Filtering the noise in consensual community detection

Antoine Huchet, Jean-Loup Guillaume, Yacine Ghamri-Doudane

Noname manuscript No.
(will be inserted by the editor)

Filtering the noise in consensual community detection

Antoine Huchet, Jean-Loup Guillaume · Yacine Ghamri-Doudane

Received: date / Accepted: date

Abstract Community detection allows understanding 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 can benefit existing algorithms.

Keywords Real Graphs, Graph Algorithms, Consensual Communities, Noise

1 Introduction

Graphs representing real-world data exhibit particular features that make them far from regular. 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*. Finding these densely connected parts is called *community detection*. In social graphs, community detection could help identify group of people such as families, friends or co-workers.

Many community detection algorithms exist like Walktrap, Infomap or Louvain [1]. Some algorithms are non-deterministic like the latter, where the communities produced are determined by the order in which the nodes are visited. Since the nodes may be visited in any order, such an algorithm may

This work has been partially funded by the ANR MITIK project, French National Research Agency (ANR), PRC AAPG2019.

A. Huchet · Jean-Loup Guillaume · Yacine Ghamri-Doudane
L3i La Rochelle Université, La Rochelle, France
E-mail: antoine.huchet@univ-lr.fr

produce different partitions of communities. To get a deterministic result, we combine the information of different partitions of communities into *consensual* communities [2].

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.

2 Consensual Communities

A *graph* $G = (V, E)$ is made of a set of nodes V and a set of edges $E \subseteq V \times V$, where $|V| = n$ and $|E| = m$. *Communities* form a partition of the nodes of the graph. That is, each node belongs to exactly one community. Lancichinetti, Fortunato and Radicchi (*LFR*) have proposed a model to generate synthetic graphs with a known community structure [3]. Those graphs are generated with a *mixing* parameter μ that defines the proportion of edges between communities. It ranges from 0 to 1, and the smaller μ, the easier it is to detect communities. The *Modularity* measures the quality of a partitions of a graph into communities. When the ground-truth communities are known, the *NMI* measures how close they are to a set of discovered communities. The Edge Clustering Coeffieicnt ($ECC(i, j)$) is a similarity metric between two nodes i and j [5].

Since most community detection algorithms are non-deterministic, running n_p times an algorithm $\mathscr{A}$ on a graph G may result in different partitions. We define a *consensus* matrix, C, where C_{ij} is the number of times that nodes i and j were put in the same community by $\mathscr{A}$ across the n_p executions, called the *consensus* coefficient of i and j. Consensual communities can then be computed by building the *consensus* graph G_C, whose adjacency matrix is C. One way to derive the consensual communities is to set a threshold λ, and remove the edges of G_C whose weight is lower than λ. The resulting connected components would be the consensual communities [6]. It is also possible to execute again $\mathscr{A}$ on G_C until convergence [2].

Note that the consensus matrix is an $n \times n$ matrix. Filling such a matrix is a lengthy process that requires a lot of memory. Different authors worked on improving this computation, as in Tandon's algorithm [7] or ECG [4], where only the entries C_{ij} that correspond to edges (i, j) of G are computed. This brings the number of entries from n^2 down to m, the number of edges in G.

3 Identifying and filtering the noise

We show that the consensus matrix C is noisy. To do so, we generate an LFR graph G (along with its ground-truth communities). We also use the ECC, which allows ordering our pairs of nodes. Next, we execute n_p times a community detection algorithm $\mathscr{A}$ on G. Finally, we build a consensus matrix C, but we fill it one entry C_{ij} at a time in increasing order of $ECC(i, j)$. 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 C, we build the associated partial consensus graph G_C and execute $\mathscr{A}$ on G_C. We then compute the NMI between the LFR ground-truth communities and the communities we just computed. We iterate this way until C is completely filled. This method allows us to study how the NMI varies based on the number of entries in C. Figure 1 shows the NMI as a function of the number of entries in C, for LFR graphs with 1 000 nodes, and 4 different values of μ. As we fill C, we observe that the NMI increases, reaches a maximum, then decreases. Notice that the maximum NMI corresponds to a consensus matrix that is far from filled.

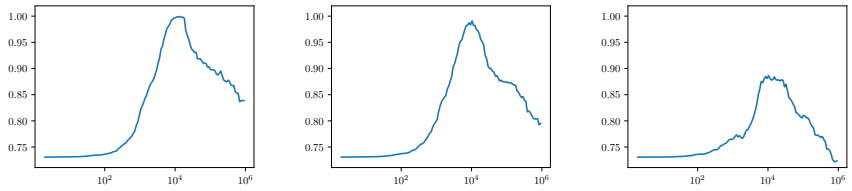

Fig. 1: NMI vs the number of entries of C, respectively for $\mu = .5$, .6 and .7

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 of C need to be computed to get as close as possible to the maximum NMI. Our experiments show that the average modularity Q of the n_p executions of $\mathscr{A}(G)$ is strongly correlated to the threshold on the ECC τ below which entries should not be added to C. Thus, from Q, we can deduce when to stop filling C. We then feed the consensus graph G_C to existing algorithms. We call Tandon_Filtered the heuristic that feeds the consensus graph G_C to Tandon, and ECG_Filtered when we feed it to ECG. We call Generic_Filter the approach that feeds G_C back to $\mathscr{A}$ one last time. Note that when the algorithm that is fed with G_C supports weighted graphs, we weight the edges of G_C with their consensus coefficient.

To validate our approach, we generate LFR graphs with 10 000 nodes, with different values of μ. We choose $\mathscr{A}$ = Louvain, and we execute our filtered and non-filtered algorithms on those graphs[1]. Figure 2 shows the NMI and the running time as a function of μ. ECG_filtered provides a higher NMI than ECG, at the cost of a higher running time, due to the computation of the ECC. Tandon_Filtered yields a comparable NMI, but takes longer than Tandon because of the computation of the ECC. The Generic_Filter provides a higher NMI than ECG, but lower than Tandon, but allows working on bigger graphs than Tandon thanks to its better running time. We observe a high NMI, then a sharp decrease for our filtered approaches. This is because

[1] All the implementations are available on Software Heritage.

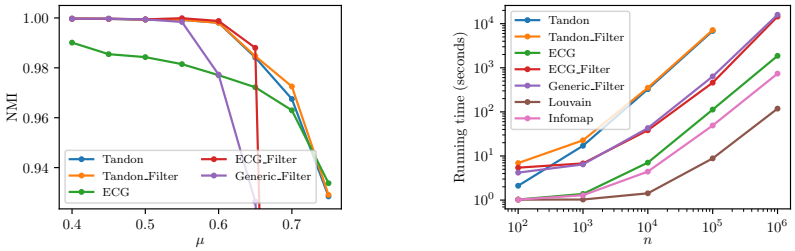

Fig. 2: NMI as a function of μ, LFR graphs with 10 000 nodes (left); Running time as a function of the size of the graph (number of nodes), for LFR graphs with $\mu = 0.6$ (right)

for high μ, the communities do not correspond to dense parts of the graphs anymore, so we tend to filter out intra-community edges.

We also applied our filters on real graphs and obtained similar results.

4 Conclusion and future work

We have shown that the information in the consensus matrix can be noisy but that it is possible to filter out some of the noise. We then used these observations to improve existing algorithms, and verified the effectiveness of our approach on synthetic and real graphs.

We believe that our noise filtering method would be useful for most algorithms that use a consensus matrix. Moreover, some algorithms perform community detection in several iterations, it could be worthwhile to study the effect of filtering the graph at each iterations.

References

1. Blondel, V., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
2. Lancichinetti, A., Fortunato, S.: Consensus clustering in complex networks. Scientific reports **2**(1), 1–7 (2012)
3. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical review E **78**(4), 046110 (2008)
4. Poulin, V., Théberge, F.: Ensemble clustering for graphs. In: International Conference on Complex Networks and their Applications, pp. 231–243. Springer (2018)
5. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the national academy of sciences **101**(9), 2658–2663 (2004)
6. Seifi, M., Guillaume, J.L.: Community cores in evolving networks. In: Proceedings of the 21st International Conference on World Wide Web, pp. 1173–1180 (2012)
7. Tandon, A., Albeshri, A., Thayananthan, V., Alhalabi, W., Fortunato, S.: Fast consensus clustering in complex networks. Physical Review E **99**(4), 042301 (2019)