Stanford

World Wide Web, blogging platforms, instant messaging and Facebook can be characterized by the interplay between rich information content, the millions of individuals and organizations who create and use it, and the technology that supports it. This thesis will cover recent research on the structure and analysis of large social and transaction networks and on models and algorithms that abstract their basic properties. Unusual ways have been explored how to practically analyze large scale network data and how to reason about it through models for network structure. Topics include methods for network community detection, their connection with transactional graphs.

Article: 0709.2938v1

Community detection and analysis is an important methodology for understanding the organization of various real-world networks and has applications in problems as diverse as consensus formation in social communities. Currently used algorithms that identify the community structures in large-scale real-world networks require a priori information such as the number and sizes of communities or are computationally expensive. I intend to rely more on algorithms, which use the network structure as their guide instead of this priori information. Finding community structures in networks is another step towards understanding the complex systems they represent. Social networks are represented by people as nodes and their relationships by edges.

Girvan-Newman algorithm

The Girvan-Newman algorithm for the detection and analysis of community structure relies on the iterative elimination of edges that have the highest number of shortest paths between nodes passing through them. By removing edges from the graph one-by-one, the network breaks down into smaller pieces, so-called communities. The algorithm was introduced by Michelle Girvan and Mark Newman. The idea was to find which edges in a network occur most frequently between other pairs of nodes by finding edges betweenness centrality. The edges joining communities are then expected to have a high edge betweenness. The underlying community structure of the network will be much more fine-grained once the edges with the highest betweenness are eliminated which means that communities will be much easier to spot.

The Girvan-Newman algorithm can be divided into four main steps:

1. For every edge in a graph, calculate the edge betweenness centrality.
2. Remove the edge with the highest betweenness centrality.
3. Calculate the betweenness centrality for every remaining edge.
4. Repeat steps 2-4 until there are no more edges left.

NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. It supports a variety of features for complex networks.

* Data structures for graphs, digraphs, and multigraphs
* Many standard graph algorithms
* Network structure and analysis measures
* Generators for classic graphs, random graphs, and synthetic networks
* Nodes can be "anything" (e.g., text, images, XML records)
* Edges can hold arbitrary data (e.g., weights, time-series)
* Open source 3-clause BSD license
* Well tested with over 90% code coverage
* Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

Sixtep software:

The software has been released in 2007 by network theory researchers and CRM advisors. It is able to load a graph and visualize it. Several algorithms are implemented to clusterize or detect communities such as the Newman-Girvan algorithm or the Markov chain model. The UI representation of the graph is user friendly; the location of the nodes can be easily modified by clicking one by one or select a targeted area. Several built in function helps to make the graph more interpretable. The user can select unique modules like clusters or communities and display only the selected ones. The source code of the software cannot be accessed, but the export function made it possible to use the calculated clusters and communities. Sadly, it was not enough to provide the information about the edges between the clusters, however valuable data can be found while measuring that area.

Markov clustering algorithm:

The MCL algorithm is short for the Markov Cluster Algorithm, a fast and scalable unsupervised cluster algorithm for graphs (also known as networks) based on simulation of (stochastic) flow in graphs. The algorithm was invented/discovered by Stijn van Dongen at the Centre for Mathematics and Computer Science (also known as CWI) in the Netherlands.

Community detection, clique problem: wikipedia

In computer science, the clique problem is the computational problem of finding cliques (subsets of vertices, all adjacent to each other, also called complete subgraphs) in a graph. It has several different formulations depending on which cliques, and what information about the cliques, should be found. Common formulations of the clique problem include finding a maximum clique (a clique with the largest possible number of vertices), finding a maximum weight clique in a weighted graph, listing all maximal cliques (cliques that cannot be enlarged), and solving the decision problem of testing whether a graph contains a clique larger than a given size.

The clique problem arises in the following real-world setting. Consider a social network, where the graph's vertices represent people, and the graph's edges represent mutual acquaintance. Then a clique represents a subset of people who all know each other, and algorithms for finding cliques can be used to discover these groups of mutual friends. Along with its applications in social networks, the clique problem also has many applications in bioinformatics, and computational chemistry.

Most versions of the clique problem are hard. The clique decision problem is NP-complete (one of Karp's 21 NP-complete problems). The problem of finding the maximum clique is both fixed-parameter intractable and hard to approximate. And, listing all maximal cliques may require exponential time as there exist graphs with exponentially many maximal cliques. Therefore, much of the theory about the clique problem is devoted to identifying special types of graph that admit more efficient algorithms, or to establishing the computational difficulty of the general problem in various models of computation.

To find a maximum clique, one can systematically inspect all subsets, but this sort of brute-force search is too time-consuming to be practical for networks comprising more than a few dozen vertices. Although no polynomial time algorithm is known for this problem, more efficient algorithms than the brute-force search are known. For instance, the Bron–Kerbosch algorithm can be used to list all maximal cliques in worst-case optimal time, and it is also possible to list them in polynomial time per clique.

Experiences:

The networkx implementations of the algorithms have been used for the python script. The code of the algorithms can be found in the coloring.py file. The colormap and the coloring logic is self-implemented and it can be found in the Utils.py file.

The Newman-Girvan algorithm is fairly slow on medium sized graphs, but the result is more accurate and all the nodes are classified.

The well known Zachary community is divided into 2 main part due to a conflict of interest. The graph coloring represent the two new community.



The following figure is a result of a Newman-Girvan run on a graph made by facebook anonymized data. There are 100 nodes and several edges represented.

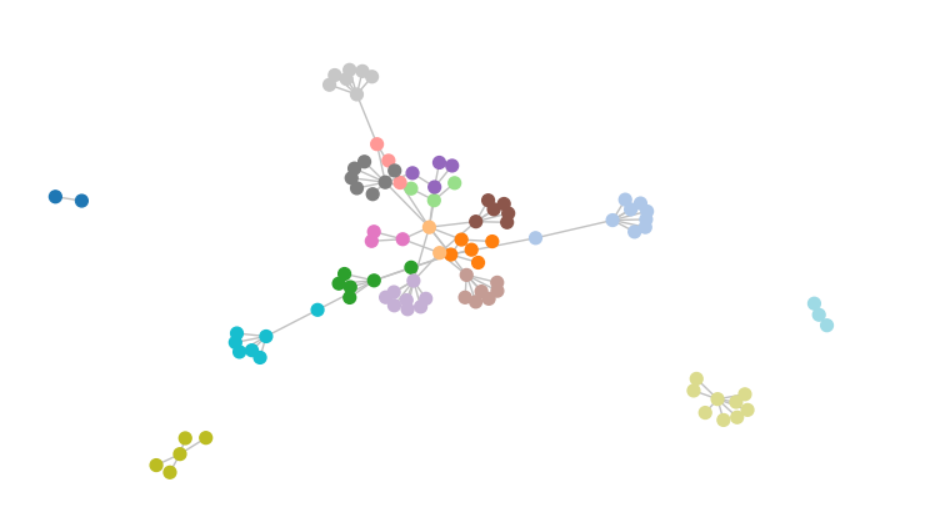


Finding cliques is an important corner of the community detection problem, but there is an implementation of it based on the Bron–Kerbosch solution. The result was the following.



It is important to determine the size of the cliques should be detected. For example a clique with size 3 has lower importance than 5 or above.

Transaction graphs:



GRAPH DATABASE!!!!

Efficiency, solutions, etc.

Further studies:

Make a ui to represent small graphs, but the software should provide opportunity to manually color graph nodes, modify the location of the nodes.

This solution is scalable and modularized which makes further implementation more easier. The code and be found in the github repository liked below:

<https://github.com/daniellanikov/Community-detector>