Visualization of Hierarchical Communities in Large Scale Networks

Adrien Baland* KU Leuven

Raghvendra Mall† ESAT-STADIUS KU Leuven

Rocco Langone‡ ESAT-STADIUS KU Leuven

Johan Suykens§ ESAT-STADIUS KU Leuven

ABSTRACT

In this paper we design a tool that allows visualization of hierarchical communities extracted from complex networks similar to Google maps i.e. the tool allows to zoom in and zoom out of communities at different levels of hierarchy. This tool uses the hierarchical structure of the communities to attack this problem in a recursive manner in order to avoid an ever-increasing complexity. We incorporated several additional functionalities that allows the user to modify the communities and/or their layout in the tool. The tool is implemented with the **R** library **Shiny**.

The proposed tool requires the community membership of each node for the different levels of hierarchy as well as the network configuration i.e. the set of connections between all nodes. Based on these data, a few choices regarding the layout can be made by the user and the tool then proceeds to render these data in a 2D layout.

Index Terms: I.3.1 [Computer Graphics]: Visualization Tool—Hierarchical Community Visualization;

1 INTRODUCTION

In the modern era complex networks are ubiquitous. Their omnipresence is reflected in domains like social networks, web graphs, road graphs, communication networks, biological networks and financial networks. Complex networks can be represented as graphs $(G(V,E))$, where the nodes (V) represent vertices in the graph and edges (E) depict the relationship between these nodes. Real world networks exhibit community like structure, where nodes within a community are densely connected and the connections are sparse between the communities. The problem of community detection has rich literature including [1], [2] and [3].

A subclass of these detection algorithms attempts to create a hierarchical community structure, divided in H levels of hierarchy. For each level h ($h = 1,\ldots,H$), the algorithm returns a set of communities C_i^h ($i = 1,\ldots,N_h$) such that three conditions are ensured:

- Each node belongs to exactly one community at each level.
- Each community for a given level (except for $h = 1$) is strictly contained in one and only *parent* community ($\forall h' = 2,\ldots,H, j = 1,\ldots,N_{h'}, \exists i \in \{1,\ldots,N_{h'-1}\} : C_j^{h'} \subset C_i^{h'-1}$).
- The size of a *parent* community corresponds to the sum of the size of its children communities.

Some algorithms which create such a hierarchical structure are the Louvain [1], OSLOM [2] and AH-KSC [3] methods.

Based on the list of connections between the nodes these algorithms return a set of community memberships for each level of hierarchy. These memberships along with the connection list is all that our tool requires as input. With this information the user selects one layout out of several possible layouts for the communities at that level of hierarchy and then the tool proceeds to place every community in the 2-D space. The hierarchical structure is used to recursively place communities based on the parent location in order to avoid a higher computational complexity. The users are provided with a range of possibilities that can be explored to improve the visualization if required.

To the best of our knowledge no software exists currently that allows the user to navigate through more than two hierarchical levels on a given visualization palette. The primary contribution of this work is to design a tool which overcomes this drawback and allows hierarchical visualization of communities in complex networks.

*e-mail: adrien@baland.be

†e-mail:Raghvendra.Mall@esat.kuleuven.be

‡e-mail:Rocco.Langone@esat.kuleuven.be

§e-mail:Johan.Suykens@esat.kuleuven.be

2 EXPOSITION

Several points have to be taken into account during the construction of this tool, from the placement method of the communities for a given level to the additional functionalities a user might desire.

2.1 Community layout

To place communities in a two-dimensional space for the top hierarchy level (largest communities), our tool currently offers various methods with different computational complexities. These methods take the similarity between each pair of communities into account, where this similarity is proportional to the ratio of the number of connections between the communities and their respective sizes:

- Circular/Spiral layout : The layout is divided into a user-defined number of layers all surrounding a *sun*, chosen as the largest community. Based on the number of layers, the number of communities inside a single layer say n is also given. On the first layer (aside from the sun), the n most-similar communities to the sun are selected and placed circularly at the same distance d from this sun

$$i\text{-th position} : \left[d\cos\left(\frac{2\pi i}{n+1}\right), d\sin\left(\frac{2\pi i}{n+1}\right)\right] \quad i = 1,\ldots,n$$

  where a similarity-based greedy approach is used to determine the order of the communities on that layer. On every outer layer, the communities chosen are picked as those who are the most similar to the community in the inner layer. The difference between spiral and circular layout lies in a slight rotation that is applied between two successive layers.

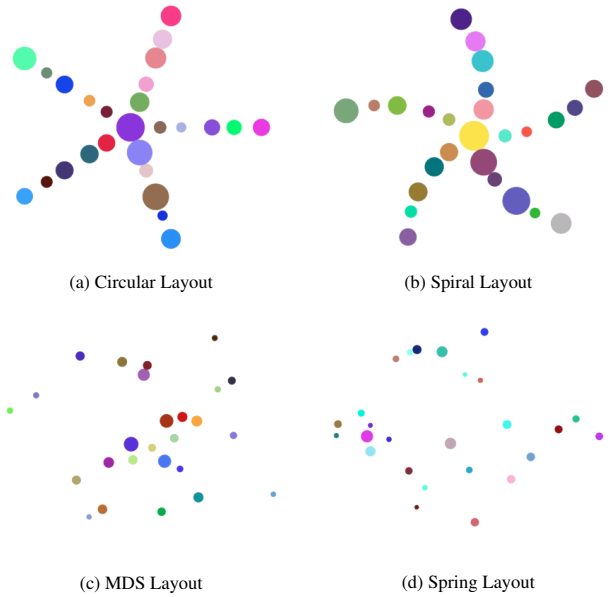
- Multidimensional scaling (MDS) : Using exclusively the similarity between each pair of community (which can be converted to a distance matrix), MDS techniques are used to project these communities into a lower-dimensional space (here, two dimensions) while trying to retain distances between communities.

- Spring system : Force-directed graph drawing algorithms are very useful in graph visualization. In these algorithms, all connections between vertices are replaced by springs $s \in S$, characterized by their natural length r_s and their stiffness coefficient k_s, and the node's optimal position minimizes the potential energy of the system

$$\min \sum_{s\in S} k_s (r_s - ||\vec{x}_{s_1} - \vec{x}_{s_2}||)^2$$

with $\vec{x}_{s_1}, \vec{x}_{s_2}$ the two end-points of spring s. Projecting our problem into this framework, our communities become the vertices and the springs parameters between two vertices are functions of both the size of the clusters and their similarity.

- *Others* : Our tool also offers the possibility to easily use a custom positioning method.



(a) Circular Layout (b) Spiral Layout

(c) MDS Layout (d) Spring Layout

Figure 1: Some layouts offered by our tool.

Based on the position computed by these methods for the top hierarchy level, the communities are placed as circles on the 2D plane (an example layout is shown in Figure 1), with the area of the circle being proportional to the size of the community and are placed in a fashion s.t. the communities do not overlap. After that, the hierarchical structure is used to position the children communities in the following top-down approach: For every parent-community (1) find all of its immediate children communities; (2) compute their position with the same method as presented above; (3) re-scale and translate these positions so that all children are located inside of it. Because of this, several small positioning problems are solved sequentially instead of one massive problems.

2.2 Visualization

With all these communities positioned on the grid, the user can navigate through the layout by means of the following ways:

1. *Level-zoom* : Completely zoom inside a given community of the current hierarchy level and switch the hierarchy level. All communities associated with the parent are displayed on the palette and the borders of the parent community disappears. Now, only its descendants become visible, with its children as the new *top-level*. At any point the user can go back to the previous level.

2. *Spreading-zoom* : A drawback from our recursive approach and from the fact that we use circles to represent communities is that children communities overlap inside their parent as the size of the parent corresponds to the sum of the children sizes. Because of this, the user can apply partial zooms to the layout, which progressively spread the children outside of their parents to reduce overlaps between descendants.

3. Community information : Upon placing the cursor on a given community (on the top-level or one of its descendants), a series of information about this community (its ancestors if the selection is not on the 1^{st} level) appear on the screen (size, name, most-related community) etc.

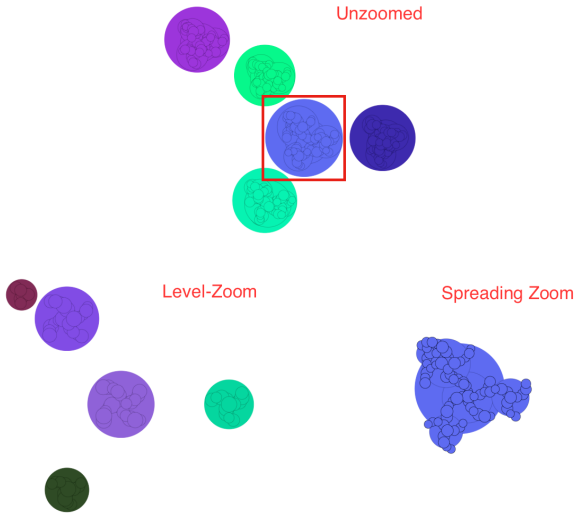


Figure 2: Difference between zoom types for the red-framed community : change of hierarchy level vs spread of children outside the parent.

2.3 Additional Functionalities

Next to these compulsory elements of our tool given its purpose, a few other handy functionalities have been added to make it more practical. These functionalities include (1) removal of a community from the data (only available on the top-most hierarchy level to avoid a mis-match between parent size and children sizes); (2) manual relocation of a community on the grid; (3) merger of two communities together (while re-positioning their children inside this new community); (4) change of a community name; (5) display of the top m (user-defined) connections for some communities; (6) save all of the current data into a *.zip* to resume the work later on.

3 Conclusion

In this work we presented a tool that allows the user to visualize hierarchical communities in a two-dimensional grid over any number of hierarchy levels. This tool takes advantage of the hierarchical structure to decrease the complexity of the placement problem and allows user to apply some changes to either the community information or to the results given by the algorithm.

The hierarchy is also used to allow the user to switch from one level to another to focus on sub-communities of a given parent and provides basic configuration details about any community that the user selects. This allows the user to go through this community information in a relatively user-friendly way.

Acknowledgements

The work is supported by Research Council KUL, ERC AdG A-DATADRIVE-B, GOA/10/09MaNet, CoE EF/05/006, FWO G.0588.09, G.0377.12, SBO POM, IUAP P6/04 DYSCO.

References

[1] V. Blondel, J. Guillaume, R. Lambiotte and L. Lefebvre, *Fast unfolding of communities in large networks.*, Journal of Statistical Mechanics: Theory and Experiment, 10:P10008, 2008

[2] A. Lanchichinetti, F. Radicchi, J. Ramasco, and S. Fortunato, *Finding statistically significant communities in networks*, PLOS One, 6(e18961), 2011.

[3] R. Mall, R. Langone and J. A. K. Suykens, *Multilevel Hierarchical Kernel Spectral Clustering for Real-life Large Scale Complex Networks*, PLOS One, e99966, 9(6), pp. 1-18, 2014.