Chapter 1

A FUSION CONDITION FOR COMMUNITY DETECTION WITH MODULARITY

Olivier Gach[1], ***Jin-Kao Hao***[1,2]
1) LERIA, Université d’Angers, 2 bd Lavoisier, 49045 Angers, France
2) Institut Universitaire de France, 1 rue Descartes, 75231 Paris, France
Email: olivier.gach@univ-tours.fr; jin-kao.hao@univ-angers.fr

Abstract

Community detection is an important research topic for complex network analysis. Modularity is probably the most popular measure of clustering quality for community detection. Unfortunately, modularity has a default, called the resolution limit, which tends to make disappear small communities especially in large networks. Several algorithms for community detection rely on community fusions by considering the modularity. We establish a fusion condition which aims to prevent wrong fusions and alleviate the effect of the resolution limit. To verify the usefulness of the proposed fusion condition, we implement it within the popular Louvain method that uses intensively merging operations while optimizing the modularity. We evaluate the resulting method on LFR benchmark graphs and show that the proposed fusion condition strongly reduces the effect of the resolution limit. The proposed condition can also be used in other algorithms that make community fusions.

Keywords: community detection, community fusion, modularity, resolution limit, complex networks, heuristics

1. INTRODUCTION

Complex networks formed by human activity or observed in nature have nontrivial topological features that distinguish them from simpler networks, especially those generated by random models [Newman, 2010]. Complex networks have wide applications in diverse domains like biology [Guimerà and Nunes Amaral, 2005, Ito et al., 2001], collaboration modeling [Durugbo et al., 2011], technological networks [Flake et al., 2002], texture analysis and classification [Backes et al., 2013], and web [Girvan and Newman, 2002]. Given their relevance, studies of common properties of these complex networks become more and more intense in recent years under the generic term of *complex network analysis*. One of these remarkable properties is the presence of high density areas described as groups, communities or modules [Newman, 2010] depending on the application. Informally, a community is defined as a set of network elements that are highly connected within the group and weakly connected to the outside. The study of community structures inspires intense research activities to visualize and understand the dynamics of a network at different scales [Didimo and Montecchiani, 2014, Gulbahce and Lehmann, 2008, Newman, 2012].

Graph-based models and techniques are the dominant tools to the study of complex networks [Buluç et al., 2016] where a network element is designated by a vertex (node) and a connection between two elements is represented by an edge (or arc). In the simplest form of network analysis, each vertex is considered to belong to a single community, communities being disjoint subsets of vertices. For a graph $G = (V,E)$ with set V of n vertices and set E of m edges, we use $\mathcal{P} = \{C_1,C_2,...,C_k\}$ to represent a partition or clustering of the vertices of set V into communities C_i. The community detection problem studied in this paper is to find such a partition according to a given criterion, which is close to the conventional graph partitioning problem [Benlic and Hao, 2013], but without knowing the number of groups.

There are a wide variety of quality measures for community detection [Aldecoa and Marín, 2011, Ghosh and Lerman, 2010, Li et al., 2008, Xiang et al., 2016]. Among them, modularity introduced in 2004 [Newman and Girvan, 2004] is certainly the most popular and most studied measure. Let C be a community and call an edge ’internal edge’ if both its endpoints are in C. Let $l(C)$ be the number of internal edges of C, $d(C)$ be the degree of community defined as the sum of the degrees of nodes belonging to C. Then the modularity $Q(\mathcal{P})$ of partition $\mathcal{P}$ is given by:

$$Q(\mathcal{P}) = \sum_{C\in\mathcal{P}} \left(\frac{l(C)}{m} - \left(\frac{d(C)}{2m} \right)^2 \right) \tag{1}$$

This global measure opens ways of using optimization algorithms to detect communities. In other words, the problem of community detection comes down to finding a partition of the vertex set V of graph G such that the above modularity criterion is maximized. However, this problem is NP-hard [Brandes et al., 2008] in general. To handle large graphs, heuristics become indispensable and are indeed the dominant approach to find approximate partitions [Blondel et al., 2008, Gach and Hao, 2012, Liu and Murata, 2010, Lü and Huang, 2009, Newman, 2004, Pons and Latapy, 2005, Rotta and Noack, 2011, Schuetz and Caflisch, 2008]. Among the many methods for Q optimization, approaches based on local search that moves vertices and merges communities prove to be both effective and fast. One representative local search heuristic is the well-known Louvain algorithm [Blondel et al., 2008] that is based on a multi-level technique [Benlic and Hao, 2011]. Compared to other methods, the Louvain algorithm offers an interesting compromise between solution quality and computing efficiency, with a weak complexity of $O(m)$ for sparse graphs.

Unfortunately, the modularity criterion suffers from an intrinsic drawback, i.e., the resolution limit that is established by Fortunato and Barthemely in 2007 [Fortunato and Barthélemy, 2007]. The resolution limit states that under certain conditions, optimizing the modularity criterion makes small communities vanish especially in large graphs with the presence of relative large communities (i.e., small communities are

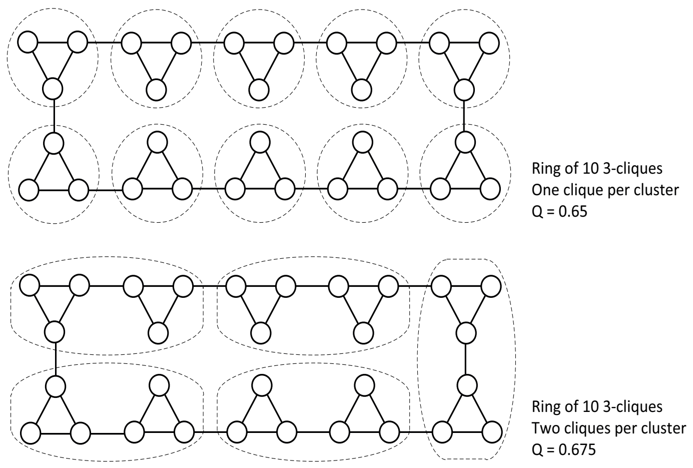

Figure 1. Illustration of the resolution limit of modularity [Fortunato and Barthélemy, 2007]. From a ring of 3-cliques connected in pairs by a single link, the natural clustering associates a community to each clique, leading to a modularity value of about 0.65. However, modularity can be improved by combining cliques by connected pairs while making some communities disappear.

'absorbed' by large communities). This drawback is first demonstrated on extreme cases of cliques which are interconnected minimally, e.g., a ring of cliques connected in pairs by a single edge (see Fig. 1). In this case, if the graph is large enough, the cliques of size less than or equal to $\sqrt{m}$ are merged with other cliques while they should be identified as independent communities. This phenomenon is also observed in more general situations, e.g., for random graphs [Lancichinetti et al., 2008, Lancichinetti and Fortunato, 2009].

The Louvain algorithm [Blondel et al., 2008] optimizes the modularity criterion by operating in two phases: a phase of building a clustering with the *vertex mover* (VM) procedure [Schuetz and Caflisch, 2008], then a phase of consolidation of communities at higher levels, similar to community fusions. By running the algorithm on graphs generated with the LFR model [Lancichinetti et al., 2008], the number of communities obtained from the VM procedure is greater than the number of communities in the reference solution, then the number of communities becomes smaller after the second phase. This is because some small communities are merged together during the second phase when the algorithm seeks to optimize (maximize) the modularity criterion. This is the most obvious illustration of the resolution limit. Although intrinsically optimizing modularity generates this default, we can imagine that some community fusions are justified and necessary while others should be avoided. In this work, we will use the Louvain algorithm as our reference algorithm.

The objective of this work is to establish a fusion condition (criterion) that allows one to know whether two communities should be merged or not. From two communities C and C' and the subgraph $G_{C,C'}$ induced by $C \cup C'$, the fusion condition $\alpha(C,C')$ is true if C and C' should be merged and false otherwise. Our intuitive idea is to base the fusion decision on the modularity calculated for the partition $\{C,C'\}$ in the subgraph $G_{C,C'}$.

One first possibility is that the two communities must be put together (i.e., the fusion condition is true) if they separately are not two strong communities (i.e., their modularity is low). Unfortunately, according to our experiments, there is no a significant correlation between the value of modularity for the partition (C, C') and the correct decision to merge C and C' or not. In contrast, the parametric modularity Q_λ [Reichardt and Bornholdt, 2006] reveals a greater correlation when the parameter λ is close to 0.25, which is the resolution value at which the graph is partitioned. For this reason, we will elaborate our fusion condition by considering the parametric modularity (See Section 2.1.). The idea is still to check if the modularity between two separate communities is too high for a fusion, but with a weight λ put on the *null model* [Molloy and Reed, 1995] part of the modularity formulation. With λ set to a value between 0 and 1, the internal edges score (first term of modularity) is more or less important compared to the null model term, which is weighted by the λ parameter.

To assess the usefulness of the proposed fusion condition, we implement it within the very popular Louvain algorithm [Blondel et al., 2008] and carry out experiments on graphs generated by the LFR model [Lancichinetti et al., 2008]. These graphs are particularly appropriate for our evaluation since the set of 'ideal' communities is known for each of these graphs. By using the known communities as our reference, we are able to evaluate precisely the clusterings obtained by the Louvain algorithm with and without the proposed fusion condition.

The paper is organized as follows. In Section 2., we establish a short formulation of the fusion condition with only one parameter, followed by a characterization of the possible values for this parameter (Section 3.). Section 4. is dedicated to the assessment of the proposed fusion condition. We discuss in Section 5. the effect of fusion condition on the resolution limit and conclude the paper in the last section.

2. FORMULATION OF THE FUSION CONDITION

2.1. General formula

Given two communities C and C' of a clustering $\mathcal{P}$ of the graph $G = (V, E)$, we introduce the following notations and definitions which are useful for the description of the proposed fusion condition.

- $G_{C,C'} = (C \cup C', E_{C,C'})$: The subgraph of G induced by the vertex subset $C \cup C'$, where $E_{C,C'}$ is the edge set such that $E_{C,C'} = \{\{u,v\} \in E : u \in C \cup C' \text{ and } v \in C \cup C'\}$
- $e = |E_{C,C'}|$: The cardinality of $E_{C,C'}$
- $d(v,S)$: The degree of a vertex v in a subgraph S of G
- $d(C,S)$: The community degree defined as the sum of degrees $d(v,S)$ of vertices v belonging to C, i.e., $d(C,S) = \sum_{v \in C} d(v,S)$

Then, the parametric modularity $Q_\lambda(C,C')$ of the clustering $\{C,C'\}$ in the graph $G_{C,C'}$ is given by [Reichardt and Bornholdt, 2006]:

$$Q_\lambda(C,C') = \frac{l(C) + l(C')}{e} - \lambda \frac{d(C, G_{C,C'})^2 + d(C', G_{C,C'})^2}{4e^2} \tag{2}$$

Parameter λ goes from 0 to potentially an infinite positive value. A special case is $\lambda = 1$, corresponding to the standard modularity $Q = Q_1$.

The general idea of the fusion condition is to define a criterion that prohibits or authorizes a merge between two communities. Our preliminary tests showed that a fusion condition based on the parametric modularity is relevant. The principle is to prohibit the fusion if the parametric modularity exceeds a threshold to be fixed. For a constant θ (a threshold) between 0 and 1 inclusive, the fusion condition $M_{\lambda,\theta(C,C')}$ that we propose is defined as follows:

$$Q_\lambda(C,C') < \theta \tag{3}$$

This indicates that communities C and C' are allowed to be merged if the parameter modularity for the clustering $\{C,C'\}$ in the subgraph $G_{C,C'}$ is strictly below the threshold θ. Otherwise (i.e., if $Q_\lambda(C,C') \geq \theta$), C and C' must not be merged.

This formulation requires two parameters: λ the resolution of Q and θ the threshold, and five variables: $l(C), l(C'), d(C,G_{C,C'}), d(C',G_{C,C'}), e$. To characterize the values of θ, we first need to reformulate the fusion condition by reducing the number of variables and parameters.

2.2. Reformulation

Let $d_{out}(C)$ be the external degree of community C, i.e., the number of edges having an end point in C and the other outside. We have $d(C) = 2l(C) + d_{out}(C)$ since in $d(C)$, the internal edges are counted twice. In the context of subgraph $G_{C,C'}$, edges counted in $d_{out}(C)$ necessarily connect a node in C to a node in C', thus $d_{out}(C) = d_{out}(C') = l(C,C')$ which is the number of edges between C and C'. We deduce three important relations which are used below.

$$\begin{cases} d(C,G_{C,C'}) = 2l(C) + l(C,C') \\ d(C',G_{C,C'}) = 2l(C') + l(C,C') \\ l(C) + l(C') + l(C,C') = e \end{cases} \tag{4}$$

As $d(C,G_{C,C'}) = e + l(C) - l(C')$ and $d(C',G_{C,C'}) = e - l(C) + l(C')$, the parametric modularity can be rewritten as:

$$\begin{aligned} Q_\lambda(C,C') &= \frac{l(C)+l(C')}{e} - \frac{\lambda}{4e^2}\left((e+l(C)-l(C'))^2 + (e-l(C)+l(C'))^2\right) \\ &= \frac{l(C)+l(C')}{e} - \frac{\lambda}{2e^2}(l(C)-l(C'))^2 - \frac{\lambda}{2} \end{aligned} \tag{5}$$

This form only requires three variables: $l(C)$, $l(C')$ and e.

2.3. Formulation of θ

To bound θ, we consider two extreme cases. The first case corresponds to the situation where communities C and C' are not connected by an edge while the other case occurs when one community has only one vertex. In both cases, we consider e, the number of edges in the subgraph induced by $C \cup C'$, as a constant and we study all the possible variations of $l(C)$ and $l(C')$ between 0 and e.

2.3.1. Upper bound of θ

Suppose that communities C and C' are not connected, i.e., $l(C,C') = 0$. By Eq. (4) we have $l(C) + l(C) = e$, so $l(C) - l(C) = 2l(C) - e$. The parametric modularity of Eq. (5) becomes:

$$\begin{aligned} Q_\lambda(C,C') &= 1 - \frac{\lambda}{2e^2}(2l(C)-e)^2 - \frac{\lambda}{2} \\ &= \frac{-2\lambda l(C)^2}{e^2} + \frac{2\lambda l(C)}{e} + 1 - \lambda \end{aligned} \tag{6}$$

For a fixed value of e, the modularity can be seen as a function of $l(C)$ defined on $\{0,\ldots,e\}$. This function is increasing on the interval $[0\ldots e/2]$ and decreasing on $[e/2\ldots e]$. Moreover, for $l(C) = 0$ and $l(C) = e$, $Q_\lambda(C,C') = 1-\lambda$. Thus we have $Q_\lambda(C,C') \geq 1-\lambda$. Under the given hypothesis ($l(C,C') = 0$), the criterion must be unsatisfied and thus C and C' should never be merged. We must choose θ such that $Q_\lambda(C,C') \geq \theta$ for two unconnected communities C and C'. This holds when $\theta \leq 1-\lambda$ because in this case $Q_\lambda(C,C') \geq 1-\lambda \geq \theta$.

2.3.2. Lower bound of θ

Now suppose that C and C' are connected communities, thus $l(C,C') > 0$. By this hypothesis, the two communities could be merged. Suppose also that one community, say C', contains only one vertex. Thus C' has no internal edge, i.e., $l(C') = 0$. It follows that $e = l(C) + l(C,C')$ and, according to the first hypothesis, $l(C) < e$. The modularity used in the fusion condition becomes:

$$Q_\lambda(C,C') = -\frac{\lambda}{2e^2}l(C)^2 + \frac{l(C)}{e} - \frac{\lambda}{2} \tag{7}$$

As previously, the modularity can be seen as a function of $l(C)$ defined on $\{0,\ldots,e\}$. This function is strictly increasing with bounds $-e/\lambda$ for $l(C) = 0$ and $1-\lambda$ for $l(C) = e$. So $Q_\lambda(C,C') \leq 1-\lambda$ and, with the hypothesis $l(C) < e$, we have $Q_\lambda(C,C') < 1-\lambda$. According to the second hypothesis that C' contains a single vertex, it is preferable not to forbid a fusion between C and C' and to base the fusion decision on the examination of the modularity gain. It is possible that the single vertex of C' has to be in C with a justified gain of modularity. Thus, the fusion condition has to be positive and we must choose θ such that $Q_\lambda(C,C') < \theta$. It is satisfied if $\theta \geq 1-\lambda$ because in this case $Q_\lambda(C,C') < 1-\lambda \leq \theta$.

2.3.3. Value of θ

Thanks to the hypothesis of connection between C and C', we have $\theta \geq 1-\lambda$ instead of $\theta > 1-\lambda$. According to the bounds of θ, the only valid value is $\theta = 1-\lambda$. The fusion condition becomes thus:

$$Q_\lambda(C,C') < 1-\lambda, \text{ with } \lambda > 0 \tag{8}$$

The hypothesis of connection between C and C' is not restrictive because in the case that C and C' are not connected by an edge, $l(C) = e$ so $Q_\lambda(C,C') = 1-\lambda$. By the fusion condition, the fusion is correctly rejected, avoiding unnecessary fusion examinations between two communities sharing no links.

3. GENERAL STUDY

Our fusion condition now relies on a single parameter λ. It is important to study under what conditions the fusion condition performs well and reduces effectively the effect of the resolution limit. The general idea is to use the fusion condition as a filter before evaluating the modularity gain induced by a potential fusion between two communities.

We use in the following a formulation of the fusion condition without e in the denominator:

$$\frac{l(C)+l(C')}{e} - \frac{\lambda}{2e^2}(l(C)-l(C'))^2 - \frac{\lambda}{2} < 1-\lambda$$
$$2e(l(C)+l(C')) - \lambda(l(C)-l(C'))^2 + (\lambda-2)e^2 < 0 \tag{9}$$

3.1. Admissible values of λ

We first examine the minimum value of the parametric modularity $Q_\lambda(C,C')$, based on Eq. (9). Let $l(C)-l(C') = h$, h ranging from $-e$ to e. The left term in Eq. (9) becomes:

$$Q_\lambda(C,C') = 4el(C) - 2eh - \lambda h^2 + (\lambda-2)e^2 \tag{10}$$

For a constant h, this function of $l(C)$ is increasing, so its minimum is reached for $l(C) = 0$ and equals to $2eh - \lambda h^2 + (\lambda-2)e^2$. By varying h from $-e$ to 0 (under the condition $l(C) = 0$, this is the range of h), this term has two minimum: 0 for $h = -e$ and $(\lambda-2)e^2$ for $h = 0$. The first minimum is irrelevant because the fusion condition should be always false, according to Eq. (9). With the second minimum, the fusion condition could be true only if $\lambda < 2$. This is an important bound for λ parameter.

3.2. Malformed communities

The weakest condition for the existence of a community [Hu et al., 2008] requires that the community has at least twice internal edges of the edges shared with any other community. For the case of only two communities C and C', this condition requires that $2l(C)$ and $2l(C')$ be strictly greater than $l(C,C')$.

We focus on one of these communities, for instance C, the other situation being symmetric. Two cases may arise inducing an expected behavior of the fusion condition.

1. Community C is well formed: $2l(C) > l(C,C')$. The fusion condition must fully operate independently of its value. At the extreme case, when $l(C,C') = 1$ (i.e., C and C' are linked by a single edge), we can expect the fusion to be prohibited (see Section 5.).

2. Community C est malformed: $2l(C) \leq l(C,C')$. In this case, it is useless to prevent the resolution limit because the fusion of C and C' is *a priori* legitimate. The final decision for the fusion relies on the modularity gain induced by the fusion.

We check now in which condition the fusion condition is always satisfied if $2l(C) \leq l(C,C')$. Let $l(C,C') = hl(C)$ with $h \geq 2$. As $e = (1+h)l(C) + l(C')$, the fusion condition is given jointly by the following three inequalities:

$$\begin{aligned}
&[(2\lambda-2)h+4\lambda]l(C)l(C') + [(\lambda-2)h^2+(2\lambda-2)h]l(C)^2 < 0\\
&[(2\lambda-2)h+4\lambda]l(C') + [(\lambda-2)h^2+(2\lambda-2)h]l(C) < 0\\
&[(2\lambda-2)h+4\lambda]l(C') < [(2-\lambda)h^2+(2-2\lambda)h]l(C) \qquad (11)
\end{aligned}$$

The fusion condition must be true for any positive values of $l(C)$ and $l(C')$. For that, in the previous inequality, $(2\lambda-2)h+4\lambda$ must be negative and $(2-\lambda)h+2-2\lambda$ must be positive.

The first inequality is equivalent to $h < 4\lambda/(2-2\lambda)$ if $\lambda \geq 1$ or $h > 4\lambda/(2-2\lambda)$ if $\lambda < 1$. The first condition is impossible because if $\lambda \geq 1$ then $4\lambda/(2-2\lambda) < 0$, that requires $h < 0$. It remains $h > 4\lambda/(2-2\lambda)$. By the same reasoning, as $h \geq 2$, the second condition is true making λ such that $2 > 4\lambda/(2-2\lambda)$, i.e., $\lambda < 1/2$ (compatible with the initial condition $\lambda < 1$).

The second inequality is equivalent to $h > (2\lambda-2)/(2-\lambda)$ if $\lambda \leq 2$ (the other possibility is impossible because $\lambda < 2$ holds). By the same method, we obtain $\lambda < 3/2$.

Given these two joint inequalities ($\lambda < 1/2$ and $\lambda < 3/2$), one must adopt the most restrictive inequality (i.e., $\lambda < 1/2$) in order for the fusion condition to operate and allow a malformed community to pass the filter of the fusion condition and possibly to take part in a fusion. Notice that the best λ value for the fusion condition is unknown and will be approximated experimentally in Section 4..

4. EXPERIMENTATION

We have established the following fusion condition for two candidate communities: C and C' can be merged if $Q_\lambda(C,C') < 1-\lambda$ holds for some $0 < \lambda < 1/2$. The lower limit 0 for λ is trivial because the fusion condition is falsified for $\lambda < 0$. On the other hand, as discussed in Section 3.2., the upper limit of 1/2 implies that beyond this value a malformed community can no more be merged with a neighboring community because of the fusion condition. In other words, this value must not be exceeded if we want to authorize the fusion of a malformed community.

As shown in Sections 2. and 3., the bounds $0 < \lambda < 1/2$ are sound since they have been derived formally. However, the best (optimal) value of λ cannot be determined theoretically. In this section, we will carry out experiments to study the effect of the fusion condition for community identification with various values of the λ parameter. This allows us to identify an approximate best λ value for the fusion condition. For this, we will test λ with values going from 0 to 1 inclusive with a step of 0.1 and observe their effects on the fusion condition. Using a range of values larger than $0 < \lambda < 1/2$ also aims to study the λ parameter within its theoretical bounds as well as some values around these bounds.

Notice that we also consider the two special cases $\lambda = 0$ and $\lambda = 1$. If $\lambda = 0$, we have $Q_\lambda(C,C') < 1$ and thus the fusion condition is trivially satisfied. On the other hand, if $\lambda = 1$, we have $Q_1 = Q$, i.e., the fusion condition degenerates to the standard modularity. Moreover, the fusion condition becomes $Q(C,C') < 0$, enabling very few fusions. In practice, according to our observations with the Louvain algorithm, this later case corresponds to a quasi lack of fusion in the second phase, i.e., the algorithm reduces to the vertex mover procedure (i.e., the first phase of the Louvain algorithm).

4.1. Testing the fusion condition within the Louvain algorithm

One of the fastest and most popular methods for modularity optimization is the Louvain algorithm [Blondel et al., 2008]. Like other agglomerative methods, this algorithm relies on two operators that are applied in two phases, starting with a trivial clustering where each vertex of the initial graph defines a community. In the first phase, each vertex is examined and moved to another community if this move increases the modularity. This vertex move procedure is repeated until no modularity increase is possible. Then in the second phase, a coarsen graph G^1 is created where each vertex of G^1 represents a community and two vertices are linked by an edge if the corresponding communities are connected. The Louvain algorithm repeats these two phases until no vertex can be moved in the last coarsen graph.

Since the Louvain algorithm operates on multi-level coarsen graphs, each vertex move corresponds to a possible fusion of two communities. In other words, community fusion is the key operation of the Louvain algorithm, making it very appropriate for our purpose of assessing the usefulness of the proposed fusion condition.

For our experimentation, we implemented the fusion condition with an improved Louvain algorithm presented in [Gach and Hao, 2013]. Compared to the standard Louvain algorithm, the improved version integrates a (third) uncoarsening phase which unfolds each vertex (a composed community) of a coarsen graph into its composing communities. Then the vertex move procedure is again applied to the uncoarsened graph to further improve the modularity. We use the fusion condition with this improved Louvain algorithm such that during the second phase, each vertex move is first filtered by the fusion condition. Indeed, in the coarsen graph, a vertex stands for a composite community from the previous level. The move of such a vertex is equivalent to a fusion between two communities. We call this improved Louvain algorithm ’Louvain’ while we call the algorithm integrating the fusion condition ’Louvain-CF’ (conditional fusion).

4.2. LFR benchmark graphs

As put forward in [De Meo et al., 2013], testing an algorithm on real-life networks is problematic because neither the ground truth communities nor the features of these networks are known. For our tests, we need to consider a set of artificially generated networks with known structural properties being compliant with those existing in real networks. For this, we use graphs generated according to the LFR model presented in [Lancichinetti et al., 2008]. The LFR model provides realistic complex networks with power law distribution of vertex degree. Based on a few parameters, we use the LFR model to generate a wide variety of graph types with special characteristics like low or high density, small or large range of community sizes, good or poor internal density in communities. To generate a LFR graph we have to define:

1. n and m, numbers of vertices and edges of G;
2. avg_deg and max_deg average and maximum vertex degrees;
3. δ_v and δ_c exponents of power law distribution of vertex degree (δ_v) and community size (δ_c);
4. μ, the mixing parameter, which designates for each vertex, the fraction of links shared with vertices in other communities (the vertex has the fraction $1-\mu$ of its links with other vertices of its community);
5. $min|S_i|$ and $max|S_i|$ sizes of the smallest and largest community;

Other characteristics are automatically determined by the LFR generator like min_deg the lowest vertex degree and k the number of communities.

Instead of generating many graphs with same parameters, we choose to use a small number of 20 LFR graphs which represent different types of complex networks. Table 1 summarizes the characteristics of the complexe networks covered by the tested instances.

As shown in Table 1, each graph is defined by 11 parameters among which avg_deg, μ, $min|S_i|$ and $max|S_i|$ are the most important ones. These instances cover three large families of graphs (networks).

- Graphs rather dense ($avg\_deg \geq 15$) whose communities are not well defined ($\mu > 0.5$): $\#5, \#8, \#11, \#12, \#15, \#16, \#18$;
- Graphs rather sparse ($avg\_deg \leq 10$) whose communities are well defined ($\mu \leq 0.5$): #1 to #4, $\#7, \#9, \#14, \#17$;
- Graphs with large communities, average density and average belongingness of node to community ($10 \geq avg\_deg \leq 20, 0.4 \geq \mu \leq 0.5$): $\#10, \#13, \#19, \#20$.

Graph #6 cannot be classified and does not belong to the above cases, ours tests show this graph is easy to partition.

4.3. Experimental protocol

We implement the tested algorithm in Free Pascal and executed on a PC equipped with a Pentium Core i7. In the LFR model, a graph and its (optimal) solution (i.e., the best clustering) are generated together according to the fixed parameters. To compare a clustering from our algorithm and the reference solution, we use as our main indicator the Normalized Mutual Information (NMI) [Ana and Jain, 2003], which is based on the information theory and has proven its reliability in complex network applications [Danon et al., 2005].

Table 1. The set of 20 LFR benchmark graphs with detailed parameters and characteristics: graph identification, number of vertices, number of edges, minimum, average and maximum degree, number of communities (k), lowest and largest community size, distribution exponent of vertex degree (δ_v) and community size (δ_c) and mixing parameter μ (average fraction of external links for a vertex)

Graph	n	m	*min_deg*	*avg_deg*	*max_deg*	k	$\min\lvert S_i\rvert$	$\max\lvert S_i\rvert$	δ_v	δ_c	μ	comments
#1	80	237	4	6	12	10	4	18	2	1	0.3	
#2	1000	3353	3	7	30	32	3	100	2	1	0.3	
#3	1000	3436	3	7	30	92	3	30	2	2	0.4	
#4	1000	5000	10	10	10	100	10	10	2	1	0.5	same degree for all vertices, same size for all communities
#5	1000	7625	7	15	51	32	15	80	2	1	0.6	
#6	1000	7692	7	15	50	36	15	57	2	1	0.4	comparable to #5 but with better definition of communities
#7	5000	21911	5	9	100	50	10	328	3	1	0.5	
#8	2000	29878	13	30	100	58	10	81	2	1	0.7	
#9	10000	34742	3	7	30	906	2	30	2	2	0.4	comparable to #3, 10 times bigger
#10	10000	76710	7	15	50	13	510	988	2	1	0.4	only large communities
#11	10000	77051	7	15	50	193	20	118	2	1	0.7	
#12	5000	124908	29	50	100	65	20	182	2	1	0.8	
#13	20000	146850	8	15	50	275	10	459	3	2	0.4	
#14	50000	172854	3	7	30	984	4	200	2	2	0.3	low density with many small communities
#15	8000	201088	34	50	100	35	57	534	3	1	0.8	very bad definition of communities but high density and none small communities
#16	30000	219864	5	15	100	699	10	912	2	2	0.6	broad extent of community size, many small communities, bad definition of communities
#17	50000	248663	6	10	50	1144	10	938	3	2	0.5	close to #16 but many vertices have a small degree
#18	8000	320836	55	80	150	78	20	287	3	1	0.85	
#19	50000	476468	7	19	100	1224	5	10371	2	2	0.5	
#20	100000	977547	10	20	50	252	100	999	2	1	0.5	

Given two clustering solutions P and P', the NMI indicator quantifies the similarity between the two solutions based on the shared information. The NMI value which varies from 0 (totally different) to 1 (identical) is calculated as follows.

$$NMI(\mathcal{P},\mathcal{P}') = \frac{2I(\mathcal{P},\mathcal{P}')}{H(\mathcal{P})+H(\mathcal{P}')} \tag{12}$$

In this formula, $I(\mathcal{P},\mathcal{P}') = H(\mathcal{P}) - H(\mathcal{P}|\mathcal{P}')$ measures, according to the information theory, how much we know about $\mathcal{P}$, knowing $\mathcal{P}'$. For a given partition $\mathcal{P} = \{C_1, C_2, ..., C_k\}$, $H(\mathcal{P})$ $= -\sum_{i=1}^{k} \frac{C_i}{n} \log(\frac{C_i}{n})$ denotes the entropy applied to P.

In addition to the NMI global measure, we also observe three structural indicators: number of communities k , size of the smallest community $min|C_i|$ and size of the largest community $max|C_i|$. Like many other algorithms, Louvain is sensitive to the order of processing the vertices. To ensure a reliable comparison between the two algorithms (Louvain and Louvain-CF), we generate 100 instances of each graph with a random reordering of vertices. For all presented tests, the quantity (NMI, k, $max|C_i|$ and $min|C_i|$) associated with a graph is the average of these values over these 100 instances.

4.4. Time complexity

In the best implementation of the Louvain algorithm, the value of $l(C)$ which represents the number of edges within community C is recorded for each community and updated incrementally. This allows the gain of a vertex move or a community merge to be calculated quickly, leading to a low complexity $O(m)$ (m = nummber of edges) of the whole algorithm. In the formulation (5) of the fusion condition, there are only two variables $l(C)$ and $l(C')$. Therefore, the fusion condition can be evaluated in linear time and does not change the complexity of the Louvain algorithm with the fusion condition (i.e., always in $O(m)$). In practice, our tests confirm this observation and indicate an increase of some 7% of the run time when the fusion condition (with $\lambda = 0.3$) is used compared to the standard Louvain algorithm ($\lambda = 0$).

4.5. General results

Figure 2 shows the average NMI and average relative errors over 100 executions on all graphs. For a clustering $\{C_1, C_2, ...C_k\}$ found by an execution, relative errors are computed by the formula '(value_found - expected_value) / expected_value' for the number of communities k, the minimum $min(|C_i|)$ and maximum $max(|C_i|)$ of community sizes. Thus a perfect error rate is 0, a positive (negative) value indicates a too big (small) value according to the reference solution.

About the relevance of the found clustering, measured by NMI, we observe a maximum and positive effect of the fusion condition with $\lambda = 0.3$. Comparing to the Louvain algorithm ($\lambda = 0$), the increase of NMI is almost 0.1 for $\lambda = 0.3$. After this threshold, NMI goes down but still remains above the NMI of Louvain. This observation is confirmed by a best compromise between the number of communities and the minimum size on the interval $[0.2, 0.3]$ on λ, near an ideal relative error of 0. The most significant effect is observed on the minimum size going from a relative error of 7 for $\lambda = 0$ to almost 0 for λ in this interval.

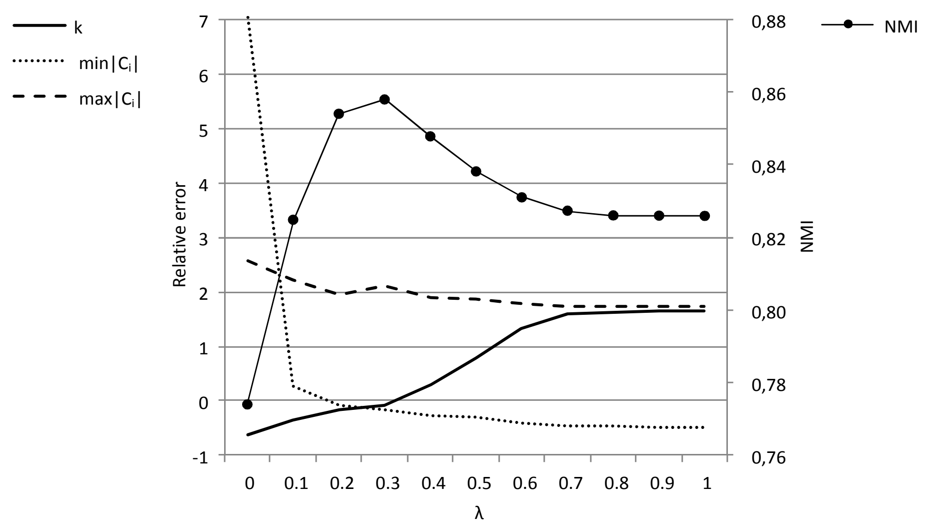

Figure 2. Influence of the λ parameter on community range and NMI. Relative errors represent the relative distance between a value found by an algorithm and the expected value from the reference solution. We indicate on the left x-axis the number of communities (k), the size of the minimum community ($min|C_i|$) and the size of the maximum community of communities ($max|C_i|$). The NMI between a found clustering and the reference solution is represented on the right x-axis. All represented values are an average over 100 instances of each of the 20 benchmark graphs. Note that $\lambda = 0$ corresponds to a total permissive criterion i.e., the classic Louvain algorithm, while $\lambda = 1$ corresponds to a lack of fusion i.e., an execution with only the first phase of Louvain.

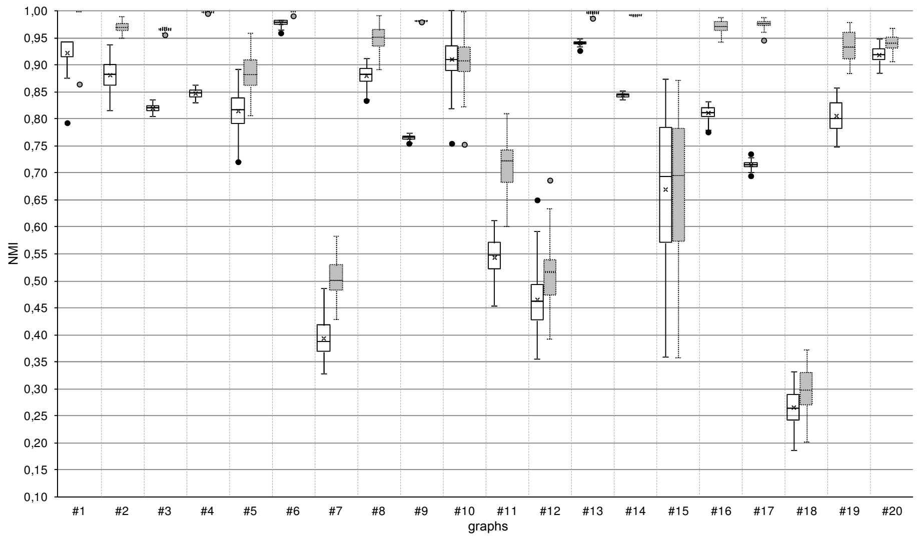

Figure 3. Comparison results based on NMI between the Louvain algorithm with and without the fusion condition. The box and whisker plot show distributions of NMI over the 100 instances for the 20 benchmark graphs. For each graph (indicated in a column), the white box plot (left) corresponds to the Louvain algorithm without the fusion condition and the gray box plot (right) is for the Louvain version with the fusion condition (Louvain-CF) using $\lambda = 0.285$.

This demonstrates the positive effect of the fusion condition to reduce the resolution limit on the tested graphs.

A problem remains on large communities because although the relative error is slightly improved by the fusion condition, this error still remains around 2 (i.e., the biggest community is, on average, 3 times bigger than the reference solution). The important fact to note is that this under-performance is present from the first *vertex-mover* phase of the Louvain algorithm as evidenced by the result for $\lambda = 1$. Errors on big communities can appear in a clustering during the first phase, and the fusion condition is helpless in this case as we discuss later in Section 4.7..

4.6. Detailed results

We now explore detailed and structural results for each graph with parameter $\lambda = 0.285$, near the best approximate value found here ($\lambda = 0.3$), which represents an important value as we discuss in this section.

From the detailed results presented on Fig. 3 we observe three situations about the performance of the fusion condition:

1. No improvement with the same range of NMI for 2 graphs #10 and #15,

2. A significant but relative improvement with overlap distributions of NMI for 7 graphs #5, #7, #8, #11, #12, #18 and #20,

3. A significant improvement with a tight distribution of NMI between 0.9 and 1 for the 11 remaining graphs.

To explain the relatively small improvements on the 9 graphs (cases 1 and 2), we highlight the fact that these graphs are generated with $\mu > 0.5$, and thus have only weak communities. As such, during the first phase of the Louvain algorithm, ’faulty’ (too big) communities are formed by moving vertices poorly attached to a community. Unfortunately, the resolution limit occurs during this phase and the Louvain algorithm alone has no way to prevent the appearance of these ’faulty’ big communities. In Section 4.8., we provide more discussions about this issue.

4.7. Structural results

In this section, we are interested in an analysis of the results from the structural point of view in terms of three indicators: the number of communities, the size of the minimal community and the size of the maximal community (see Fig. 4). As we can observe easily, for most of the graphs, the fusion condition helps improve the number of communities and the size of the minimum community. For 14 graphs, the error on the number of communities is almost 0, i.e., the algorithm with the fusion condition (Louvain-CF) finds the correct number of communities. For all graphs, the number found with the fusion condition is better (close to 0) than Louvain. Precisely, the relative error is between -0.61 (graph #18) and 0.55 (graph #7) for Louvain-CF, while it is between -0.92 (graph #9) and -0.17 (graph #6) for Louvain.

On the second graphic of Fig. 4, concerning the minimum size of communities, we observe clearly the bad effects of the resolution limit with Louvain (all errors are above 0, 10 graphs with an error greater than 2). Here again, the fusion condition brings a very good correction with almost no error for 13 graphs and an error ranges between -0.88 (graph #11) and 0.50 (graph #9). Poor results with the fusion condition (graphs #7, #9, #11, #14, #16 and #17) are due to a low density revealed by a relatively low average vertex degree (see Table 1). However, the improvement over Louvain is remarkable because small communities stay in the clustering thanks to the fusion condition.

The third graphic of Fig. 4 shows that the curve of the vertex mover procedure is close to the curve with the fusion condition. This is caused by the phenomenon described above such that large communities may be formed early in the algorithm, during the first phase. The fusion condition is unable to break these too large communities. Despite this comment, we observe a good improvement with the fusion condition for graphs #2, #3, #4, #6, #9 and #14, compared to Louvain. For all other graphs, except #17, the results are the same between Louvain and Louvain-CF. Graph #17 is the only problematic graph for the fusion condition with an average size of the largest community twice greater than for Louvain, which itself is twice greater than the reference size. Strangely, the fusion condition, which is supposed to prevent some wrong fusions, produces a larger community. It is delicate to give an explanation of this behavior, but we guess that the condition, while prohibiting some wrong fusions, could promote other community fusions that lead to a very big community. Note that the global performance on #17 graph, measured by an average NMI around 0.967 (1 is the best possible value), is very good. We observe that the problem of big community concerns only one or two communities that contains well formed sub-communities existing in the reference solution. This observation of an under-performance means that the fusion

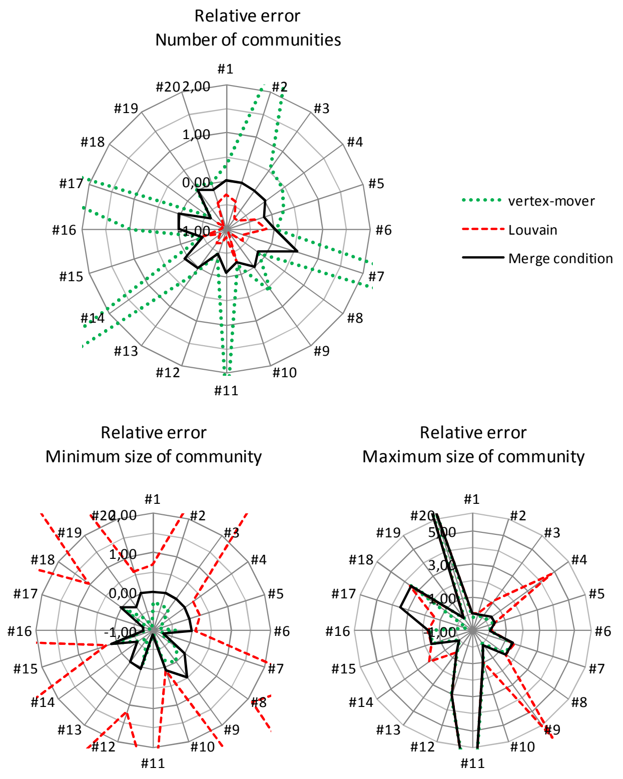

Figure 4. Compared structural elements between vertex-move (i.e., Louvain’s first phase), Louvain and Louvain-CF. A radar graphic shows the average value, over the 100 instances, for each graph in disk section. Structural values are measured on clusterings found after the first phase of the Louvain algorithm (vertex-move), the second and last phase (Louvain) and by the Louvain algorithm with the fusion condition (Louvain-CF). For each clustering, we measure the relative error (see Fig. 2) for the number of communities and the sizes of the minimum and maximum communities.

condition is not an absolute condition for good fusions. An exploration of hierarchical clusterings given by the algorithm could be necessary.

4.8. Inconsistency of solutions

Methods based on modularity also suffer from, in addition to the resolution limit, another problem, i.e., inconsistency of solutions. For the same graph, the scores of modularity obtained by changing the examination order of the nodes can be very different. These differences generally affect the peripheral nodes whose belongingness to a community is uncertain, as shown in studies on stable community cores [Seifi et al., 2013]. All agglomerative algorithms like Louvain are sensitive to the examination order of nodes and thus may suffer from this inconsistency problem. This makes the interpretation of a partition provided by the algorithm difficult and requires a more detailed analysis, for example through the community cores. The fusion condition proposed in this work has the advantage of greatly alleviating this inconsistency problem because the fusion condition reduces the possible combinations of merged communities and the number of really merged communities. However, the examination order of the nodes still impacts the outcomes of a Louvain-like algorithm even with the fusion condition. To study the sensitivity of this impact, we conducted an experiment on 20 real-life graphs including those used in [Gach and Hao, 2013] with 20 runs of the standard Louvain algorithm (Louvain) and the Louvain algorithm with the fusion condition (Louvain-CF), each run being based on a random node examination order. This leads to 20 different solutions for each graph and each algorithm. Then we made a pairwised comparison and calculated the average NMI using the $20*19 = 380$ comparison scores for each graph. Finally, we summarize the calculation by the average NMI over the 20 graphs. Ideally, if the partitions are the same, then we have NMI = 1. With Louvain, We obtain an average NMI of 0.768 while this value becomes 0.879 with the fusion condition ($\lambda = 0.3$), indicating a significant reduction in the inconsistency of solutions.

4.9. A case study on a real network

It is generally difficult to observe or measure the effect of a method on real graphs for which ground truth solution is unknown [De Meo et al., 2013]. Yet, it is interesting to investigate some real networks. To show the positive effect of the fusion condition on real graphs, we chose to plot *erdos* collaboration graph containing 6,927 nodes and 11,850 edges [Grossman, 2007]. The test we show here has no demonstration value because it covers only a single run on a single instance, but it still allows to measure the positive effects of the fusion conditions. In Figure 5, we show the distribution of the sizes communities contained in the solutions obtained respectively by Louvain and Louvain-CF. Louvain finds 33 communities while Louvain-CF proposes more than 300 communities. The range of community sizes is 16-705 for Louvain against 3-216 for Louvain-CF. The figure clearly shows a power law distribution of the community sizes with many small communities and very few large ones.

We found, with artificial graphs, that poor communities identified by Louvain are relatively large compared to smaller communities and have especially a very low density. The density of the subgraph forming a community is an interesting measure to assess the internal cohesion of the community. This density must remain high (i.e., close to be a clique) while having few links with other neighboring communities. This requirement is supposed to be ensured in the Louvain algorithm by the modularity provided that the maximum modularity is not reached. In our tests, Louvain reaches a modularity value of 0.6969 (probably close to the maximum) against 0.6035 for Louvain-CF. As to the community densities, as shown in Figure 6, they are much higher for Louvain-CF than for Louvain. The highest community density from Louvain-CF is nearly 13 times greater than that from Louvain (0.25 against 0.0183). Similarly, the lowest density with Louvain-CF, characterizing in general the large communities, is still 5 times larger compared to Louvain.

The fusion condition therefore significantly improves the community density by providing much smaller communities, and consequently the internal coherence of the identified communities. These communities probably make more sense to the user who could merge some neighboring communities if needed. Moreover, the λ parameter plays the role of cursor to the desired size and cohesion of communities from 0 (the same behavior as Louvain)

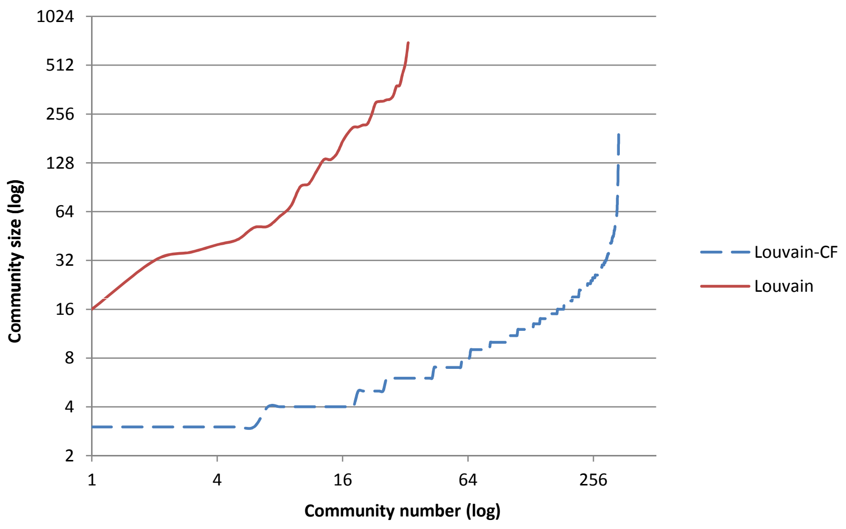

Figure 5. Distribution of the community sizes of solutions obtained by Louvain and Louvain-CF ($\lambda = 0.285$) on *erdos* collaboration graph [Grossman, 2007].

and 1 (no merge, only *vertex-mover* applies).

5. RESOLUTION LIMIT REVISITED

We now study the case where the fusion condition is false for two communities C and C' linked by only one edge. We have $l(C,C') = 1$, i.e., $e = l(C) + l(C') + 1$. Let h be a value such that $l(C') = hl(C)$. Then from Eq. (9), we obtain the condition where the fusion condition becomes false:

$$(4\lambda l(C) + 2\lambda - 2)h + 4\lambda l(C)^2 + 4(\lambda - 1)l(C) + \lambda - 2 \geq 0 \tag{13}$$

The form of the left expression is $Ah + B$ with $A = 4\lambda l(C) + 2\lambda - 2$ and $B = 4\lambda l(C)^2 + 4(\lambda - 1)l(C) + \lambda - 2$.

Without loss of generality, we consider the case where C is the smallest community, thus $h \geq 0$. If $h = 0$, the sign of B is the sign of $2(l(C)+1)(2\lambda l(C) + \lambda - 2)$ by factorization, i.e., the sign of $(2\lambda l(C) + \lambda - 2)$ as $l(C) + 1$ is positive. Thus, assuming that h is zero, the fusion condition is false if $(2\lambda l(C) + \lambda - 2) \geq 0$, i.e., $l(C) \geq 1/\lambda - 1/2$.

Now if h is increasing (by hypothesis, h is positive), the function $Ah + B$ remains positive only if A is positive, i.e., $l(C) \geq 1/(2\lambda) - 1/2$. This condition is already covered by the previous one, $l(C) \geq 1/\lambda - 1/2$, because $\lambda > 0$. Therefore, the fusion condition is false if $l(C) \geq 1/\lambda - 1/2$. It is impossible to satisfy this condition for any value of λ because $l(C)$ has to be as much greater as λ is close to zero. Practically, we must set a bound K such that $l(C) \geq K$. Under that condition, the fusion of C and C', linked by only one edge, is forbidden if $\lambda \geq 2/(2K+1)$. A reasonable value for K is 3, which requires $\lambda \geq 2/7 \approx 0.285$. This value, which is theoretically obtained to prevent the resolution limit in an extreme case, confirms the experimental results (best behavior around $\lambda = 0.3$, see Sections 4.5.–4.7.).

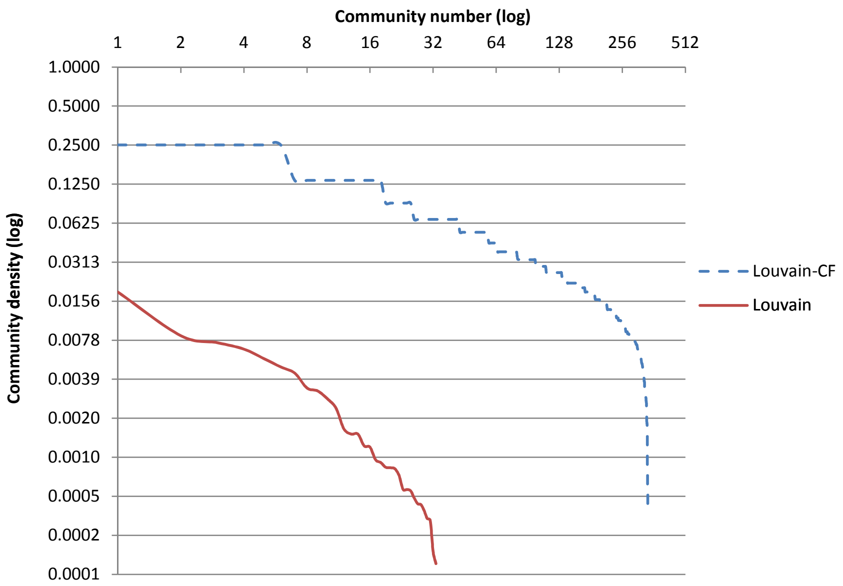

Figure 6. Distribution of the community densities of solutions obtained by Louvain and Louvain-CF ($\lambda = 0.285$) on *erdos* collaboration graph [Grossman, 2007].

It is important to note that C is not necessarily a clique. In the worst case, the community C with K edges could have $K+1$ nodes of a degree no greater than 2 (the community is a chain). The prohibition of any fusion of C is meaningless and probably wrong. An algorithm implementing the fusion condition should be able to provide communities with a strong density. Louvain is a good candidate because its first phase produces some well formed communities, unlike for instance the FastGreedy [Clauset et al., 2004] algorithm that uses fusions at the beginning with very primitives communities.

6. PERSPECTIVES

We showed the effectiveness of the fusion condition to reduce the effect of the resolution limit in the Louvain algorithm. On the other hand, the detailed structural analysis of Section 4.7. shows that the algorithm equipped with the fusion condition cannot fully avoid the resolution limit. This is because for a sparse graph, large communities can still be formed by the vertex mover procedure (first phase of the Louvain algorithm). To further alleviate the resolution limit, other solutions can be envisaged. First, we can modify the first phase of the Louvain algorithm (vertex mover) by using the parametric modularity (instead of the standard modularity) to reduce the size of the communities. The second solution is based on the idea of breaking a large community into two (smaller) communities and then applying the fusion condition to know if these smaller communities should really be merged. According to our preliminary tests, this method seems promising. Yet, it increases the time complexity of the algorithm that is no longer linear. Another possibility is to limit the size of communities when two communities are merged like some recent studies [Ciglan and Nørvåg, 2010, Meyerhenke et al., 2014]. Finally, it would be interesting to study the possibility of combining the fusion condition with other community detection approaches like graph-based techniques. In any case, the fusion condition could be beneficially applied to decide whether two communities could be merged whenever such a decision is to be made.

7. CONCLUSION

Algorithms which optimize the modularity for community detection in complex networks suffer from the resolution limit which tends to dissolve small communities in big ones. Analyses of the well-known Louvain algorithm disclose that it makes wrong fusions of communities in the second phase of this algorithm.

We have established a fusion condition that helps decide if two communities should be merged. This condition uses the parametric modularity λ computed between two candidate communities for fusion. Our theoretical study and computational assessment of the fusion condition applied to Louvain show a significant reduction of the resolution limit around the value 0.3 for λ. The relevance of clustering, measured by NMI (from 0 bad to 1 perfect), goes from 0.77 for Louvain to 0.86 for Louvain with the fusion condition. The most important effect concerns small communities. The relative error on the size of the smallest community goes from 7 to nearly 0.

This study shows that the fusion condition can alleviate the resolution limit by producing denser and more cohesive communities when it is combined with an agglomerative algorithm like Louvain. The fusion condition is effective only if communities to merge are well formed with a good density. In the Louvain algorithm, such communities are built during the first phase by the vertex mover procedure. As such, wrong fusions of communities are avoided thanks to the fusion condition which is applied during the second phase. Notice that if a cluster is already too big at the end of the Louvain algorithm, the problem can only be solved by a slit operation (not by a fusion operation). Such a situation appears mostly in sparse graphs which may contain some very large and loosely formed communities where the fusion condition alone will be helpless.

In this work, we have focused on the popular Louvain algorithm for the study of the fusion condition. It would be interesting to test the proposed fusion condition with other agglomerative approaches to verify its effectiveness. More generally, the fusion condition could be used jointly with other community detection approaches as a filter to decide whether two communities could be merged whenever one faces such a decision.

REFERENCES

[Aldecoa and Marín, 2011] Aldecoa, R. and Marín, I. (2011). Deciphering network community structure by surprise. *PLoS ONE*, 6(9).

[Ana and Jain, 2003] Ana, L. and Jain, A. K. (2003). Robust data clustering. Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition, Madison, WI, USA, pages 128–136.

[Backes et al., 2013] Backes, A. R., Casanova, D., and Bruno, O. M. (2013). Texture analysis and classification: A complex network-based approach. *Information Sciences*, 219:168–180.

[Benlic and Hao, 2011] Benlic, U. and Hao, J.K. (2011). A multilevel memetic approach for improving graph k-partitions. *IEEE Transactions on Evolutionary Computation*, 15(5):624–642.

[Benlic and Hao, 2013] Benlic, U. and Hao, J.K. (2013). Hybrid metaheuristics for the graph partitioning problem. In: Talbi EG. (eds) Hybrid Metaheuristics. Studies in Computational Intelligence, vol 434. Springer, Berlin, Heidelberg pages 157–185.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008+.

[Brandes et al., 2008] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2008). On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2):172–188.

[Buluç et al., 2016] Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., and Schulz, C. (2016). Recent advances in graph partitioning. In: Kliemann L., Sanders P. (eds) Algorithm Engineering. Lecture Notes in Computer Science, vol 9220. Springer, pages 117–158.

[Ciglan and Nørvåg, 2010] Ciglan, M. and Nørvåg, K. (2010). Fast detection of size-constrained communities in large networks. In: Chen L., Triantafillou P., Suel T. (eds) Web Information Systems Engineering. Lecture Notes in Computer Science, vol 6488. Springer, Berlin, Heidelberg, pages 91–104.

[Clauset et al., 2004] Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6):066111.

[Danon et al., 2005] Danon, L., Díaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008.

[De Meo et al., 2013] De Meo, P., Ferrara, E., Fiumara, G., and Provetti, A. (2013). Enhancing community detection using a network weighting strategy. *Information Sciences*, 222:648–668.

[Didimo and Montecchiani, 2014] Didimo, W. and Montecchiani, F. (2014). Fast layout computation of clustered networks: Algorithmic advances and experimental analysis. *Information Sciences*, 260:185–199.

[Durugbo et al., 2011] Durugbo, C., Hutabarat, W., Tiwari, A., and Alcock, J. R. (2011). Modelling collaboration using complex networks. *Information Sciences*, 181(15):3143–3161.

[Flake et al., 2002] Flake, G. W., Lawrence, S., Giles, C. L., and Coetzee, F. M. (2002). Self-organization and identification of web communities. *Computer*, 35(3):66–70.

[Fortunato and Barthélemy, 2007] Fortunato, S. and Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41.

[Gach and Hao, 2012] Gach, O. and Hao, J.K. (2012). A memetic algorithm for community detection in complex networks. In: Coello C.A.C., Cutello V., Deb K., Forrest S., Nicosia G., Pavone M. (eds) Parallel Problem Solving from Nature - PPSN XII. Lecture Notes in Computer Science, vol 7492. Springer, Berlin, Heidelberg, pages 327–336.

[Gach and Hao, 2013] Gach, O. and Hao, J.K. (2013). Improving the Louvain algorithm for community detection with modularity maximization. In: Legrand P., Corsini MM., Hao J.K., Monmarché N., Lutton E., Schoenauer M. (eds) Artificial Evolution. EA 2013. Lecture Notes in Computer Science, vol 8752. Springer, pages 145–156.

[Ghosh and Lerman, 2010] Ghosh, R. and Lerman, K. (2010). Community detection using a measure of global influence. In: Giles L., Smith M., Yen J., Zhang H. (eds) Advances in Social Network Mining and Analysis. Lecture Notes in Computer Science, vol 5498. Springer, Berlin, Heidelberg, pages 20–35.

[Girvan and Newman, 2002] Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7821–7826.

[Grossman, 2007] Grossman, J. (2007). The erdös number project. `http://www.oakland.edu/enp/`.

[Guimerà and Nunes Amaral, 2005] Guimerà, R. and Nunes Amaral, L. A. (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900.

[Gulbahce and Lehmann, 2008] Gulbahce, N. and Lehmann, S. (2008). The art of community detection. *BioEssays*, 30(10):934–938.

[Hu et al., 2008] Hu, Y., Chen, H., Zhang, P., Li, M., Di, Z., and Fan, Y. (2008). Comparative definition of community and corresponding identifying algorithm. *Physical Review E*, 78(2):026121.

[Ito et al., 2001] Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., and Sakaki, Y. (2001). A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574.

[Lancichinetti and Fortunato, 2009] Lancichinetti, A. and Fortunato, S. (2009). Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5 Pt 2):056117.

[Lancichinetti et al., 2008] Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110.

[Li et al., 2008] Li, Z., Zhang, S., Wang, R.-S., Zhang, X.-S., and Chen, L. (2008). Quantitative function for community detection. *Physical Review E*, 77(3):036109.

[Liu and Murata, 2010] Liu, X. and Murata, T. (2010). Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications*, 389(7):1493–1500.

[Lü and Huang, 2009] Lü, Z. and Huang, W. (2009). Iterated tabu search for identifying community structure in complex networks. *Physical Review E*, 80(2):026130.

[Meyerhenke et al., 2014] Meyerhenke, H., Sanders, P., and Schulz, C. (2014). Partitioning complex networks via size-constrained clustering. In: Gudmundsson J., Katajainen J. (eds) Experimental Algorithms. SEA 2014. Lecture Notes in Computer Science, vol 8504. Springer, pages 351–363.

[Molloy and Reed, 1995] Molloy, M. and Reed, B. (1995). A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6(2-3):161–180.

[Newman, 2012] Newman, M. (2012). Communities, modules and large-scale structure in networks. *Nature Physics*, 8(1):25–31.

[Newman, 2004] Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133.

[Newman, 2010] Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

[Newman and Girvan, 2004] Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113.

[Pons and Latapy, 2005] Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks. In: Yolum ., Güngör T., Gürgen F., Özturan C. (eds) Computer and Information Sciences5. Lecture Notes in Computer Science, vol 3733. Springer, Berlin, Heidelberg, pages 284–293.

[Reichardt and Bornholdt, 2006] Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, 74(1):1–14.

[Rotta and Noack, 2011] Rotta, R. and Noack, A. (2011). Multilevel local search algorithms for modularity clustering. *Journal of Experimental Algorithmics*, 16:2.1.

[Schuetz and Caflisch, 2008] Schuetz, P. and Caflisch, A. (2008). Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 77(4):046112.

[Seifi et al., 2013] Seifi, M., Junier, I., Rouquier, J.-B., Iskrov, S., and Guillaume, J.-L. (2013). Stable community cores in complex networks. In: Menezes R., Evsukoff A., González M. (eds) Complex Networks. Studies in Computational Intelligence, vol 424. Springer, Berlin, Heidelberg, pages 87–98. Springer.

[Xiang et al., 2016] Xiang, J., Hu, T., Zhang, Y., Hu, K., Li, J.-M., Xu, X.-K., Liu, C.-C., and Chen, S. (2016). Local modularity for community detection in complex networks. *Physica A: Statistical Mechanics and its Applications*, 443:451–459.