Visual Mining of Communities in Complex Networks: Bringing Humans into the Loop

Jack Murtagh
Tufts University
jack.murtagh@tufts.edu

Florentina Ferati
Texas Lutheran University
fferati@tlu.edu

James Abello
Rutgers University
abello@dimacs.rutgers.edu

Tina-Eliassi Rad
Rutgers University
eliassi@cs.rutgers.edu

Monica Babes Vroman
Rutgers University
babes@cs.rutgers.edu

Abstract

Community discovery is an important area of research in the study of real-world networks (graphs). Many visualization tools use clustering to display graphs and allow users to interactively explore the structure of networks. Many metrics exist to determine the quality of a particular community division of a graph. We used a popular measure called modularity, which essentially compares the number of edges within each community to the number of edges leaving each community. Finding the optimal clustering of a given graph with respect to any of these measures is an NP-hard problem, so researchers develop algorithms to approximate good community divisions. Our goal is to incorporate user feedback by allowing him/her to change the community assignment of nodes and create brand new clusters. This feature adds a new layer of information to the clustering process and may lead to better community structure. We also implement a "Suggest Changes" function to tell the user which changes will improve the modularity of the clustering (for small networks). We demonstrate a useful application of these features by modifying a variety of datasets until no single change will increase the modularity measure. Finding these local maxima leads to new interesting research avenues in community discovery.

Keywords: Community discovery; Modularity; Visual analytics; User Interaction.

1. Introduction

Many real-world phenomena can be represented as networks or graphs where sets of edges connect vertices. Examples include social networks, the World Wide Web, citation networks, and biological networks. A popular task on such networks is the discovery of communities. A community (a.k.a. a cluster or a group) is a subset of nodes that are deemed to be similar. The most popular notion of a community in networks is one where nodes in a community have higher intra link density either compared to their inter link density or compared to their intra link density in a random graph with the same degree distribution. Searching for clusters in a network often reveals relationships that may not have been apparent from the raw data. A community in a social network, for example might represent a close group of friends because they all know each other and have fewer common links to the rest of their social network.

Researchers have a variety of measures to deem a particular clustering of a graph successful or not, all of which in some way must consider the distribution of edges in the current community structure. The difficulty in this line of research lies in the fact that graphs are complex structures and finding the optimal clustering with respect to the most useful metrics is computationally hard. To approach this problem, researchers have developed algorithms that approximate good community splits rather than check all possibilities through brute force.

Another field of research in network analysis is the development of visual analysis tools to display graphs on a screen so users can explore these structures more easily. Community discovery and visual analytics are closely linked as clustering of a graph makes it far more intelligible to the eye and aides their study by exposing closely related agents in the network. For our research we used a program that combines clustering with visualization by approximating the best community split and then displaying the result to the screen. Since the output of this algorithm will not give us the optimal community structure of the graph, our task was to improve the clustering by bringing the human into the loop.

Real-world graphs will likely come with semantic information attached. Vertices and the edges between them represent actual relationships in the physical world (of which the computer is completely ignorant). Our goal was to allow users to change the community of any node and update the modularity after each move. This way, users can incorporate personal knowledge about their networks to add another layer of information to the clustering process. In addition to adding the change-of-community feature, we also give the user the ability to create an entirely new

community and track the details of the changes he has made. We implemented a "suggest changes" feature that indicates which changes are most likely to improve the modularity of the communities. This feature is currently useful but inefficient for larger networks and will be a main focus of our future research.

The rest of this paper is structured as follows. Next, we survey the related work and motivation. We present our approach and results in Section 3, followed by future work. We conclude the paper in Section 5.

2. Related Work and Motivation

In this section we review related work and background. We then emphasize the motivation of this paper.

2.1 Visual Analytics

Various graph visualization and analysis tools have been developed throughout the course of the growth of network science. A few examples include Cytoscape [9] ASKGraphView [2], and OntoVis [3]. All these tools allow users to display complex topologies and expose the hidden relationships among the elements in a network. Thus, they enable the investigation and understanding of networks in which nodes could represent different concepts and edges represent various relations between those concepts. For example, Figure 1 presents a graph of characters in the novel "Les Miserables" [4]. An edge connects two characters if they co-appear in any chapter of the story. In this graph, it is hard to discover any meaningful relation between the nodes. On the other hand, Figure 2 displays the same graph clustered into five communities, which reveals the underlying structure and relationships between the nodes. There are several challenges in the visualization of networks. First of all, networks can have millions of nodes; and displaying massive networks efficiently is computationally beyond the bounds of possibility. In their work, Zeqian Shen, Kwan-liu Ma, and Tina Eliassi–Rad [3] address these issues stressing that the mismatch between the density of information and screen resolution available is also troublesome. James Abello, Frank van Ham, and Neeraj Krishnan [4], allow the users to cluster and interactively navigate large graphs, ranging in size with up to 16 million edges.

Although many graph visualization tools exist, none of them incorporate users' knowledge of semantic information in the clustering process. Therefore, the design of new human-computer interfaces for understanding, navigating, and revising community structures is needed. The goal of these new interfaces is to obtain and use feedback from users to improve the community discovery process.

2.2 Community Structure

The definition of a "community" in networks is not formalized and currently there are different notions of what a community is. We can create a mathematical expression describing an ideal clustering of a graph but it would be too complicated to compute efficiently so we use surrogate measures instead. All of these measures have their own strengths, weaknesses, and peculiarities and there is not necessarily a "best" shortcut for indicating good community structure. For our research, we used a popular measure called modularity [1], which compares the number of edges inside each community to the expected number of edges if they were distributed randomly. According to modularity, the more the internal connectivity of a subset of nodes exceeds random expectation, the better fit it is for a community. Here is the technical definition:

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

where vw is a pair of nodes in the graph, m is the total number of edges in the graph, and k_v is the degree of vertex v. A_{vw} equals to 1 if v and w are connected and is 0 otherwise. Similarly, $\delta(c_v, c_w)$ equals 1 if v and w are in the same community and is 0 otherwise. The term $k_v k_w / 2m$ represents the probability that two nodes (with the given degrees) will share a link in a graph with m edges. Modularity can have values from -1 to 1 where a score of 0 means the number of edges within each cluster exactly matches what we would expect from a random subset of nodes with the same degrees. The closer it gets to 1, the stronger the community structure of the graph.

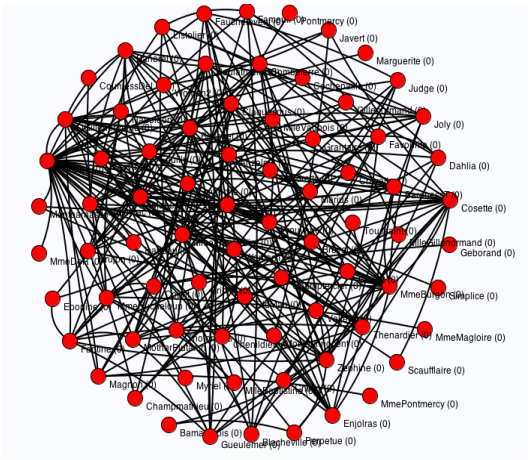

Figure 1. Les Miserables Characters Co-occurrence Network

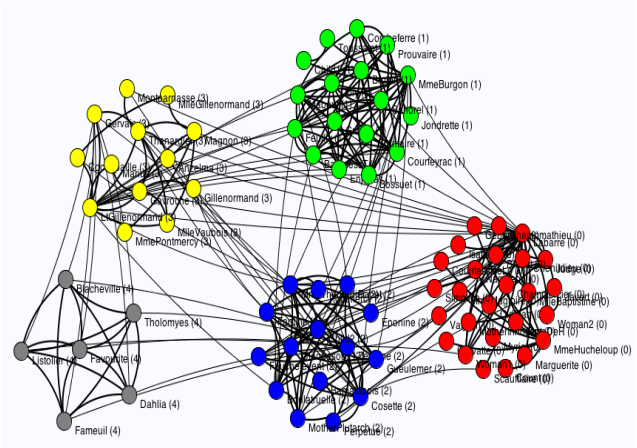

Figure 2. Clustered Les Miserables Characters Co-occurrence Network

2.3 Community Analysis Tool (CAT)

We based our work on an existing program called Community Analysis Tool (CAT), which combines community discovery and graph visualization. It was created in the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. CAT is implemented in the Java programming language. It clusters graphs using Clauset, Newman, and Moore's *Fast Modularity* algorithm [1]. As mentioned before, finding the true maximum modularity for a given graph is computationally hard, so Clauset *et al.* developed a greedy algorithm to approximate good community divisions for networks. The algorithm begins with each vertex as the sole member of a community. Then two nodes are merged into one community if this blending produces the largest increase in modularity score. This process continues until there is one cluster that contains every node in the graph. At some point in this hierarchy, a maximum score is reached and this is the stage that gets displayed to the user.

2.4 Our Goal

Finding the best community structure of a graph with respect to the most useful metrics is an NP-hard problem. To overcome this, we would like to bring users into the loop of the clustering process. The added step in the clustering process involves humans, who through visual analytics are able to manipulate various graphs using their knowledge and semantic information. The user will be suggested to make changes that bring on a higher modularity score, which consequently means that a state of enhanced community division has been reached.

To achieve this we have implemented several functions and added them to CAT. These features provide a more robust user interaction environment, which allows the user to make modifications to the displayed graphs. Hence, we can obtain feedback from the users and improve the clustering process. We have tested these functions by modifying and analyzing different networks of various sizes. These networks include a characters' co-appearance network of Victor Hugo's novel "Les Miserables" [4] consisting of 77 nodes, a co-authorship network of scientists working on network theory and experiment, comprising 1589 nodes [5], and a network representing books about US politics sold on Amazon.com, containing 105 nodes [6].

3. Approach and Results

In this section we describe in detail the features that were added to CAT and the results we obtained by using these functions.

3.1 Approach

3.1.1 Change the community of a node

The graph in CAT will initially be displayed with the number of communities where the modularity score is the highest. Each community is assigned a different color, which allows the user to distinguish visually between them. The user will be able to select certain nodes and place them into a different community. The user can accomplish this by first double-clicking on the node he/she wishes to move and then double-clicking on any node in the community he/she would like to place it. The new position of the moved node will be the average coordinates of its neighbors in that cluster. Upon this action, the color of the moved node will update to match the color of the new community. In order to keep some sort of recent history of changes, the rim (i.e. border) and the label of the moved node will remain the same color as its original community. This way the user can visualize what changes he/she has made to the graph. After the node is moved to the desired community, we recalculate modularity and display it on the screen so the user can see the effects of the changes he/she has made. Additionally, users will be warned if their action will result in a decrease of modularity by 10% or more.

3.1.2 Create a new community

If the user thinks that a node should not belong to any of the existent communities, but should rather be on its own, he/she will be able to add a brand new community to the network. The user can perform this by right-clicking (or holding control while clicking) on the particular node he/she would like to move to its own community. The new community will be assigned a number and a color and the modularity will update once again.

3.1.3 Suggest changes

We think that it is important to aid the user in knowing which nodes, if moved to a different community, would have the best effect on the modularity score. To accomplish this visually, the nodes that would fit better elsewhere will blink the color of their desired community. The current implementation of this feature observes the effect on modularity for every possible change in the graph. This method is satisfactory for the smaller datasets that we

tested, but is far too slow for graphs of even moderate sizes. To overcome this problem, we will develop a surrogate measure to indicate those changes that are most likely to improve the modularity of the graph (see the Future Work Section).

3.1.4 Detailed history of changes

The networks that are analyzed may contain thousands or more nodes and if many changes are made, it is difficult to remember what modifications have been made to the network. Therefore, a more detailed history of changes is provided to the user. This history will consist of data pertinent to the moved nodes, such as the node label or node ID, the community number, the new community number, the old modularity, the modularity change, and the new modularity.

3.2 Results

We have tested the functionality and efficiency of the added features using several datasets of different sizes. One of the networks we used is a network of co-appearance of characters in the novel "Les Miserables" with 77 vertices and 254 edges, mentioned earlier. We have displayed this graph in CAT and modified it with changes that have consequently led to better modularity scores. After numerous changes were made to this graph, we have been able to reach a state in which any additional change would not increase the modularity. We achieved the same state with a dataset of books about US politics sold by Amazon.com, comprised of 105 nodes and 441 edges[6]. The nodes in this graph correspond to books that can be liberal, neutral or conservative, and the edges signify frequent co-purchasing of books by the same buyers, as indicated by Amazon's feature "customers who bought this book also bought these other books". Finding these local maxima sparked several interesting questions and observations. While modifying the Les Miserables graph, there were only 3 suggested changes at first, but these gave rise to more suggestions after they were moved. Overall the graph required 7 changes to reach a maximum while the larger political science books network needed only two changes. It would be useful to investigate why some graphs are closer to a local maximum after clustering than others and if we could quantify an upper bound of this distance. The software slowed down too much when we loaded a graph with over 1500 nodes and over 2700 edges. The largest dataset we tested that could smoothly run all of the features had 286 nodes and 554 edges, but this is likely far from the biggest graph the code can handle because it performed just as quickly as our smallest network.

4. Future Work

4.1 Constraints Before Clustering

In the coming weeks, we will add a feature to CAT to incorporate user constraints before clustering begins. This will allow users to specify that certain nodes must be placed in the same community or others must be in separate communities. This improvement will add yet a third layer of information to the clustering process and again hopefully increase the modularity of the ultimate community split of the graph.

To insist that two nodes be placed together, we can treat them as a single vertex in the code. To ensure that specific nodes do not end up in the same community, it makes sense to wait for clustering to finish and then check if the nodes in question landed together. If they did, we will make the community change that decreases modularity the least.

We hope that this feature will not only add convenience to one's analysis, but could also have large effects on how the program actually clusters the network, giving the user a completely different base with which to begin his/her changes.

4.2 "Suggest Changes" Indicators

As mentioned before, with our "Suggest Changes" feature, we are able to reach a point with our datasets where no repositioning of a single node would improve the modularity of the clustering. Again, the current implementation is far too slow for graphs of even moderate sizes and we would like to develop a surrogate measure to indicate those changes that are most likely to improve the modularity of the graph. This shortcut must be easy to compute and have a high accuracy in predicting helpful adjustments.

One evolving idea labels a vertex as a "potential mover" if it has more links to a community other than its own. Then we must scale these nodes according to the sizes of the respective clusters. For example, a vertex may have 10 edges in its own community and 15 in another, but if its current home has 11 nodes and the other is of size 1000, suddenly those 15 links are less likely to indicate a better fit for that node. This technique has seemed promising in test runs, recommending almost all of the same changes as the exhaustive approach. However, this research is still in its nascent phases and requires much more careful thought.

4.3 Move Groups of Nodes

As the size of a network grows, single node changes begin to have less and less influence over the modularity of the entire graph. In a graph with millions of vertices, the degree of each node shrinks to a miniscule fraction of the total number of edges in the network. At this point, the contribution of a single node to the modularity of the graph becomes negligible unless it happens to be a major player in the network. The natural next step in our research is to allow users to change the community of a group of nodes all at once.

This addition increases the complexity of the "Suggest Changes" feature. Instead of searching for single potential movers, we would have to develop quick ways to identify groups of arbitrary sizes that would fit better elsewhere. One possible strategy might be to run the same analysis as

mentioned above for single nodes and then identify groups that would like to move to the same location. Once again, this avenue of thought is still in a preliminary stage and must be explored in greater depth.

4.4 Comparison of Clustering Metrics and Algorithms

As discussed above, there are several objective functions to determine the quality of a particular clustering of a graph and many community discovery algorithms that try to optimize these various functions. Comparisons of these algorithms in different contexts can provide important insights into their behaviors. The more we understand about the trends of particular clustering methods, the more equipped we are to select the appropriate algorithm for a given task [7].

Our work can serve as a useful tool in this endeavor. We could allow users to easily connect our implementations to different community discovery algorithms and track the effects of a change on several cluster-quality metrics. For example, a user could run two clustering algorithms on the same network and alter the communities of the nodes in one graph to resemble the other. As this happens, the objective functions continuously update, revealing trends in their relative behaviors. A user could also run the Fast Modularity algorithm and select the "Suggest Changes" feature to enhance the clustering with respect to modularity and observe if these improvements have similar effects on other objective measures.

There is no reason that this work must link with modularity or the algorithm that CAT implements. By providing the freedom to choose which algorithms to use and which functions to study, we would greatly widen the versatility of this tool.

4.5 User Feedback as "Hints"

Henderson *et al.* presented a clever strategy to improve the performance of two community discovery algorithms by combining them. They ran one clustering and then fed the results to another algorithm as "hints" [8]. To expand on their idea, bringing the user into the process could add another layer of useful information to these hints. We could run fast modularity (or a different algorithm as discussed in the previous section), pause while the user provides his/her suggestions, and then feed the product to a final clustering for an enhanced result. When this "hybrid" algorithm completes, the user would be free once again to make whatever changes he/she sees fit.

5. Conclusion

In this paper we present a mechanism for improving the community discovery process in real-world networks. Specifically by allowing users to modify a node's community assignment, we add another layer of information to the clustering process, which leads to improved community structure.

6. Acknowledgments

We would like to thank Nischal Devanur for his help. This work was performed under the Perceptual Science and Technology REU at Rutgers University.

7. References

[1] A. Clauset, M.E.J. Newman and C. Moore, "Finding community structure in very large networks."

Phys. Rev. E 70, 066111 (2004).

[2] Zeqian Shen, Kwan-Liu Ma, Tina Eliassi-Rad: Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction.

[3] J. Abello, F. van Ham, and N. Krishnan, "Ask-graphview: A large scale graph visualization system", IEEE TVCG journal, Vol. 12, No. 5, pp. 669–676, 2006

[4] D. E. Knuth, The Stanford GraphBase: A Platform for Combinatorial Computing, Addison-Wesley, Reading, MA (1993). http://www-personal.umich.edu/~mejn/netdata/lesmis.zip

[5] M. E. J. Newman, Phys. Rev. E 74, 036104 (2006). http://www-personal.umich.edu/~mejn/netdata/netscience.zip

[6] V. Krebs. http://www-personal.umich.edu/~mejn/netdata/polbooks.zip

[7] Jure Leskovec, Kevin Lang, Michael Mahoney: Empirical Comparison of Algorithms for Network Community Detection. WWW 2010:631-640.

[8] Keith Henderson, Tina Eliassi-Rad, Spiros Papadimitriou, Christos Faloutsos: HCDF: A Hybrid Community Discovery Framework. SDM 2010: 754-765.

[9] Michael E. Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, Trey Ideker: Cytoscape 2.8: new features for data integration and network visualization. Bioinformatics 2011: 431-432.