Visualizing Large Networks with Aggregated Communities

Kousuke Tachida

1. Introduction

Networks, also known as graphs, are used to represent complex systems that consist of entities and relationships between these entities. These entities are represented as nodes/vertices and the relationships between them are represented as links/edges. Networks can come from many different domains like social networks, biological networks, infrastructure networks, and more. The target audience for my visualization is people who are interested in networks, since my visualization should work for most networks. Visualizing a network can lead to insights about the network and the relationships between entities. Some of these insights include global and local structure, the presence or absence of communities, and where connections are dense and sparse. The goal for my visualization was to be able to show these insights for large networks in an efficient way. However, large networks (my arbitrary definition of large for this project is 1,000 - 1,000,000 edges) are difficult to visualize because there are many elements and not enough ink or space on the screen to show them all. I decided to focus on visualizing the largest connected component of an unweighted, undirected network, which makes it so that I do not have to worry about visualizing disjoint components, directions on edges, or edge weights. Instead of showing all of individual nodes and edges, my approach was to simplify the structure of the network with aggregated communities. Using the Louvain and Leiden algorithms to continually group nodes and form a hierarchy of communities, the network is simplified into an aggregated community hierarchy structure, which is then visualized in an interactive browser page. Network and community statistics are also displayed in my visualization, and communities and subcommunities can be focused on through hovering or clicking. The research question for this project was: how efficient and effective is this visualization in showing insights about the network? My hypothesis was that the visualization method would work well for every kind of network, at least up to million-node and million-edge networks.

2. Related Work

There are numerous techniques for visualizing networks. Many of them do not scale well to large amounts of nodes and edges. Hu et al. [1] goes over layout algorithms, visual abstraction/compression, and interactive exploration techniques for large graphs. Jonker et al. [2] also used the Louvain algorithm for visualization, but instead of simplifying the structure with the aggregated community hierarchy, they used the community hierarchy along with centrality measures to inform node placement and used various zoom levels to deal with the large number of elements.

3. Visualization Methodology

3.1. Finding Aggregated Communities with Louvain/Leiden Community Detection

Community detection is a method to find groups of nodes which are densely connected within their groups and more sparsely connected between groups. The Louvain [3] and Leiden [4] algorithms are very similar community detection algorithms. The Louvain algorithm is a widely used community detection algorithm that is fast and forms good communities. The Leiden algorithm is an improvement to the Louvain algorithm, which adds and extra refinement step to the Louvain algorithm. It leads to guaranteed connected communities, better partitions, and runs faster than the Louvain algorithm. Both the Louvain and Leiden algorithms try to optimize a quality function, commonly maximizing modularity, which is used in this project. Modularity measures the density of edges inside of communities compared to the expected number of edges. A parameter of modularity calculation is resolution, which results in more communities when larger and fewer communities when smaller. In both the Louvain and Leiden algorithms, each node starts in their own community, then nodes are repeatedly moved to the neighbors community with the biggest modularity gain. The movement step is repeated until modularity no longer increases. After this step, the Leiden algorithm does a refinement step, which can separate a community into separate subcommunities. Then, for both algorithms, a new aggregated network is constructed, where the

new nodes are the communities found from the previous steps. One iteration is now complete, and iterations are repeated until maximum modularity. Because of the iterative aggregation of communities, both the Louvain and Leiden algorithms form hierarchies of communities. To force this hierarchical structure, I reduce the resolution parameter after each community detection iteration. I also terminate the algorithm when the number of top-level communities goes below a threshold, which I usually set to 30. I used an open-source implementation of the Louvain and Leiden algorithms from cwts.networkanalysis Java package [5]. An implementation challenge was running out of heap memory when trying to store information about the found communities for every iteration of the Louvain/Leiden algorithm. To solve this problem, I stored only information about the last couple of community detection iterations. The stored information includes subcommunity assignment, node-to-cluster and cluster-to-node maps, which are necessary to visualize the network.

3.2. Exported Data Format

After the Louvain/Leiden algorithm stops running, I needed to export the results to read into the web interface. The JSON file format works well because its structure can contain hierarchical objects and is supported by D3.js. The JSON file has 6 base fields: numNodes (int), numEdges (int), bounds (object), nodes (array of objects), links (array of objects), degreeDist (array of degree-frequency object pairs). Each node object represents a community, and contains id (int), numNodes, numEdges, degreeDist, and subclusters (object which contains nodes and links) fields. Each edge object contains source (int, id of source node), target (int, id of target node). The bounds object contains the largest and smallest community sizes, and largest edge weight.

3.3. Laying Out the Aggregated Communities

Using D3.js library with d3-force, the JSON file is read and top-level communities are created and put in a force-directed layout. Each community is represented as a circle, the radius of which increases linearly with the number of nodes in the community. The edges between communities are represented as a line, the width of which increases linearly with the weight, which is the total number of edges in the original network that connect across the two communities. Subcommunities are placed as circles inside of their parent communities, with their radius on the same radius scale and edges between them represented as lines on the same width scale. For the top-level communities, a many-body repulsion force, a centering force, a link force, and a collision force are applied. The link force is based on the edge weight, where a higher weight leads to more attraction between nodes, which makes it so that communities that have many connections between them tend to be closer together. For the second-level communities, two centering forces, a many-body attraction force, a link force, and a collision force is applied. This makes it so that subcommunities are centered inside their parent communities, and are as close together as possible without colliding.

3.4. Visualizing Network and Community Statistics

The statistics for the whole network and communities include the number of nodes, number of edges, and a scatter plot of the degree distribution. The scatter plot is shown on a log-log scale which is commonly used to show degree distributions. The plot was created with the Chart.js library.

4. Design

4.1. Design Evolution

The initial storyboard, shown in Figure 1, of my visualization contained a section for laying out the network and its clusters and a section for viewing statistical information about the network and selected clusters. I had a good idea about what the network would look like in its aggregated community structure, so that did not change much from the storyboard.

4.2. Feedback

The initial feedback that I got from my project idea was that it was too broad. I was originally unsure of the types of networks I would try to visualize, but decided to focus on the largest connected component of undirected, unweighted networks. Another piece of feedback I received was to try a temporal dataset, which I did not do, but would be interesting to see. I also received feedback on not using color, which I was not able to get working given the time constraints. My visualization would definitely improve if I used to color to show something meaningful.

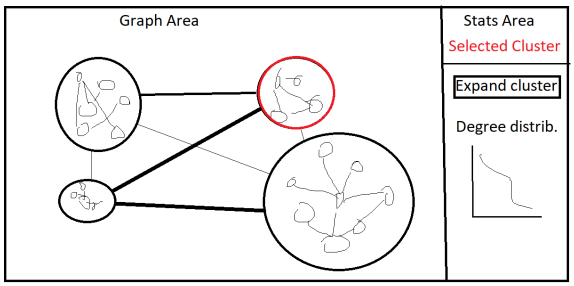

Figure 1: The initial project storyboard, which looks similar to the final visualization design

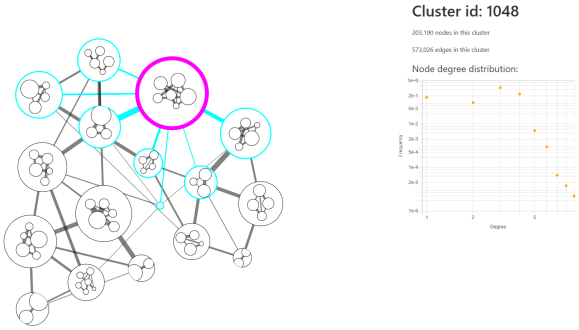

Figure 2: The final visualization design, where the selected community is highlighted in magenta

4.3. Final Design

My final visualization design is shown in Figure 2. The selected community is highlighted in magenta with connecting edges and communities highlighted in cyan. The degree distribution is shown for the nodes in the selected community.

5. Results

5.1. Data Collection

I tested my visualization on real-world networks of various types and sizes. The datasets used in this section were retrieved from Stanford's Large Network Dataset Collection [6]. The networks in this collection come from various sources and can be downloaded as edge list text files. All of these networks were unweighted and undirected.

5.1.1. Amazon Product Network [7]

For this network, nodes are products on Amazon and edges between two products means they are frequently purchased together. This network contains around 300000 nodes and 900000 edges. The initial result only had 9 top-level clusters (Figure 3), while stopping at a previous iteration gives 45 top-level clusters (Figure 4).

5.1.2. Road Networks of California, Pennsylvania, and Texas [8]

For these three networks, nodes are intersections and endpoints, and edges are roads between the nodes. The visualization, especially because of the force-directed layout, brings out the global structure of the network well, resulting in the road network visualizations being similar in shape to their respective states, as shown in Figure 5.

5.1.3. Human Protein-Protein Network [9]

For this network, nodes are proteins and edges are interactions between the proteins. This network is quite dense with around 19,000 nodes but 750,000 edges. As shown in Figure 6, many of the subcommunities do not fit inside of their parent communities and some of the edges within the top-level communities are difficult to interpret because there are so many.

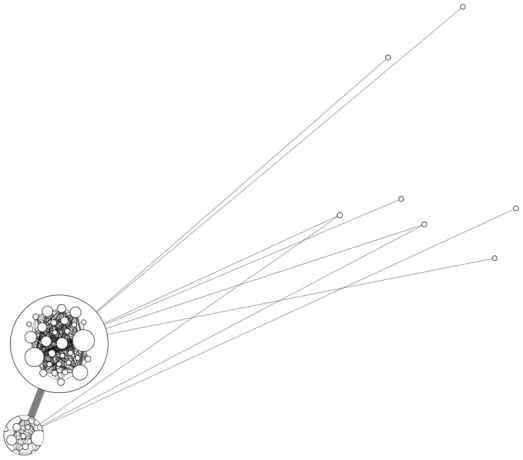

Figure 3: The Amazon Product network with 9 top-level communities

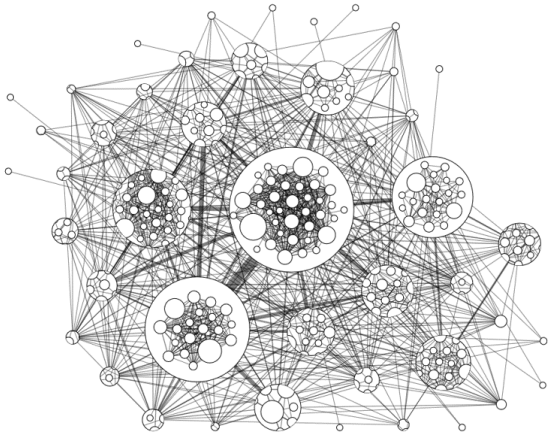

Figure 4: The Amazon Product network with 45 top-level communitie

6. Limitations

My visualization method has several limitations. One limitation is the amount of edge overlaps, especially for denser networks. In the worst case, so many edges overlap that they show as black. >2 nodes can also be placed collinearly, leading to confusing overlapping edges. A possible solution to the edge overlap issue is to implement edge bundling, curved edges, or using a different layout technique. Another limitation is that some subcommunities can be hidden when there are too many of them inside the same parent community, or if the parent community is too small. A possible solution could be expanding the parent community on hover or click. There may also be some bugs with determining subcommunities and their edges, since there can be poorly-connected subcommunities (for example a subcommunity with 50 nodes and only 2 edges) and subcommunities that are disconnected from other subcommunities.

7. Conclusion and Future Work

For future work, I would like explore more of the visualization results by running on random generated networks and trying different parameters. Trying the visualization on various randomly generated networks would help understand and evaluate the visualization, since the structure of randomly generated networks are known. Trying different parameters on the same network would help to understand how they effect the visualization. It would also be nice to compare my visualization results to other visualization results, in part to verify that my visualizations are reasonable, and to see if my visualization seems to show insights better. I would like to add more information about the edges between communities, since the only information available are the widths. I would also like to show labels of the top 5 highest degree nodes in every community, for datasets that have node labels. In conclusion, I think my visualization is able to efficiently show insights for very sparse million-edge

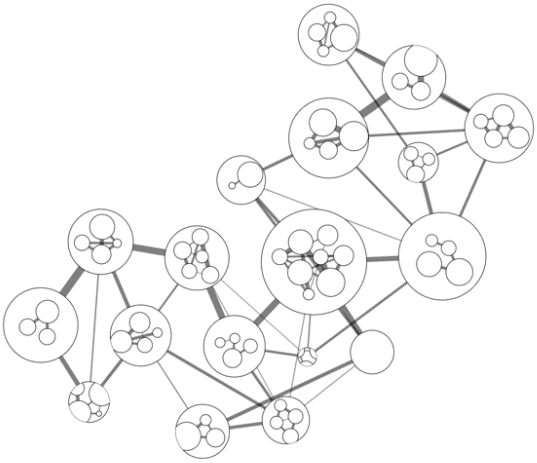

Figure 5: The road network of California, whose structure loosely resembles the shape of California

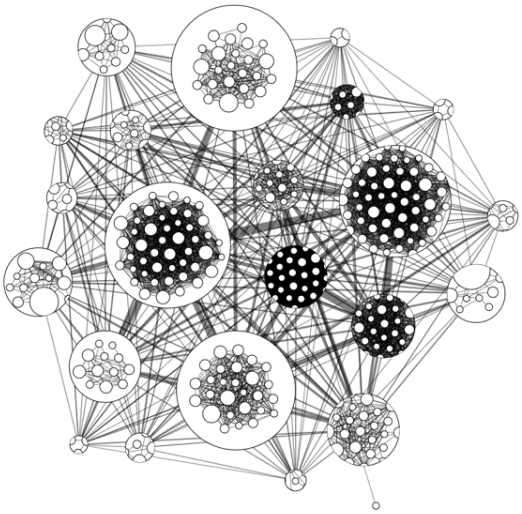

Figure 6: The proten-protein network, with very dense edges and many subcommunities

networks, but should not be blindly used for denser networks. Pruning edges based on weight or other factors before my method may lead to better visualization results. I think my visualization method has a lot of room for improvement and can eventually work well for denser networks.

References

[1] *Y. Hu, L. Shi, Visualizing large graphs, Wiley Interdisciplinary Reviews: Computational Statistics 7 (2) (2015) 115–136.*

[2] *D. Jonker, S. Langevin, D. Giesbrecht, M. Crouch, N. Kronenfeld, Graph mapping: Multi-scale community visualization of massive graph data, Information Visualization 16 (3) (2017) 190–204.*

[3] *V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of statistical mechanics: theory and experiment 2008 (10) (2008) P10008.*

[4] *V. A. Traag, L. Waltman, N. J. van Eck, From louvain to leiden: guaranteeing well-connected communities, Scientific reports 9 (1) (2019) 1–12.*

[5] *V. Traag, Cwtsleiden/networkanalysis: 1.0.0 (Oct. 2018). doi:10.5281/zenodo.1466831.*
URL https://doi.org/10.5281/zenodo.1466831

[6] *J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, http://snap.stanford.edu/data (Jun. 2014).*

[7] *J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowledge and Information Systems 42 (1) (2015) 181–213.*

[8] *J. Leskovec, K. J. Lang, A. Dasgupta, M. W. Mahoney, Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, Internet Mathematics 6 (1) (2009) 29–123.*

[9] *M. Zitnik, M. Agrawal, J. Leskovec, Modeling polypharmacy side effects with graph convolutional networks, Bioinformatics 34 (13) (2018) i457–i466.*