National University of Mongolia
School of Engineering and Applied Sciences
Faculty of Information, Computer Science

Erkhembayar Gankhuyag
Supervisor: Ph.D P.Dalaijargal

Betweenness Centrality for community detection in social networks

Software engineering (D061302)
Bachelor thesis work

Ulaanbaatar

2022

CONTENTS

List of figures

List of Tables

INTRODUCTION

These days, the use of networks and their analysis is rapidly increasing in almost all fields such as biology where we can see protein structure as a kind of network and extract useful information after researching, analyzing, etc. One of the main problems in the network analysis field is community detection. In community detection, there are many types of algorithms that use different approaches to detect the desired communities in the network. In this work, we focused on the betweenness method of community detection. The biggest disadvantage of the betweenness method is its time complexity and we suggested two approaches for decreasing the execution time.

1. THEORY

1.1 Community in social networks

Many studies focus on different factors of the networks such as small-world property, power-law degree distribution, and network transitivity[2]. Then this thesis work is focused on the concept of community in the network as detecting communities is an important problem in many fields. This problem is relevant for social tasks (objective analysis of relationships on the web), biological inquiries (functional studies in metabolic and protein networks), or technological problems (optimization of large infrastructures)[7]. For instance, in a friend list, a network community will be a group of people that are friends. And if the communities are connected, it means there is at least one person who is friends with people of both groups. A community is in short, a group of nodes or objects that interacts with each other.

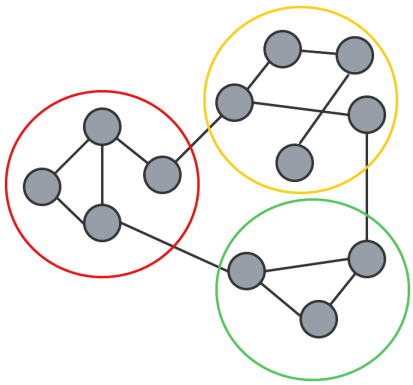


Figure 1.1 Communities in network

Communities allow detecting objects, and nodes that interact with each other. For example, a community can express people who attended the same school, papers that cite the same article, functional modules of interacting proteins, etc. depending on the networks[11]. In the following figure we can see the most classic example in the network analysis field which is Zachary karate club.

Detecting communities from a network means the distinguishing group of nodes and these nodes

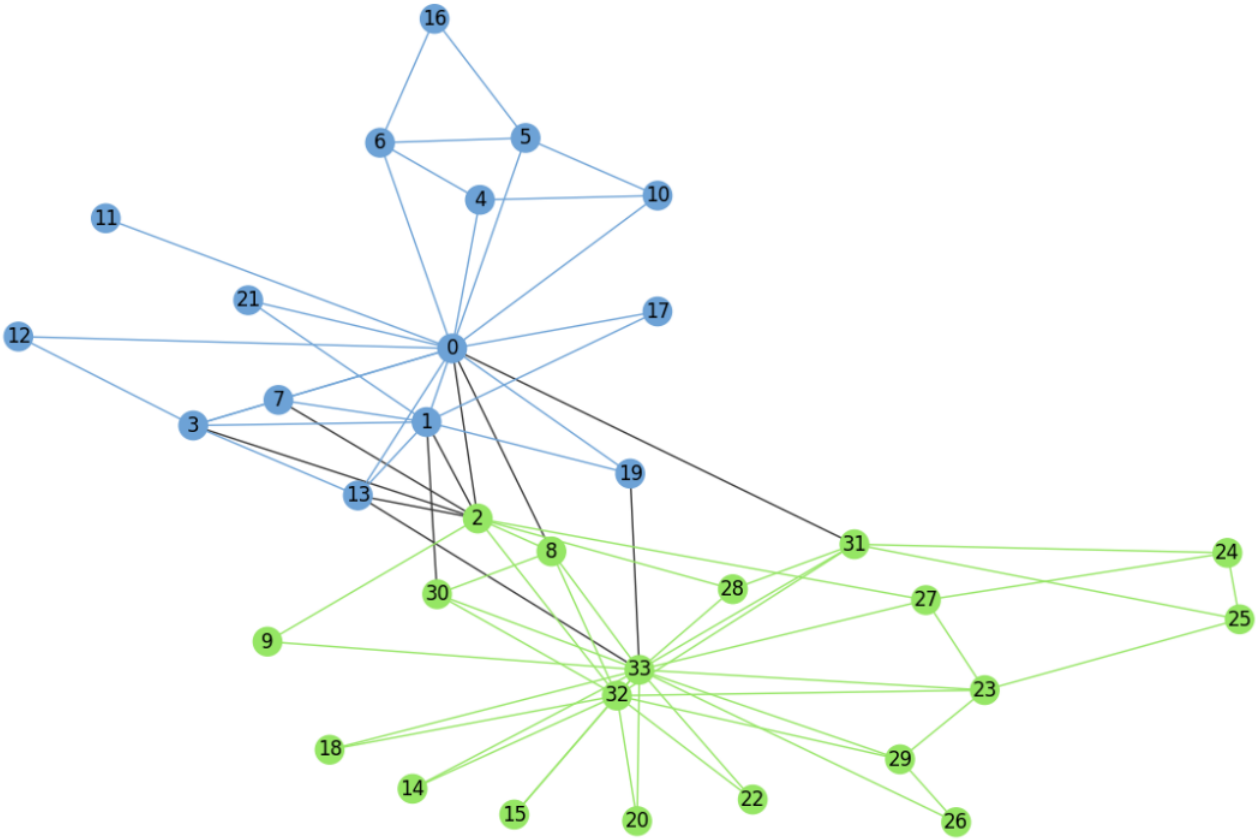
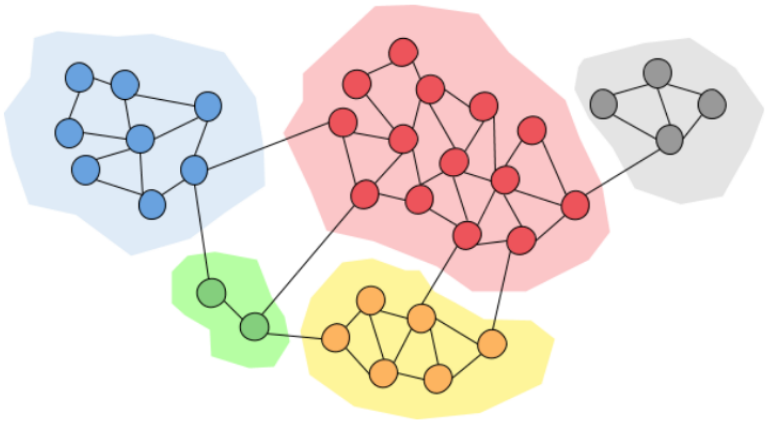


Figure 1.2 It represents the karate club social network which consists of 34 nodes which are its members and 78 edges that represents the interaction between the members.

can exist in multiple communities. We can use two sources of data to perform the task. The first is the characteristic of the objects and attributes. Known properties of proteins, people's contact information, social network profiles, etc. The second source focuses purely on network structure and the set of connections between the objects. In terms of characteristics, the first method divides nodes whose attributes are similar. On the other hand community detection algorithms mainly focus on the network structure. And this work is following the second method which is to detect communities based on the concept of betweenness.



[4]

Figure 1.3 Communities in network

1.2 Betweenness centrality

One of the most common problems in network analysis is to determine whether the edge or node is important and if so, by how much. Many measurements indicate the importance such as closeness, stress, and betweenness. Betweenness /BC/ is used in many fields such as social interaction networks. Vertex betweenness has been studied in the past as a measure of the centrality and influence of nodes in networks. First proposed by Freeman [1], the betweenness centrality of a vertex i is defined as the number of shortest paths between pairs of other vertices that run through i. It is a measure of the influence of a node in the network, especially in cases where information flow over a network primarily follows the shortest available path. Then the betweenness centrality of an edge is defined as the number of shortest paths that pass the desired edge. And these edges with high BC /betweenness/ are the most important because high betweenness means these edges are the bridges that connect these communities in the graph. Node betweenness is defined as[3]:

$$NB(v) = \sum_{v_i \in V} \sum_{v_j \in V/v_i} \frac{\sigma_{v_i v_{j(v)}}}{\sigma_{v_i v_j}}$$

Here $\sigma_{v_i v_{j(v)}}$ is the number of paths that passes through the node v and $\sigma_{v_i v_j}$ is the total number of possible paths.

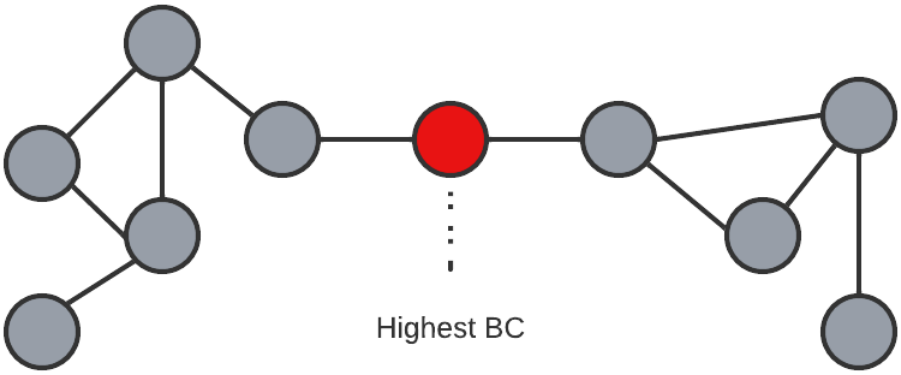


Figure 1.4 Node betweenness

Edge betweenness is defined as [3]:

$$EB(e) = \sum_{v_i \in V} \sum_{v_j \in V/v_i} \frac{\sigma_{v_i v_{j(e)}}}{\sigma_{v_i v_j}}$$

Here, $\sigma_{v_i v_{j(e)}}$ is number of paths pass through the edge v and $\sigma_{v_i v_j}$ is total number of possible paths. From this, the higher the BC, there is a high probability that the edge or node is connecting

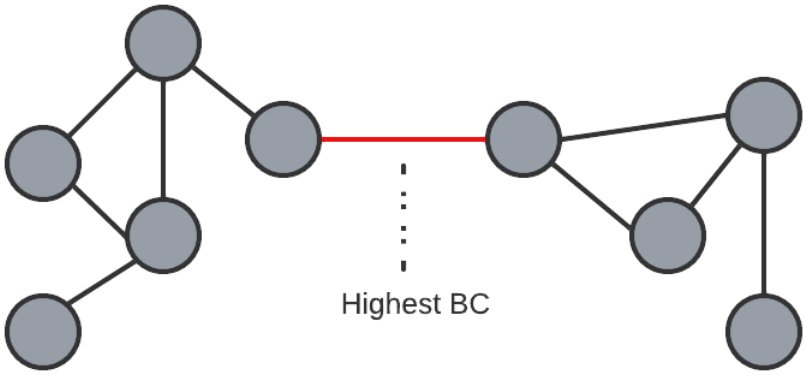
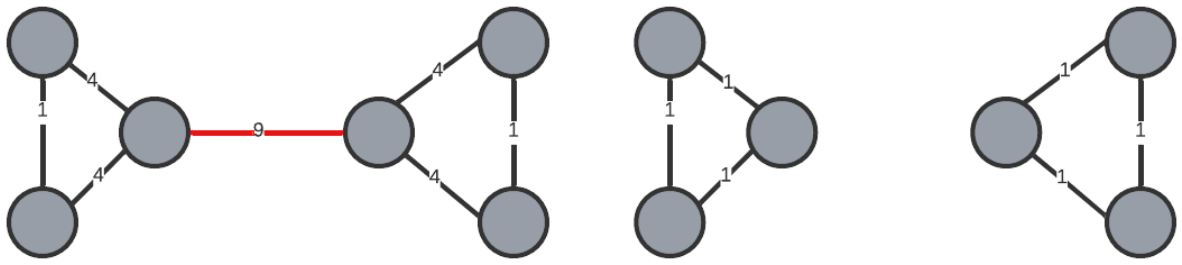


Figure 1.5 Edge betweenness

communities, and groups in the network. For instance, in a friend network, a person that is friends with multiple groups has the highest BC, thus it means he is an important person in terms of this measurement. This edge is called inter-community edge.

1.3 A community detection algorithm based on betweenness (Girvan Newman)

In this era, where technologies are advancing faster than ever, various algorithms are proposed with unique approaches such as fast greedy, infomap, leading eigenvector, walktrap, spinglass, betweenness, etc. Girvan Newman et al. invented an algorithm that divides a network into communities based on edge betweenness[2] in 2002. The algorithm was unique at the time because it focused on the edge's betweenness instead of its centrality. It removes the most between edges from the graph as the network is divided into communities. As mentioned above, node betweenness is proposed by Freeman et al. and Girvan Newman et al. used betweenness on edge for their algorithm. If there are multiple shortest paths between two nodes, then the algorithm counts every edge that has the same weight. If there are a few edges that connect communities in the graph then these edges have a better probability to have a high betweenness. As we remove these edges, the communities will disconnect from the rest of the graph over time. The algorithm removes the edge with the highest BC until there is no edge left. To calculate every edge betweenness in the graph with m edges and n nodes, it takes O(mn) time complexity. For every edge, we have to repeat the same step so the time complexity of an algorithm is estimated as $O(m^2n)$. However, in reality, the time complexity will be smaller because the number of the edge is decreased one by one in a loop.

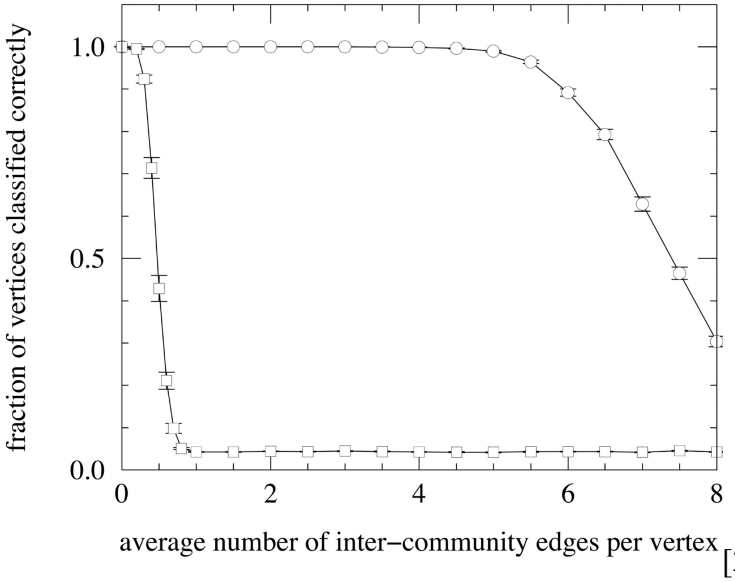


(a) Before Girvan Newman algorithm (b) After Girvan Newman algorithm

Figure 1.6 Result of Girvan Newman

Girvan Newman et al. implemented and experimented with their algorithm on various social

networks and artificial networks and compared it with the traditional hierarchical clustering method.



[2]

Figure 1.7 Vertical alignment is the number of correctly classified nodes and horizontal alignment is the number of inter-community edges. The Square represents the traditional hierarchical clustering method and the round represents the Girvan Newman algorithm

From the figure above, we can infer that GN (Girvan Newman) method reserves its quality as the number of inter-community edges increases. However, the main problem with the GN method is its very expensive execution time. It calculates all edge BC in the graph in one loop with O(mn) time complexity which is not optimal and it makes it impossible to use the algorithm for bigger graphs. Our proposed idea is to decrease the execution time without losing its main principle which is to focus on inter-community edges.

2. GN IMPROVEMENT

2.1 Girvan Newman (GN)

The algorithm calculates edge BC in every loop until there is no edge in the graph. As mentioned above, this move is expensive as the time complexity is linear.

Algorithm 1 GN algorithm

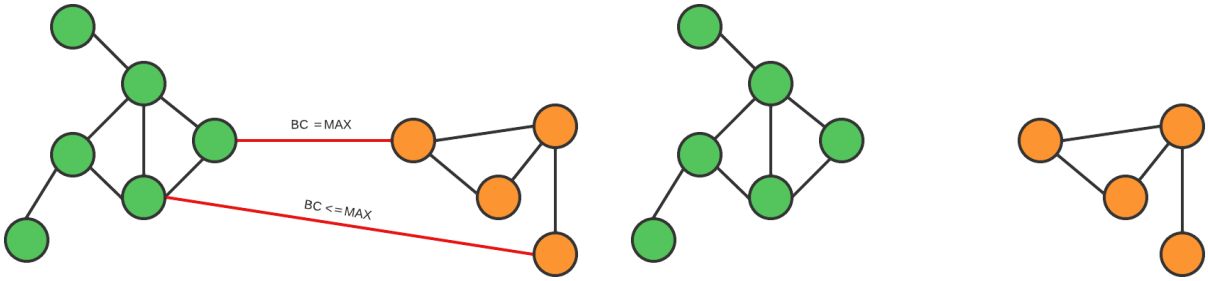
```
1: function girvanNewman(G)                          ▷ G–input graph
2:     while edgeNumber is not 0 do
3:         edges ←calculateBetweenness(G)            ▷ removes an edge with highest BC
4:         removeEdge(edges.max)
5:     end while
6: end function
```

Table 2.1 Execution time of Girvan Newman experimented on various graphs.[5].

Input	**Type**	**Vertex number**	**Edge number**	**BC calculation time**	**GN**
Graph 1	*Undirected*	*500*	*2551*	*13sec*	*0.906 sec*
Graph 2	*Undirected*	*1000*	*9894*	*39sec*	*8.769 sec*
Graph 3	*Undirected*	*1500*	*22515*	*157sec*	*31.090 sec*

We can see that when the number of nodes and edges reaches 1000 execution time multiples 4 times.

2.2 Multiple edge removal



(a) Before GNM
(Multiple edge removing Girvan Newman)

(b) After GNM
(Multiple edge removing Girvan Newman)

Figure 2.1 Multiple edge removal

We proposed an idea to remove multiple edges with top N highest betweenness since the algorithm calculates BC for every edge and reduces time complexity by the number of edges we remove. Even if the algorithm removes undesired edges, it will still eliminate important edges as well. However, the number of edges to eliminate greatly affects the quality of the algorithm and we took the square root of the network's edge number as the optimal range. The GNM (Girvan Newman multiple edge removal) algorithm:

Algorithm 2 GNM (multiple edge removal)

```
1: function girvanNewman(G)                          ▷ G–input graph
2:     while edgeNumber is not 0 do
3:         edges ←calculateBetweenness(G)
4:         removeEdges(edges, percent)               ▷ eliminates multiple edges
5:     end while
6: end function
```

2.3 Approximating betweenness

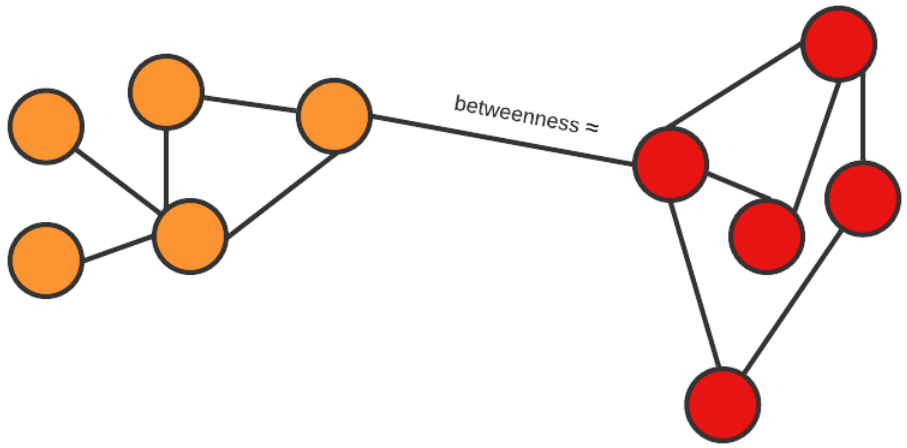


Figure 2.2 Approximating edge betweenness

GN calculates BC (betweenness) by taking all possible shortest paths between every pair node. This calculation process is mainly responsible for GN's high time complexity. In GN, the order of which edges have the highest BC is necessary but not the exact BC values. There are many types of efficient BC approximating algorithms and our second approach was to use one of these approximating algorithms instead of calculating exact edge BC values so we could preserve the quality and reduce the time complexity at the same time. We chose a BC approximating algorithm "Fast approximation of betweenness centrality through sampling"[8] invented by Matteo Riondato et al. to use in the GN algorithm and experiment on. The algorithm takes pair nodes randomly r sufficient times and increases every edge's BC by 1/r that is on the shortest path between the pair. The r relies on various variables such as delta, epsilon, universal constant, and graph diameter. All approximate values are within an additive factor $\varepsilon \in (0, 1)$ from the real values, with probability at least $1-\delta$ [8]. In the article, the authors mentioned that calculating graph diameter would also have high time complexity so they suggested a graph diameter approximating algorithm which takes graph diameter as the sum of the two longest paths between a randomly chosen node and the rest of the nodes in the graph. In the result, we implemented the algorithm and combined it with the

multiple edge removal approach which was later introduced in the experimenting section as GNC (Girvan Newman combined).

Algorithm 3 BC aproximation algorithm

Input: Graph G = (V,E) |V| = n, ϵ**,** $\delta \in$**(0,1)**

Outpit: List containing edge bc ordered as highest to lowest

1: **function** bcApprx(G) ▷ G–input graph

2: $b(v) \leftarrow 0$ ▷ Initalize all edge BC as 0

3: $VD(G) \leftarrow$getVertexDiameter(G) ▷ Approximates diameter

4: r $\leftarrow (c/\epsilon^2)(log_2(VD(G) - 2)) + \ln 1/\delta)$ ▷ calculate the r

5: **for** `i ←1 to r` **do**

6: (u,v) ←sampleUniformVertexPair(V) ▷ Randomly choose a pair of nodes

7: $S_{uv} \leftarrow$computeAllShortestPaths(u,v) ▷ Randomly choose a shortest path between the pair

8: $t \in S_{uv}$ ▷ t edge in S_{uv} path

9: $b(t) \leftarrow b(t) + 1/r$ ▷ Increase edge BC by 1/r

10: **end for**

11: **end function**

3. IMPLEMENTATION, EXPERIMENS

3.1 Used datasets

- Karate club: The dataset contains social ties among the members of a university karate club collected by Wayne Zachary in 1977.[9]
- Dolphine dataset: A social network of bottlenose dolphins. The dataset contains a list of all of links, where a link represents frequent associations between dolphins. [9]
- Data collected about Facebook pages (November 2017). These datasets represent blue verified Facebook page networks of different categories. Nodes represent the pages and edges are mutual likes among them.[9]
- Facebook social dataset: This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. The dataset includes node features (profiles), circles, and ego networks. [6]
- LastFM dataset: A social network of LastFM users which was collected from the public API in March 2020. Nodes are LastFM users from Asian countries and edges are mutual follower relationships between them. The vertex features are extracted based on the artists liked by the users. The task related to the graph is multinomial node classification - one has to predict the location of users. This target feature was derived from the country field for each user.[10]
- Advogato: Advogato is a social community platform where users can explicitly express weighted trust relationships among themselves. The dataset contains a list of all of the user-to-user links. аруулна.[9]

We used datasets varying from small to middle-sized networks from networkrepository.com and Stanford University's snap dataset website to test and experiment with our algorithms.

Table 3.1 Used datasets

Dataset	Type	Node number	Edge number	Diameter
Karate club dataset	*Undirected*	*34*	*78*	*5*
Dolphine dataset	*Undirected*	*62*	*159*	*8*
Facebook food page dataset	*Undirected*	*620*	*27,806*	*17*
Facebook dataset	*Undirected*	4,039	*88,234*	*8*
LastFM dataset	*Undirected*	*7,624*	*27,806*	*15*
Advogato dataset	*Undirected*	*5,200*	*47,300*	*14*

3.2 Result

3.2.1 Results from real life networks

The table below shows the time execution of algorithm GNC (combined), GNM (multiple edge removal), and GN (Original Girvan Newman). We took the square root of edge number as the number of edges to eliminate in one loop and took constant and variables accordingly: epsilon = 0.3, delta = 0.5, and universal constant are 0.5 as the authors recommended.

Table 3.2 Time execution result. GNC (Girvan Newman combined as epsilon = 0.3, delta = 0.5). GNM (Girvan Newman multiple edge removal)

Өгөгдөл	GNC/combined/	GNM/multiple edge/	GN
Karate club dataset	*0.00498sec*	*0.0149sec*	*0.0368sec*
Dolphine dataset	*0.03792sec*	*0.0518sec*	*0.0738sec*
Facebook food page dataset	*0.09175sec*	*14.2979sec*	*56.6508sec*
Facebook dataset	*79.3771sec*	*1356.1541sec*	*>=18000sec*
LastFM dataset	*137.0539sec*	*3618.751sec*	*>=18000sec*
Advogato dataset	*41.1248sec*	*2283.1243sec*	*>=18000sec*

3.2.2 Algorithm quality

We have measured GNC quality by comparing it with GN output which is a list array using SequenceMatcher which is a method of python default difflib class. SequenceMatcher is a method that compares two lists and outputs how many percent of lists are matched with each other.

Table 3.3 The table shows how many percent of GNC matches with GN in the following cases. Output is measured by 2 communities due to GN's high execution time.

Algorithm	Karate club	Dolphine dataset	Facebook food page dataset	Facebook dataset
GNC (e=0.1 delta=0.3)	*0.95*	*0.96*	*0.97*	*0.73*
GNC (e=0.2 delta=0.4)	*0.88*	*0.94*	*0.91*	*0.67*
GNC (e=0.3 delta=0.5)	*0.78*	*0.90*	*0.87*	*0.67*

3.2.3 Karate club graphic results

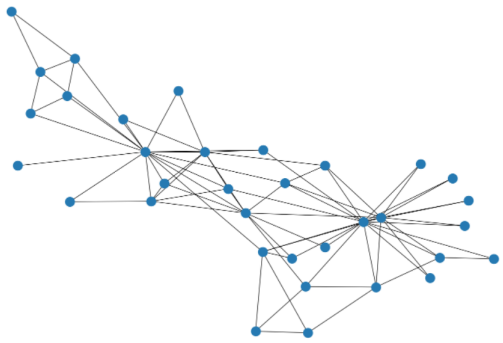
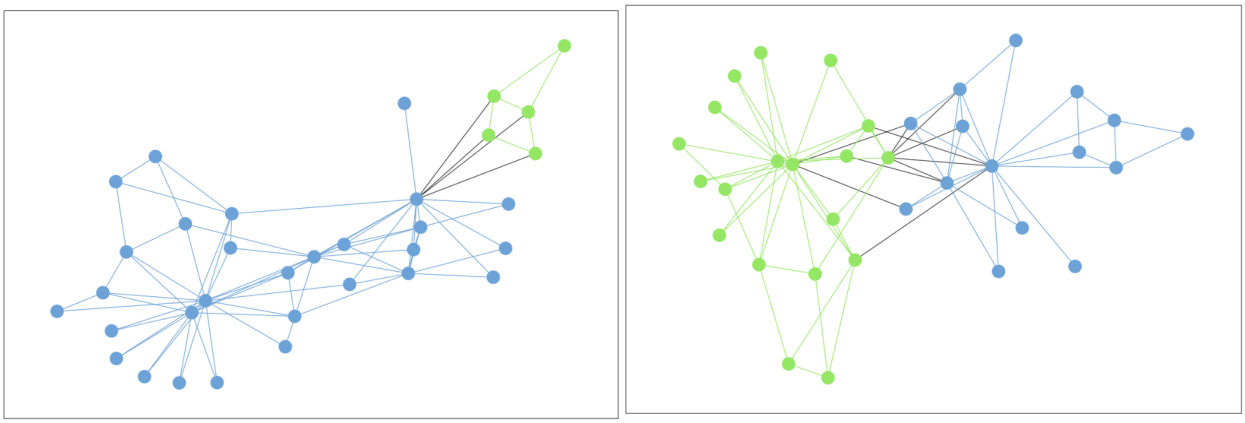


Figure 3.1 Karate club network

Figure 3.1 demonstrates the structure of the network and Figure 3.2(a) demonstrates the GNC algorithm after dividing the network into 2 communities. As you can see, the GNC algorithm didn't return input identical or close results to the GN algorithm. It is because the edge elimination number is not in the optimal range. After all, the network is very small. If the edge removal number is deducted to 2 or 3 in this case, the output becomes almost identical or close as it is demonstrated in Figure 3.3.



(a) GNC algorithm (b) GN algorithm

Figure 3.2 Karate club result

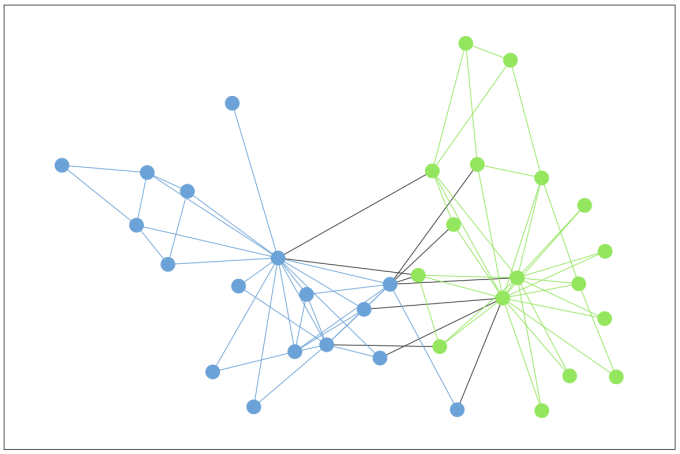


Figure 3.3 GNC after reducing edge removal number to 3.

3.2.4 Dolphine network graphic result

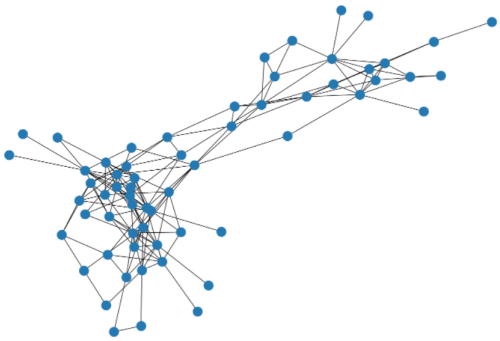
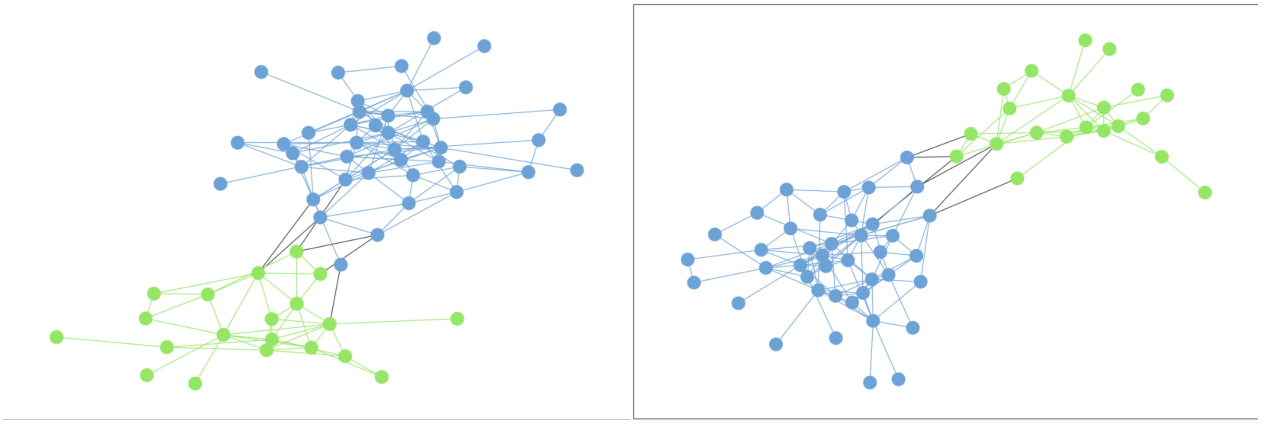


Figure 3.4 Dolphine social interaction network

Figure 3.4 demonstrates the overall structure of the dolphins' social interaction network and Figure 3.5(a) shows the GNC algorithm result after dividing the network into 2 communities and Figure 3.5(b) shows the GN algorithm result. The dolphins' network is two times bigger than the karate club dataset, hence, removing the square root of edge number edges is effective.



(a) GNC algorithm (b) GN algorithm

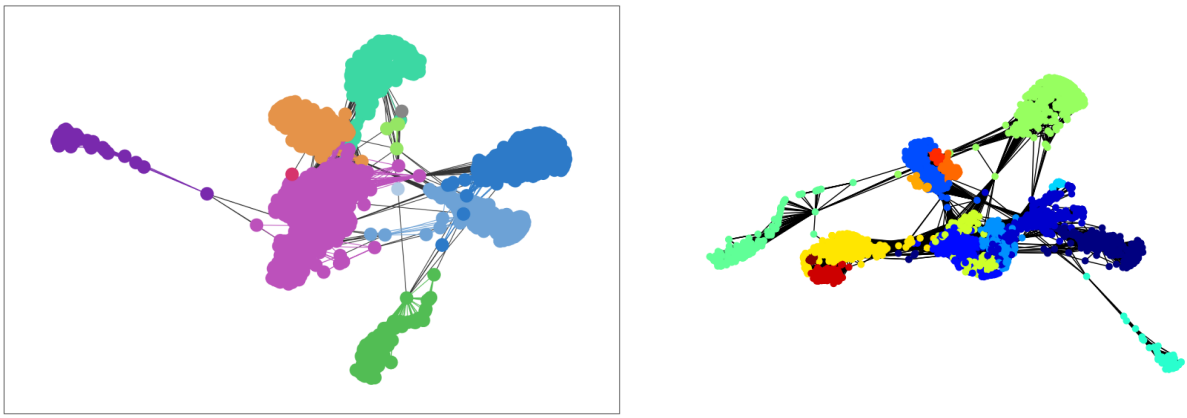
Figure 3.5 The dolphine network graphic result

3.2.5 Facebook social network graphic result

Figure 3.6 Facebook friend list network

Figure 3.6 demonstrates the overall structure of the facebook frien list network and Figure 3.7(a) shows the GNC algorithm resul after dividing the network into multiple communities and Figure 3.7(b) shows the GN algorithm result. The network is relatively bigger than previous networks and it contains over 4 thousand vertexes and 80 thousand edges. GNC eliminates square root of the total

edge number which is around 290 in this case.



(a) GNC algorithm (b) GN algorithm

Figure 3.7 Facebook social network graphic results

Conclusion

In this thesis work, we have tried to reduce the Girvan Newman community detection algorithm's time complexity by 2 approaches. The first approach, multiple edge removal, rely on how many edges to remove, thus, quality drops as the number of edges to remove increases. The second approach, however, while maintaining stable quality, greatly reduces the algorithm execution time. For further improvements, Matteo Riondato et al. proposed another algorithm that outputs a list of top-K edges with the highest betweenness. This algorithm greatly matches Girvan Newman's algorithm purpose which is to find top-k edges with the highest betweenness.

Bibliography

[1] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, 1977. Publisher: [American Sociological Association, Sage Publications, Inc.].

[2] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[3] Tatsunori B. Hashimoto, Masao Nagasaki, Kaname Kojima, and Satoru Miyano. BFL: a node and edge betweenness based fast layout algorithm for large scale networks. *BMC Bioinformatics*, 10(1):19, January 2009.

[4] Thamindu Dilshan Jayawickrama. Community Detection Algorithms, February 2021.

[5] Tripo Matijević, Tijana Vujicic, Jelena Ljucović, Petar Radunović, and Adis Balota. Performance Analysis of Girvan-Newman Algorithm on Different Types of Random Graphs. In *CECIIS - 2016*, August 2016.

[6] Julian McAuley and Jure Leskovec. Learning to discover social circles in ego networks. page 9.

[7] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, March 2004. Publisher: Proceedings of the National Academy of Sciences.

[8] Matteo Riondato and Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, March 2016.

[9] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[10] Benedek Rozemberczki and Rik Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. *arXiv:2005.07959 [cs, stat]*, August 2020. arXiv: 2005.07959.

[11] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community Detection in Networks with Node Attributes. *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156, December 2013. arXiv: 1401.7267.