Static community detection algorithms for evolving networks

Thomas Aynaud
LIP6 – CNRS – Université Pierre et Marie Curie
104 avenue du président Kennedy
75016 Paris, France
thomas.aynaud@lip6.fr

Jean-Loup Guillaume
LIP6 – CNRS – Université Pierre et Marie Curie
104 avenue du président Kennedy
75016 Paris, France
jean-loup.guillaume@lip6.fr

***Abstract*—Complex networks can often be divided in dense sub-networks called communities. Using a partition edit distance, we study how three community detection algorithms transform their outputs if the input network is slightly modified. The instabilities appear to be important and we propose a modification of one algorithm to stabilize it and to allow the tracking of the communities in an evolving network. This modification has one parameter which is a tradeoff between stability and quality. The resulting algorithm appears to be very effective. We finally use it on an evolving network of blogs.**

***Index Terms*—complex networks, evolving communities, stability, tracking, blogs**

INTRODUCTION

Complex networks arise naturally in many different fields. They describe a variety of systems by modeling the interactions between constituents. For instance, the internet is a network of computers linked by physical or wireless links. Similarly, the web can be seen as a network of web pages linked by hyperlinks.

A common feature of many real networks is the existence of groups of nodes, or *communities*, which are crucial in understanding the underlying structure of large networks. Communities can be, for example, groups of friends in a social network, websites dealing with the same subject or scientists working on similar topics in co-authorship networks. Formalizing the notion of community and detecting communities is a harsh task which has attracted a lot of attention in the recent years. In particular, many algorithms based on the network topology have been proposed.

However, most studies usually deal with a single static network which is a snapshot of the data. Very few studies have given attention to temporal features. As real data are always evolving (in phone networks, people do not call each other permanently; on the web, pages appear, disappear or are updated continuously for instance), a lot of information is lost.

In this paper, we will try to track communities between successive snapshots of the evolution of a network. We will first use classical community detection algorithms whose main issue in this context is stability. Indeed, we will show how little modifications of the input network often lead to strong transformations of the detected community structure, making them untraceable. We will thus propose a modification of one very effective algorithm to obtain much more stable communities and then use it to study a truly dynamic dataset: a network of blogs.

The paper is organized as follows: in the first section we will present classical results about community detection, evolving networks and community tracking. Then, we will describe our experiments and results concerning stability in the second section. In the third section, we will propose a modified version of the Louvain algorithm to achieve better stability and the results of our experiments on a real graph.

I. BACKGROUND AND RELATED WORK

A. Communities

The detection of communities in complex networks has attracted a lot of interest and many definitions of a community have been proposed. Usually, algorithms are looking for a *good* partition of the nodes. This implies that no node belongs to more than one community and the main issue is to define what *good* means. People dealing with visualization problems often define a good partition as a partition where parts are drawn separately [1]. Good partitions can also be obtained through k-means algorithms using the classical distance in graphs as a distance between nodes [2]. Another classical approach consists in defining a *quality function* which gives a score to a partition: the good partition

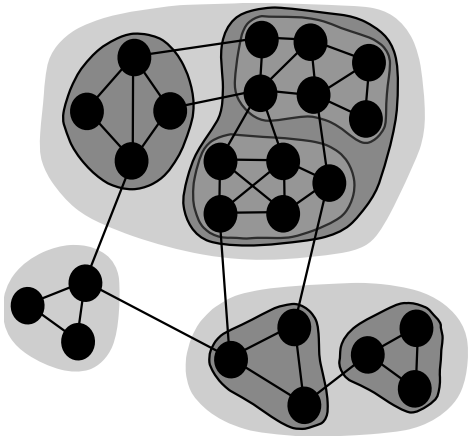


Figure 1. Example of a network with communities.

is the one which maximizes this quality function. One of the most used quality function is the *modularity*, defined in [3]. Ranging between -1 and 1, the modularity compares for each parts (or community) the proportion of links inside the community with a null model. The higher it is, the better the partition is.

B. Detection algorithms

Finding the partition which maximizes the modularity is NP-hard [4] and therefore many approximation algorithms have been proposed, see [5]–[7] for comprehensive surveys. In this paper, we will use three different algorithms:

- the first one, Walktrap [8], is based on the idea that a small random walk will stay inside the community from where it's originating because there are many links inside and few bridges leading outside.
- the second one, Fast Greedy [9], is a greedy optimization algorithm for modularity: each node is initially in its own community and then, at every step, the algorithm groups two communities in order to maximize the gain of modularity.
- the last one, Louvain Method [10] is described more precisely in the next paragraph since we will modify it in section III-A.

Louvain Method is a hierarchical greedy algorithm. It is composed of two phases, executed alternatively. Initially, each node is in its own community. Then, during phase 1, nodes are considered one by one, and each one is placed in the neighboring community (including its own community) which maximizes the modularity gain. This phase is repeated until no node is moved (the obtained decomposition is therefore a local maxima). Then, phase 2 consists in building the graph between communities obtained during phase 1: there is a node in the new graph for each community and, for two communities C and C', there is a link of weight w where $w = \sum_{v,v' \in C \times C'} weight(v, v')$. There is also a loop on C of weight $\sum_{v,v' \in C \times C} weight(v, v')$ [1]. Then, the algorithm starts the phase 1 again with the new graph, grouping communities together, and then phase 2, and so on until the modularity does not improve. The whole process is described on algorithm 1.

Algorithm 1 Pseudo-code of Louvain Method

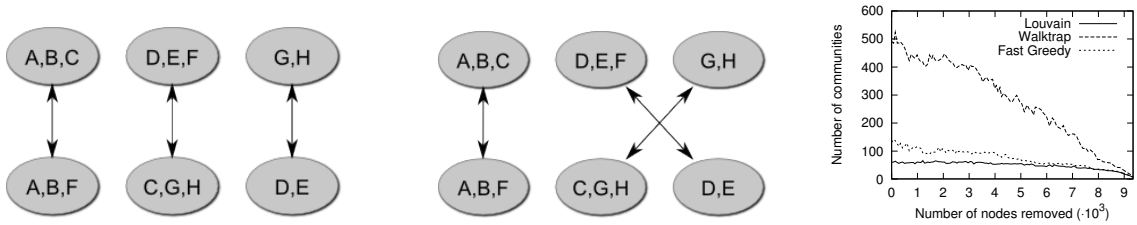
```
G the initial network
repeat
    Put each node of G in its own community
    while some nodes are moved do
        for all node n of G do
            place n in its neighboring community including its own which maximizes the modularity gain
        end for
    end while
    if the new modularity is higher than the initial then
        G = the network between communities of G
    else
        Terminate
    end if
until
```

These three algorithms perform well on static networks. However, Louvain Method is the fastest and most accurate method (it achieves better modularity) and walktrap the slowest and less accurate. However, we are willing to deal with evolving networks and less efficient static algorithm can be more efficient for evolving networks. This is the reason why we did not restrict ourselves to the Louvain Method.

C. Evolving communities

The study of the evolution of communities has lead to two main approaches: using the temporal information directly in the detection or tracking communities among different snapshots, often obtained with algorithms suited for static graphs. The first direction has not been widely followed. To take into account the temporal information, authors of [11] have for example modified the quality

[1] This ensures that the partition found after phase 1 has the same modularity as the partition of the new graph where each node is put in its own community



(a) Matching example

(b) Number of communities

Figure 2. Example of the matching between parts of different partitions and number of communities during one run

function to integrate dynamics inside it. In [12], a probabilistic model integrating dynamic features is defined and used to detect communities.

On the other hand, the main underlying problem when computing communities on different snapshots is to decide which community at time t has evolved into which community at time $t+1$. Considering that communities may merge, split, appear or disappear, many solutions have been proposed. In [13], Hopcroft et al. look for two parts sharing the most nodes and having similar size. In [14], this idea is generalized by introducing many rules to handle split, merge, etc. This method works but finding a consensus on rules is difficult, and there are always some parameters that must be fixed arbitrarily. When do two communities are the same? Is it when they share 70% of the nodes? Or maybe 80%, 90% or more? To avoid this kind of rules, [15] uses a modification of their own community detection algorithm to do the matching. However, this requires a property on the algorithm that is barely fulfilled: by adding links to a network, the communities can only grow, merge or remain unchanged. Thus, community tracking and detection on evolving networks is still a challenge.

D. Comparing partitions and tracking

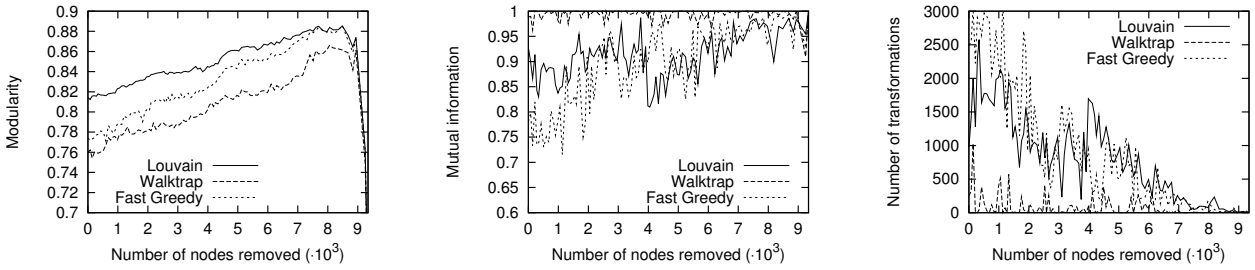
The most widely used metric to evaluate the distance between two partitions is the mutual information [16]. It ranges from 0 to 1, where a value of 0 means that partitions are independent and 1 that they are equal. Defined by the information theory, it counts the number of bits shared by two random variables. This metric is really interesting but have a few issues. First, there are many ways to normalize it, leading to different values, and no normalization is clearly the good one. Secondly, it has been proven that there is a correlation between mutual information and the size of the partitions [17]. Consequently, it is not well-suited to compare algorithms. It is also rather difficult to interpret and, extreme values excepted, it does not allow to really decide if partitions are close, really close or far from each others.

To address these issues, we have decided to use both mutual information and an edit distance between partitions. This distance counts the number of transformations needed to move from a partition A to a partition B. For example, figure 2(a) presents two partitions with two possible matchings. In the first one, nodes C, D, E, F, G and H must be moved whereas, only C and F have to move with the second matching. The matching which minimizes the number of transformations can be computed in $O(n^3)$ with n the number of parts using the Kuhn-Munkres algorithm [18]. This distance has two advantages and one disadvantage: it gives the matching and it is more intelligible but its matching is a one to one association. There are no merge, split or apparition of new communities, and thus the association is only reliable when such one to one association exists, i.e. only when the communities are really stable.

Using these algorithms and definitions, we are able to compute communities at different time steps of the evolution of a network and compute the similarities between the partitions. This will be first used to study the stability of the algorithms in case of small modifications of the networks.

II. Stability of the three algorithms

To study the stability of the three algorithms, we first simulate an evolution on a classical static network. The evolution simply consists in the removal, one by one, of a random node, keeping only the largest connected component to always have a connected network. This is clearly not realistic, but removing one node should not change the structure of the network and thus partitions computed with any algorithm should be very similar (both in terms of mutual information or edit distance) before and after removal.



(a) Modularity (b) Mutual information (c) Edit distance

Figure 3. Modularities during one run and mutual information or partition edit distance between two consecutive partitions.

We present hereafter results using the arxiv dataset which is a network of scientists defined by Newman in [19]. Two scientists are connected if they have written a paper together. To simplify the network and as they do not modify the communities, we have removed the nodes connected by a single link. This gives a network of 9377 nodes and 24107 edges. The obtained results are qualitatively similar on other networks.

The first metric computed at each time step is the modularity. The results are presented on the figure 3(a). The modularity is in each case growing slowly until a break point where it falls quickly. The Louvain Method performs better than Fast Greedy which is itself better than Walktrap in the sense that the partition found has higher modularity. The order remains the same when considering the speed. Another remark is that the modularity is almost constant between two consecutive steps. Removing random nodes does not modify much the structure of the network and the quality is therefore really stable.

The number of communities in the partition, shown on the figure 2(b), is also stable for the Louvain Method and the Fast Greedy algorithm. Walktrap is less stable during the whole run, but the number of communities does not change very quickly. Thus we are in the case where the partition edit distance is reliable.

The figure 3(b) shows the mutual information between every partition and its predecessor. Walktrap algorithm has values close to 1 and seems really stable. On the opposite, the two other algorithms take many different values of mutual information ranging from 0.75 to 0.98 for Louvain Method, and from 0.65 to 0.97 for Fast Greedy. The distance between partitions varies a lot between different steps. But a value of 0.9 of the mutual information, which is common for Louvain Method, seems to indicate that the partitions are not so different. The partition edit distance, shown on figure 3(c), gives a really different insight: after the removal of only one node among (initially) 9377, between 2000 and 3000 nodes are moved in the community structure with the Fast Greedy algorithm, and between 1500 and 2500 with the Louvain Method. This means that the communities detected at each step are completely different and that both algorithms are extremely instable. Walktrap's results are clearly better, but there are still often 500 moves which represent too many transformations for only one node removed.

Therefore, none of these algorithms is suitable to compute evolving communities due to this instability. We will next propose a new version of the Louvain method which is far more stable.

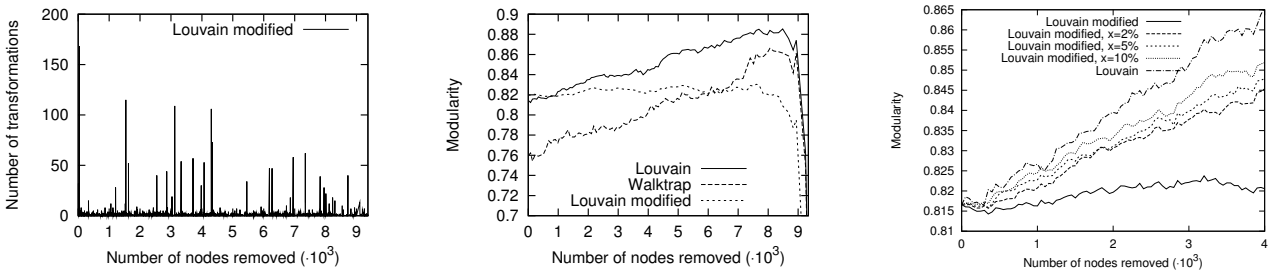
III. A STABILIZED VERSION OF THE LOUVAIN METHOD

A. Stabilized algorithm

To be able to track communities we propose another much more stable version of the Louvain Method. The idea is to constrain the initial partition in order to force stability. This can be simply done by changing the initialization of the algorithm. Whereas the classical Louvain Method starts with every node in its own community, we will now start the computation at time t by grouping nodes using the partition found at time $t-1$. We then apply the exact same algorithm, allowing nodes to move if the initial constraint is not pertinent.

The figure 4(a) is similar to the figure 3(c) but with the modified algorithm. The communities are far more stable. Most of the time, the communities are not modified, but there are some pikes where small modifications happen. This is far more satisfactory since the removal of one node does not change the structure of the graph, except for a few central nodes.

The pikes on figure 4(a) can be caused by true events on the network or be artifacts caused by the algorithm.



(a) Stability (b) Modularity (c) Modularity using the quality parameter

Figure 4. Stability and modularity of the stabilized version of the Louvain Method and modularity using the quality parameter.

Therefore we have removed each node independently, starting from the initial network and compared the communities obtained with the initial decomposition. We then tried, for each node, to correlate the distance between partitions and properties of the removed node (degree, pagerank, closeness or betweeness centrality computed on the whole graph or using only the graph inside the removed node community). None of the computed metrics is clearly correlated with the effect on communities' stability. Therefore we are not able at the moment to explain why the removal of some specific nodes generates more modifications of the partition.

However, stability is not sufficient to have good partition. The figure 4(b) presents the modularity of Louvain Method, Walktrap and Louvain Method modified. Most of the time, Louvain Method modified is better than Walktrap and competes with Louvain Method. It is only when the network topology has changed a lot, at the end of the run, that Walktrap outpasses Louvain Method modified. As we remove one node, every node in the initial partition is in a community, and we can assume that it is in a good one as it is the result of the algorithm. Thus, the initial partition is a strong constraint for the algorithm and we will next study some possibilities to limit this constraint.

B. A stability vs. quality parameter.

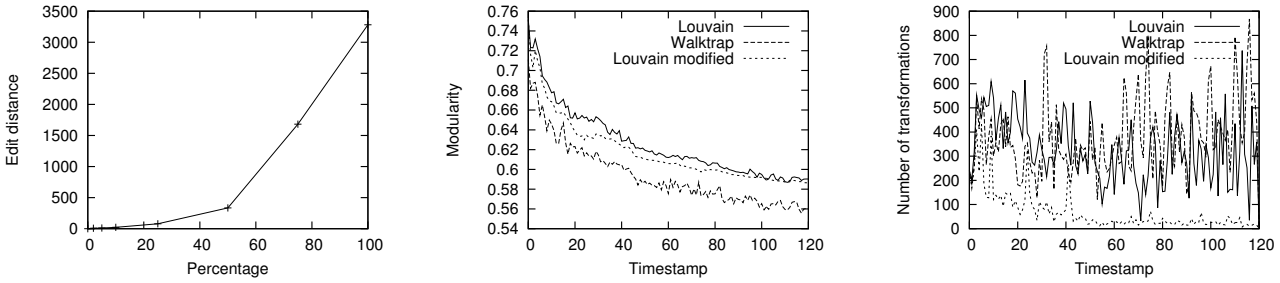
To limit the constraints of the modified algorithm, we choose randomly $x\%$ of the nodes and place them alone in their own community instead of putting them in their previous community. The higher x is, the more the algorithm can modify the communities since the nodes placed alone in their communities will be moved (it is always a bad choice to leave a node alone). If $x = 100\%$, the algorithm is the classical Louvain method and if $x = 0\%$, the algorithm is the stabilized version presented before. There is obviously a tradeoff between stability and quality. Figure 4(c) shows the modularity for different values of x. Even low values of x like 2% make an important improvement and change the shape of the plot.

Figure 5(a) shows the average partition edit distance during the first 1000 removals. For small values of x, the stability is only slightly damaged. For larger values, above $x = 50\%$, the stability increases linearly to reach the classical Louvain stability. Using $x = 2\%$ seems to be a good compromise here: it gives a good stability and a very high modularity. This parameter depends on the network evolution. For instance, if there are many modifications, using the previous partition can be sufficient. Indeed, the initial partition is not so good and therefore does not impose much constraints. The choice of x depends on the number of modifications between two consecutive snapshots. If snapshots are very similar, x should be increased to give more freedom, but if they are far enough, it may remain set to 0.

C. A real dynamic

As the evolution used before is completely artificial, we have run the same tests on a truly evolving network: a network of blogs. During four months, around five thousands blogs have been monitored to track posts and comments. Let's consider that time 0 corresponds to the beginning of the measurement. The nodes of the network at time t are the blogs which contain a post between time 0 and t. There is a link between two blogs at time t if one of them make a link to the other between time 0 and t. The resolution is of one day, so we have 120 snapshots of the network, one for each day, which all represent the aggregation of the posts and links until the considered day. To simplify community detection and tracking, we take only the largest connected component for each snapshot.

Starting first with an empty network, we applied



(a) Stability using the quality parameter (b) Modularity (c) Distance

Figure 5. Average edit distance during the first 1000 removals with different values of the quality parameter. Modularity and partition edit distance of the blogs at each time step

the same algorithms and tracking techniques than in section II using the modified Louvain Method. As we are only adding nodes and not removing them, we put all new nodes alone when starting the algorithm.

Figure 5(b) shows the modularity of the different decompositions at each time step. The Louvain Method is still the best but the modified version is really close. It performs better than in section III-A since the algorithm has more freedom. Enough nodes are alone or clearly misclassified, so the decomposition here is not too constrained and there is no need of the parameter defined in the previous sub-section. It can be set to 0 without any loss of quality.

The figure 5(c) shows the partition edit distance between each successive partition. The modified Louvain Method is more stable than the others, allowing an easier tracking of the communities. First, the network evolves a lot during its first construction and then stabilizes. There is also a detected event at the time step 40 corresponding to an event in the measurement (some blogs where added to the initial list).

Thus, starting from an empty network, the modified Louvain Method is fast, stable, and produce high quality results. To start with a network more structured and to see how the modified method reacts, we run the same tests starting with the network at time 60, and the results are very similar. The modified Louvain Method is still able to adapt the initial partition to achieve high modularity, with a better stability.

Conclusion and discussion

We have shown that classical community detection algorithms are very instable. This instability has been precisely evaluated using a partition edit distance which gives understandable values and a matching between partitions. It raises questions about the reliability of the results of these algorithms. They produce many different results, which are all evaluated as good since they have high modularity, but which one is the best? We have no answer to this question, and the proposed solution is a workaround.

The instabilities also forbid to make an effective tracking of the communities at different time steps: everything change, so it becomes impossible to identify trends and events. We have proposed a modification of the Louvain Method which achieves a really good stability without major loss of quality. This modified version also includes a parameter that allows to controll the loss of quality and the stability. If the evolution is tiny, like the removal of one node, setting x to small values like 2% is enough to achieve better modularity with still a good stability. It is one of the first algorithms taking into account the dynamic by using the previous computation and it now allows to track several communities at different time steps.

The gap between partitions cannot be considered as true events but are often due to the algorithm and removing the last artifacts is still a challenge. The possible transitions are also very limited. There is always a one to one matching, with new empty sets if needed. The transitions are still rough because of the lack of overlapping communities. We can imagine that a community does not disappear instantaneously but is slowly absorbed by some other ones and our method cannot handle this kind of transformations.

References

[1] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon, "Multiscale visualization of small world networks," in *Proc. IEEE Symposium on Information Visualization*, vol. pages, 2003, pp. 75–81.

[2] A. Schenker, M. Last, H. Bunke, and A. Kandel, "Graph representations for web document clustering," *Pattern Recognition and Image Analysis*, no. 1, pp. 935–942, 2003.

[3] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 26113, 2004.

[4] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner, "Maximizing Modularity is hard," *ArXiv Physics e-prints*, 2006.

[5] M. A. Porter, P. J. Mucha, and J.-p. Onnela, "Communities in Networks," *Notices of the American Mathematical Society*, vol. 56-9, 2009.

[6] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, pp. 27–64, 2007.

[7] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75 – 174, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6TVP-4XPYXF1-1/2/99061fac6435db4343b2374d26e64ac1

[8] P. Pons and M. Latapy, "Computing communities in large networks using random walks," *Journal of Graph Algorithms and Applications*, vol. 10, pp. 191–218, 2006.

[9] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70066111, 2004.

[10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech*, vol. 10008, pp. 1–12, 2008.

[11] R. Kumar, A. Tomkins, and D. Chakrabarti, "Evolutionary clustering," in *In Proc. of the 12th ACM SIGKDD Conference*, 2006.

[12] B. L. Tseng, Y.-R. Lin, Y. Chi, S. Zhu, and H. Sundaram, "Analyzing communities and their evolutions in dynamic social networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 2, pp. 1–31, 2009.

[13] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Tracking evolving communities in large linked networks," in *National Academy of Sciences of the United States of America*, vol. 101, no. Suppl 1. National Acad Sciences, 2004, p. 5249.

[14] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult, "Monic: modeling and monitoring cluster transitions," in *Proceedings of the 12th ACM SIGKDD*. ACM New York, NY, USA, 2006, pp. 706–711.

[15] G. Palla, A.-L. Barabasi, and T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, pp. 664–667, 2007.

[16] D. J. Fenn, M. A. Porter, P. J. Mucha, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones, "Dynamical Clustering of Exchange Rates," *Exchange Organizational Behavior Teaching Journal*, no. 21, pp. 1–35, 2009.

[17] J. Epps and J. Bailey, "Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[18] H. Kuhn, P. Haas, I. Ilyas, G. Lohman, and V. Markl, "The Hungarian method for the assignment problem," *Masthead*, vol. 23, no. 3, p. 151210, 1993.

[19] M. E. J. Newman, "The structure of scientific collaboration networks." in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 2, 2000, pp. 404–9.