Scientific Modeling Computer Laboratory

Project: Time Evolving Networks

Fourth Bi-weekly Report

Ádám Gergely Szabó

25th of April, 2022

1 Introduction

This project is about exploring MTMT's co-partnership network, which evolves in time as more publications get submitted to the site. MTMT's main goal is to host a site that maintains high quality publications, meaning the submitted works are often checked and rated for their quality. The data stored by the site is publicly available, thus data can be gathered from the site without registration.

This project's goal is to explore the co-partnership network of MTMT. In this work, we will look at this network in different given states or its subsets, see how it develops in time, calculate different central indicators, apply different group searhcing methods and embeddings. The work will be mostly done in Python3 language that will be utilized in the Jupyter Notebook framework.

2 Progress

2.1 Motif Finding and Z-score

Previously, I have brought up motif finding as an entrance level of group searching. This is due to motifs being predefined and as such we only get information about how many of the motifs we find. This means that we have to compare the results: in this case, the randomisation of the network is needed. In my work, I randomize the edges in the graph by double edge swap, which keeps the degree distribution unchanged. In the randomization, only 30% of the edges will be randomized. Comparing the original network to its randomized counterpart is not good enough, multiple randomized networks are needed. With this, the z-score can be defined:

$$z = \frac{\langle m_i \rangle_g - \langle m_i \rangle_{rand}}{\sigma_{rand}} \tag{1}$$

where $\langle m_i \rangle_g$ is the number of times that motif i was found in the original network, $\langle m_i \rangle_{rand}$ is the average of the number of times motif i was found and σ_{rand} the deviation of the number of times motif i was found in the network.

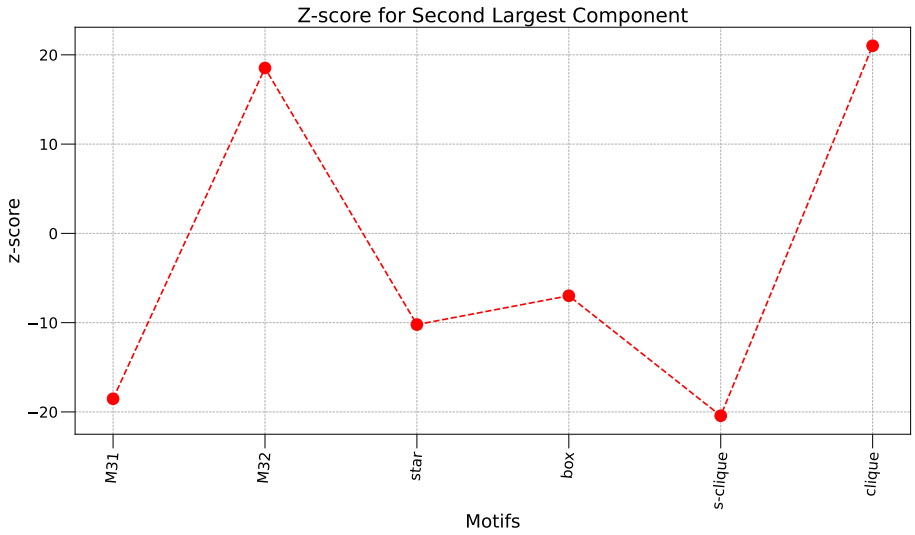

Figure 1: Z-score for the second largest component. Negative values mean that it was found more times in the randomized networks, positive means the opposite.

There are only two more problems remaining: the second largest component is not even comparable to the network size and searching algorithm for motifs as the VF2 algorithm scale with the network size as $O(n^3)$ which is bad for large, sparse graphs. I will discuss the solution or work-arounds in the following.

2.1.1 Parallel Processing

The first and most obvious solution is to utilize more processing power as python runs on a single core. With the multiprocessing library in python, multiple jobs can be given out to the CPU. These makes the individual processing time for a single calculation longer, but due to having multiple calculations done together, the total processing time decrease significantly. When tested on the second largest component, a speed up of 2.5 times was reached. Indeed, this is major step forward for faster calculations, but does not scale well with the network size.

2.1.2 Cutting of Part of the Network

Furthermore narrowing could speed up the motif searching even more. The subgraph of the network was created by searching for given authors, in this case, lecturers from the university, and searching for their neighbors and neighbors of neighbors. The following

people were choosen: Gergely Palla, István Csabai, Tamás Vicsek, Oroszlány László, József Stéger. Suprisingly the subgraph consists of one component, it has 7148 nodes and 17417 edges, which is comparable in size of the original network. After 3 hours, the results were the following:

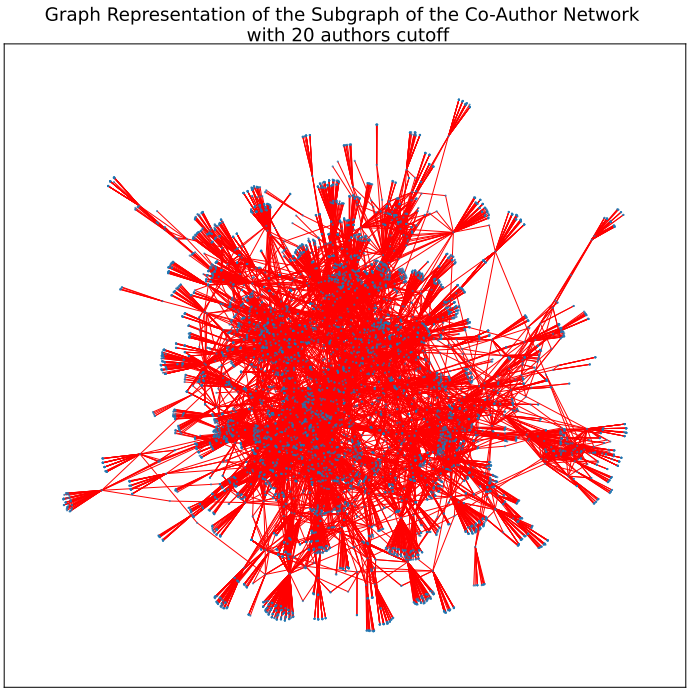

Figure 2: Graph representation of the cut-off subgraph which was created by the choosen authors as starting point of their neighborhood exploration.

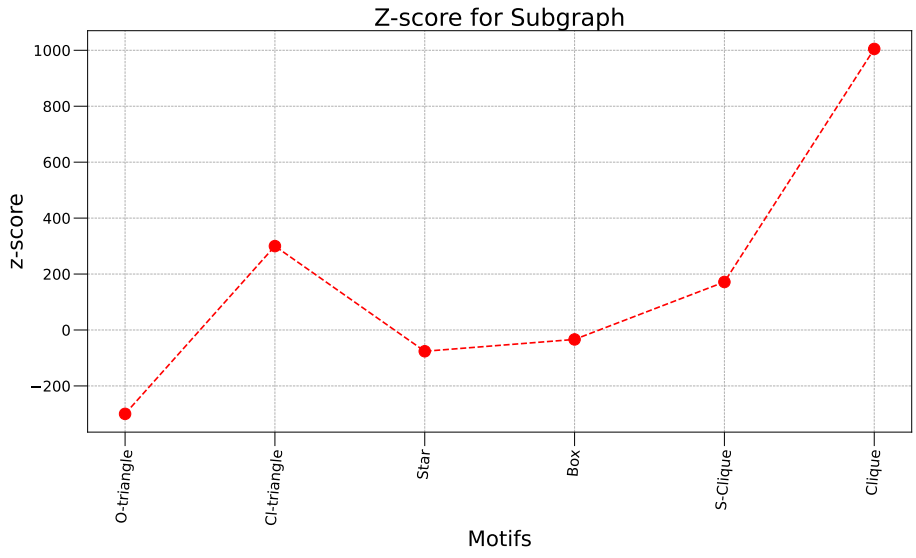

Figure 3: Z-score for the cut-off subgraph. It seems that the fully connected motifs are presented more times in the subgraph.

2.2 Greedy Modularity

This method is a hierarchical clustering which means that the nodes are joined joined together into a community depending on the distance. But if we do this for more steps, in the end we will get a single community containing all the nodes. We have to stop this with the introduction of modularity:

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

where A is the adjacency matrix k_i is the degree of i, γ is the resolution parameter, $\delta(c_i, c_j)$ is 1 of i and j is the same community and zero otherwise.

We can go further with this:

$$Q = \sum_{c=1}^{n} \left[\frac{L_c}{m} - \gamma \left(\frac{k_c}{2m} \right)^2 \right] \tag{3}$$

Where L_c is the number of intracommunity links for community c, k_c is the sum of degrees in the nodes in community c and we iterate through over all communities c. After this, we have to look for the maximum of modularity as show by [2] which defines were to stop the community finding. With this method, there were 43 communities being found, which is many and hard to distinguish. The other problem is that lightly connected nodes in hierarchical clustering still being pulled into communities and as a result all nodes are part of a community or another.

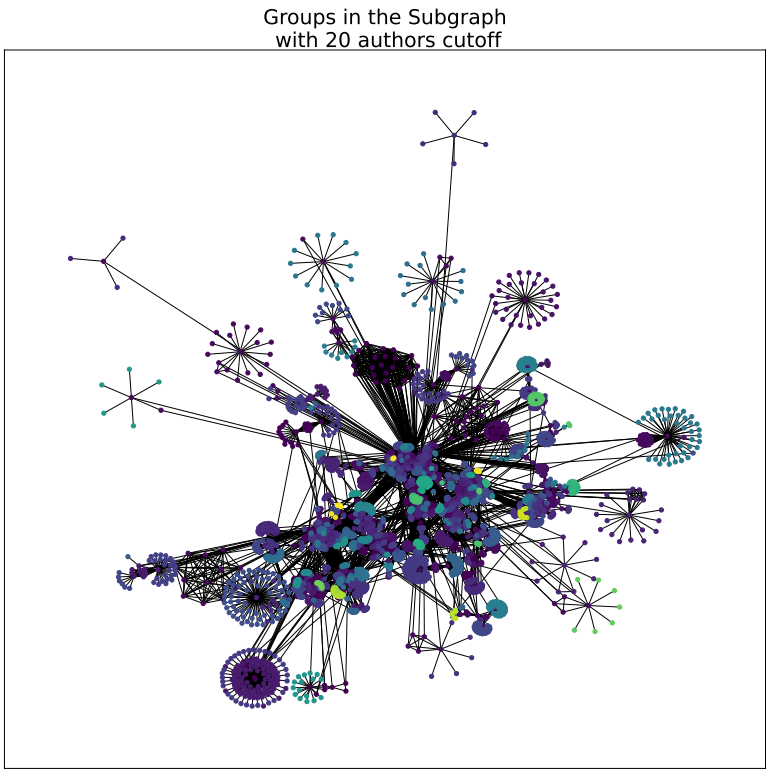

Figure 4: Communities in the subgraph. Because there are 43 communities, it is hard to identify them.

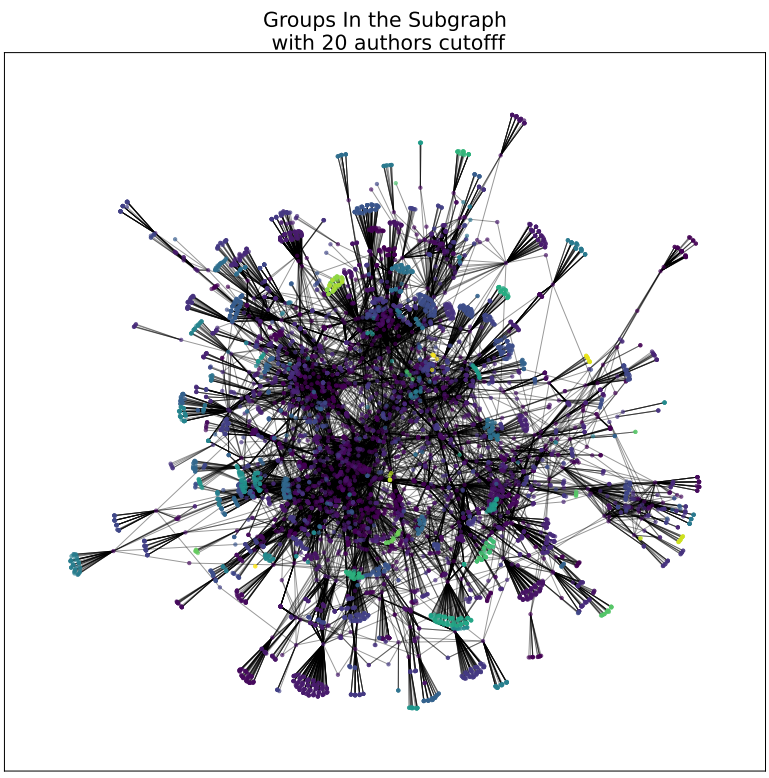

Figure 5: Communities in the original layout.

3 Discussion

Till the making of this report, a good amount of progress was made: a subgraph was prepared to make analysis much faster, the multiprocessing library was utilized to speed up calculation and thus lower processing time and more of the available processing power got used. Sadly, there is still improvement with the representations: its hard to identify each community due to having 43 of them. This calls for further exploration of the chosen subgraph, as the nodes in the periphery of the network are visible coming from one fully connected graph, but they are not connected to each other and as such, it may give wrong results.

Furthermore, more community finding methods need to be applied to the subgraph and the implementation of embeddings are still waiting.

References

[1] Albert-László Barabási. Network science. *http://networksciencebook.com*, 2012.

[2] Aaron Clauset, M. E. J. Neumann, and Cristopher Moore. Finding community structure in very large networks. 2004.

[3] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx, in proceedings of the 7th python in science conference (scipy 2008), 2008.

[4] Xiaoming Liu, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. Co-authorship networks in the digital library research community, 2005.