On filtering the noise in consensual communities[⋆]

Antoine Huchet[1,2], Jean-Loup Guillaume[1,2], and Yacine Ghamri-Doudane[1,2]

[1] L3i, La Rochelle University, La Rochelle, France
[2] {antoine.huchet,jean-loup.guillaume.yacine.ghamri}@univ-lr.fr

Abstract. Community detection is a tool to understand how networks are organised. Ranging from social, technological, information or biological networks, many real-world networks exhibit a community structure. *Consensual* community detection fixes some of the issues of classical community detection like non-determinism. This is often done through what is called a *consensus* matrix. We show that this consensus matrix is not filled with relevant information only, it is noisy. We then show how to filter out some of the noise and how it could benefit existing algorithms.

Keywords: Consensual Community Detection, Noise, Complex Networks

1 Introduction

Biological, social, technological and information networks can be represented by graphs [1]. Thus, graph analysis has become an important tool to understand such networks. Graphs representing real-world data exhibit particular features that make them far from regular. Few nodes tend to have a lot of neighbours while many nodes tend to have few neighbours. The distribution of edges is not homogeneous: parts of the graph are densely connected, while between such dense parts, there tend to be only a few edges. Such feature of real-world graphs is called *community structure* [2] and finding such densely connected parts of a graph is called *community detection*. In a network of purchase relationship between customers and products, community detection can identify communities of customers with similar interests, thus improve recommendation systems. In social graphs, it could help identify group of people: families, friends or co-workers.

Many community detection algorithms exists like Walktrap [3], Infomap [4] or Louvain [5]. Some are non-deterministic like Louvain, in which the result is determined by the order in which the nodes are visited. Since the nodes may be visited in any order, such algorithm may produce different partitions of communities. Therefore, we need a way to pick a "good" partition. In order to do so, we would need a criterion to sort the partitions and pick the one that is the most representative of the actual community structure of the network.

[⋆] The authors would like to sincerely thank Bivas Mitra (from IIT Kharagpur, India) for the insightful discussions in the early stages of the article. This work has been partially funded by the ANR MITIK project, French National Research Agency (ANR), PRC AAPG2019.

In the absence of such criterion, we combine the information of different partitions of communities into *consensual* communities [6] (also known in the literature as community *cores* or *constant* communities). Such communities allow working with dynamic graphs as it becomes easier to follow communities in a timestamped network when the computation is deterministic [7].

Our contribution is twofold: we show that the information from the different partitions of communities is noisy. Then when combining the partitions into consensual communities, we show that some of the noise can be avoided.

This article is organised as follows: Section 2 formally defines general graphs concepts needed in Section 3, which gives an overview of related works in the area of consensual community detection. Then, Section 4 shows that the computation carries noise, and 5 shows a way to filter out some of the noise. Section 6 shows how our ideas can improve some state of the art algorithms, on some synthetic and real data. The last section concludes the article.

2 Definitions

A *graph* $G = (V, E)$ is made of a set of nodes V and a set of edges $E \in V \times V$, where $|V| = n$ and $|E| = m$. Within a graph, *communities* are defined as a partition of the nodes. That is, each node belongs to exactly one community.

To evaluate the quality of a set of communities, the *Modularity* [8] compares the number of intra-community edges to the expected number of intra-community edges in a random graph with the same number of nodes, same number of edges and same degree distribution as the starting graph. It is defined as

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

where i and j are nodes of the graph, and A_{ij} is 1 if there is an edge between i and j, 0 otherwise. m is the number of edges of the graph, k_i is the degree of i, and $\delta(c_i, c_j)$ is 1 if nodes i and j are in the same community, 0 otherwise.

If we have ground-truth communities, we can directly compare a partition given by any community detection algorithm to this ground-truth. In this case we can use the Normalised Mutual Information (*NMI*) [9]. It is based on a *confusion matrix* N where N_{ij} is the number of nodes of the ground-truth community i that also belong to the computed community j. It if formally defined as

$$I(A,B) = \frac{-2 \sum_{i=1}^{c_A} \sum_{j=1}^{c_B} N_{ij} \log\left(\frac{N_{ij} N}{N_{i.} N_{.j}}\right)}{\sum_{i=1}^{c_A} N_{i.} \log\left(\frac{N_{i.}}{N}\right) + \sum_{j=1}^{c_B} N_{.j} \log\left(\frac{N_{.j}}{N}\right)},$$

where c_A (resp. c_B) is the number of ground-truth (resp. computed) communities. The sum over row i (resp. column j) of N is denoted $N_{i.}$ (resp. $N_{.j}$). The NMI takes values from 0 (independent partitions) to 1 (same partitions).

Finally it is also possible to use synthetic graphs generated using the Lancichinetti, Fortunato and Radicchi (*LFR*) model [10]. This model creates synthetic graphs with a known community structure. They are generated with a *mixing* parameter μ that ranges from 0 to 1, and denotes the fraction of edges that a node shares with other communities of the graph. Thus, the smaller μ, the stronger the communities are and the easier it will be to detect them.

3 Related Work

Most community detection algorithms are non-deterministic. Thus, running any such algorithm $\mathscr{A}$ multiple times on the same graph G may result in different partitions. Moreover, these partitions cannot be discriminated on the basis of their modularity alone, as it has been shown that there are significantly different partitions with similar modularity [11]. To address this issue, a solution is to search similarities between different partitions to obtain *consensual communities*.

These similarities has also been studied for random graphs. It has been shown that random graphs contain good communities (or at least partitions of high modularity) which is very unnatural. However, different partitions are very dissimilar from each other. The absence of similarities might therefore indicate that the communities are not meaningful [12]. Peixoto et al. showed that when there is too much diversity in several partitions, it is in general not possible to obtain a consistent answer. Therefore, consensual communities are expected to yield good results only when there is a community structure in the graph.

3.1 The consensus matrix

To record similarities between partitions, most algorithms rely, in one way or another, on a *consensus matrix* P. First, a (classical) community detection algorithm $\mathscr{A}$ is ran n_p times on a graph G. Since $\mathscr{A}$ is non-deterministic, the n_p results will most likely be different. Then, the consensus matrix P is built as follows: the *consensus coefficient* P_{ij} is the number of times that nodes i and j were placed in the same community by $\mathscr{A}$ during the n_p executions. P_{ij} is thus between 0 and n_p. Then, the weighted *consensus graph* G_P whose adjacency matrix is P is built: nodes i and j are linked in G_P by an edge whose weight is P_{ij}. If $P_{ij} = 0$, then nodes i and j are not connected [6,7].

Rather than using $\mathscr{A}$ multiple times, some authors obtain diversity by using different algorithms to generate the first n_p. The Azar method [13] picks n_p different algorithms and each of them is run only once on G. Then the information from the n_p partitions is aggregated in a consensus matrix as described before, from which they build the associated consensus graph.

Liu et al. [14] chose to avoid local optimum of modularity by applying perturbations to the initial graph, then used a modified version of Louvain [5] that may explores a wider range of possible solutions. Finally, they aggregate the result in a consensus matrix. Burgess et al. [15] focuses on adding intra-community edges to the starting graph G, hoping to increase the efficiency of community detection algorithm. They do so by computing the similarity between pairs of nodes, based on metrics such as Jaccard [16] or the number of common neighbours. The edges are then added in a non-deterministic way to G, and they execute $\mathscr{A}$ on G. The process is repeated n_p times to obtain a consensus matrix.

Rasero et al. [17] use another version of consensus matrix that is not obtained through community detection but from the original data. They study brain connectivity graphs from different persons and aggregate the different graphs into a consensus matrix.

3.2 From consensus matrix to consensual communities

The first type of approach developed by Seifi et al. [7] does not require additional community detection. It considers that low values of P_{ij} are not significant and therefore keep only the values P_{ij} higher than a given threshold τ and set all the others to 0. This gives a thresholded matrix P_τ and the associated graph G_{P_τ}. The connected components of G_{P_τ} are then directly the consensual communities. Work has been done to get rid of τ and decide whether the P_{ij} entry is statistically significant before keeping it or not [18]. In the same vein, Chakraborty et al. [19] consider that constant communities are groups of nodes that are always in the same community in several executions (i.e., $P_{ij} = 1$).

Another approach consists in executing one final community detection algorithm on the consensus graph. Liang et al. [20] use the Label Propagation algorithm (LPA) [21] n_p times to build the consensus matrix, then perform a final single execution of a weighted version of LPA on the consensus graph. This has been generalized for instance in the LF procedure [6] that runs $\mathscr{A}$ n_p times again on G_P. At this point, if the n_p partition just computed are all the same, they are considered consensual and the algorithm stops. Otherwise, a new consensus matrix is built, and the process is repeated until all n_p partitions are the same.

Wang and Fleury developed an algorithm for overlapping communities that builds a consensus graph, merges nodes when their consensus coefficient is greater than a threshold α and repeat the process [22]. A similar idea using a consensus matrix has been developed by Yang and Leskovec [23].

3.3 Complexity issues

The consensus matrix P is an $n \times n$ matrix. The computation and storage of such a matrix quickly becomes infeasible on large graphs. This is unfortunately regularly the case with complex networks with up to several billions of nodes [1]. Several studies have therefore worked on improving the computation of the consensus matrix, as in the algorithms of Tandon [24] and ECG [25] where they only compute the entries P_{ij} for the edges $\{i, j\}$ of G. Instead of having n^2 entries in P, they only have m, the number of edges in the graph and all other entries are 0. Tandon chooses to construct a consensus graph G_P on which $\mathscr{A}$ is run n_p times until convergence. ECG, on the other hand, runs $\mathscr{A}$ one last time on the consensus graph G_P and considers the result as final.

The presence of edges between communities limits the efficiency of community detection algorithms. If there were only intra-community edges, the communities would be the connected components of the graph and identifying them would be trivial. Therefore, adding all the entries that correspond to the edges of the graph into the consensus matrix adds noise. The two main contributions of this article are to highlight the presence of noise and to find another criterion that would decrease further the number of entries, hence limit this noise. In addition, decreasing the number of entries in P could also reduce the temporal and spatial complexity of the consensus matrix computation.

4 Information is noisy

This section shows that the consensus matrix P is noisy. We do this by showing that the quality of the community is maximal when the consensus matrix is only partially filled. This means that there is some noise in P when it is full.

First, we generate a synthetic LFR graph G (along with its ground-truth communities). We use a distance DIST between nodes that will be described in more details in the following subsections. Next, we execute n_p times a community detection algorithm $\mathscr{A}$ on G. Finally, we build a consensus matrix P, but we fill it one entry P_{ij} at a time in increasing order of $\textsc{dist}(i, j)$. In case of a tie, we break it by selecting an arbitrary pair of nodes among the tie. After each addition in the incomplete consensus matrix P, we build the associated partial consensus graph G_P and execute $\mathscr{A}$ on G_P. We then compute the NMI between the LFR ground-truth communities and the communities we just computed. We iterate this way until P is completely filled. This method allows us to study how the NMI varies based on the number of entries in P.

Our methodology (see Algorithm 1) uses several parameters: the LFR graph and its mixing parameter μ that defines the proportion of links between communities; the community detection algorithm $\mathscr{A}$, the number n_p of executions of $\mathscr{A}$ and the distance DIST. In the rest of the paper we use $\mathscr{A}$ = Louvain as it is very fast, $n_p = 12$ as it has been shown that the consensus matrix rapidly converges as n_p grows [25] and we will vary μ and use several DIST.

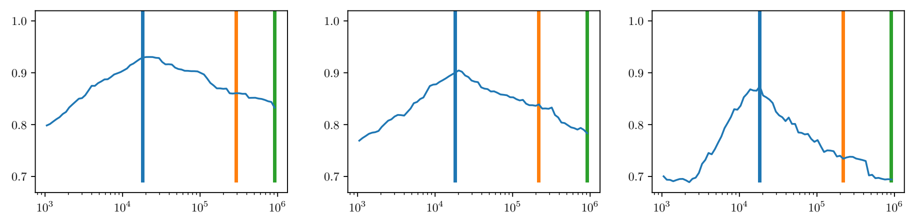
Algorithm 1 ORDERING (an LFR graph G, $\mathscr{A}$, n_p, DIST)

```
1: Build an ordered list $L$ of the pairs of nodes $(i, j)$, following the order given by DIST
2: Execute $n_p$ times algorithm $\mathscr{A}$ on $G$
3: while $L$ is not empty do
4:     Pick the pair $(i, j)$ whose dist$(i, j)$ is the smallest among $L$
5:     Compute the consensus coefficient $P_{ij}$ and add it to $P$
6:     Execute $\mathscr{A}(G_P)$
7:     Compute the NMI between the ground truth communities, and those found at
   the previous step
8: end while
9: return The plot with the NMI as a function of the number of entries in P
```

4.1 Graph distance

First we consider the graph distance (smallest number of consecutive edges) between two nodes as DIST, and apply Algorithm 1 for different values of μ. We use the graph distance since in Tandon [24] and ECG [25] the authors restrict themselves to connected pairs of nodes, i.e., nodes at distance 1.

Figure 1 shows the NMI as a function of the number of entries in P, for LFR graphs with 10 000 nodes and 3 different values of the mixing parameter μ. The graphs were generated with the implementation available online. The parameters used for LFR are: number of nodes $n = 10.000$, average degree $k = 20$, maximum degree $\max_k = 50$, degree sequence exponent $t_1 = 2$, community size distribution exponent $t_2 = 3$, minimum community size $\min_c = 10$, maximum community size $\max_c = 50$, weight mixing parameter $\mu_w = 0.6$.



(a) $\mu = 0.5$ (b) $\mu = 0.6$ (c) $\mu = 0.7$

Fig. 1: NMI vs number of entries of P filled using the graph distance ordering. LFR graphs with 10.000 nodes, values of the mixing parameter μ from .5 to .7. The vertical bars represent change of distance: entries that are added before the blue bar correspond to pairs of nodes at distance 1 (using a random order as a tie-breaker). Between the blue and orange bars are pairs at distance 2…

Figures 1 provide us with three main information. First of all, higher values of μ correspond to lower values of NMI. Since higher values of μ creates less pronounced communities, they are harder to find. The maximum values obtained are respectively .93, .904 and .874 respectively for μ from .5 to .7.

Then, the shape of the curves indicates that entries need to be filled. Indeed, after a certain number of entries, the NMI decreases. Intuitively, community detection algorithms would work better if there were no inter-community edges, and if all intra-community edges were present. The non-determinism of $\mathscr{A}$ makes the n_p partitions potentially different and thus, some non-zero consensus coefficients correspond to pairs of nodes that are in different communities in the ground-truth. The more we add such values, the harder it is for the algorithm to find the real communities, even though these values are very close to 0.

Finally, we can observe that, using the graph distance, the NMI is maximised when all entries corresponding to pairs of nodes at distance 1 are present in the consensus matrix, plus some of the entries at distance 2 (the peak is located a little after the vertical blue bar). Even if Tandon [24] and EGC [25] do not use exactly the same method as we do, it might be possible to do a little better than these algorithms by adding a few more entries in the consensus matrix.

To improve these algorithms we would therefore need to select some at distance 2. However, since the graph distance can only take integer values, we are limited by its precision and we therefore need another criterion that would help discriminate such entries.

4.2 Edge Clustering Coefficient

Intuitively we want to use a distance that will first add entries corresponding to intra-community pairs of nodes. Several criteria have been used to identify such pairs and they are often used to find communities. We can cite, among others, the Jaccard coefficient [16], the Cosine similarity, the Hamming distance [26] or the Edge Clustering Coefficient (ECC) [27] that measures the similarity between nodes i and j, and is defined as

$$ECC(i,j) = \frac{nb\_common\_neigh}{\min(\deg(i), \deg(j))},$$

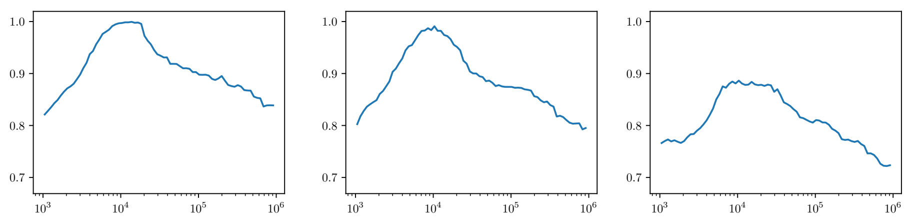
where nb_common_neigh is the number of neighbours that nodes i and j have in common, while deg() is the degree of a node.

Most of these similarities give good results, even though we ruled out Cosine because of its higher computing time. We present here the results for the ECC.

We proceed as before: we generate an LFR graph G, but time we use the ECC as DIST. We compute the ECC of all pairs of nodes that are at distance one or two. Nodes at distance three or higher cannot have any neighbour in common, thus their ECC will be zero. They should be avoided to maximise the NMI anyway, as shown in figures 1. Finally, we use algorithm 1 but insert values in decreasing order of the ECC.

Figure 2 shows the NMI as a function of the number of entries in P, for 3 different values of μ of LFR graphs with 10 000 nodes. The graphs were generated with the same implementation and the same parameters as figure 1. The ECC value being a real number, we can now precisely choose which entry to add in the consensus matrix, based on their ECC. We also notice that the maximum NMI is much greater for the ECC than graph distance. They are respectively .9994, .9910 and .8864 for μ from .5 to .7 (compared to .93, .904 and .874).

In real scenarios, we do not know the ground truth communities, so we cannot compute the NMI. We therefore need to decide beforehand how many entries need to be inserted in P to get as close as possible to the maximum NMI. The next section focuses on finding such an optimum number of entries.



(a) $\mu = 0.5$ (b) $\mu = 0.6$ (c) $\mu = 0.7$

Fig. 2: NMI vs number of entries of P using the ECC in decreasing order. LFR graphs with 10.000 nodes and different values of μ.

5 Finding an optimum number of entries – Filtering out the noise

This subsection is devoted to showing how, based on some correlations, we can estimate the number of entries to insert in the consensus matrix.

Figure 3a shows that the average modularity obtained across multiple executions of $\mathscr{A}$ is linearly correlated with the mixing parameters μ on LFR graphs. We use the correlation:

$$Q = 1 - \mu$$

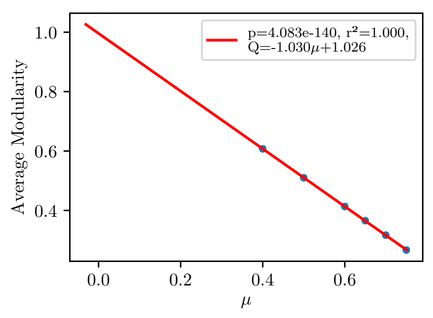
Kaminski et al studied such phenomenon on the ABCD graph model, which is very close to LFR [28]. They shown that the modularity of the ground truth partition of an ABCD graph with a mixing parameter μ asymptotically reaches $1 - \mu$ as the number of nodes n grows. They also observe that the modularity is smaller for smaller graphs, and converges as n grows.

Figure 3b shows that the mixing parameter μ is correlated to the ECC value τ above which pairs of nodes should be added to the consensus matrix to maximise the NMI. That is, all the pairs of nodes $\{i, j\}$ whose ECC is greater than τ should have their consensus coefficient put in the consensus matrix.

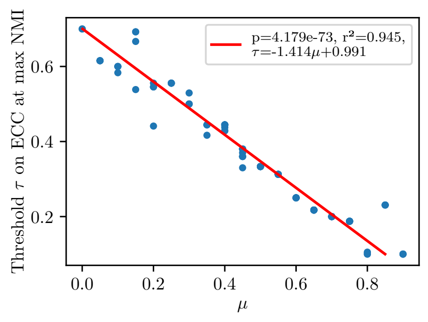
$$\tau = -1.414\mu + 0.991$$

Combining these two correlations it is therefore possible to deduce the optimum number of entries in the consensus matrix from the first n_p executions of $\mathscr{A}$. We compute several partitions (which is anyway the first step to compute the consensus matrix), we compute the average modularity and deduce the value of μ from which we deduce the lower limit on ECC. These correlations are made on LFR synthetic graphs and may not be valid for real graphs. However, we will see in Section 6 that they give good results even in these situations.

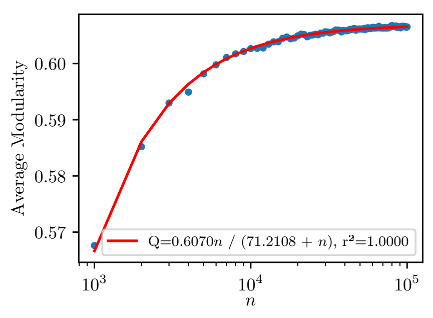
Another potential limitation to these correlations is that the average modularity over several executions increases with the size of the graph and stabilizes



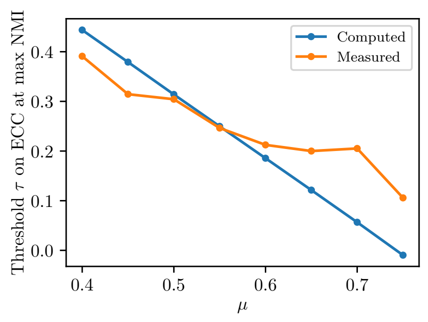
(a)



(b)



(c)



(d)

Fig. 3: Average modularity as a function of μ (top left). Threshold τ as a function of μ (top right). Q vs the number of nodes n (bottom left). τ vs μ at max NMI, computed with our filter and measured (bottom right)

once the number of nodes of a graph reaches 20 000 (see Figure 3c). A solution would be to artificially increase the average modularity for graphs with less than 20 000 nodes. Figure 3c shows the modularity Q as a function of n, for LFR graphs with $\mu = 0.4$. We repeat the operation for different values of μ. We get a fit in the general case $\frac{Q_{\max} \times n}{(K+n)}$, where $Q_{\max}$ is the modularity for a sufficiently large graph, and $K = 71.21$. Therefore:

$$Q_{\max} = \frac{KQ}{n} + Q$$

The threshold τ should therefore be deduced from $Q_{\max}$ if the graph has less than 20 000 nodes.

Finally, figure 3d shows the *computed* threshold τ along with the *measured* threshold, with which the NMI would have been maximised, for different values of μ.

5.1 Improving existing algorithms

In this subsection, we show how our observations can be included into current algorithms. To do so, we pick an existing algorithm that is representative of existing consensual community detection algorithms and modify it. We also build a generic algorithm that uses our filtering method.

Consider algorithm 2, a generic algorithm implementing our filter. It filters an input graph G based on the ECC and our correlation (that depends on the modularity of n_p executions of $\mathscr{A}(G)$). It outputs a *filtered* graph G_L.

Algorithm 2 FILTER (G, $\mathscr{A}$, n_p)

```
1: Execute $n_p$ times algorithm $\mathscr{A}$ on $G$
2: Compute the average modularity $Q$ of the $n_p$ partitions
3: Deduce a threshold $\tau$ from $Q$
4: Compute the ECC of the pairs of nodes $(i, j)$ whose distance is $\leq 2$
5: Build a list $L$ of pairs $(i, j)$ whose $ECC(i, j) > \tau$
6: Build a graph $G_L$ whose edges set is $L$. The weight of each edge is its corresponding
   consensus coefficient
7: return $G_L$
```

This filtered graph G_L can be used as input for other consensual community detection algorithms. We call the GENERIC_FILTER approach, which consists in running $\mathscr{A}(G_L)$, where the edges of G_L are weighted by their consensus coefficient. Consider also the ECG_FILTER algorithm which improves on ECG [25]. After applying the FILTER algorithm on G, we feed G_L into ECG, then return the consensual communities found by ECG. We also study the TANDON_FILTER algorithm, which builds G_L in the same way, then feeds it to TANDON. Note that since TANDON end ECG work on unweighted graphs, we don't bother weighting them. The next section focuses on studying the performance of both algorithms.

6 Experiments

First we run our algorithm on several LFR graphs and compare the NMI along with the running time compared to other state of the art algorithms, then we repeat the experiment on real-world graphs. In this section, we chose $\mathscr{A}$ = Louvain and $n_p = 64$ as we chose to parallelize the executions of $\mathscr{A}$ and our machine has got 64 cores.

Implementation details with correlations and correction for small graphs can be found on Software Heritage. The ordering criteria (based on the ECC) along with such correlations allows to improve on some current consensual community detection algorithms. The implementation behind figures 1 and 2 is available on Software Heritage

6.1 Synthetic Graphs

First, we generate LFR graphs with 10 000 nodes, using the parameters from Section 4.1 and μ varies from 0.4 to 0.7. Figure 4a shows the NMI as a function of μ, and figure 4b shows the running time as a function of the number of nodes. We compare the filtered version of Tandon and ECG against their regular version. We also display our generic filter, and Louvain and Infomap as baselines. For low values of μ, the filtered version of ECG provides a higher NMI, at the cost of a higher running time compared the regular ECG algorithm. For high values of μ, the NMI rapidly decreases, providing an NMI of 0.42 and 0.30 for $\mu = 0.7$ and 0.75 respectively, which is a lot lower than the regular ECG. Our generic filter algorithm gives a higher NMI compared to ECG, but is outperformed by Tandon's algorithm. However, it uses a lot less time and memory, allowing it to work on bigger graphs. We also observe a sharp decrease in NMI for high values of μ. For Tandon's algorithm, the filtered version provides a slim NMI improvement at the cost of a slightly higher running time.

Note that since a fair amount of the running time is spent filling the consensus matrix, the memory consumption grows with the running time.

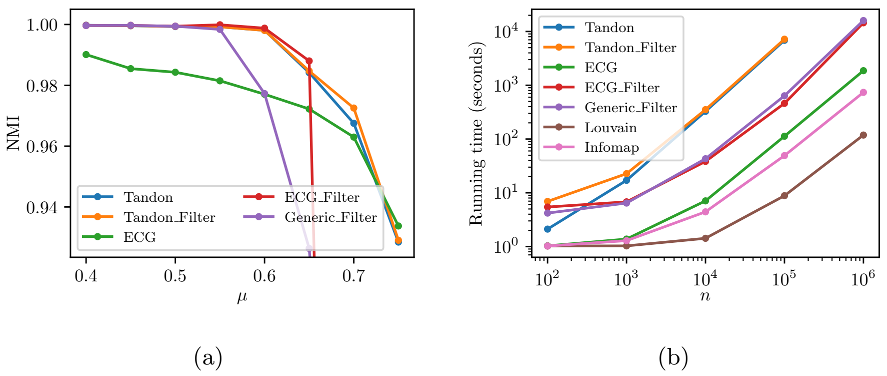


Fig. 4: NMI as a function of μ, LFR graphs with 10 000 nodes, with different algorithms (left). Running time as a function of n, LFR graphs, $\mu = 0.6$ (right)

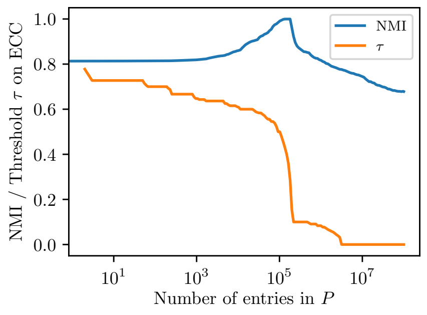
Even though classical community detection algorithms suffer from the limits described before, we also applied this protocol to Louvain and Infomap. The NMI is naturraly much lower with these two algorithms: Louvain (resp. Infomap) goes from 0.84 to 0.76 (resp. from 0.81 to 0.78) for μ from 0.4 to 0.7.

6.2 Limitations

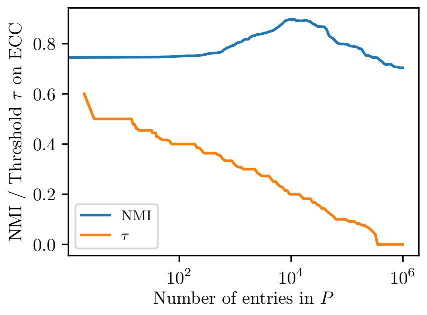
Overall, figure 4a shows that the filtered algorithms tend to perform better than their non-filtered counterparts for low values of μ, and perform worse on high values of μ. We believe that this is due to a combination of several effects.

As μ grows, the number of edges inside communities decreases, while the number of inter-community edges increases. This means that the ECC of pairs of nodes located in different communities will also tend to increase and our filter is more likely to add noise in the consensus matrix, which will amplify the problem, if μ is sufficiently large. Second, as μ grows, the modularity of the ground-truth partition tends to decrease. Therefore, as observed by Aynaud et al. [29] a community detection algorithm that focuses on modularity maximisation may find a partition with a higher modularity than the ground-truth. In this case it might be interesting to use non-modularity based algorithms.

Last, figure 5a shows the NMI as a function of the number of entries added in the consensus matrix, along with the value of the ECC pair added to the matrix. For low values of μ, we observe a sharp decrease in the ECC values, which corresponds to the maximum NMI value. This allows some imprecision in our threshold on the ECC value. As long as the threshold falls within the sharp decrease, we should be close to the maximum NMI possible. However, as μ grows, the sharp decrease tends to happen after the maximum NMI (see figure 5b). The decrease tends to be less sharp, thus requiring the threshold to be more precise. This makes the ECC threshold harder to set as a small imprecision might make a big change in the number of entries added to the consensus matrix. All in all, communities are harder to find and the threshold needs to be more precise.



(a)



(b)

Fig. 5: NMI and ECC threshold as a function of the number of entries in the consensus matrix, for $\mu = 0.4$ (left). NMI and ECC threshold as a function of the number of entries in the consensus matrix, for $\mu = 0.7$ (right)

According to Orman et al [30], in LFR graphs, when μ is greater than 0.5, the communities are less well defined, and as μ increases we are left with a scale-free network with little to no community structure. At one extreme, if μ is very low, the partition is very easy to find. At the other extreme, there is no more community structure in the graph. Therefore, the consensus methods are particularly useful in intermediate ranges of μ.

6.3 Real Graphs

Football Dataset In the football dataset [2], nodes represent US football teams, and the edges represent games played between the teams. The communities are the conference in which the team play. There are 115 nodes, 613 games and 12 communities in this dataset. The results are summarized in table 1. We see that our generic filter algorithm provides the highest NMI along with the lowest running time, while our filtered version of ECG provides a higher NMI, but a higher running time than the classical ECG algorithm. We compare the performance against two classical community detection algorithms: the Louvain method [5], which gives a lower NMI than the other algorithms we tested, and the Infomap algorithm [3], which gives the second highest NMI on this dataset.

DBLP Dataset The Digital Bibliography & Library Project (DBLP) dataset [31] is a co-authorship network where nodes are authors and two authors are linked by an edge if they co-published a paper. It is a larger graph than football, with 317 thousands nodes and about 1 million edges. The ground truth communities are the publication venues. Results are summarized in table 1. First, we note a big difference in running time between the different algorithms, ranging from 12 seconds to 35 minutes. The *filtered* algorithms are the longest because of the computation of the ECC and the generation of the consensus graph. The highest NMI is provided by Infomap, at the cost of a high running time. The filtered version of the ECG algorithm provides the second highest NMI, but also has a high running time. Our generic filter algorithm performs poorly on this dataset, while Louvain gives the lowest NMI but is the fastest by far. We could not run Tandon's algorithm because the running time was too high.

	Football		DBLP	
Algorithm	NMI	Running Time	NMI	Running Time
ECG	0.9079	1.10 s	0.6030	171 s
Tandon	0.8976	1.29 s	–	–
Tandon_Filter	0.8981	2.54 s	–	–
Generic_Filter	0.6823	1.21 s	0.4613	2236 s
ECG_Filter	0.9349	1.99 s	0.6288	1278 s
Louvain	0.8879	1.02 s	0,4769	12 s
Infomap	0.9242	1.12 s	0.7738	1305 s

Table 1: NMI and running time of several algorithms for football and DBLP

7 Conclusion

Most community detection algorithms are non-deterministic and it is often necessary to aggregate the results of multiple runs to find a consensus, summarized in a consensus matrix. However, in real data, community are not perfectly defined.

There are inter-community edges that carry noise. We have shown that the information in the consensus matrix is therefore noisy but that it is possible to filter out some of this noise. We then used these observations to improve existing algorithms, and verified the effectiveness of our approach on synthetic and real graphs. ECG_FILTER tends to yield a higher NMI than ECG at the cost of a higher running time. Our generic filter, GENERIC_FILTER, also provides a high NMI in most of our tests, while being more scalable than Tandon's algorithm.

A closer look at existing consensus community detection algorithms would also be relevant. Indeed, we believe that our general observations would be useful for most algorithms that use a consensus matrix and that could therefore benefit from noise filtering. Moreover, some algorithms like the LF procedure perform community detection in several steps, it could be worthwhile to filter the graph at each step. Finally, the correlations of figure 3 would be worth proving, the same way it was done for figure 3a [28].

References

1. S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.
2. M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
3. P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences.* Springer, 2005, pp. 284–293.
4. M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *The European Physical Journal Special Topics*, vol. 178, no. 1, pp. 13–23, 2009.
5. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
6. A. Lancichinetti and S. Fortunato, "Consensus clustering in complex networks," *Scientific reports*, vol. 2, no. 1, pp. 1–7, 2012.
7. M. Seifi and J.-L. Guillaume, "Community cores in evolving networks," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 1173–1180.
8. M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.
9. L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of statistical mechanics: Theory and experiment*, vol. 2005, no. 09, p. P09008, 2005.
10. A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.
11. B. Good, Y.-A. de Montjoye, and A. Clauset, "Performance of modularity maximization in practical contexts," *Phys. Rev. E*, vol. 81, p. 046106, Apr 2010. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.81.046106
12. R. Campigotto, J.-L. Guillaume, and M. Seifi, "The power of consensus: Random graphs have no communities," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2013, pp. 272–276.

13. M. Kheirkhahzadeh and M. Analoui, "A consensus clustering method for clustering social networks," *Statistics, Optimization & Information Computing*, vol. 8, no. 1, pp. 254–271, 2020.
14. Q. Liu, Z. Hou, and J. Yang, "Detecting spatial communities in vehicle movements by combining multi-level merging and consensus clustering," *Remote Sensing*, vol. 14, no. 17, p. 4144, 2022.
15. M. Burgess, E. Adar, and M. Cafarella, "Link-prediction enhanced consensus clustering for complex networks," *PloS one*, vol. 11, no. 5, p. e0153384, 2016.
16. P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901.
17. J. Rasero, M. Pellicoro, L. Angelini, J. M. Cortes, D. Marinazzo, and S. Stramaglia, "Consensus clustering approach to group brain connectivity matrices," *Network Neuroscience*, vol. 1, no. 3, pp. 242–253, 2017.
18. D. Mandaglio, A. Amelio, and A. Tagarelli, "Consensus community detection in multilayer networks using parameter-free graph pruning," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2018, pp. 193–205.
19. T. Chakraborty, S. Srinivasan, N. Ganguly, S. Bhowmick, and A. Mukherjee, "Constant communities in complex networks," *Scientific reports*, vol. 3, no. 1, pp. 1–9, 2013.
20. Z.-W. Liang, J.-P. Li, F. Yang, and A. Petropulu, "Detecting community structure using label propagation with consensus weight in complex network," *Chinese Physics B*, vol. 23, no. 9, p. 098902, 2014.
21. U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.
22. Q. Wang and E. Fleury, "Detecting overlapping communities in graphs," in *European Conference on Complex Systems 2009 (ECCS 2009)*, 2009.
23. L. Yang, Z. Yu, J. Qian, and S. Liu, "Overlapping community detection using weighted consensus clustering," *Pramana*, vol. 87, no. 4, pp. 1–6, 2016.
24. A. Tandon, A. Albeshri, V. Thayananthan, W. Alhalabi, and S. Fortunato, "Fast consensus clustering in complex networks," *Physical Review E*, vol. 99, no. 4, p. 042301, 2019.
25. V. Poulin and F. Théberge, "Ensemble clustering for graphs," in *International Conference on Complex Networks and their Applications.* Springer, 2018, pp. 231–243.
26. R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.
27. F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the national academy of sciences*, vol. 101, no. 9, pp. 2658–2663, 2004.
28. B. Kamiński, B. Pankratz, P. Prałat, and F. Théberge, "Modularity of the abcd random graph model with community structure," *Journal of Complex Networks*, vol. 10, no. 6, p. cnac050, 2022.
29. T. Aynaud, V. D. Blondel, J.-L. Guillaume, and R. Lambiotte, "Multilevel local optimization of modularity," *Graph Partitioning*, pp. 315–345, 2013.
30. G. K. Orman and V. Labatut, "A comparison of community detection algorithms on artificial networks," in *Discovery Science: 12th International Conference, DS 2009, Porto, Portugal, October 3-5, 2009 12.* Springer, 2009, pp. 242–256.
31. J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, 2012, pp. 1–8.