Community Cores in Evolving Networks

Massoud Seifi
Pierre and Marie Curie University, LIP6
4, place Jussieu 75005 Paris, France
massoud.seifi@lip6.fr

Jean-Loup Guillaume
Pierre and Marie Curie University, LIP6
4, place Jussieu 75005 Paris, France
jean-loup.guillaume@lip6.fr

ABSTRACT

Community structure is a key property of complex networks. Many algorithms have been proposed to automatically detect communities in static networks but few studies have considered the detection and tracking of communities in an evolving network. Tracking the evolution of a given community over time requires a clustering algorithm that produces stable clusters. However, most community detection algorithms are very unstable and therefore unusable for evolving networks. In this paper, we apply the methodology proposed in [14] to detect what we call *community cores* in evolving networks. We show that cores are much more stable than "classical" communities and that we can overcome the disadvantages of the stabilized methods.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Graph Theory

Keywords

Complex networks, community structure, community cores, evolving networks

1. INTRODUCTION

The study of complex systems from a network perspective has attracted much attention since the discovery of common structural properties among such networks. Complex systems can be generally modeled by graphs, in which related entities are represented by nodes and relationships between these entities by edges. They may be encountered in the real world in various fields such as social sciences (online social networks, collaborative networks), computer sciences (Internet, Web, peer-to-peer exchange networks), biology (protein-protein interactions networks, neural networks), transportation (road networks, airline networks), linguistics (synonymy graphs, graphs of word co-occurrence in texts), etc. (see [4] for a survey).

Most complex networks of the real world evolve over time. Representing networks with graphs, this evolution can be modeled as the creation or removal of nodes and/or edges over time. For example, 700 000 new users register every day on Facebook and at the same time many users delete or deactivate their accounts. In addition, users may find new friends or remove some people from their contact list of. This dynamics plays an essential role and must be taken into account in the analysis of such networks.

One important feature of complex networks is that they are naturally composed of groups of nodes that are more densely connected with one another than with the rest of the network. These groups are generally called *communities* [7]. The identification of such communities can provide a macroscopic view on the structure of the global network but also a microscopic view inside these communities. Many algorithms have been proposed to automatically detect communities in static networks (see [10] for a survey) but few studies have considered the detection and tracking of communities in an evolving network, see [6] for instance. A typical approach for evolving networks consists in representing the state of the system at different time steps: communities are identified independently at each time step and are studied over time. Indeed, given two time steps t and $t + \delta t$ in the evolution of a graph, we can partition this graph into communities at each of these moments. Each community of the graph at time t may remain the same at the moment $t + \delta t$, disappear, split into sub-communities or merge with other communities (or a combination of everything).

Tracking the evolution of a given community over time requires a clustering algorithm that produces stable clusters i.e. which do not significantly change under small perturbations of the input data. However, in most community detection algorithms, the output changes dramatically if the input network is (even slightly) modified [1]. Some recent work [1] proposes stabilized methods but they are generally unable to track communities and to detect events because of the non-determinism that is present in their initialization.

In this paper, we apply the methodology proposed in [14] to detect what we call *community cores* in evolving networks. We show that cores are much more stable than communities and that they can overcome the disadvantages of stabilized methods. In the next section we present related work on community detection and tracking in evolving networks. We briefly explain our methodology in section 3. Then, we describe our experiments and results concerning stability on a simulated evolution and a real one in sections 4 and 5 before concluding in section 6.

2. RELATED WORK

Two main approaches are proposed in the literature for

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

identifying communities in evolving networks. The first one is based on tracking communities among different snapshots, using a community detection algorithm suited for static graphs. The major challenge after extracting communities at each snapshot is to identify which community at time t has evolved into which community at time $t + \delta t$. Many solutions have been proposed to follow the communities which may merge, split, appear or disappear over time. A simple approach to follow communities at each time step is that a community C_i at time t has become the community C_j at time $t + \delta t$ if C_i and C_j share a certain number of nodes and have similar sizes [9]. In [15], this idea is generalized by introducing many rules to handle split, merge, etc. The other approach, which has not been widely followed, consists in using temporal information directly in the detection process. For example, in [3] the quality function is modified to take into account dynamic features. Likewise, in [11] a probabilistic model integrating dynamic features is used to identify communities.

As mentioned before, the main underlying problem of both above approaches described above is that small perturbations of the input graph can greatly influence the output of classical community detection algorithms. This instability therefore makes such algorithms unusable for evolving networks.

3. METHODOLOGY

The identification of community cores is based on the idea that if several community detection algorithms, or multiple executions of a non-deterministic algorithm, agree on certain sets of nodes, then these sets of nodes are certainly more significant. More precisely, given a graph $G = (V, E)$ with $n = |V|$ vertices, we apply $\mathcal{N}$ times a non-deterministic community detection algorithm to G. In the following we use the non-deterministic algorithm known as Louvain method [2]. At the end of an execution, each pair of nodes $(i, j) \subseteq V \times V$ can be classified either in the same community or in different communities. We keep track of this in a matrix of size $n \times n$, which we denote by $P_{ij}^{\mathcal{N}} = [p_{ij}]_{n\times n}^{\mathcal{N}}$, where p_{ij} represents the fraction of the $\mathcal{N}$ executions in which i and j have been classified in the same community. Note that $p_{ij} = p_{ji}$, and we set $p_{ii} = 0$. From $P_{ij}^{\mathcal{N}}$, we create a complete weighted graph $G' = (V, E', W)$, where the weight of the link (i, j) is p_{ij}. Finally, given a threshold $\alpha \in [0, 1]$, we remove from G' all links with $p_{ij} < \alpha$ to obtain the *thresholded virtual graph*, ${G''}_{\alpha}$. The connected components in ${G''}_{\alpha}$ obtained with a given α are called α-cores, which are non-overlapping sets of nodes (see [14] for more details).

3.1 Stability evaluation

Given a dynamic graph G, consisting of a sequence of graphs G_t for each time t of its evolution, we consider $P_{ij\ G_t}^{\mathcal{N}} = [p_{ij_{G_t}}]_{n\times n}^{\mathcal{N}}$ the p_{ij} matrix of graph G_t at time t and $P_{ij\ G_{t+1}}^{\mathcal{N}} = [p_{ij_{G_{t+1}}}]_{n\times n}^{\mathcal{N}}$ the p_{ij} matrix of graph G_{t+1} at time $t + 1$.

We can comprehensively assess changes between times t and $t+1$ by calculating the Euclidean distance between these two matrices:

$$d(P_{ij\ G_t}^{\mathcal{N}}, P_{ij\ G_{t+1}}^{\mathcal{N}}) = \sqrt{\sum_{ij\in(G_t\cap G_{t+1})} |p_{ij_{G_t}} - p_{ij_{G_{t+1}}}|^2}$$

In order to study local in a simpler way, we use the matrix of changes in p_{ij}, denoted $[\Delta p_{ij}]_{n\times n}^{\mathcal{N}}$, where $\Delta p_{ij} = p_{ij_{G_t}} - p_{ij_{G_{t+1}}}$. Finally, if we consider a single node i, we can assess the impact of the evolution on this node by calculating the sum of absolute values of the i-th row of the matrix $\Delta p_{ij} = p_{ij_{G_t}} - p_{ij_{G_{t+1}}}$:

$$I(i) = \sum_{j=1}^{n} |p_{ij_{G_t}} - p_{ij_{G_{t+1}}}| = \sum_{j=1}^{n} |\Delta p_{ij}|$$

Note that when $\mathcal{N} = 1$, the core detection algorithm leads to a partition into communities that can be obtained with the Louvain method. We can therefore evaluate the changes in the community structure of the graph by setting $\mathcal{N} = 1$.

4. SIMPLE DYNAMICS

Before studying real dynamic networks, we simulate dynamics from a real static network. This dynamics must be as simple as possible, controllable and ensure that the network community structure remains stable despite this evolution. The method that we used consists in removing a single node from a static network and to study the impact of this removal on the community structure, as a function of the removed node. Although this dynamics is not realistic, it can help us estimate the cores stability and compare it with "classical" communities stability. We can also study the impact of the removal of any given node on the network structure.

We applied this simple dynamics to an Email network consisting of 1133 nodes and 5451 edges [8], and to the NetSci collaboration network, which gathers 379 researchers in the field of networks [12]. In both cases we have considered for each node u the initial network before and after the removal of this node. We note $G_u = G\backslash u$ the graph obtained by simply removing the node u.

4.1 Cores stability

To assess the cores stability and to compare it with the communities stability, we first calculated the p_{ij} matrix of the initial graph $G(V, E)$, denoted $P_{ij\ G}^{\mathcal{N}}$, and the matrices of graphs G_u for all nodes $u \in V$, denoted $P_{ij\ G_u}^{\mathcal{N}}$, by applying the core detection algorithm with $\mathcal{N} = 1000$. Then we calculated Euclidean distances between $P_{ij\ G}^{\mathcal{N}}$ and all $P_{ij\ G_u}^{\mathcal{N}}$ matrices. Figure 1 shows the cumulative distributions of these distances for the Email network. For $\mathcal{N} = 1$ we executed the algorithm five times to ensure that no pathological case occurred. We may observe that distances between cores matrices are much lower than those of the community structure.

In order to have a better understanding of these results, we also calculated the distance between the p_{ij} matrices for two different runs with $\mathcal{N} = 1$ and $\mathcal{N} = 1000$, see Figure 2. This figure shows a scatter plot illustrating the similarity between the results of two executions: each point corresponds to a node and coordinates (x, y) imply that the deletion of this node has had an impact of x in the first execution in terms of distance and y in the second one. With $\mathcal{N} = 1$ there is a high volatility: deleting a node has sometimes a strong impact and sometimes a low impact, without any visible correlation. Instead, for $\mathcal{N} = 1000$ there is a strong correlation that comes from the fact that p_{ij} values have nearly converged when $\mathcal{N} = 1000$.

However, core p_{ij} matrices contain real values between 0 and 1 while communities p_{ij} matrices are binary and contain only 0 or 1. For a more accurate comparison of these two

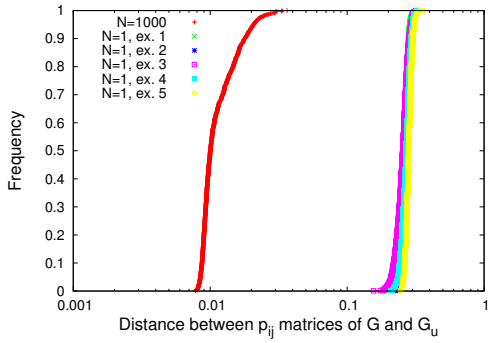
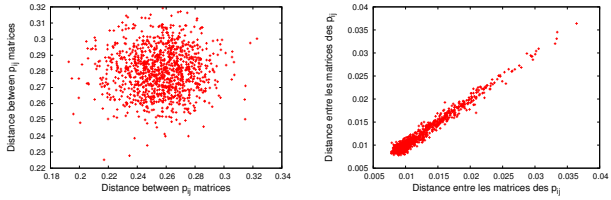


Figure 1: Cumulative distributions of Euclidian distances between matrices before and after removing a node for cores and communities in Email network.



(a) Communities (N=1) (b) Cores (N=1000)

Figure 2: Correlations between two executions of cores with $\mathcal{N} = 1$ and $\mathcal{N} = 1000$. Each point of coordinates (x, y) corresponds to a node and means that the Euclidean distance between the p_{ij} matrices before and after its removal is x for the first execution and y for the second one.

types of matrices, we can transform cores matrices in binary matrices using thresholds. For this, given a threshold α and we assign 0 to all elements with a $p_{ij} < \alpha$ and 1 to all others:

$$P'^{\mathcal{N}}_{ij\,G_u} = [p'_{ij}]^{\mathcal{N}}_{n\times n} \quad with \;\; p'_{ij} = \begin{cases} 0 & p_{ij} < \alpha \\ 1 & p_{ij} \geq \alpha \end{cases}$$

We computed the Euclidean distances between the thresholded matrices of p_{ij} of G and G_u i.e. $P'^{\mathcal{N}}_{ij\,G}$ and $P'^{\mathcal{N}}_{ij\,G_u}$ for all $u \in V$ for different thresholds α. As illustrated in Figure 3, we observe that the variation of p_{ij} for cores is still lower than the variation for communities (see Figure 1) but higher than non-thresholded matrices. This means that thresholding includes information loss.

In this experiment, the thresholds chosen for matrices of G and G_u, α and α_u were the same, but it is possible that the cores of G with a threshold α are very similar to the cores of G_u with a completely different threshold α_u. We therefore also studied the distances between matrices when α differs from α_u. In order to do this, we tested all combinations of α and α_u from 0.5 to 1.0 with an interval of 0.01. We computed the distances between the thresholded matrices of p_{ij} of G and G_u with α and α_u for all $u \in V$. Figure 4 illustrates distances for all α and α_u for two graphs. This figure shows that distance is generally minimal when α and α_u are close.

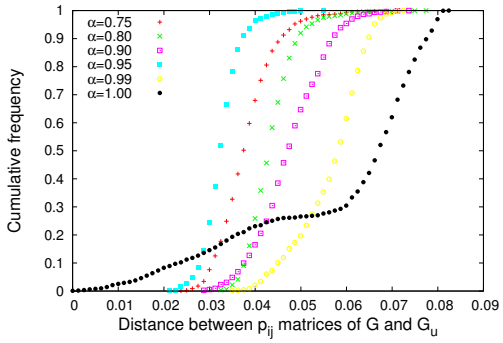
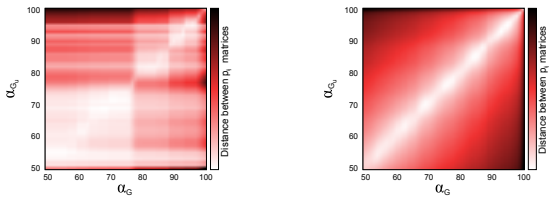


Figure 3: Cumulative distributions of Euclidean distances between matrices before and after removal of a node for thresholded cores in Email Network.

In our case, it is therefore more interesting (and easier) to keep the same threshold to track communities.



(a) NetSci (b) Email

Figure 4: Distance between p_{ij} thresholded matrices of G and G_u with different thresholds α and α_u. The clearer the color, the smaller the distance.

4.2 Impact on the structure of cores

We also calculated the matrix of variations $\Delta p_{ij} = p_{ij\,G} - p_{ij\,G_u}$ and studied the distribution of values in this matrix. We may observe two different types of distributions, as shown in Figure 5 for a simulated dynamics of the Email network. Figure 5(a) shows the first type of distribution of p_{ij} variation before and after the removal a node. We see that the Δp_{ij} values in this figure are very close to 0, which means that removing this type of nodes has little influence on the structure of the cores. However, for some other nodes (Figure 5(b)) we may observe much higher values, close to -1 and 1.

A Δp_{ij} value of 1 means that i and j have never been set together before the removal and are always together after, which means i joined the core of j or vice versa. Similarly, a Δp_{ij} of -1 means that i and j have always been together before deletion and are no longer together after, which means that the core containing i and j has been split. Many values close to -1 or 1 in the distribution thus mean that the deletion of the considered node has a large impact on the structure of the cores.

Automatic detection of strong impacts is not obvious, we therefore decided to study the maximum and minimum values of the distribution. Figure 6 shows these extreme values after removing a node, for all nodes in the Email network

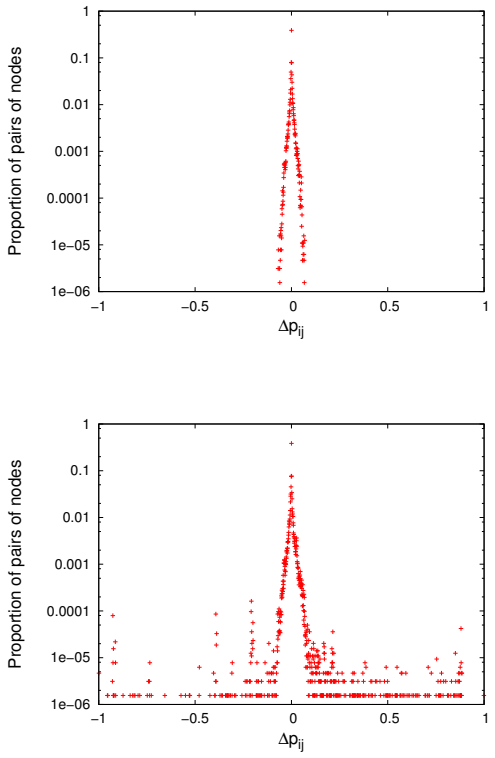
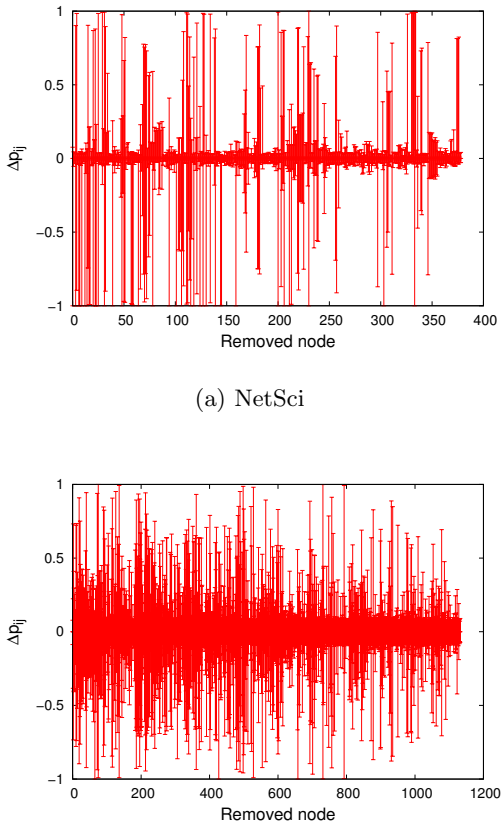


Figure 5: Distributions of Δp_{ij} values for (a) a node with a low impact (distribution peaked around 0), and (b) a node with a strong impact (distribution with values close to 1 and −1).



(a) NetSci

(b) Email

Figure 6: Extreme variation of Δp_{ij} values on the Email network and the collaboration network. Each bar connects the minium Δp_{ij} to the maximum Δp_{ij}.

and researchers network. We see cases where the removal of a node has a strong impact (high extreme Δp_{ij} values) and others where the impact is limited and close to 0. In particular the average values are higher in the Email network than in the collaboration network, where most impacts are very low.

We can also observe specific behaviors for nodes that have a strong impact in one direction only. For example, when node 376 is removed from the researchers network, we only see a positive variation of p_{ij} (it is not directly visible on the curve given the scale). Figure 7(a) shows the distribution of p_{ij} variations before and after the removal of this node. We observe that the majority of Δp_{ij} values are close to 0 and a few values are close to 1, which corresponds to cores merging. Figure 7(b) shows the distribution of impacts $I(i) = \sum_{j=1}^{n} |\Delta p_{ij}|$ on nodes. We observe that 8 nodes are very strongly impacted (with a total impact of 26) and a number of nodes are quite severely affected (with a total impact close to 7).

Figure 8 illustrates the cores identified with $\alpha = 0.75$ before and after the removal of node 376 (pointed by an arrow). We can clearly identify the 8 nodes that are merged with another core.

Figure 9 shows this result with a different type of visualization and shows how a node is affected by the deletion of another node in the graph. Each (i, j) pixel indicates the total impact of the removal of node i on node j. A light horizontal line for a node means that it is never impacted by the removal of other nodes. These nodes are more resistant to changes in the network. On the contrary, a dark horizontal line for a node indicates that this node is more sensitive and that whatever the deleted node, it is strongly impacted. Conversely, a dark vertical line indicates the deleted node has an important role in the structure of the core because its deletion affects all other nodes. On the contrary, a clear vertical line means that the deleted node does not impact anything. There are no light or dark vertical lines, which means that no node has an impact on all nodes, which seems natural, but we see horizontal lines for important nodes.

4.3 Impact locality

We have shown that cores are much more stable than traditional communities when the graph is slightly modified, although the removal of some specific nodes has a stronger impact. We now study whether the changes in the structure of cores and communities are local or not.

To illustrate the quality of cores versus communities, we have removed an off-centered node of degree 1 (node 36). Figure 10(a) shows the distribution of p_{ij} variation before and after the removal of this node. Deleting this node should not change the community structure of the network. How-

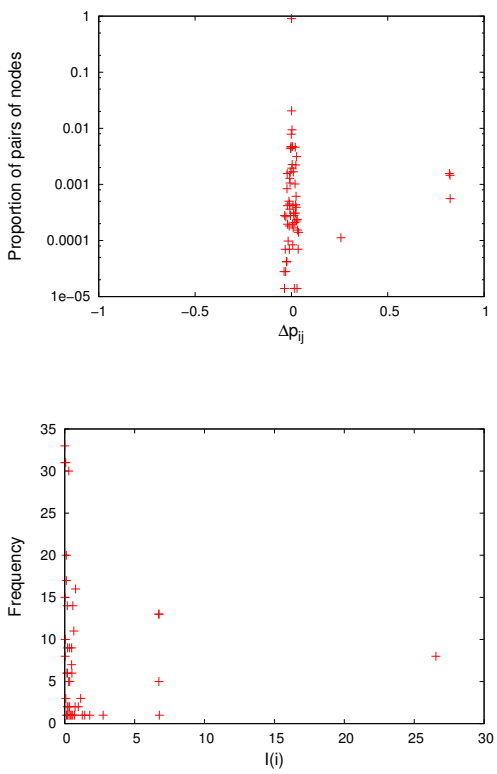
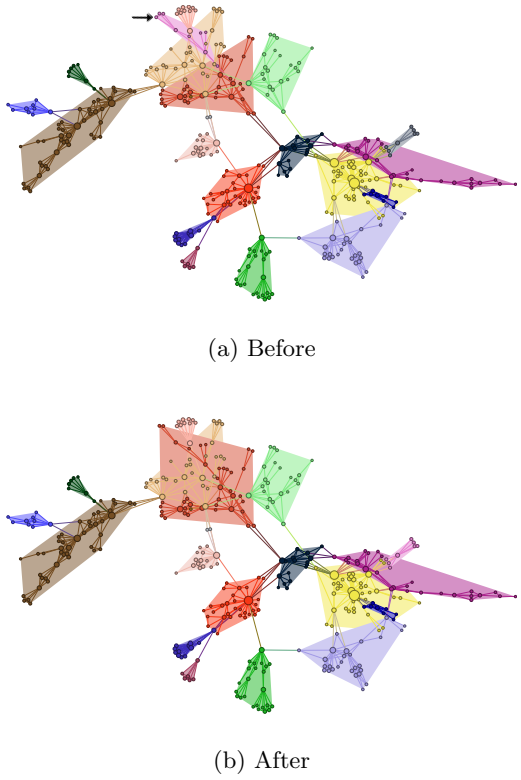


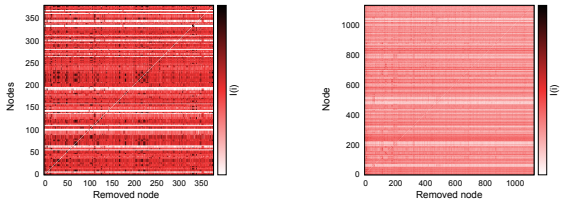
Figure 7: (a) Distribution of Δp_{ij} before and after the removal of node 376 from the researchers network. (b) Distribution of total impacts, $I(i)$, for the deletion of each node i.



(a) Before

(b) After

Figure 8: Cores identified (a) before and (b) after removing node 376, pointed by an arrow in the upper left corner.



(a) NetSci (b) Email

Figure 9: The pixel (i, j) indicates the total impact of the removal of the node i on the node j, for two networks.

ever Figure 10(b) shows that many nodes are severely affected by this deletion for communities ($\mathcal{N} = 1$) while no node is affected for the cores ($\mathcal{N} = 1000$).

Moreover, figure 11 shows the average impact of removing a node on other nodes of the graph, according to their distance from the deleted node (for two different networks). We observe that this impact is low for cores and occurs generally near the removed node.

However, we sometimes see strong impact on more distant nodes. For example, as illustrated in Figure 12, removing the node 112 from the researchers network has a strong impact on nodes that have a distance of 10 and 11 from it. Figure 13 shows this network where node colors indicate how it is impacted by the removal of the node 112 (indicated by an arrow). The more a node is impacted the more "red" is. The nodes in a distance of 10 and 11 from node 112, which are significantly impacted, are displayed in a circle.

This result is both negative for cores and also very counter-intuitive since cores are supposed to absorb non-local variations due to instabilities of the algorithm. We tried to understand the cause and we have therefore studied several cases of nodes whose removal causes a clearly non-local impact. The conclusion is that what we observe here is related to the problem of resolution limit [5]. This problem implies that the larger the network, the larger the communities and we see that this problem exists also for cores. As Figure 14 shows, nodes affected by the removal of node 112 split into two subcores. The same effect occurs with the removal of other nodes of high degree. On the other hand, removing low degree nodes does not cause this breakage.

Validating these results and defining a methodology to distinguish cases of resolution limit from cases of actual impact is an important perspective of this work.

5. REAL DYNAMICS

Finally, we considered a more realistic case of dynamics. One major application of our work consists in automatically detecting events in the evolution of real networks. Although it is difficult to clearly define what an event is,

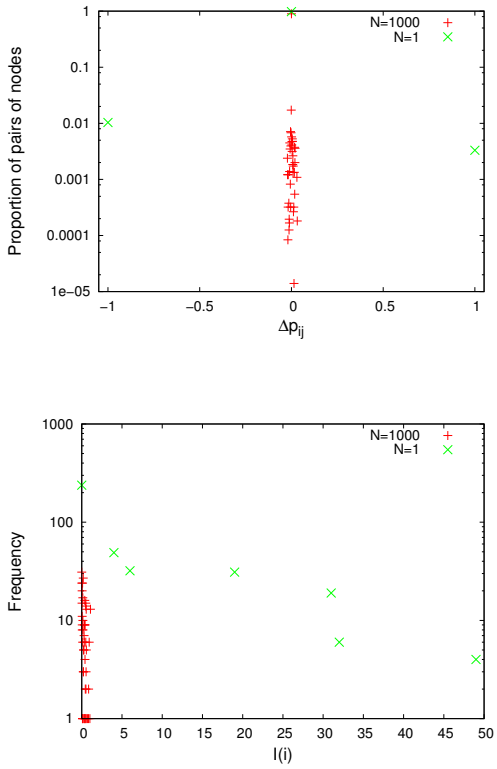


Figure 10: (a) Distribution of p_{ij} variation before and after the removal of node 36 from the researchers network. (b) Distribution of the number of nodes with a given impact, for this removal for $N = 1$ and $N = 1000$.

we aim at identifying fundamental, occasional and unusual changes that do not conform to the "expected behavior". In the context of cores it may be a significant variation in the structure.

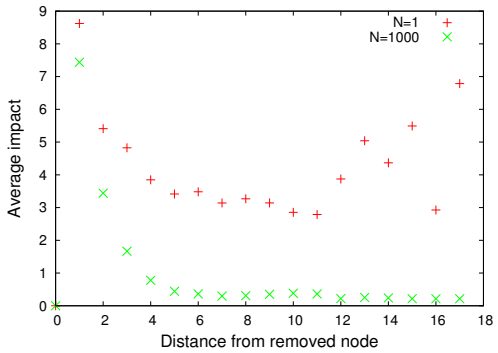
In this section, we use the `mrinfo` network as an example of dynamic network. This network has the advantage of having a quite slow evolution but with some clearly identified events.

5.1 Mrinfo network

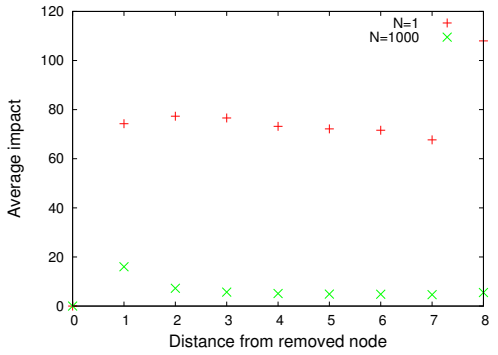
This network is a map of the topology of multicast routers on the Internet, measured using the tool `mrinfo`. This tool allows to request the list of neighbors from a multicast router. Every day, `mrinfo` is launched on a first router then recursively on each of its neighbors in a breadth-first search manner. This measurement was conducted over several years and resulted in a dynamic map of multicast routers (see [13] for more details concerning the measurement). We studied the data of 2005 which represent 365 time steps (days) containing 3114 nodes and 7523 edges in average (see figure 15).

5.2 Evolution of cores

It is clearly shown in [1] that the direct application of classical community detection algorithms, such as the Louvain method, independently at each time step is not suitable for monitoring of communities because of their instability. We



(a) NetSci



(b) Email

Figure 11: Node deletion. Average impact on the nodes of distance k from the removed node depending on k.

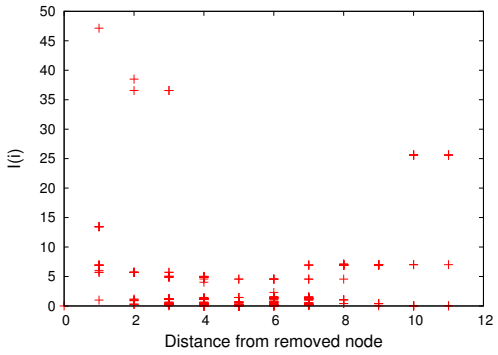


Figure 12: Impact of removing the node 112. Each point of coordinates (x, y) is a node i of distance x from the node 112 where $I(i) = y$.

will compare our approach to the Louvain method and the stabilized one proposed in [1].

Figure 17 illustrates the Euclidean distance between two partitions of successive time steps for the Louvain method, the stabilized Louvain method and cores. We observe that cores are much more stable than the Louvain method, but

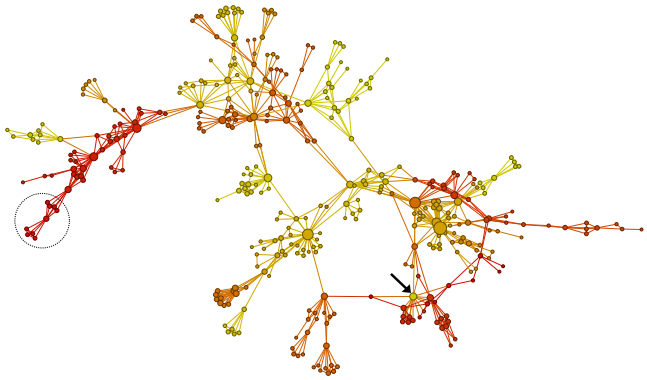
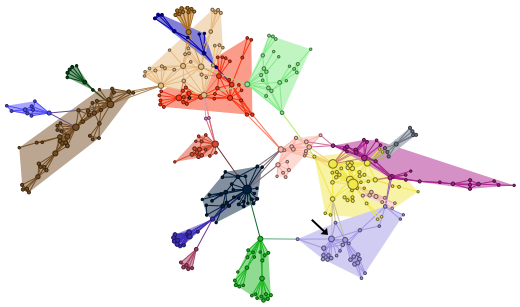
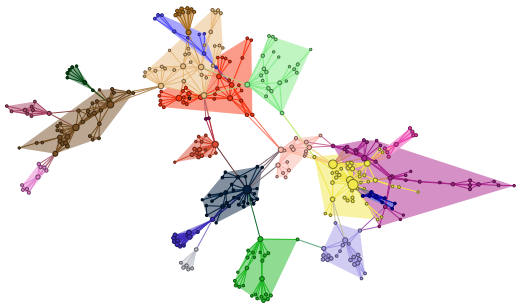


Figure 13: Impact of removing node 112 (pointed by arrow) on the other nodes of the researchers network. The color of a node indicates how strongly it is affected by the deletion, the intense red indicates the strongest impact.



(a) Before



(b) After

Figure 14: The cores identified before and after the removal of node 112 with demonstration of the problem resolution limit for the brown cores at left.

less stable than the stabilized Louvain method. We chose a threshold 0.86 for cores, which gives the better stability.

Although the stabilized Louvain method is more stable, it has one major flaw which is that it uses the partition of the previous time as the seed of the current time. This helps the algorithm but simultaneously imposes a constraint because it can be difficult to radically change the imposed partition. The results of the algorithm can be strongly associated with

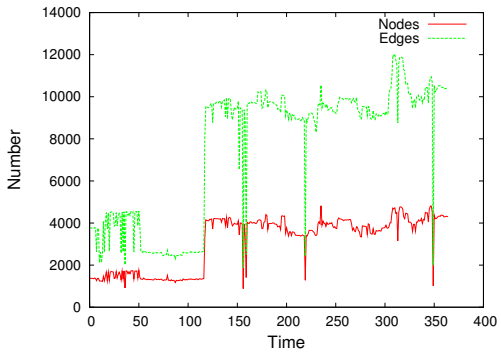


Figure 15: Changes in the number of nodes and links over time in the `mrinfo` network.

the first initial partition whose significance is limited. As As illustrated in Figure 16, when applying the stabilized Louvain method several times to the same network, some peaks of modifications remain in place and others happen at different instants in time. This means that the events observed with this method are quite dependent on the first identified partition and are therefore unreliable. For cores, peaks indicating large variations are always located in the same time steps since the calculations are time independent.

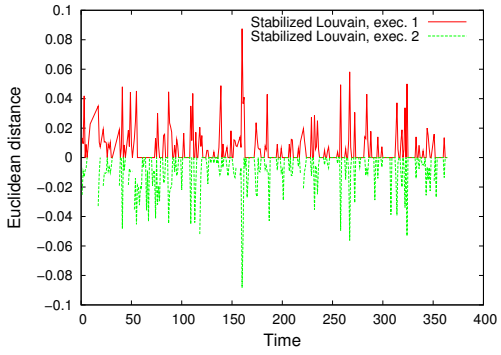


Figure 16: Euclidean distance between two successive time steps for two different executions of the stabilized Louvain method on the `mrinfo` network. The results of the second execution are shown with negative values to increase curves readability.

In all cases, we observe regular changes in the structure of cores. We should then be able to eliminate the effects related to the resolution limit that can pollute the results and then to validate the quality of the obtained decompositions.

6. CONCLUSION

We have shown that classical community detection algorithms are too unstable and therefore unsuitable to be applied to dynamic graphs. Indeed, very low system disturbances can produce a major transformation of the results. Stabilized approaches exist but are generally unable to identify events and to follow communities because of non-determinism in their initialization.

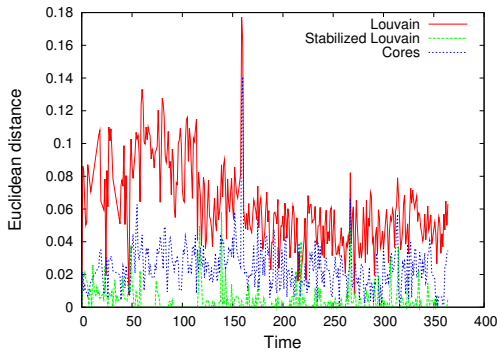


Figure 17: Euclidean distance between p_{ij} matrices of between two successive time steps for the `mrinfo` network.

We studied the cores stability in dynamic graphs, first with a simple and controllable dynamics where we showed that cores are much more stable than communities. We also showed that the influence of changes in the structure of the cores is overwhelmingly local, with the exception of resolution limit, while impacts are usually global for communities and therefore difficult to interpret. We then showed the effectiveness of our approach by applying it to real dynamic graphs on which we have shown that cores are more stable than conventional approaches and although less stable than some stabilized algorithms, but at least they are deterministic.

Research on communities in dynamic graphs is still in its infancy and many opportunities exist. First, we should clearly validate cores dynamics, which requires finding formal arguments to distinguish real changes from those due to the resolution limit, or even find a way to eliminate this problem. Second, we should validate the dynamics of real cores on graphs, for example by comparing detected events with high Euclidean distances to known and unknown events. Of course this requires multiple graphs with several types of dynamics (fast or slow, ...). Finally, it is possible to follow other conventional approaches for community detection in dynamic graphs and apply the methodology of cores to them. For example the stabilized Louvain method is non-deterministic and provides a partition per time step. Rather than calculating cores at every moment, it would be possible to calculate dynamic cores seeking similarities between different executions of the stabilized Louvain method for example.

7. ACKNOWLEDGMENTS

This work is supported in part by the French National Research Agency contract DynGraph ANR-10-JCJC-0202 and by the DiRe project, funded by the city of Paris Émergence program.

8. REFERENCES

[1] T. Aynaud and J. Guillaume. Static community detection algorithms for evolving networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 513–519. IEEE, 2010.

[2] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.

[3] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560. ACM, 2006.

[4] L. Costa, O. Oliveira Jr, G. Travieso, F. Rodrigues, P. Boas, L. Antiqueira, M. Viana, and L. Da Rocha. Analyzing and modeling real-world phenomena with complex networks: A survey of applications. *Arxiv preprint arXiv:0711.3199*, 2007.

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

[6] M. Giatsoglou and A. Vakali. Capturing social data evolution via graph clustering. *IEEE Internet Computing*, 99(PrePrints), 2012.

[7] M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821, 2002.

[8] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68(6):065103, 2003.

[9] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *Proceedings of the national academy of sciences of the United States of America*, 101(Suppl 1):5249, 2004.

[10] A. Lancichinetti. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[11] Y. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. Tseng. Analyzing communities and their evolutions in dynamic social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(2):8, 2009.

[12] M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[13] J. Pansiot, P. Mérindol, B. Donnet, and O. Bonaventure. Extracting intra-domain topology from mrinfo probing. In *Passive and Active Measurement*, pages 81–90. Springer, 2010.

[14] M. Seifi, S. Iskrov, J.-B. Rouquier, I. Junier, and J.-L. Guillaume. Stable community cores in complex networks. In *Studies in Computational Intelligence*. Springer, 2012.

[15] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. Monic: modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 706–711. ACM, 2006.