Abstract: A prototype console application has been implemented to detect social networks and clustering transactional type of the graphs. Several algorithms have been implemented to run against the given databases, such as Girvan-Newman, Markov chain, etc. The application is able to map a graph (also known as network) with any given delimiter, in any text format furthermore there is an option to connect to database if necessary. The results of the algorithms are plotted and displayed after the run. The coloring logic has been implemented in a fairly naive and greedy colormap. The application is scalable and modularized.

Introduction

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 the processing of 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 and their connection with transactional graphs. [1]

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 [2]. Social networks are represented by people as nodes and their relationships by edges therefore they contain more triangles than a random graph with similar edge density or degree properties. In contrast technological or transaction graphs contain fewer triangles and often display tree-like structures [15].

I gained knowledge about some of the prerequisites of social graph studies, for example algorithms of community detection have been used, along with the extracted data from an earlier implemented software, the Sixtep program, as well.

Based on the research written in article [15] and some earlier studies I intend to give an explanation about the implementation of algorithms for generalized coloring of transactional graphs. As the study goes, I follow the heuristics and possible solutions for the clustering problem.

There was an attempt to find the embedded pattern in the provided transaction-like graphs.

NetworkX [4]

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 functions help 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 exploring that area.

3 algorithms included:

* Markov chain clustering
* Maximized modularity
* Community detection

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. 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 graphs 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.

Technology:

Data sources:

Social:

* Iwiw
* Facebook
* Karate club graph

Transaction:

* OTP transaction graph

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.

Karate Club [5]

A social network of a karate club was studied by Wayne W. Zachary. The network became a popular example of community structure in networks after its use by Michelle Girvan and Mark Newman. It captures 34 members of a karate club, documenting links between pairs of members who interacted outside the club. During the study a conflict arose between the administrator and instructor, which led to the split of the club into two. Half of the members formed a new club around the instructor; members from the other part found a new instructor or gave up karate. Based on collected data Zachary correctly assigned all but one member of the club to the groups they actually joined after the split. The coloring of the graph represents the two new community.



Figure : The well-known karate club community is divided into 2 main part due to a conflict of interest.

Maximized modularity:

A képen égbolt látható

Automatikusan generált leírás

This algorithm finds communities in graph using Clauset-Newman-Moore greedy modularity maximization. This method currently does not consider edge weights. Greedy modularity maximization begins with each node in its own community and joins the pair of communities that most increases modularity until no such pair exists.

Source code:

<https://networkx.org/documentation/stable/_modules/networkx/algorithms/community/modularity_max.html>

Markov-chain: [4]



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

Girvan-Newman:



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.

Clique maximization:

For each node n, a maximal clique for n is a largest complete subgraph containing n. The largest maximal clique is sometimes called the maximum clique.

This function returns an iterator over cliques, each of which is a list of nodes. It is an iterative implementation, so should not suffer from recursion depth issues.

This function accepts a list of nodes and only the maximal cliques containing all of these nodes are returned. It can considerably speed up the running time if some specific cliques are desired.

A list output of the function find\_cliques(G) has been used to obtain all maximal cliques. However, in the worst-case scenario, the length of this list can be exponential in the number of nodes in the graph. This function avoids storing all cliques in memory by only keeping current candidate node lists in memory during its search.

This implementation is based on the algorithm published by Bron and Kerbosch (1973) [6], as adapted by Tomita, Tanaka and Takahashi (2006) [7] and discussed in Cazals and Karande (2008) [8].

This algorithm ignores self-loops and parallel edges, since cliques are not conventionally defined with such edges.



Figure 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.

Source Coloring:

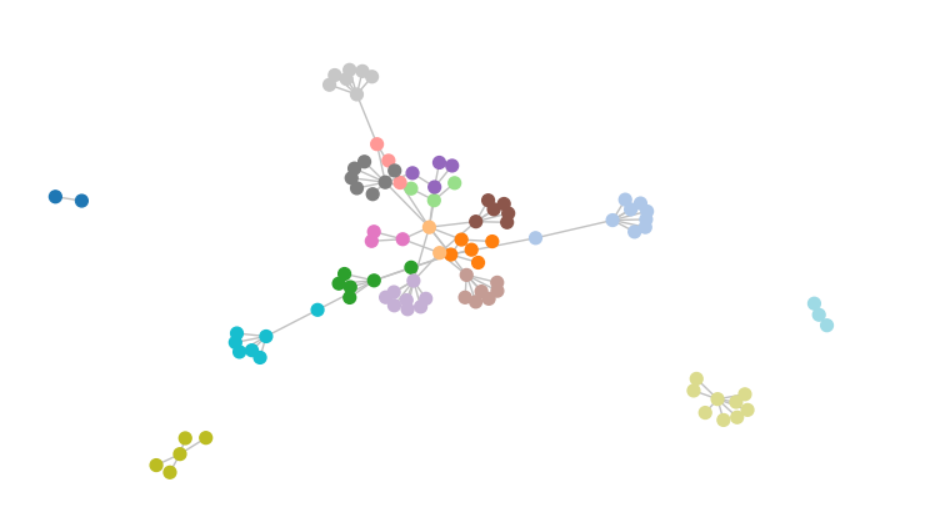
A képen narancs látható

Automatikusan generált leírás

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.



Transaction graphs:



Modularity

One of the most sensitive detection methods is optimization of the quality function known as modularity over the possible divisions of a network, but direct application of this method using, for instance, simulated annealing is computationally costly. A community structure in a network corresponds to a statistically surprising arrangement of edges, can be quantified using the measure known as modularity [9]. The modularity is, up to a multiplicative constant, the number of edges falling within groups minus the expected number in an equivalent network with edges placed at random. The modularity can be either positive or negative, with positive values indicating the possible presence of community structure [10].

Modularity is defined in [11] as

Where m is the number of edges, A is the adjacency matrix of G, is the degree of i, γ is the resolution parameter, and is 1 if i and j are in the same community, else 0.

According to [12] (and verified some algebra) this can be reduced to

Where the sum iterates over all communities c, m is the number of edges, is the number of intra-community links for community c, is the sum of degrees of the nodes in community c, and γ is the resolution parameter.

The resolution parameter sets an arbitrary tradeoff between intra-group edges and inter-group edges. More complex grouping patterns can be discovered by analyzing the same network with multiple values of gamma and then combining the results [13]. That said, it is very common to simply use gamma=1. More on the choice of gamma is in [14].

This NetworkX version has been used in the community detector repository. The parameters are the following: G represent the NetworkX graph. Communities are a list or iterable of set of nodes. These node sets must represent a partition of G’s nodes. Weight is an edge attribute that holds the numerical value used as a weight. It is an optional parameter, if the value is None or an edge does not have that attribute, then that edge has weight 1. Resolution is an optional parameter as well. If resolution is less than 1, modularity favors larger communities. Greater than 1 favors smaller communities. The function returns Q, the modularity of the partition. In case of the communities are not a partition of G, the function raises NotAPartition exception.

The second formula is the one actually used in calculation of the modularity. For directed graphs the second formula replaces with .

Additional results: Modularities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Modularity | | | |
|  | newman | Girvan\* | markov | Greedy modularity |
| Facebook (0.edges) | 0.3615 | (6) 0.2523 | -0.8080 | **0.4429** |
|  | (20) 0.3729 |  |  |
|  | (30) 0.4136 |  |  |
|  | (35) 0.4129 |  |  |
|  | (40) 0.4111 |  |  |
|  | (50) 0.4043 |  |  |



Figure : Facebook communities detected by greedy modularity algorithm

Size of nodes: 324

\* The numbers before the modularity values represent the number of the iteration the algorithm took.

Transaction graph of the OTP database:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Modularity | | | |
|  | newman | Girvan | markov | Greedy modularity |
| OTP (smaller) |  |  | 0.6116 |  |
| First quarter year |  |  | 0.6234 |  |
| Middle slice |  |  | 0.5745 |  |
| total |  |  | 0.3900 |  |
| Second quarter year |  |  | 0.3794 |  |
| Third quarter year |  |  | 0.3690 |  |

The smaller OTP graph has been represented by 96 nodes, the first quarter year by 675 nodes and the middle slice is about 18032 nodes. The smaller one is a subset of the first quarter year. The total amount of nodes in this dataset is 34992. The graph of the second quarter year contains 37008 and the third one has 39155. At this size there is not much sense to visualize the graphs in 2D. No information can be gained with naked eyes or manual clustering.



Figure . First quarter year of the OTP transactions clustered by Markov chain algorithm. 675 nodes have been colorized.

Wordgraph

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Modularity | | | |
|  | newman | Girvan | markov | Greedy modularity |
| wordgraph |  |  | 0.2241 |  |

The number of nodes in this graph is 11381.

Condensed graphs



Each community/cluster has been contracted to single node.

If there is a path between the modules, the algorithm puts an edge between the nodes.

The algorithm works well, if the source graph is strongly connected.



OTP transaction leveling by condensation

This graph is a slice of the OTP transactions from the first quarter year.



And the condensated one…



This is interesting because technical graphs usually represent tree-like structure (like said before), but this seems more like a social graph. Some of the layers in the otp graph look like communities. Originally the suppliers supposed to be in a competitive relation by restrict the edges between them.



But based on this, some cooperation can be supposed.

The modularity of the original graph is 0.7632.

Embeddedness of bipartite subgraphs in transaction networks

Certain bipartite graphs, for example pollinator networks or trade networks suggest the presence of different structures, like the notion of embeddedness. The vertices of each color class can be ordered and the smaller ranked vertex neighbourhood contains the neighbourhood of any higher ranked one. A binary matrix *A* is fully nested if its rows and columns can be reordered such that the ones are in echelon form. Let be the bipartite graph whose adjacency matrix is A. Then A being fully nested is equivalent to satisfying embeddedness.

Let X (the columns) and Y (the rows) be the bipartition of . The matrix A and the graph are each said to be k-nested with respect to X if X can be partitioned as such that all subgraphs spanned by are fully nested for . The quantity of interest for any is smallest k for which is k-nested.

H avoiding coloring of graph G

Given a “forbidden graph” H and an integer k, an H-avoiding k-coloring of a graph G is a k-coloring of the vertices of G such that no maximal H-free subgraph of G is monochromatic.

A monochromatic graph is a colored graph (either vertex-colored or edge-colored, depending on the context) in which each of the vertices or edges is assigned the same color.



Figure 5. Proper 2K2 avoiding coloring of a bipartite graph

*Colormap: {'1': 0.82, '2': 0.44999999999999996, '4': 0.72, '6': 0.94, '8': 0.15000000000000002, '10': 0.15000000000000002, '3': 0.82, '5': 0.82}*

A new kind of clustering of general (that is, not necessary bipartite) transaction graphs has been presented via a certain class of proper colorings. The clusters are the color classes, since no edges are desired inside a cluster. The structure of the edges is restricted between the pairs of classes. The above examples suggest that in some cases there should be a fully nested or, equivalently, embeddedness relation among any two color classes. This notion is generalized to an arbitrary host graph G and a forbidden bipartite subgraph H as follows.

Definition 1. Fix a bipartite graph H. A proper coloring of a graph G is an H-avoiding coloring if the union of any two color classes spans an induced H-free graph. Let be the minimum number of colors in an H-avoiding coloring of G.

Observation 2. For any graphs H and G, . If G is H-free, then

The computation of is NP-hard for some graphs and polynomially computable for others. The most interesting case is, when , gives back embeddedness as described above. For these generalized chromatic numbers some theoretical extremal results have been derived as well as results on complexity.

Matek!

GRAPH DATABASE!!!!

Patterns!

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>

Acknowledgements

Many thanks to all the scientists and colleagues, who contributed to this very old but rather fascinating topic of graph clustering in network theory and special thanks to my advisor, András Pluhár, who provided the help with patience and understanding through my learning path.

References:

[1] <https://snap.stanford.edu/>

[2] Usha Nandini Raghavan, Réka Albert and Soundar Kumara, 2007, Near linear time algorithm to detect community structures in large-scale networks

[3] <https://networkx.org/>

Modularity page

[4] <https://github.com/guyallard/markov_clustering>

[5] Zachary, W. W., 1977, An Information Flow Model for Conflict and Fission in Small Groups

[6] Bron, C. and Kerbosch, J, 1973, Algorithm 457: finding all cliques of an undirected graph

[7] Etsuji Tomita, Akira Tanaka, Haruhisa Takahashi, 2006, The worst-case time complexity for generating all maximal cliques and computational experiments

[8] F. Cazals, C. Karande, 2008, A note on the problem of reporting maximal cliques

[9] M. E. J. Newman and M. Girvan, Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004).

[10] M. E. J. Newman, 2006, Modularity and community structure in networks

[11] M. E. J. Newman “Networks: An Introduction”, page 224. Oxford University Press, 2011.

[12] Clauset, Aaron, Mark EJ Newman, and Cristopher Moore. “Finding community structure in very large networks.” Phys. Rev. E 70.6 (2004). <<https://arxiv.org/abs/cond-mat/0408187>>

[13] Reichardt and Bornholdt “Statistical Mechanics of Community Detection” Phys. Rev. E 74, 016110, 2006. <https://doi.org/10.1103/PhysRevE.74.016110>

[14] M. E. J. Newman, “Equivalence between modularity optimization and maximum likelihood methods for community detection” Phys. Rev. E 94, 052315, 2016. https://doi.org/10.1103/PhysRevE.94.052315

[15] A London, Ryan R. Martin, A. Pluhár, 2021, Graph clustering via generalized colorings