clusters already formed (to maximize n_avgdeg(C)). The pre-cluster includes v and neighbors that have more edges with other neighbors of v than the rest of the vertices, see Fig. 1. On the other hand, we assume that this pre-cluster will have more intra-cluster edges if it is started from a vertex with a high degree. The experiments performed show that these assumptions are correct for random graphs and many real life networks.

If other clusters are present from a former iteration of the algorithm, the pre-cluster will be joined with one of them if the number of inter-cluster edges is higher than the expected value number of edges between the two clusters, if all edges were distributed at random in the graph, see Algorithm 1 for details. The general process is as follows:

1. Create a list L with all vertices ordered according to their degree (decreasing order).
2. Check L and look for the first element f of this list such that at least half of its neighbors are also in the list L. We form an auxiliary cluster, C_{pre}, by selecting vertices from the set given by this vertex f and its neighbors in L, and such that at least half of their adjacencies are to vertices in L, and at the same time at least half of the adjacent vertices in L are also adjacent to neighbors of f.
3. Extract from L all vertices which are now in C_{pre}. If this results in an empty list, then there are no new clusters. Go to the last step.
4. Check if C_{pre} can be added to any existing cluster. If this is not possible, and it has more than two vertices, create a new cluster with it. The condition to add the auxiliary cluster to an already existing cluster is the following: the normalized average degree of the union of the two clusters must be at least the same than the normalized average degree for each of the clusters, and the number of adjacencies between them has to be greater than the average for all the vertices of the graph.
5. Repeat from the second step until we obtain an empty list L or isolated vertices.
6. Each remaining vertex is added to an existing cluster.

Algorithm 1 provides a pseudocode version of our algorithm with the details which allow a practical implementation. Clustering algorithm.

Algorithm 1. Clustering algorithm.

Input: $G = (V,E)$ a graph. V, set of vertices; E set of edges.
Output: $C_1, C_2, \ldots C_k$ Subsets of vertices (clusters) partitioning V.
// VERTEX ORDERING.
$L \leftarrow V$; // $n(=|V|)$ *vertices in decreasing degree*
//CLUSTER FORMATION.
$L_{pre} := \emptyset$, $k := 0$; // *Number of clusters*
while $L \neq \emptyset$ **do**
  $C_{pre} := \emptyset$, $L_1 := \emptyset$, $L_2 := \emptyset$
  if $k = 0$ **then** $f := l_1$; //*Start cluster from highest degree vertex.*
  if $k > 0$ **then**
    //*Select vertex to start a new cluster.*
    for i from 1 to $|L|$ **do**
      if l_i *has more than* $\frac{|\Gamma(l_i)|}{2}$ *adjacent vertices in* L **then** $f := l_i$;
      else $f := l_1$
  $L_1 \leftarrow \{f, \Gamma_1(f)\}$
  $L_2 \leftarrow \{\Gamma_2(f)\}$
  foreach $v_i \in L_1$ **do**
    if v_i *verifies* $\delta^{in}_{L_1}(v_i) \geqslant \delta^{in}_{L_2}(v_i)$ *and* $\delta^{in}_{L_1 \cup L_2}(v_i) \geqslant \frac{\delta(v_i)}{2}$ **then**
      $C_{pre} \leftarrow v_i$ //*Add the vertex to* C_{pre}.
  if $C_{pre} = \emptyset$ **then break** // *Exit from the* **while** *loop*
  if $k = 0$ **then**
    $C_1 := C_{pre}$
    $L := L \backslash C_{pre}$ //*Remove from L the vertices in* C_{pre}.
    $k := 1$
  else
    for i *from* 1 *to* k **do**
      if (n_avgdeg($C_{pre} \cup C_i$) $\geqslant$ n_avgdeg(C_{pre}) **and**
      n_avgdeg($C_{pre} \cup C_i$) $\geqslant$ n_avgdeg(C_i) **and** the number of edges between
      C_{pre} and C_i is greater than $|C_{pre}| \cdot |C_i| \cdot \text{avgdeg}(G)/(|V| - 1)$) **then**
        $C_i = C_i \cup C_{pre}$ //*Add vertices of* C_{pre} *to* C_i.
        $L := L \backslash C_{pre}$
      else
        if $|C_{pre}| > 2$ **then**
          $C_{k+1} := C_{pre}$ //*Form new cluster with vertices of* C_{pre}.
          $L := L \backslash C_{pre}$
          $k := k + 1$
        else
          $L_{pre} := L_{pre} \cup C_{pre}$
          $L := L \backslash C_{pre}$
//REDISTRIBUTION OF THE REMAINING VERTICES
foreach $v \in L \cup L_{pre}$ **do**
  v is added to the cluster that has more adjacencies to it.
  If several clusters exist, it is added to the first according to the creation order.

4. Results

4.1. Computer-generated clustered networks

To test our algorithm we first use the same method as in the Girvan and Newman paper [1]: We generate a number of random graphs with a given cluster structure and then we run the algorithm to check its performance. We have performed two sets of tests. In the first set, as in [1], each graph has 128 vertices divided into four groups of 32. The second set also considers graphs with 128 vertices but divided into fours groups with 64, 32, 16 and 16 vertices, respectively. Each vertex on an average has δ^{in} edges connecting it to members of the same group and δ^{out} edges to members of other groups, with δ^{in} and δ^{out} chosen such that the total expected degree is 16. The performance results from the average count of the vertices that are misclassified for different values of δ^{in} (40 instances) (See Figs. 2,3).

We see that our algorithm performs similar to the Girvan and Newman method, correctly identifying more than 90% of vertices for values of δ^{in} greater than 6. When δ^{in} approaches the value 8, i.e. when the average number of intra- and

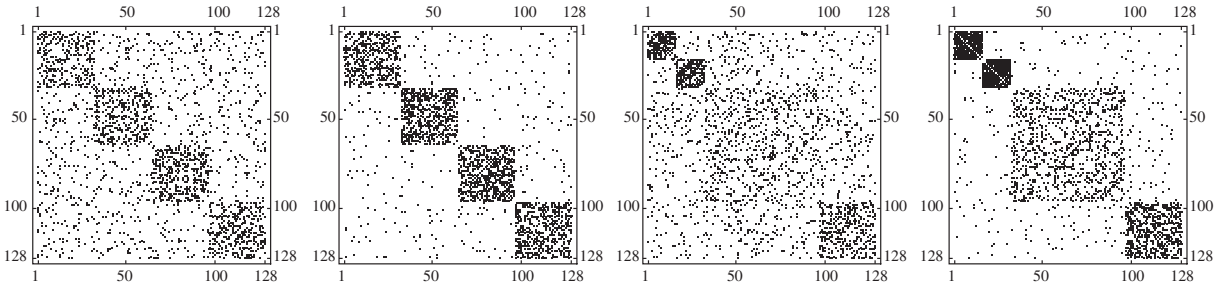

Fig. 2. Adjacency matrices of sample random graphs. The two left matrices correspond to graphs with four clusters of 32 vertices and δ^{in} 10 and 14, and the two other matrices to graphs with four clusters of size 16, 16, 64 and 32 and δ^{in} 10 and 14.

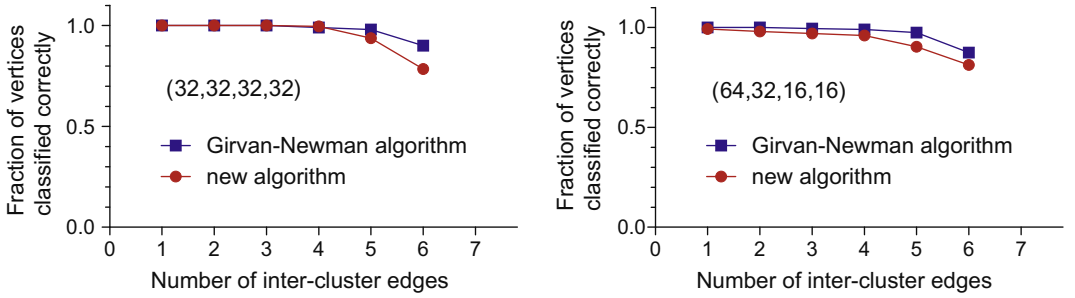

Fig. 3. Fraction of vertices correctly identified by our algorithm considering computer-generated clustered graphs of order 128 and average degree 16 (left: (32,32,32,32), right: (64,32,16,16). The values obtained with the new algorithm are represented by circles and those of the algorithm of Girvan and Newman [1] by squares. Each data point is an average over 40 graphs.

inter-cluster edges per vertex is the same, the algorithm fails, as it also happens for all the clustering algorithms described in the literature.

We have considered relatively small graphs for this comparative test as the running time for the GN algorithm increases very quickly with the order of the graph. Our algorithm, on the contrary, can classify correctly very large graphs (we have tested graphs with ten thousand vertices and several hundred thousand edges in minutes, when the GN method is totally impractical in the same conditions). At the end of this section we discuss the running complexity of our algorithm.

4.2. Real-life networks

Computer-generated clustered networks are of interest as we know beforehand the number of clusters and their characteristics. These graphs are very useful to check the limitations of a clustering method. However the interest of introducing a new clustering algorithm lies on its application to find cluster in real data. Therefore we have used our algorithm with well-known benchmarks clustered networks: the Zachary karate club network, a scientific collaboration network, a dolphins network and the network of characters in the novel from Victor Hugo "Les Misérables".

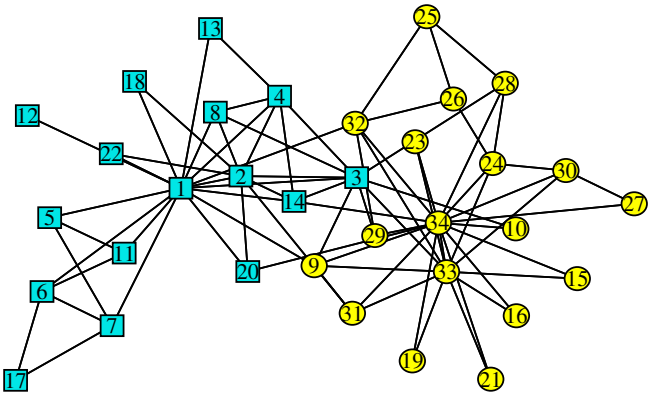

Fig. 4. Clusters found by the algorithm described in this paper for the friendship network associated to Zachary's karate club [8]. Our algorithm finds exactly the same partition as in the original paper.

4.2.1. Zachary karate club

This has become a classical test as it has been considered in many studies on clustering methods. The graph was constructed by Wayne W. Zachary [8] after collecting data on the social relations between the members of a karate club. In Fig. 4 we show the clusters obtained with our method, which correspond exactly to the split of the club after a dispute between the club's administrator and the karate instructor.

4.2.2. Scientific collaborations network

This example considers the collaboration network of scientists who perform research on networks (as reflected by their publications). The graph was generated by Mark Newman using names drawn from the large bibliography of his interesting review on complex networks [9] and coauthorship data from preprints submitted to the condensed matter section of Archive at `arxiv.org` between January 1, 1995 and April 30, 2003. Fig. 5 shows only the largest connected component of his graph with 142 vertices. The clusters found with our algorithm have only minor differences from those found with the GN method in [1,10].

4.2.3. Dolphins network

In this example we study a community of 62 bottlenose dolphins living in Doubtful Sound, New Zealand. This network was compiled by Lusseau [11] from statistical observations of associations along many years. He reported a split into two groups which can be identified from the results of our algorithm (Fig. 6). See [10] for more details.

4.2.4. Les Misérables network

This network reflects the interactions between major characters in the novel Les Misérables, by Victor Hugo, considering their appearances by scene as compiled by Knuth [12]. An edge in the graph represents the simultaneous appearance of the corresponding characters in one or more chapters of the novel. The clusters obtained agree (with minor modifications, see Fig. 7) with those obtained with other methods, see for example [13].

4.3. Algorithm complexity

To analyze the running time of our algorithm we should notice that it relies on local neighborhood information, exploring adjacent vertices up to distance two, while the GN algorithm uses global information derived from the betweenness factor and involves, at each step, computing the shortest path between all pair of vertices. With respect to other local algorithms, the consideration of a neighborhood up to distance two, allows a much better performance.

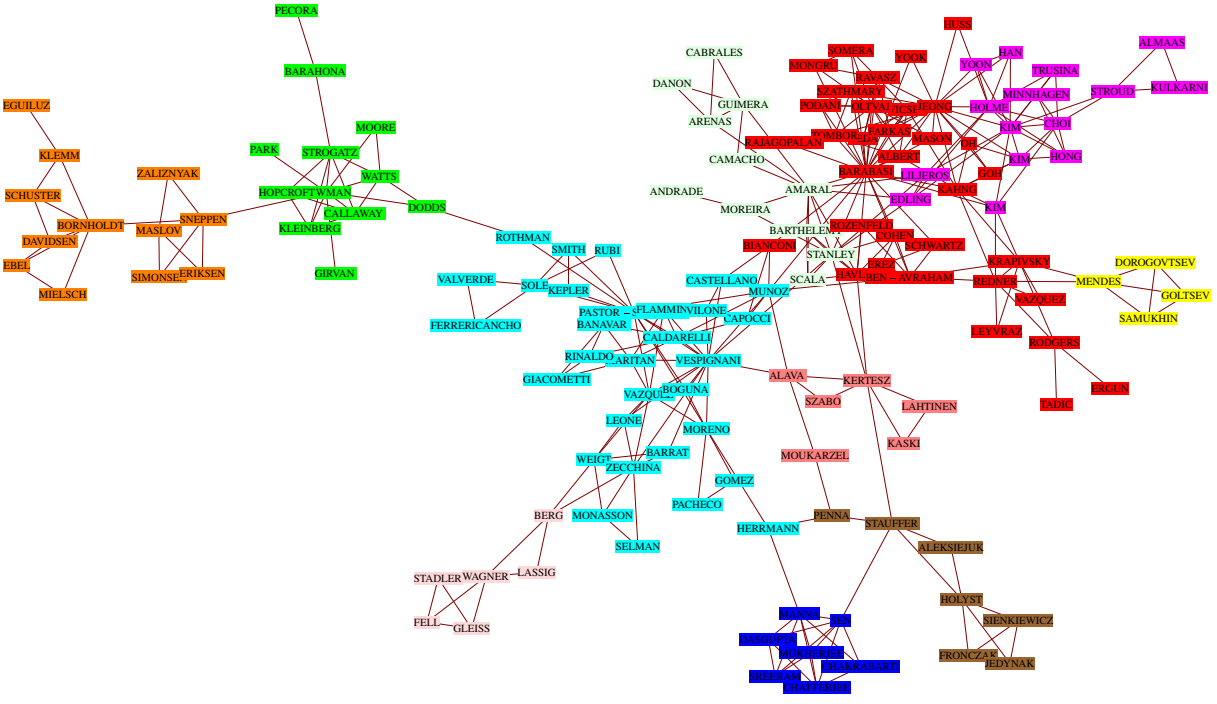

Fig. 5. Largest connected component of the network of coauthorships between scientists publishing on networks (data provided by M. Newman [10]). This component contains 142 scientists. We have colored the vertices according to the clusters found using our algorithm. The resulting clusters are close to those found with the GN method [1].

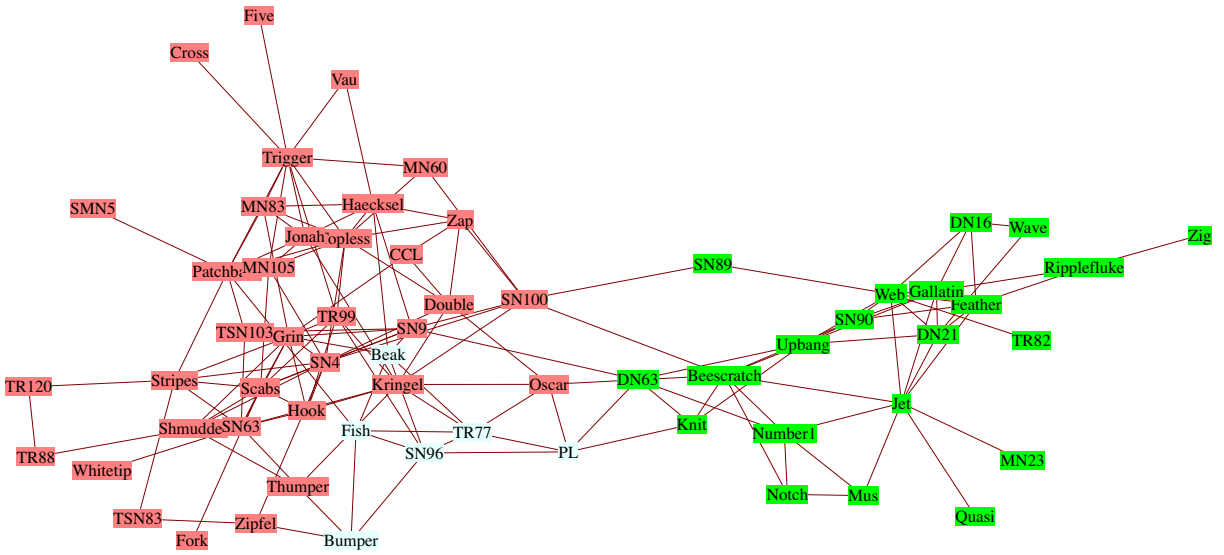

Fig. 6. Cluster structure in the bottlenose dolphins of Doubtful Sound [11], found with our algorithm. We find two large communities and a small one (6 individuals, lighter color). The split of the network into two groups observed by Lusseau matches this division if we add the small group to the largest one. Clustering algorithms by other authors produce more subdivisions in the large group. [10].

From the pseudocode we see that the new algorithm is a fast method, $O(mn)$, suitable for very large networks and not difficult to implement. Note than the algorithm involves exploring the 2-neighborhood of all the vertices of the graph (taking for each, on average, Δ^2 steps), but this action has to be performed only once and therefore the calculation is $O(n\Delta^2) \approx O(mn)$, at most.

5. Conclusion

The detection and study of the cluster structure of very large networks is an area of increasing interest due of the expanding importance of networks like the WWW, Internet, transportation systems, ad-hoc networks, etc. Although the concept of

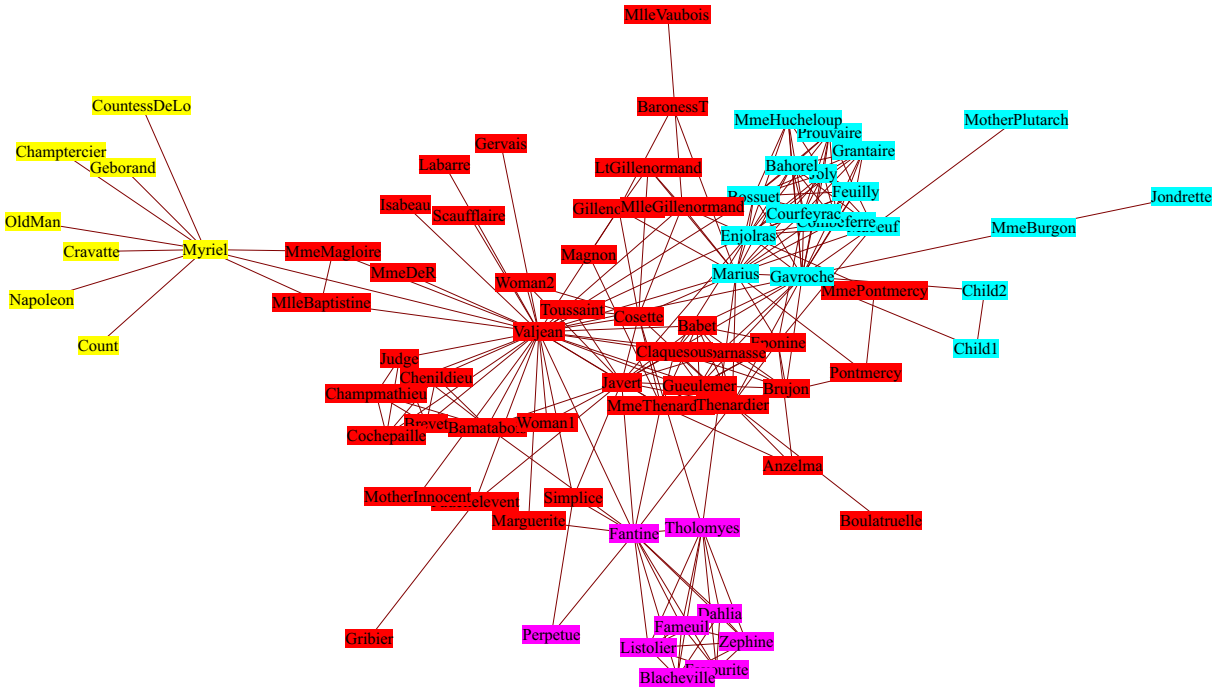

Fig. 7. Clusters resulting from our algorithm for the interactions between major characters in the novel Les Misérables by Victor Hugo.

cluster is clear from an intuitive point of view, to perform a correct analysis of a network it is essential to use an unambiguous and quantitative definition of a cluster, and more important to provide constructive algorithms useful to deal with these large networks. Notice that the determination of all the subgraphs of a given network that form a cluster according to a given definition is an NP-complete problem. Therefore it is useful to deal with the problem using heuristic and probabilistic methods.

In this paper we propose a new constructive deterministic clustering algorithm. It relies on the local structure of the graph and does not need the determination of global parameters, resulting in a very fast method with a performance comparable to well established algorithms.

Acknowledgement

The authors would like to thank Mark Newman for providing the data on the networks considered here.

References

[1] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (2002) 7821–7826.
[2] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proc. Natl. Acad. Sci. USA 101 (2004) 2658–2663.
[3] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (2004) 066133.
[4] A. Lancichinetti, S. Fortunato, Community detection algorithms: A comparative analysis, Phys. Rev. E 80 (2009) 056117.
[5] G.W. Flake, S.R. Lawrence, C.L. Giles, F.M. Coetzee, Self-organization and identification of web communities, IEEE Comput. 35 (2002) 66–71.
[6] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, D. Wagner, On finding graph clusterings with maximum modularity, Lect. Notes Comput. Sci. 4769 (2007) 121–132.
[7] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. USA 104 (2007) 36–41.
[8] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Antropol. Res. 33 (1977) 452–473.
[9] M.E.J. Newman, The structure and function of complex networks, SIAM Rev. 45 (2003) 167–256.
[10] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.
[11] D. Lusseau, The emergent properties of a dolphin social network, Biol. Lett., Proc. R. Soc. Lond. B (Suppl.) 270 (2003) S186–S188.
[12] D.E. Knuth, The Stanford GraphBase: A Platform for Combinatorial Computing, Addison–Wesley, Reading, MA, 1993. ISBN 0-201-54275-7.
[13] A. Medus, G. Acuña, C.O. Dorso, Detection of community structures in networks via global optimization, Phys. A 58 (2005) 593–604.