RAIRO-Oper. Res. 45 (2011) 339–352
DOI: 10.1051/ro/2012001

RAIRO Operations Research
www.rairo-ro.org

BOOTSTRAP CLUSTERING FOR GRAPH PARTITIONING *

Philippe Gambette[1] and Alain Guénoche[1]

Abstract. Given a simple undirected weighted or unweighted graph, we try to cluster the vertex set into communities and also to quantify the robustness of these clusters. For that task, we propose a new method, called *bootstrap clustering* which consists in (i) defining a new clustering algorithm for graphs, (ii) building a set of graphs similar to the initial one, (iii) applying the clustering method to each of them, making a *profile* (set) of partitions, (iv) computing a consensus partition for this profile, which is the final graph partitioning. This allows to evaluate the robustness of a cluster as the average percentage of partitions in the profile joining its element pairs ; this notion can be extended to partitions. Doing so, the initial and consensus partitions can be compared. A simulation protocol, based on random graphs structured in communities is designed to evaluate the efficiency of the Bootstrap Clustering approach.

Keywords. Graph partitioning, clustering, modularity, consensus of partitions, bootstrap.

Mathematics Subject Classification. 05C85, 90C35, 62F40.

1. Introduction

The topic of assessing the quality of the classes of a partition, or the quality of the partition itself, appeared with the beginning of the clustering methodology and still remains an open subject. It has often been reduced to the seek for a partition that optimizes some criterion. However, these criteria are very diverse, and none of

Received June 14, 2011. Accepted January 26, 2012.

* *This work is supported by a PiriBio ANR grant (Moonlight project R09127AA). We would like to thank C. Brun (TAGC, Marseille) and A. Baudot (IML, Marseille) for fruitful discussions.*

[1] IML – CNRS, 163 Av. de Luminy, 13009 Marseille, France. guenoche@iml.univ-mrs.fr

them can be chosen as the best one to evaluate the quality of partitions computed according to different principles ([12] or [11]). Furthermore, they do not indicate the reliability of the computed classes and/or partition.

To quantify the robustness of the classes, we define a *bootstrap* methodology. The principle of bootstrap, used in machine learning [7] or in molecular phylogeny [10] consists in: (i) generating several data-sets by altering, *i.e.* sampling or noising, the initial data, (ii) applying the same treatment algorithm, and (iii) comparing the results to the initial one to evaluate its robustness. This process has already been used for clustering from data arrays [15], and in this article we apply it to graphs.

Hence, our bootstrap clustering approach for graph partitioning consists in the following steps:

- generating q altered graphs that are similar to the initial one (*i.e.* they share a large proportion of edges with it), by modifying the weights and/or the edge list;
- defining a fast partitioning method for graphs, for which it is not necessary to fix the number of classes, which is applied to partition the q graphs;
- applying a partition consensus method [13] which gives a median partition of the q obtained partitions.

Given the constraints of the partitioning method, we choose to use partitioning based on *modularity* optimization. Modularity has been introduced by [17] to describe whether a simple unweighted graph displays a community structure. Indeed, the modularity value of a graph partition increases with the percentage of intra-class edges. A very fast multi-level algorithm which can handle networks with millions of vertices was introduced in [4] to find a vertex partition optimizing the modularity function. We have developed a new heuristic for modularity optimization, which is not as fast, but is still able to deal with graphs with thousands of vertices (suitable for protein-protein interaction graphs), and gives results with better modularity. This approach was implemented in C and the source code is available at `http://bioinformatics.lif.univ-mrs.fr`.

In Section 2 we introduce the modularity formulas and the corresponding optimization problem. In Section 3, we describe two methods to generate altered graphs similar to the input graph, which will be used as the input of the bootstrap clustering method. In Section 4 we develop the consensus formalization, leading to the same kind of optimization problem. Then, in Section 5, we give a new heuristic algorithm, called TFit, for graph partitioning by modularity optimization. We compare its performance on benchmark graphs with other methods optimizing modularity. Other experimental results are detailed in Section 6: we evaluate the bootstrap procedure on simulated data, and quantify partition robustness.

2. Partitions optimizing modularity

Let $G = (V, E)$ be a simple, undirected, connected graph, with $|V| = n$ vertices and $|E| = m$ edges, weighted by a function $A : E \rightarrow \mathbb{R}_+$. If G is not weighted, A is the adjacency matrix, otherwise it is the non negative matrix of the edge weights. To detect communities in G, we look for a vertex partition $P = \{V_1, V_2, \dots V_p\}$ (into separate non empty classes, such that their union is equal to V) with a high modularity value. The partition modularity reflects the difference between the percentage of internal edges and this percentage under the hypothesis there is no community ; in that case, the edges are randomly distributed according to degrees. The modularity value is the gap between what is observed and what is due to randomness. Formally, let e_{ij} be the percentage of edges having one end in class V_i and the other one in class V_j: $(e_{ij} = |E \cap (V_i \times V_j)|/m)$. The probability for a random edge to have one of its ends in class V_i is

$$a_i = e_{ii} + 1/2 \sum_{j \neq i} e_{ij}$$

and the modularity of partition P is given by:

$$M(P) = \sum_{i=1..p} (e_{ii} - a_i^2). \tag{2.1}$$

This formula has been equivalently reformulated in the literature (see for instance [9, 16, 18]). In the following, we refer to Newman's formula:

$$M(P) = \frac{1}{2m} \sum_{x=1}^{n} \sum_{y=1}^{n} \left(A_{xy} - \frac{d_x d_y}{2m} \right) \alpha_{xy}, \tag{2.2}$$

where (A_{xy}) is the adjacency matrix of G, d_x is the degree of vertex x and α_{xy} is the square matrix of order n such that

$$\alpha_{xy} = \begin{cases} 1 \text{ if vertices } x \text{ and } y \text{ belong to the same class in } P, \\ 0 \text{ otherwise.} \end{cases}$$

Observing that A and α are symmetrical, $A_{xx} = 0$, $\alpha_{xx} = 1$, and setting $w_{xy} = 2mA_{xy} - d_x d_y$, maximizing $M(P)$ is equivalent to maximize

$$Q(P) = \sum_{x=2}^{n} \sum_{y=1}^{x-1} \alpha_{xy} \big(2mA_{xy} - d_x d_y \big) = \sum_{x=2}^{n} \sum_{y=1}^{x-1} \alpha_{xy} w_{xy}. \tag{2.3}$$

It can be seen that $Q(P)$ is the sum of the values w_{xy} of joined pairs (x, y) of the partition. These values are certainly negatives if $(x, y) \notin E$ but they are positive as soon as $d_x d_y < 2m$. So, maximizing the modularity Q over the set of all the partitions of V reduces to a Clique Partitioning problem of the complete graph on V weighted by the positive or negative values of $W = (w_{xy})$.

This formulation allows to extend the above formulas to weighted graphs. The adjacency matrix corresponds to the edge weights, admitting $A(x, y) = 0$ if

$(x, y) \notin E$. The vertex degrees coincide with the sums over the rows of this matrix ($d_x = \sum_{y|(x,y)\in E} A(x, y)$) while formula (2) still defines the weights, positive or negative, of a complete graph. In that case, the modularity Q is no longer an integer value.

3. Bootstrapped graphs

In many domains, such as protein-protein interaction (PPI) graphs or social networks, there is some uncertainty about the edges. A "true" graph possibly exists, but the one we have depends on temporal informations or on experiments. For the PPI networks, the interactions are far to be known and even some of them could not exist. The given graph can be considered as an instance of a true graph. Other ones are expected, that's why we generate graphs from the initial one, which is the only one we have as input. We will denote these randomly generated graphs as *bootstrapped graphs.*

As the partition profile is obtained, in our bootstrap clustering framework, by building a partition on a bootstrapped graph, those graphs should not be too close to G. Otherwise, the same partition would always be obtained, giving 100% robust identical classes. On the contrary, the generated graphs should not be too far from G. This would give a non homogeneous profile of partitions leading to the atomic partition as consensus. So, starting from G, we will not introduce edges between distant vertices, or erase edges making distant new pairs of vertices previously adjacent. Actually, we would like to perform some local rearrangements, by slightly modifying the graph, and the resulting partition.

For that task, we use the Czekanovski-Dice (C-D) distance [8] which is a local measure of dissimilarity. Let $G(x)$ be the set of all adjacent vertices of x, let $\overline{G}(x) = \{x\} \cup G(x)$ and let Δ denote the symmetric difference between two sets. We have

$$D(x, y) = \frac{|\Delta(\overline{G}(x), \overline{G}(y))|}{|\overline{G}(x)| + |\overline{G}(y)|}.$$

It is easily seen that the distance values decrease as vertices share more common adjacent vertices and that two vertices having distance 1.0 are separated by more than 2 edges. The C-D distance allows, on the one hand to weight the edges (a strong weight corresponding to edges whose ends belong to the same community), and on the other hand to prevent from connections between distant vertices. This distance has been established for unweighted graphs, but equivalent formulas have been adapted for weighted ones [2].

Now, we describe two generating strategies for bootstrapped graphs, which will be tested afterwards to fix parameters. Namely:

(1) Randomly modify the edge weights. It is the simplest procedure, fixing the *elongation rate* τ_e as the only parameter. Each edge weight is multiplied by a random coefficient selected in $\tau \in [1 - \tau_e, 1 + \tau_e]$. Thus, all the edges are kept

and none are added.

$$\text{If } (x,y) \in E, \text{ then } a(x,y) \leftarrow a(x,y)\tau \text{ else } a(x,y) = 0;$$

(2) add supplementary edges and assign new weights to all the edges, according to the C-D distance. In the first step, the C-D distance is computed and let F be the set of non adjacent pairs of vertices having a distance strictly lower than 1.0:

$$F = \{(x,y) \notin E, \text{ such that } D(x,y) < 1.0\}.$$

The supplementary edges are selected in F by the way of a parameter τ_a which corresponds to the probability of any pair to be selected. All these edges are weighted by $a(x,y) = 1 - D(x,y)$.

From these two types of bootstrapped graphs, a set of partitions of V is established. Since graphs are not identical, partitions will be different but not too much. The frequently joined vertices will define robust classes of V making a consensus partition.

4. Consensus partition

4.1. Definition of the problem

Former works on consensus partitions were motivated by the problem of clustering items described by nominal variables. In his pioneer paper, Régnier [19] introduced the notion of *partition centrale*, defined as the partition with minimum sum of distances to the partitions in the profile. Hereafter, we focus on this notion of consensus. Indeed it has been empirically assessed that other definitions, more strict or formal, do not lead to satisfying practical results. For the numerous works axiomatically tackling this problem or seeking for a node in the partition lattice, we refer to [3].

Let $\mathcal{P}$ be the set of all partitions of V and $\Pi \subset \mathcal{P}$ be a *profile* of q partitions. For partition $P \in \mathcal{P}$, any element $x \in V$ belongs to the class denoted $P(x)$. Using δ as the usual Kronecker symbol, $\delta_{P(x)P(y)} = 1$ if x and y are joined in P and $\delta_{P(x)P(y)} = 0$ otherwise. Given a profile Π, the *consensus partition problem* consists in finding $\pi \in \mathcal{P}$ minimizing the sum of symmetric difference distances to Π. In other words, it is a median partition. In terms of similarity, rather than distance, it consists of enumerating pairs of items that are commonly joined or separated.

$$S(P,Q) = \sum_{x<y} \left(\delta_{P(x)P(y)}\delta_{Q(x)Q(y)} + (1-\delta_{P(x)P(y)})(1-\delta_{Q(x)Q(y)})\right). \tag{4.1}$$

Then, the *score* of a partition P with respect to a profile $\Pi = (P_1, \dots, P_q)$ is defined as the sum of the similarity values between P and any partition in Π:

$$S_\Pi(P) = \sum_{k=1}^{q} S(P, P_k). \tag{4.2}$$

Given a profile, let T_{xy} be the number of partitions joining x and y in the same class. So:

$$\begin{aligned} S_\Pi(P) &= \sum_{x<y} \left(\delta_{P(x)P(y)} T_{xy} + (1-\delta_{P(x)P(y)})(q - T_{xy})\right) \\ &= 2\sum_{x<y} \delta_{P(x)P(y)} T_{xy} + \sum_{x<y} q - \sum_{x<y} \delta_{P(x)P(y)} q - \sum_{x<y} T_{xy}. \end{aligned}$$

Quantities $\sum_{x<y} q$ and $\sum_{x<y} T_{xy}$ only depend on the profile Π and not on P. Thus, maximizing $S_\Pi(P)$ is equivalent to maximize:

$$\sum_{x<y} \delta_{P(x)P(y)} T_{xy} - \frac{1}{2}\sum_{x<y} \delta_{P(x)P(y)} q.$$

Let $J(P)$ be the set of joined pairs in P. An equivalent criterion to $S_\Pi(P)$ is:

$$W_\Pi(P) = \sum_{(x,y)\in J(P)} \left(T_{xy} - \frac{q}{2}\right)\cdot \tag{4.3}$$

Criterion W_Π can be intuitively interpreted as follows: for a partition P, a joined pair in $J(P)$ has a positive contribution (resp. negative) when both elements are joined in more (resp. less) than half the partitions in Π.

Let $\mathbf{K}_n$ be the complete graph on V, in which the pairs are weighted by $w : V \times V \to \mathbb{R}$, where $w(x,y) = T_{xy} - q/2$ and let P be a partition into p classes $P = (V_1, \ldots, V_p)$. The quantity $W(V_k) = \sum_{(x,y)\in V_k} w(x,y)$ is the weight of all the pairs (a clique) in V_k. We have,

$$W_\Pi(P) = \sum_{k=1,..p} W(V_k) = \sum_{k=1,..p} \sum_{(x,y)\in V_k} \left(T_{xy} - \frac{q}{2}\right)\cdot \tag{4.4}$$

Thus the consensus of partitions problem can be seen as a Clique Partitioning problem on the complete graph on V, weighted by $w(x,y) = T_{xy} - \frac{q}{2}$. The weights $w(x,y)$ are positive or negative, according to the number of times x and y are joined.

4.2. A Common optimization problem

Both problems, modularity maximization and consensus of partitions, come down to Clique Partitioning problems on the complete graph on V, positively and negatively weighted. So we are looking for a set of separated cliques in $(\mathbf{K}_n, w)$ with a maximum sum of weights. This problem is an extension to weighted graphs of Zahn problem [20], which has been proved to be NP-hard [3]. Therefore no polynomial algorithm is able to ensure an optimal solution. As noted by Régnier, the consensus of partitions problem can be solved by integer linear programming, and so it is for modularity optimization [5]. For n elements there are $n(n-1)/2$ binary variables and $3\binom{n}{3}$ linear constraints. Even if there exist optimal methods optimizing W over $\mathcal{P}$, we have for $n = 100$, 4950 variables and 485 100 constraints, which constitutes a limitation for GLPK (GNU Linear Programming Kit). Recently, Aloise *et al.* [1] have computed, by column generation, optimal partitions maximizing the modularity for graphs up to 512 vertices. For larger ones, heuristics must be used.

5. The iterated transfer-fusion method (TFit)

5.1. Description of the algorithm

Those heuristics correspond mostly to agglomerative hierarchical clustering methods, where cluster fusion is the basic operation. In the fast and popular Louvain method [4], each step before cluster fusion consists in creating clusters of clusters, and transferring the clusters of vertices from one cluster of cluster to another, as long as modularity increases. We call this operation "cluster transfer". Once this step is over, clusters of vertices belonging to the same cluster of cluster are merged.

The function which decides which pair will be fusioned next is called "merge prioritizer" in a comparative study of heuristics for modularity optimization by Noack and Rotta [18]. In the Louvain method as well as in ours, we use no "merge prioritizer" (those may create unbalanced clusters), and just consider in turn all clusters or vertices to transfer.

Noack and Rotta also mention possible post-processings, called "refinements", based on vertex transfers, to improve modularity. They claim that this step increases the running time. However, for some graphs like protein-protein interaction graphs which have a few thousands of vertices, we are not limited by this running time issue, and we can use the "fast greedy" version of this improvement heuristic, which consists in transferring each vertex to the cluster with best modularity improvement, if any. Furthermore, we do not only apply it at the end of the algorithm, but before each cluster fusion step, which explains the name of our algorithm. Briefly speaking, TFit is a multi-level algorithm in which an element transfer procedure has been inserted at each level change.

More formally, algorithm TFit is described in Figure 1, and illustrated on an example in Section 5.2. Note that with the chosen version of the modularity formula, where each edge (x, y) has a modularity score $w(x, y)$, it is easy to compute the potential modularity increase for each vertex transfer, and to update the modularity values. Indeed, the contribution of vertex x to the modularity of its class can be expressed as $K(x, C_i) = \sum_{y \in C_i} w(x, y)$, and the gain of modularity resulting from a transfer of x to cluster C_j can be expressed as $K(x, C_j) - K(x, C_i)$.

Note that as both partition consensus and modularity optimization are special cases of Clique Partitioning, we can solve both problems with the TFit algorithm. We checked experimentally that TFit obtains similar results (or slightly better, depending on the graph density) than the Transfer-Fusion method [13] for

```
TFit ($G = (V, E)$: graph)
1.  $P \leftarrow \{\{x\}, x \in V\}$; $mod \leftarrow 0$; $currentmod \leftarrow mod$; $continue \leftarrow$ true;
2.  While $continue$ Do
3.      While $\exists v \in V, C_i \in P \cup \{\emptyset\}$ such that
        modularity($G$,$P$) < modularity($G$,Transfer($v$,$C_i$,$P$)) Do
4.          $C_i \leftarrow$ argmax(modularity($G$,Transfer($v$,$C_i$,$P$)));
5.          $P \leftarrow$ Transfer($v$,$C_i$,$P$);
6.          $currentmod \leftarrow$ modularity(G,$P$);
7.      $P' \leftarrow \{\{C_i\}\}$;
8.      While $\exists C_i \in P, P'_j \in P' \cup \{\emptyset\}$ such that
        modularity($G$,$P'$) < modularity($G$,Fusion(Transfer($C_i$,$P'_j$,$P'$))) Do
9.          $P'_j \leftarrow$ argmax[modularity($G$,Fusion(Transfer($C_i$,$P'_j$,$P'$)))];
10.         $P' \leftarrow$ Transfer($C_i$,$P'_j$,$P'$);
11.         $currentmod \leftarrow$ modularity(G,Fusion($P'$));
12.     $P \leftarrow$ Fusion($P'$);
13.     If $currentmod \leq mod$ Then
14.         $continue \leftarrow$ false;
15.     Else
16.         $continue \leftarrow$ true;
17.         $mod \leftarrow currentmod$;
18. Return $P$;
```

FIGURE 1. TFit algorithm. A call to **Transfer**(v,P_i,P) deletes the element v (either a vertex or a cluster) from its cluster in P, adds it to the cluster $P_i \in P$, and returns P. A call to **Fusion**($P' = \{P'_i\}$) returns $P = \{\bigcup_{C_i \in P'_i} C_i\}$.

consensus partition. We also proposed, in this article, a stochastic procedure based on element transfers. It starts from a random partition established by exchanging some elements in the optimized partition and applies single element transfers improving W. Its efficiency for consensus has been proved, but for fairness of comparison, we do not use it in our tests against other algorithms.

5.2. Example

To illustrate algorithm TFit, we apply it on the graph illustrated in Figure 2. Initially, $P = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}, \{10\}, \{11\}, \{12\}, \{13\}, \{14\}\}$.

The first execution of the while loop of line 2 occurs as follows:

- following the first series of transfers due to the while loop of line 3, we obtain partition $P = \{\{1, 6, 7\}, \{2, 3, 10, 11\}, \{4, 5, 13, 14\}, \{8, 9, 12\}\}$, with a modularity of 0.33;
- on line 7, we create $P' = \{\{\{1, 6, 7\}\}, \{\{2, 3, 10, 11\}\}, \{\{4, 5, 13, 14\}\}, \{\{8, 9, 12\}\}\}$;

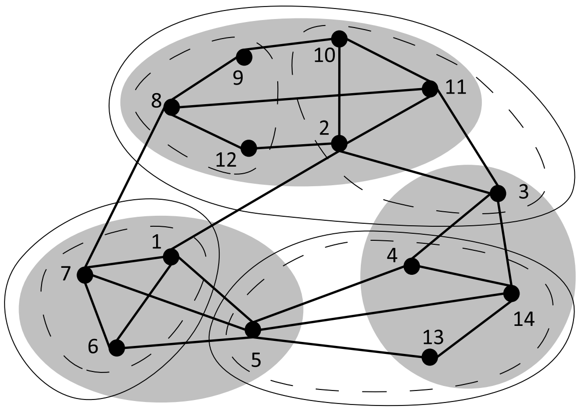

FIGURE 2. An example of graph, where TFit computes the partition $\{\{1,5,6,7\},\{2,8,9,10,11,12\},\{3,4,13,14\}\}$, which is represented by the gray ovals. Starting from singleton clusters, it is obtained after a first vertex transfer step (the dotted line clusters), a cluster transfer and fusion step (the full line clusters), and another vertex transfer step (the gray clusters).

- the second while loop (on line 8) only performs one cluster transfer: cluster $C_2 = \{2,3,10,11\}$ is transfered to $P'_4 = \{\{8,9,12\}\}$, so we obtain $P' = \{\{\{1,6,7\}\},\{\{4,5,13,14\}\},\{\{8,9,12\},\{2,3,10,11\}\}\}$;
- on line 12, we obtain $P = \{\{1,6,7\},\{4,5,13,14\},\{2,3,8,9,10,11,12\}\}$, with an improved modularity of 0.35.

The second execution of the while loop of line 2 only performs two vertex transfers:

- in the while loop of line 3, vertex 3 is transfered to $C_2 = \{4,5,13,14\}$, and vertex 5 is transfered to $C_1 = \{1,6,7\}$, which provides partition $P = \{\{1,5,6,7\},\{3,4,13,14\},\{2,8,9,10,11,12\}\}$, with an improved modularity of 0.38;
- the while loop of line 8 does not perform any cluster transfer.

Finally, we obtain partition $P = \{\{1,5,6,7\},\{3,4,13,14\},\{2,8,9,10,11,12\}\}$.

5.3. Performance of TFit on benchmark graphs

A set of graphs corresponding to real data have been used as benchmarks in many articles which compare graph clustering algorithms through modularity optimization. For some of them (with at most a few hundred vertices), an optimal solution was computed by integer linear programming [1]. We compared our heuristic to the Louvain method and to the best modularity obtained by a set of heuristics described by Noack and Rotta. As shown in Table 1, our method always gives better results than the Louvain method, and sometimes outperforms the ten Noack

Table 1. Comparison of modularity optimization heuristics on benchmark graphs.

Graph	n	m	Opt	Louvain	N-R	TFit
Dolphins	62	159	.5285	.5185	**.5276**	.5268
polBooks	105	441	.5272	.5266	**.5272**	.5268
afootball	115	613	.6046	**.6046**	.6045	**.6046**
A01	249	635	.6329	.6145	**.6293**	.6284
USAir97	332	2126	.3682	.3541	**.3678**	.3595
netscience	379	914	.8486	**.8475**	.8474	**.8475**
s388	512	819	.8194	.7962	.8143	**.8148**
emails	1133	5452		.5438	**.5816**	.5722

and Rotta heuristics. Note that if we compare TFit with each of those heuristics, the modularity found is equal or better in at least 5 of the 8 benchmark graphs.

5.4. Robustness of classes and partitions

The score of a partition $W_{\Pi}(\pi)$ is defined as the sum of joined pair weights. So the score of a class is all the more high that its pairs are often joined in the profile. One can evaluate the robustness of a class by the percentage of partitions in the profile joining its elements. As $T_{xy} = |\{P \in \Pi = \{P_1, \dots, P_q\}$ such that $P(x) = P(y)\}|$, we set:

$$Rob(V_k) = \frac{2\sum_{x,y \in V_k} T_{xy}}{q \times |V_k| \times (|V_k| - 1)}.$$

This quantity (between 0 and 1) is the average percentage of partitions joining the pairs of elements in the class. So, one can compare classes using $Rob(V_k)$, the best ones containing only pairs often joined in the profile.

This definition can be extended to partitions. Their robustness is the average, over joined pairs (x, y), of the percentage of partitions joining them. Recall that $J(P)$ is the set of joined pairs in P. We obtain

$$Rob(P) = \frac{1}{q \times |J(P)|} \sum_{(x,y) \in J(P)} T_{xy}.$$

6. Evaluation of bootstrap clustering

6.1. Simulation protocol

We have developed a simulation protocol with unweighted random graphs made of 200 vertices spread in 5 balanced classes defining a *seed partition*, P_{seed}. Each graph is generated by an Erdös-Reyni procedure with two parameters, the internal density (intra-class edges) d_i and the external density (inter-class edges) d_e. There are three families of graphs corresponding to densities $(d_i = .30, d_e = .10)$,

Table 2. Corrected Rand index and robustness of initial and consensus partitions, for 30 type 1 bootstrapped graphs.

		Rand				Robustness					
		P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
d_i	d_e		.01	.02	.03	.01		.02		.03	
.30	.10	.825	.881	.880	.880	.888	.939	.886	.938	.886	.938
.20	.05	.689	.793	.798	.797	.737	.842	.734	.841	.731	.842
.10	.01	.615	.682	.683	.678	.719	.835	.713	.829	.708	.828

$(d_i = .20, d_e = .05)$ and $(d_i = .10, d_e = .01)$. They generate more and more difficult problems, not because community vanishes into the graph, but the consensus classes do not fit the seed partition when the average degree decreases.

The TFit algorithm is applied to obtain an initial partition P_{ini}. Then, $q = 30$ bootstrapped graphs are generated and clustered using $TFit$ to get a profile Π containing q partitions (a larger value has been tested and does not provide any improvements). Its consensus partition P_{cons}, is computed with $TFit$ again, without applying any stochastic procedure. The two partitions, P_{ini} and P_{cons}, are compared to the seed partition P_{seed} by the way of the corrected Rand index [14]) and also their robustness values, as previously defined.

6.2. Bootstrapping by edge elongation

We try to establish the elongation rate optimizing the proximity between P_{seed} and P_{cons}, that is maximizing the corrected Rand index. This leads to small values for τ_e: .01, .02 and .03 (see Tab. 2).

It is seen in Table 2 that the corrected Rand index gets lower as the problem gets more challenging. It illustrates that problems are more and more difficult. However consensus partitions are clearly closer than the initial ones, particularly for $\tau_e = .02$. The robustness of the initial partition depends on bootstrapped graphs since it can be evaluated after the clustering of the 30 bootstrapped graphs. Finally, P_{cons} always has a larger value than P_{ini}, which is expected by consensus definition.

6.3. Bootstrapping by edge addition and weighting

Here again, we seek for the added edge rate maximizing the corrected Rand index between P_{seed} and P_{cons}. Surprisingly, the higher values for τ_a, the better the result is, suggesting that the weighting and the supplementary edges improve the homogeneity of the communities. The higher the density values are, the lower τ_a must be to get the best corrected Rand index value. The robustness values are clearly improved and reach their maximum for a large number of added edges.

TABLE 3. Corrected Rand index and robustness of initial and consensus partitions, for 30 type 2 bootstrapped graphs.

		Rand				Robustness					
		P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
d_i	d_e		.30	.50	.70	.30		.50		.70	
.30	.10	.825	.833	.828	.815	.767	.845	.777	.847	.805	.868
.20	.05	.689	.768	.775	.772	.700	.827	.732	.852	.758	.880
.10	.01	.615	.695	.723	.728	.694	.816	.744	.852	.772	.885

TABLE 4. Size and robustness of the 7 main classes computed by TFit on the *Plasmodium* network.

Size	333	176	54	219	119	193	259	$M = .1569$
Rob.	.74	.45	.57	.69	.83	.83	.65	$\overline{Rob} = .696$

TABLE 5. Size and robustness of the 11 main consensus classes computed by the bootstrap clustering algorithm on the *Plasmodium* network.

Size	325	223	239	31	113	181	161	14	30	9	14	$M = .1571$
Rob.	.81	.84	.79	.84	.92	.89	.69	82	96	97	93	$\overline{Rob} = .815$

6.4. A BIOLOGICAL NETWORK

Protein-protein interaction networks are represented by graphs whose vertices correspond to proteins and edges to the physical binding between two proteins. While these complex networks contain a tremendous amount of information about protein and cellular function, its extraction is one of the most challenging issue of systems biology.

A *Plasmodium* interactome network [6] containing 12 391 interactions between 1416 proteins has been analyzed. Algorithm TFit applied to this graph gives 33 classes, 7 of which having more than 5 elements. The size and the robustness of these classes are given in Table 4. The modularity of this partition is equal to 0.1569.

For the bootstrap clustering method, type 1 bootstrapped graphs were used with $\tau_e = .01$. The obtained consensus partition gives 44 classes, 11 of which having more than 5 elements. The modularity of this partition is practically unchanged (0.1571) and the average robustness is much higher.

Generally, the modularity is lightly decreasing, which can be expected since it is the maximized criterion in TFit, and the average robustness is always strongly increased, as it is here.

7. Conclusion

The first type of bootstrapped graphs, generated by edge length variation, seems to be the most efficient for graphs belonging to the two first families ($d_i \leq .20, d_e \leq .10$), when the τ_e parameter varies between .01 and .02. Overall, for the two types and the three families of graphs, consensus partitions are closer, on the average, to the seed partitions than the initial ones, but it is not always the case. Concerning the robustness, classes of P_{cons} are systematically more robust than those of P_{ini}.

What about the modularity and the number of classes of the consensus partition? Bootstrap clustering tends to increase the number of classes and to isolate outliers. Concerning the number of classes, we have only counted clusters with at least 3 elements (5 for the biological data). The average number of such classes does not vary much; it decreases for the first graph family, and increases by one or two units for the others. This leads to a consensus partition with lightly smaller modularity value than the initial one. However, it depends on the edge density. Concerning modularity, we observe small variations. Compared to the initial partition, modularity increases for the first family, by around 3% on the average, and decreases in the same proportion for the other family. Robustness improvement compensates for when modularity variations are negative.

In conclusion, bootstrap clustering allows to evaluate the robustness of classes and partitions. When there exists some uncertainty on graph edges, it is likely to claim that the bootstrap clustering procedure allows to improve partition quality. Moreover, it is always the case when graphs have a low edge density.

References

[1] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti and S. Perron, Column generation algorithms for exact modularity maximization in networks. *Phys. Rev. E* **82** (2010) 046112.

[2] J.B. Angelelli, A. Baudot, C. Brun and A. Guénoche, Two local dissimilarity measures for weighted graph with application to biological networks. *Adv. Data Anal. Classif.* **2** (2008) 3–16.

[3] J.P. Barthélemy and B. Leclerc, The median procedure for partitions. *DIMACS series in Discrete Mathematics and Theoretical Computer Science* **19** (1995) 3–34.

[4] V. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.* (2008) P10008.

[5] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski and D. Wagner, On modularity – NP-completeness and beyond. *Proceedings of WG 2007. Lett. Notes Comput. Sci.* **4769** (2007) 121–132.

[6] S.V. Dale and C.J. Stoeckert Jr., Computational modeling of the Plasmodium falciparum interactome reveals protein function on a genome-wide scale. *Gen. Res.* **16** (2006) 542–549.

[7] A.C. Davison and D.V. Hinkley, *Bootstrap methods and their application.* Cambridge University Press (1997).

[8] L.R. Dice, Measures of the amount of ecologic association between species. *Ecology* **26** (1945) 297–302.

[9] J. Duch and A. Arenas, Community detection in complex networks using extremal optimization. *Phys. Rev. E* **72** (2005) 027104.

[10] J. Felsenstein, *Inferring Phylogenies.* Sunderland (MA), Sinauer Associates Inc. (2003).

[11] S. Fortunato, Community detection in graphs. *Phys. Rep.* **486** (2010) 75–174.

[12] A. Guénoche, Comparison of algorithms in graph partitioning. *RAIRO* **42** (2008) 469–484.

[13] A. Guénoche, Consensus of partitions: a constructive approach. *Adv. Data Anal. Classif.* **5** (2011) 215–229.

[14] L. Hubert and P. Arabie, Comparing partitions, *J. Classif.* **2** (1985) 193–218.

[15] A.K. Jain and J.V. Moreau, Bootstrap technique in cluster analysis. *Pattern Recogn.* **20** (1987) 547–568.

[16] M.E.J. Newman, Modularity and community structure in networks. *PNAS* **103** (2006) 8577–8582.

[17] M.E.J. Newman and M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69** (2004) 026133.

[18] A. Noack and R. Rotta, Multi-level algorithms for modularity clustering, *Proceedings of SEA'2009*, edited by J. Vahrenhold. *Lett. Notes Comput. Sci.* **5526** (2009) 257–268.

[19] S. Régnier, Sur quelques aspects mathématiques des problèmes de classification automatique. *I.C.C. Bulletin* **4** (1965) 175–191. Reprint, *Math. Sci. Hum.* **82** (1983) 13–29.

[20] C.T. Zahn, Approximating symmetric relations by equivalence relations. *SIAM J. Appl. Math.* **12** (1964) 840–847.