Statistical Outlook for Community Mining in Social Networks

SURESH
GYAN VIHAR
UNIVERSITY
Mahal Jagatpura, Jaipur

A thesis submitted to the Suresh Gyan Vihar University, Jaipur,
in partial fulfillment of the requirements for the degree of

Master of Technology (*Information Communication Technology*)

By

Aditi Agrawal

(REG. No. : SGVU081090947)

Under the Guidance of

Pawan Prakash Singh

(Professor Computer Engineering Department)

GYAN VIHAR SCHOOL OF ENGG. & TECHNOLOGY
(Department of Computer Science & Engineering)
Mahal, Jagatpura, **Jaipur-302017**(Rajasthan)

MARCH, 2013

SURESH GYAN VIHAR UNIVERSITY
Mahal Jagatpura, Jaipur

This is to certify that this dissertation entitled "**Statistical Outlook for Community Mining in Social Networks**" has been submitted by **Ms. Aditi Agrawal, M.Tech (Information Communication Technology)** her **Enrollment No. SGVU 081090947** to the University.

Date: Registrar
(Evaluation)

EVALUATION CERTIFICATE

This is to certify that this dissertation has been evaluated.

	Examiner	Examiner
Signature		
Name		
College		
Date		

Ms. Savita Shiwani
Date : (Programme Coordinator)

DECLARATION BY THE CANDIDATE

I, hereby declare that this thesis entitled **"Statistical Outlook for Community Mining in Social Networks"** is a bonafide and genuine research work carried out by us under the guidance of **Mr. Pawan Prakash Singh**, Professor, Department of Computer Engineering, Gyan Vihar School of Engineering & Technology, Jaipur.

ADITI AGRAWAL
(REG. No. : SGVU081090947)
M.Tech (*Information Communication Technology*)
Gyan Vihar School of Engg. & Technology
Affiliated to Suresh Gyan Vihar University, Jaipur

Date :
Place: Jaipur

SURESH GYAN VIHAR UNIVERSITY
Mahal Jagatpura, Jaipur

CERTIFICATE BY THE GUIDE

This is to certify that the thesis entitled **"Statistical Outlook for Community Mining in Social Networks"** is a bonafide research work done by Miss Aditi Agrawal under my supervision in partial fulfillment of the requirements for the degree **Master of Technology (*Information Communication Technology*).**

(Pawan Prakash Singh)

Professor
Department of Computer Science & Engineering
Gyan Vihar School of Engineering & Technology
Jaipur – 30 20 17 (Rajasthan) India

Date :

Place : Jaipur

Dedicated to The Almighty, my Family, and Friends

ACKNOWLEDGEMENT

I am truly indebted to my thesis/dissertation Guide **Mr. Pawan Prakash Singh**, Professor Computer Engineering Department, **Gyan Vihar School of Engineering & Technology** (Affiliated to Suresh Gyan Vihar University, Jaipur) for providing me the golden opportunity to undertake this challenging project/thesis. Mr. Pawan Prakash Singh has not only given me the invaluable knowledge but also inspirational discussions throughout the course of this project/thesis.

My respect and regards goes out to all the faculty members of Computer Engineering Deptt. of Gyan Vihar School of Engg. & Technology for providing me full support during my thesis.

Finally, I thank my family and friends for their encouragement and support.

ADITI AGRAWAL

(REG. No. : SGVU081090947)

M.Tech (*Information Communication Technology*)

Gyan Vihar School of Engg. & Technology

Affiliated to Suresh Gyan Vihar University, Jaipur

Date :

Place : Jaipur

ABSTRACT

Social networking is very popular on the web and the combination with data mining techniques open up more opportunities for social intelligence on. Social Network Analysis (SNA) is one of the emergent fields of research that facilitates quantitative and qualitative analysis of social networks. Exploring the social network can assist gain further understanding on the features of the social networks.

A Social Network can be viewed as a complex interconnection of social entities, comprises of social structure of nodes tied together with one or more type of relationship namely share interests, activities, friendship, background, dislike, trade, financial exchange, etc. We can model more complex systems into social networks like neural networks, terrorist networks, image processing, market research, etc. Social entities are grouped together on the basis of their linked pattern. Mining a social is the work of grouping these social entities and there patterns for further discrimination, characterization, classification.

Much research has been done in the past on social mining algorithm. In this thesis, we will present a new algorithm Breadth First Droving (BFD) which uses statistical outlook for social mining in Social networks. The algorithm works as breadth first way and incrementally extract communities from the Network. This algorithm can be scaled easily for large Social networks and also fast and simple. The correctness of the outlook has been validated using implementation in GUESS (Graph Exploration System) tool and network illustrations.

CONTENTS

CERTIFICATE		III
ACKNOWLEDGEMENT		VI
ABSTRACT		VII
CONTENTS		VIII
LIST OF FIGURES		X
LIST OF TABLE		XI
CHAPTER I	**INTRODUCTION**	**1**
1.1	Introduction	1
1.2	Motivation	2
1.3	Issue Statement	3
1.4	Introduced Solution	4
1.5	Organization of Thesis	4
CHAPTER II	**LITERATURE SURVEY FOR SOCIAL MINING**	**5**
2.1	Introduction	5
2.2	Social Networks	5
2.3	Social Network Analysis	6
2.4	Social Mining and its Usage	8
2.5	Social Mining Algorithms	9
2.6	Comparison with Existing Algorithms	15

CHAPTER III INTRODUCED STATISTICAL SOCIAL MINING ALGORITHM 16

3.1 Introduction 16

3.2 Breadth first droving (BFD) Algorithm 16

3.3 Algorithm Evaluation 20

CHAPTER IV ALGORITHM IMPLEMENTATION FOR BREADTH FIRST DROVING 23

4.1 Introduction 23

4.2 GUESS Features 23

4.3 Loading Datasets 24

4.4 Running Script 24

4.5 Simulation Results 25

4.6 Performance Measure and Results 30

CHAPTER V CONCLUSION AND SCOPE OF FUTURE WORK 31

APPENDIX A 32

APPENDIX B 36

REFERENCES 44

LIST OF FIGURES

2.1	Illustrations that can be modeled into social networks	6
3.1	Drove Formation	17
3.2	Majority of Participation	18
3.3	A Social Network Illustrations	20
4.1	Loading dataset in GUESS tool	24
4.2	Running scripts in GUESS tool	25
4.3	Newman's Zachary karate club network	26
4.4	Communities formed in karate club	26
4.5	Network Illustrations 2	27
4.6	Outcome of Illustrations 2	27
4.7	Network Illustrations 3	28
4.8	Outcome of Illustrations 3	29
4.9	Comparison with Newman's algorithm	30

LIST OF TABLES

2.1 Comparison with algorithms in literature 15

3.1 Progress of algorithm 21

CHAPTER I

INTRODUCTION

In this chapter we will dissert in brief about this work, existing literature that motivated us to pick up this topic and the solution that we have introduced.

1.1 Introduction

Professor J. A. Barnes[4] first coined the term “Social Networking” in the 1950’s. Social Networking allows people to share ideas, pictures, posts, activities, interests and events etc. with other peoples in their network. A social network is the theoretical construct beneficial to study relationships among groups, individuals, organizations etc. More scientific and commercial usage needs patterns that are more complicated then frequent item-set and sequential patterns and require extra effort to discover. Such sophisticated patterns go beyond sets and sequence, towards trees, lattices, graphs, networks, and other complex structures. Such complex networks are extensively studied on Social Networks. Social Networks include online social networks, disease transmission networks, corporate executive networks, criminal/terrorist networks etc. Social Network Analysis (SNA) is one of the emergent fields of research that facilitates quantitative and qualitative analysis of social networks.

Real-life communities are formed by people working together, sharing a hobby, living nearby each other, etc. Social structure is main property of social networks which could provide a higher logical and structural view of network. Online social networking sites and Online communities have evolved much in past few years, providing lot of data to researchers from where the meaningful and beneficial data can be extract. More social mining algorithms have been developed in the past but there exists more issues making them perform slower and inefficient.

In section 1.2 we will present the issue statement, which we have seen from the literature. In section 1.3 we will present the outlook used to handle the issue.

1.2 Motivation

Now days, Social networks become the hot topic because of more researches and usage in this field. Social Networking is an active research area in social sciences. More usage namely social networking sites (Google+, Facebook, LinkedIn, Orkut, etc.), blog usage (e.g., Blogger, TypePad, Trumblr etc.), media sharing sites (e.g., Flickr, SlideShare, YouTube), is the reason to rise structured social networking. A lot of research has been done on social networks in past. In the literature, more algorithms have been developed to detect network communities. They can generally be divided as in [1] into three main categories:

1) *Graph theoretic ways* like Random walk ways and physics-depend ways Spectral ways

2) *Divisive algorithms* like `Betweenness' algorithms of Girvan and Newman [24] Tyler algorithm[23] and Radicchi algorithm [21] in which they divide the network into smaller subsections

3) *Agglomerative algorithms* like Modularity-depend algorithms [18] which form communities by joining nodes together.

Girvan and Newman [24] introduced ‘betweenness’ measure in 2002 which iteratively removes edges with the highest “stress” to eventually find disjoint communities. Clauset [18] in 2004 suggested a faster algorithm but the number of droves must still be specified by the user. Flake et al. [25] in 2000 used max-flow min-cut formulation to find communities around a seed node.

Kelsic [12] in 2005 introduced an agglomerative algorithm for constructing overlapping communities using local shells, and implement ways for visualizing overlap among communities. Pons and Latapy [13] in 2005 reported a social finding way applying random walk. It starts with single-node communities and again performs the merging of a pair of adjacent communities that minimizes the mean of the squared distances among each node and its social.

Hildrum [10] in 2005 submitted a cut-depend focused social search algorithm. Palla [14] in 2005 used clique percolation for the problem of identifying communities, Their way first identifies all cliques of the network and performs a standard component analysis of the clique-clique overlap matrix to discover a set of k-clique-communities.

Kim and Jeong [15] in 2005 developed a matrix block diagonalization and applied it to weighted stock networks. Their way constructs a network of stocks and identifies

stock groups with a percolation outlook depend on a filtered empirical stock correlation matrix.

Newman [9] in 2006 introduced eigen-spectrum of a matrix and calls it modularity matrix, which plays a role in social detection similar to that played by the graph Laplacian in graph partitioning calculations. Qian [6] in 2006 submitted a link mining algorithm to identify communities of practice depend on the idea that linked nodes belonging to the same social should have a larger number of 'common friends'. Ichise[7] in 2006 submitted, a social mining system which assists to find communities of researchers by using bibliography data, in this way the key feature is the modeling of papers and researchers, which enables us to eliminate the edges of large droves.

Yang and Liu [5] in 2006 submitted an incremental force-depend algorithm which allows mining communities in large scale dynamic networks, which is inspired by Newtonian gravitational law; it considers degree of vertex as mass and each edge as virtual spring. Recently Yang et al. [1] in 2007 developed a new algorithm, known as FEC, for identifying communities from signed social networks. The key idea behind it rests on an agent-depend random walk model, depend on which the FC phase can find the sink social including a specified node with a linear time complexity. Thereafter, the sink social is extracted from the entire network by the EC phase depend on some robust graph cut criteria.

In one other paper by Yang and Liu [2] in 2007 they submitted an Agent -depend AOC outlook to solving Distributed Network Social Mining Issue (D-NCMP), In this outlook, the nodes and links of distributed networks are distributed among a group of autonomous agents, who are responsible for finding all natural communities hidden in distributed networks, depend on their respective local views.

1.3 Issue Statement

Social network analysis is an emerging research area due to its wide usage. Social mining is the issue to discover the group of nodes that shares similar properties. The issue of identifying communities in a network is usually modeled as graph clustering (GC) issue or subgraph identification issue. More authors worked on this issue and have given various outlooks to mine out social structure. In Chapter 2 we will dissert most of these outlooks in detail. After studying existing algorithms we concluded that these algorithms use complex calculations and concepts to extract communities.

We are concerned to find a solution to mine communities out of social networks efficiently, to lower time complexity giving fast execution with increase in network size and that doesn't need to provide external parameter for droving, i.e., it should be fully automatic.

1.4 Introduced Solution

To mine out communities from a Social Network we have submitted a new algorithm which called BFD (Breadth First Droving) in Chapter 3 because it proceeds in breath first way to traverse and performs simple calculations to incrementally extract out communities from large Social Networks. Hence, we introduced the title of thesis as:

"Statistical Outlook for Community Mining in Social Networks"

We have implemented the algorithm on GUESS (Graph Exploration System) tool and have given the performance measure in Chapter 4 by comparing it with existing Newman's "Betweenness Algorithm" to show its effectiveness and linear time complexity.

1.5 Organization of Thesis

The thesis is organized as follows. In this *Chapter* we have disserted in brief about the existing literature, the issue statement and our outlook for its solution in this work. In ***Chapter 2***, we will dissert Social Networks, Social Network Analysis, and various existing algorithms in detail. Here, we will also dissert the main idea of these social mining algorithms and their drawbacks. In ***Chapter 3***, we will present a new algorithm BFD (Breadth First Droving) to mine communities and dissert how it works by validating it with network illustrations. In ***Chapter 4*** we will describe the implementation of algorithm in GUESS (Graph Exploration System) tool and will show the outcomes of the algorithm. ***Chapter 5*** will present the performance measure of our algorithm and rank it with respect to existing algorithms and Chapter 6 will present the summary of the thesis with the limitations and future directions of this work.

In ***Appendix A***, we will present python script of our BFD algorithm. In ***Appendix B***, we will present research resources and links available for Social Network Analysis.

CHAPTER II

LITERATURE SURVEY FOR SOCIAL MINING

2.1 Introduction

Real-life communities are formed by people working together, sharing a hobby, living nearby each other, etc. Social structure is main topological property of social networks which could provide a higher logical view of network, and will dramatically decrease the dimensionality while exploring the structure and evolution of the social network.

2.2 Social Networks

A Social Network comprises of social structure of nodes tied together with one or more type of relationship namely friendship, dislike, trade, financial exchange, etc. Bo Yang et al. [1] defined a social network as a graph G = (V, E), where V = {V_1, V_2, V_3, …. ,V_n} is the set of vertices, and E is the set of edges connecting pairs of vertices. Each edge represents the social relationships among two nodes representing people. They have heterogeneous and multi-relational dataset resubmitted by graphs. Typically these graphs are very large and both nodes and links have attributes. Social networks need not to social in context. There are more real world instances of economic, biological, technological and business social networks.

Social Networks include online social networks, electrical power grids, disease transmission networks, corporate executive networks, the spread of computer viruses, criminal/ terrorist networks etc. Customer networks and collaborative filtering issues (where product recommendation is made depend on the preferences of other customers) are other illustrations. In biology, illustrations range from epidemiological networks, cellular networks, and food webs, to the neural network of the nematado worm *Caenorhabditis elegans*(the only creature whose neural network has been completely mapped). The exchange of email messages within corporations, newsgroups, chat rooms, friendships, and the quintessential “old boy” network are illustrations from sociology.

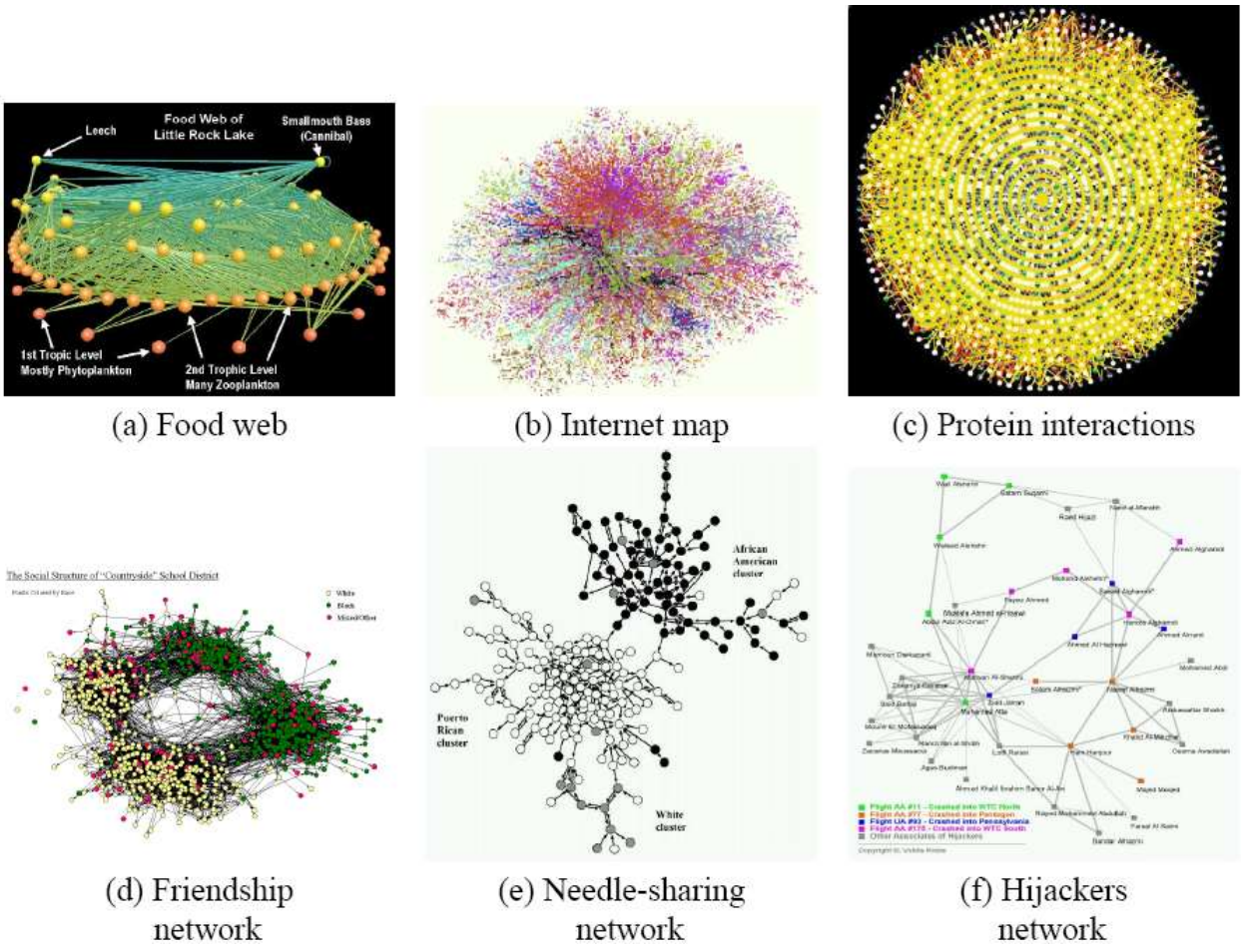


Fig. 2.1 Illustrations of graphs that can be modeled into social networks.

2.3 Social Network Analysis

As a general data structure, graphs have become increasingly main in modeling sophisticated structures and their interactions, with broad usage including bioinformatics, chemical informatics, computer vision, web analysis, video indexing, and text retrieval. Mining sub graphs and there patterns for further characterization, discrimination, classification, and drove analysis becomes a main work.

Moreover, graphs that links networks, computer networks, biological networks, and web and social networks. Because such networks have been studied extensively in the context of social networks, there analysis has often been referred as Social Network Analysis. Social Network Analysis (SNA) is one of the emergent fields of research for extracting beneficial information from social network data. Exploring the social network can assist gain further understanding on the characteristics of the social networks. M.

Jamali [8] defined SNA as the mapping and measuring of relationships and flows among people, groups, animals, computers or other information processing entities.

The graph clustering work does not try to classify, estimate, or prediction of groups. Instead, graph clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or droves, where the similarity of the records within the drove is maximized and the similarity to records outside the drove is minimized.

2.3.1 SNA Organizations and Groups

There exist more journals, organizations and groups which publishes research work related to Social Network Analysis. We have listed them in *Appendix B* with details. Some of the main journals and organizations are:

- **Social Networks Journal, Elsevier –** Journal associated with analysis, evolution and research work on social networks.
- **International Network for Social Network Analysis** (INSNA) - is a multidisciplinary scholarly organization, started in 1977 by Barry Wellman at the University of Toronto, it now has more than 1200 members.
- **Center for Computational Analysis of Social and Organizational Systems** (CASOS) at Carnegie Mellon.
- **Social Computing Group, Microsoft Research** - This group research and develop software that contributes to compelling and effective social interactions.
- **Netwiki** – a scientific wiki page devoted to social networks; maintained at University of North Carolina.
- **FAS.research** – This group researches on network visualizations produced using social network analysis

2.3.2 SNA Software Tools

More software tools are available which provide features for exploring and visualizing social networks. They are listed in details in *Appendix B.* Some of the good tools are:

- **GUESS**-Graph exploration System, Analysis and visualization tool that uses JUNG graph library and Jython (Java in Python) language.
- **UCINET**- SNA software by analytic technologies.
- **Pajek**- SNA software and visualization tool implemented in Pascal.
- **SocNetV**- Social Networks Visualize, an open source package in Linux.
- **ORA**- Network Analysis package in R language.
- **InFlow**- Business oriented SNA software**.**
- **NetMiner**- another Business oriented SNA software.

2.4 Social Mining and its Usage

The process of grouping a set of physical or abstract objects into classes of similar objects is called droving. A drove is a collection of data objects that are similar within the same drove and dissimilar to the objects in other drove. In context of Social Networks the work of grouping set of nodes exhibiting similar properties or behavior is referred as Social Mining, Sub-Graph Mining or Graph Clustering issue. Bo Yang et al. [2] defined a network social as a group of nodes, within which the links are dense but among which they are sparse. The structure of a system always affects its functionality. For illustrations, the topology of social networks affects the spread of infectious disease via a structured population. We can model more complex systems into social networks like neural networks, social networks, disease transmission networks, corporate executive networks, terrorist networks, etc. Thus Social Mining is a main aspect of social network analysis to gain insight of its structure and functionality.

Right form one childhood, Cluster analysis is main human activity. One learns how to distinguish among cats and dogs, or among animal and plants, by continuously improving subconscious classification schemes. Drove analysis has been widely used in numerous usage, including *pattern recognition, data analysis, image processing, market research, etc*. By mining communities or graph droving, one can identify crowded and

sparse regions, and therefore, discover distinct groups in their customer bases and characterize customer groups depend on purchasing patterns.

Social networks have been used to examine how organizations interact with each other, characterizing the more informal connections that link executives together, and associations and connections among individual employees at distinct organizations. In biology, it can be used to *derive plant and animal classifications*, *categorize genes with same functionality, and profit insight into structures vested in populations*. Social mining may also assist in identification of areas of similar land use in earth observation database, and in identification of groups of motor insurance policy holders with a high average claim cost, and also the identity of groups of houses in a city according to house type, appraise, and geographical location. It may also assist classify documents on the *WWW for information discovery*. As a data mining function, Social Mining can be used a standalone tool to gain insight for further analysis. More social mining algorithms have been developed in the past but there exists more issues making them perform slower and inefficient, which we will dissert in next section.

2.5 Social Mining Algorithms

There has been significant research previously done for social detection in network analysis field. This section will give a brief overview of the major algorithms introduced for social detection. More authors have introduced social mining algorithms and have given various outlooks for it. We have picked some of the main algorithms from them and submitted in this section.

2.5.1 Betweenness Algorithm

Newman and Girvan [22] introduced a set of divisive algorithms which uses 'betweenness' measure for social detection in Social Networks. The main idea behind there algorithm was to look for the edges in the network that are most among other vertices, the edges connecting different subgraphs lies more in among then other edges in the subgraph. The betweenness measure thus finds the edges which when iteratively removed decompose the network into disconnected subgraphs. The algorithms include a recalculation step in which betweenness scores are re-evaluated after the removal of every edge. The process can be resubmitted as a dendrogram depicting the successive splits of the network into smaller and smaller groups.

To calculate the betweenness measure they submitted more ways like *shortest (geodesic)-path betweenness, random-walk betweenness, current-flow betweenness* etc. along with the

algorithm which uses these measures. There algorithm is computational intensive which operates in O(n3) time on a sparse graph, which makes it usable for networks up to about 10 000 vertices, but for larger systems it becomes intractable.

2.5.2 Max Flow Min Cut Algorithm

G. W. Flake [20] introduced a simple hierarchical graph clustering ways depend on minimum cut trees of the graph. The main idea of the mincut algorithm is to expand graph G to G_α (by adding artificial sink t and edges of weight α to G), find mincut Tree $T_{G\alpha}$, remove the artificial sink t, thus resulting connected components to form clusters in G. In Hierarchical Clustering the graph is decomposed in sub-graphs of desired size and then cut clustering is applied on them. So, clusters are created that have small inter-cluster cuts and relatively large intra-cluster cuts. This guarantees strong connectedness within the droves and is also a strong criterion for a good clustering.

In this algorithm an artificial sink node is inserted in the network and connects it with all other nodes. Then maximum flows from all other nodes to sink nodes are calculated, and recursively min-cut trees are formed, thus forming droves in graph indirectly. They also submitted a framework for hierarchical droving and present usage to real-world data.

2.5.3 Modularity Algorithm

Newman [16] introduced a measure of coupling for k disjoint sets of nodes called modularity. Their algorithm provides a favorable scaling with network size. A network with n nodes can be resubmitted with an n-by-n (unweighted) `adjacency matrix', which has a 1 in the i^{th} row and j^{th} column if nodes I and j are connected, and a 0 otherwise. Modularity is a property of a network and a specific introduced division of that network into communities. It calculates when the division is a good, in the sense that there are more edges within communities and only a few among them. Modularity is defined as

$$Q = \sum_i (e_{ii} - a_i^2) \text{ and } a_i = \sum_j e_{ij}$$

where e_{ij} is the fraction of edges in the network that connect communities i and j. Each e_{ii} represents the number of edges internal to a social, and a_i^2 represents the expected appraise of internal edges assuming a uniform edge distribution. Thus, modularity measures the variance in social structure from a uniformly random graph. Initially the algorithm starts each node in its own separate social. It then conducts a greedy search by

repeatedly joining the two communities that result in the greatest increase in modularity until all of the communities are joined into one. The division with greatest calculated modularity during this search is the output of the algorithm, with run-times scaling as O(n(m + n)), where n is the number of nodes and m the number of edges.

2.5.4 Improved Modularity Algorithm

Clauset & Newman [17] submitted a fast hierarchical agglomerative algorithm for detecting social structure. The algorithm is submitted in *Appendix B.3,* its running time on a network with n vertices and m edges is O(mdlog n) where d is the depth of the dendrogram describing the social structure, More real-world networks are sparse and hierarchical, with m ~ n and d ~ log n, so this algorithm runs in essentially linear time, O(n $\log_2$ n). They improved the time complexity by exploiting some shortcuts in the optimization issue and using more sophisticated data structures.

2.5.5 Spectral Partitioning using Eigenvectors

Newman [9] introduced another way of modularity maximization as eigenspectrum of matrices and calls it modularity matrix. He submitted the modularity measure as spectral optimization work in linear algebra. He also gave a way for detecting bipartite or k-partite structure in networks, and a new social centrality measure that identifies vertices that play a central role in the communities to which they belong. They used modified benefit function Q, index vector s and eigen vectors u_i. We write s as a linear combination of the normalized eigenvectors u_i of the modularity matrix,

$$s = \sum_{i=1}^{n} a_i u_i \text{ with } a_i = u_i^T s \text{ . Then,}$$

$$Q = \frac{1}{4m} \sum_i a_i^2 \beta_i$$

Where β_i is the eigenvalue of the real symmetric matrix corresponding to eigenvertices u_i and $\beta_1 \geq \beta_2 \geq \beta_3 \quad ... \quad \beta_n$. The appraise of s is restricted to +/-1 and chosen proportional to u_i.

So,
$$s_i = \begin{cases} +1 \quad if \quad u_i^{(1)} >= 0, \\ -1 \quad if \quad u_i^{(1)} < 0 \end{cases}$$

They find the eigenvector corresponding to the most positive eigenvalue of the modularity matrix and divide the network into two groups according to the signs of the elements of

this vector. This outlook has some disadvantages too. It assumes the sizes of groups into which network is divided is fixed which is not true for real life networks. The algorithm needs to supply additional parameter α which assists in approximation process of algorithm.

2.5.6 External Optimization Algorithm

J Duch and A. Arenas [15] introduced external optimization algorithm to find the social structure in complex networks depend on an external optimization of the appraise of modularity. The algorithm is submitted in *Appendix B.4.*They considered λ_i as the local variable that characterizes an individual node, also referred as the fitness of node i.

$$\lambda_i = \frac{q_i}{k_i}$$

Where q_i is the local variable calculated as

$$k_{r(i)} - k_{r(i)} a_{r(i)}.$$

Where $k_{r(i)}$ is the no. of links that a node I belonging to a social r has with nodes into the same social, and k_i is the degree of node i. They move node having least fitness from one partition to another at each step of their process. The process finishes when the modularity Q could not be improved more.

$$Q = \frac{1}{2L} \sum_i q_i$$

Where i refer to all nodes in the network given a certain partition into communities and L is the total number of links in the network. The efficiency and accuracy of the way make it feasible to be used for the accurate identification of social structure in large complex networks. Their algorithm claims to outperform with respect to all introduced algorithms for social mining. But like betweenness algorithm, they need to recalculate fitness measure for all nodes in every loop. That makes the algorithm perform slowly as the size of network increases.

2.5.7 Force Depend Algorithm

Bo yang and D.Y. Liu introduced an incremental force depend algorithm to detect and visualize social structure using modified Newtonian laws according for graphs. They modeled network as physical system of n body simulation. Entire network is considered as 4D (E^3 x T) mechanics where E^3 is three dimensional Euclidean space and T is time dimension. Each vertex is considered as a particle having its own attributes like position (x_i), velocity, mass (m_i) and acceleration and will diffuse into such 4D world and interact with other particles via forces. Generally m_i is taken as degree of vertex, signifying that a vertex with greater mass have more impact on other vertices.

Each edge (V_i, V_j) of network is considered to be connected with each other via hypothetical spring exerting force:

$$F_r(x_i) = \sum_{v_j \in V} m_i m_j (l^0)^2 . \frac{x_i - x_j}{\left| x_i - x_j \right|^2}$$

Where l^0 is the natural length of edge spring and it represents radius of empty sphere around a vertex. l^0 is calculated as:

$$l^0 = \beta \sqrt[3]{\frac{volumeofE^3}{n}}$$

Every vertex pair repulses each other instead of attracting in contrast to Newtonian gravitation law. Only pair of vertices connected by an edge attracts each other with force:

$$Fa(Xi) = \sum_{(vi,vj) \in A} mimj \frac{\left| xi - xj \right|^2}{l^0} . \frac{xi - xj}{\left| xi - xj \right|}$$

The change in position of vertices at different times was calculated by:

$$\Delta X = \frac{1}{mi} \sum_{k=1}^{\infty} \gamma . F(xi(t + h_k))$$

The algorithm is submitted in *Appendix B.5.* So initially all vertices are added in random position in the Euclidian space with zero velocity, acceleration and force and edge set to natural length. Within a social as the edges are denser, the vertices attract each other forming communities and because of repulsive forces from other social vertices communities get separated from each other and can be visualized easily. This algorithm allows mining of large scale and dynamic networks.

2.5.8 Link Mining Depend Clustering Algorithm

Rong Qian et al. [6] submitted a fast link mining depend divisive algorithm to detect communities. The algorithm is submitted in *Appendix B.6.* The idea behind the algorithm is that linked nodes belonging to the same social should have a larger number of 'common friends'. Instead of using the link centrality appraises like used by Girvan & Newman, the algorithm works with the 'link-clustering coefficient':

$$\tilde{C}_{i,j}^{(g)} = \frac{z_{i,j}^{(g)} + 1}{S_{i,j}^{(g)}}$$

It represents the fraction of possible loops of order g that pass via a certain string. The algorithm is executed for triangles (g = 3) or squares (g = 4). The system computes the $\tilde{C}_{i,j}^{(g)}$ appraises for all links, and cuts the one with the minimum appraise. They considered informal social network from enron email corpus dataset and partition the network into communities.

2.5.9 FEC Algorithm

Yang et al. [1] recently introduced a new algorithm FEC, for identifying communities from signed social networks where both positive with-in relations and negative among-group relations coexists. The algorithm is submitted in *Appendix B.7.* The main idea behind the algorithm is an agent-depend random walk model, depend on which the ***Find Social (FC) phase*** can find the sink social including a specified node with a linear time complexity after that the sink social is extracted from the entire network by the ***Extract Social (EC) phase*** depend on some robust graph cut criteria.

In phase a sink node is placed by agent and calculates l-step transfer probability distribution function for each node. The l-step transfer probabilities are then sorted to find the nodes with least probabilities. The nodes with least l-step transfer probabilities represent the nodes outer to social and thus remove them to find the social structure.

In this chapter, we have given introduction to social networks, social network analysis and the most of the related work that has been done in this area so far. We have disserted various algorithms, the outlooks used by them to extract out the droves or communities from social network and noticed their algorithmic complexities and their dependencies on other factors. We have compared these algorithms in tabular form given below.

2.6 Comparison with Existing Algorithms

Table 2.1 Comparison with algorithms in Literature

S. No.	Algorithm	Type	Order of Time Complexity	Fully automatic
1	Newman Betweenness	Divisive	$O(N^3)$	yes
2	Max Flow Min Cut	Divisive	$O(N \log_2 N)$.	yes
3	Modularity	Agglomerative	O(N(M + N)),	yes
4	Improved Modularity	Agglomerative	$O(N \log_2 N)$.	yes
5	Spectral Partitioning	Divisive	$O(N^2)$	no
6	External Optimization	Divisive	$O(N^2 \ln N)$.	no
7	Force Depend	Graph theoretic	$O(MxN^2)$	yes
8	Link mining Depend	Divisive	$O(N^2)$	yes
9	FEC	Graph theoretic	O(N + M)	no
10	**BFD**	**Agglomerative**	**O(V+E)**	**yes**

In this table we have also compared our introduced algorithm BFD that we will dissert in next Chapter.

CHAPTER III

INTRODUCED STATISTICAL SOCIAL MINING ALGORITHM

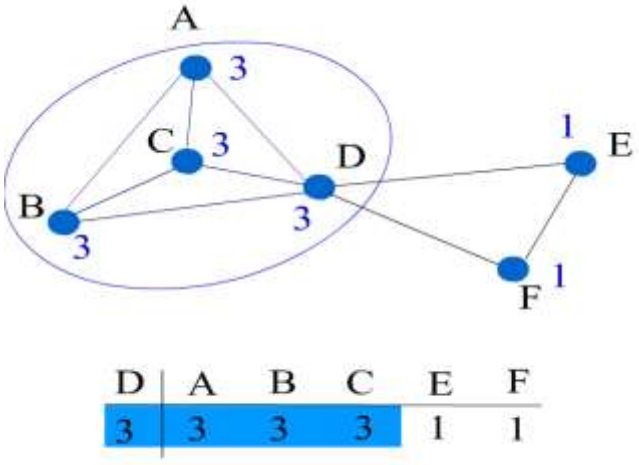
3.1 Introduction

In this chapter we present our introduced algorithm BFD and validate it using social network illustrations. Breadth first and depth first traversal techniques to traverse a graph data structure are submitted by Thomas Cormen[26]. Breadth first traversal uses FIFO (first in first out) Queue data structure for traversal and carries a main property of traversing along breadth of structure so it covers all the neighbors of node first rather than jumping from one node to another as in depth first traversal. BFD algorithm uses breadth first traversal to update statistics and detecting social structure.

3.2 BFD Algorithm

3.2.1 Main Idea

The communities in a network are formed by droving groups of nodes closely connected to each other. The algorithm uses breadth-first traversal, as disserted in Thomas Cormen [26], as its propagation way. In breadth first traversal all the neighbor nodes in social are traversed first and then nodes of other social. Whenever a node is traversed all its neighbor's visit counter is incremented, when we reach on a node having visit counter 2 or more, it signifies that a drove may exists if majority of its neighbors are traversed more than twice see fig.3.1.



D	A	B	C	E	F
3	3	3	3	1	1

Fig.3.1 Drove Formation

The case may arise for a node if its neighbors belong to more than one class then the vertex is assigned to the class with maximum common class neighbors, which we call as majority of participation rule. For any node V there exist four parameters prior to its classification:

- NAV ← No. of Adjacent Vertices of V.
- UDV ← No. of Undroved Vertices adjacent to V having visitcounter ≥ 2.
- DV ← No. of Droved Vertices adjacent to V having visitcounter ≥ 2.
- MCDN ← Max. Common Drove Number.

Let us have a look on the illustrations shown in fig. 3.2, the vertex neighbors belong to three droves (7, 6, and 4) and have two undroved neighbors.
We dissert 3 cases that will arise here:

Case 1: NAV/2 > (UDV+DV)
Case 2: UDV > DV considering case 1 is false.
Case 3: DV > UDV considering case 1 is false.

Case 1 arises when number of neighbors with visitcounter < 2 is more than remaining neighbors. In this case the vertex V is left undroved and reconsidered after, as the algorithm proceeds. Case 2 arises when number of undroved neighbors with visitcounter ≥ 2 is greater than remaining neighbors. In this case a new class is formed for vertex V along with its undroved neighbors. Case 3 arises when number of droved neighbors with visitcounter ≥ 2 is greater than remaining neighbors. In this case the class with maximum number of linked neighbors is selected and vertex V is appended into that class. So, the vertex V is assigned to drove 7.

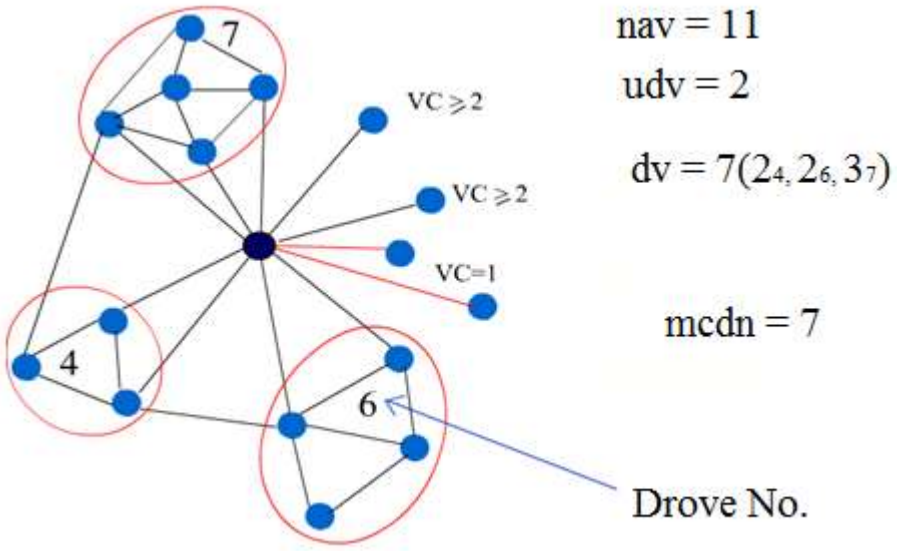


Fig. 3.2 Majority of Participation

In case there is no single class majority is there the vertex is left undroved, for illustrations in the fig. 3.2 if the node equally participates in all droves (Drove 7 also has 2 links with it) no single majority lies (so MCDN in that case should be 0).

Small tightly coupled components are detected first which merges nearby vertices together to form big drove on the basis of majority of participation incrementally. The detection of nodes of same social shows flood filling behavior. As the traversal to network grows small components drove together to form communities. The calculation is simple, fast and can be scaled for large networks.

3.2.2 Algorithm

Following is pseudo code for the BFD algorithm. The algorithm uses queue data structure is resubmitted by Q having enqueue and dequeue operations.

```
BFD(G, U)
begin
    Enqueue(Q, U)
    set U as visited
    while Q is not empty
    begin
        H ← Dequeue(Q)
        for each N ε Neighbors(h) in G
        begin
```

```
        increment VisitCounter(N)
        if N is not-visited
                enqueue(Q, N)
                set N as visited
        end
    if VisitCounter(H) > 1
    begin
            NAV ← No. of Adjacent Vertices of H
            UDV ← No. of UnDroved Vertices adjacent to H
            DV ← No. of Droved Vertices adjacent to H
            MCDN ← Max. Common Drove No.

            if (DV+UDV) > NAV/ 2
            begin
                    if UDV > DV
                            form set S_udv of UnDroved     vertices
                            set class(S_udv) =C+1                    \\ New Class Formed
                    else if DV>UDV
                    begin
                        Find MCDN                                    \\ Fig.3.2
                        set class(H) =MCDN                           \\ New Class Formed
                    end
            end
    end

  For all vertices left UnDroved
  Put them in their MCDN
end.
```

This algorithm has O(V+E) linear time complexity, which is similar to Breadth First traversal given by Thomas Cormen[26].

3.3 Algorithm Evaluation

Consider a network illustrations (fig. 3.3), we start from a randomly chosen vertex V_0, as we traverse the network and update the statics of network (see table 3.1), we will see how communities are detected incrementally out of the network.

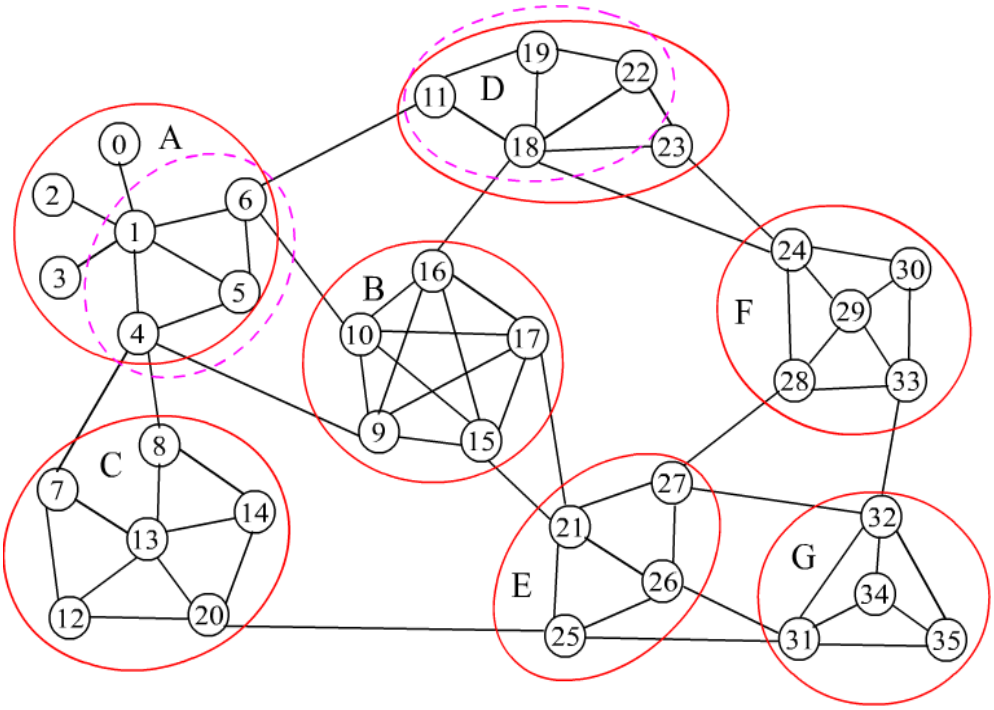


Fig. 3.3 A Social Network Illustrations

Table 3.1 Progress of Algorithm

	V(VC)	Adjacent Vertices of V(VisitCounter)	
1	0(0)	**1(1)**	
2	1(1)	0(1),**2(1),3(1),4(1),5(1),6(1)**	
3	2(1)	1(2)	
4	3(1)	1(3)	
5	4(1)	1(4),5(2),**7(1),8(1),9(1)**	
6	5(2)	1(5),4(2),6(2)	A
7	6(2)	1(6),5(3),**10(1),11(1)**	
8	7(1)	4(4),**12(1),13(1)**	
9	8(1)	4(5),13(2),**14(1)**	
10	9(1)	4(6),10(2),**15(1),16(1),17(1)**	
11	10(2)	6(2),9(2),15(2),16(2),17(2)	B
12	11(1)	6(3),**18(1),19(1)**	
13	12(1)	7(2),13(3),**20(1)**	
14	13(2)	7(3),8(2),12(2),14(2),20(2)	C
15	14(2)	8(3),13(4),20(3)	
16	15(2)	9(3),10(3),16(3),17(3),**21(1)**	
17	16(3)	9(4),10(4),15(3),17(4),18(2)	
18	17(4)	9(5),10(5),15(4),16(4),21(2)	
19	18(2)	11(2),16(5),19(2),**22(1),23(1),24(1)**	
20	19(2)	11(3),18(3),22(2)	D

21	20(3)	12(3),13(5),14(3),**25(1)**	
22	21(2)	15(5),17(5),25(2),**26(1),27(1)**	
23	22(2)	18(4),19(3),23(2)	
24	23(2)	18(5),22(3),24(2)	
25	24(2)	18(6),23(3),**28(1),29(1),30(1)**	
26	25(2)	20(4),21(3),26(2),**31(1)**	E
27	26(2)	21(4),25(3),27(2),31(2)	
28	27(2)	21(5),26(3),28(2),**32(1)**	
29	28(2)	24(3),27(3),29(2),**33(1)**	F
30	29(2)	24(4),28(3),30(2),33(2)	
31	30(2)	24(5),29(3),33(3)	
32	31(2)	25(4),26(4),32(2),**34(1),35(1)**	
33	32(2)	27(4),31(3),33(4),34(2),35(2)	G
34	33(4)	28(4),29(4),30(3),32(3)	
35	34(2)	31(4),32(4),35(3)	
36	35(3)	31(5),32(5),34(3)	

____ New drove formed in v(vc) column.

____ Merged with existing drove in v(vc) column.

The bolded text in table 3.1 represent new enqueued vertex for traversal. Lastly the nodes left undroved like in the illustrations V_0, V_2, V_3 are reconsidered again for majority to merge with existing droves.

Table 3.1 thus shows the progress of algorithm on the network taken in figure 3.3. The illustrations contain 36 nodes and 69 edges. The algorithm takes nearly .8 seconds for execution when implemented in GUESS tool. In the next chapter we disserted GUESS graph exploration tool and see the outcomes we got in its IDE. We also compared BFD algorithm with Newman's Betweenness algorithm in the next chapter to see its effectiveness.

CHAPTER IV

ALGORITHM IMPLEMENTATION FOR BREADTH FIRST DROVING

4.1 Introduction

In Chapter 3, we disserted BFD algorithm and its effectiveness. In this Chapter we dissert GUESS tool which we have used to simulate BFD algorithm. GUESS is a system that has a novel, interactive interpreter which connects the language and interface in a way that facilitates exploratory visualization works.

4.2 Features of GUESS

We have chosen GUESS (Graph Exploration System) tool to simulate our algorithm because of its following features:

- Incorporated with JUNG, a robust graph library, as a backend to represent nodes and graphs.
- Robust Jython core (Python in Java) language for selecting and managing nodes and edges.
- A database driven system. Nodes and edges have features that you can query and use to control what gets displayed.
- Ability to save state and to smoothly morph among states.
- Writes out more different types (jpg, gif, pdf, eps, svg, swf)
- Various layout algorithms
- Interface to R
- Support for subgraphs
- Allow graph analysis with users own scripts and its visualization.

4.3 Loading Datasets

GUESS uses simple comma separated format called GDF for dataset. It also supports GraphML and the Pajek file formats for importing from other usage. In addition to loading in various graph description files, users of GUESS can also create and remove nodes, edges, and fields on the fly. Various primitives support these functions and the data is appropriately saved into the backend database. It asks at start time to browse GDF, GML, or NET format datasets to load.

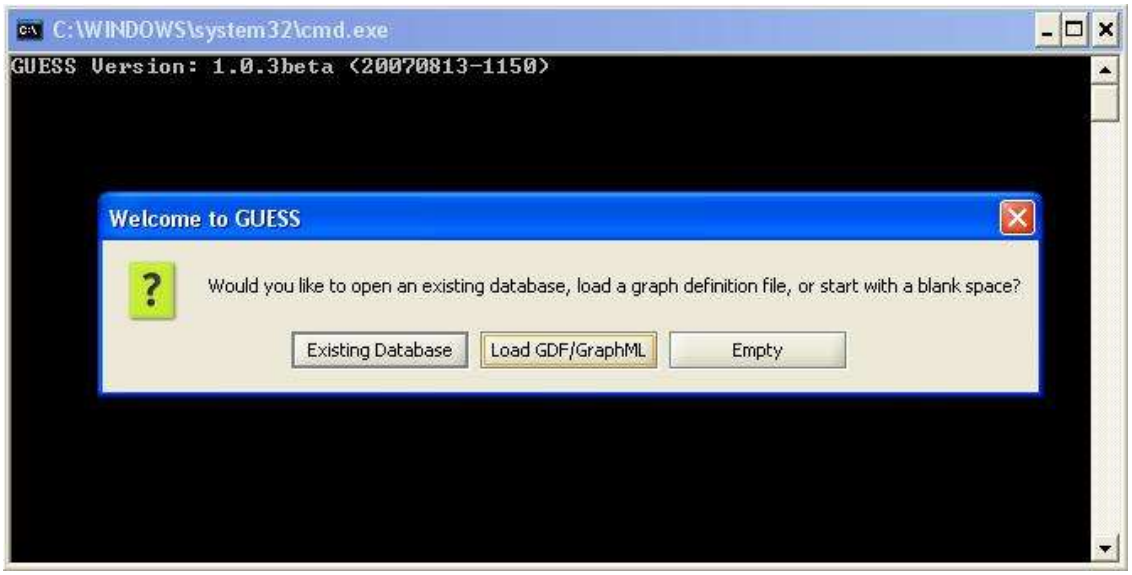


Fig. 4.1 Loading Dataset

4.4 Running Script

We have submitted implementation script of BFD algorithm in Appendix A. To run user defined scripts on dataset for analysis select Run Script option and browse the script in ".py" format. We can run the script from GUESS interpreter window by typing the command:

execfile("BFDrove.py")

In case of BFD algorithm after selecting runscript option from file menu browse to BFDrove.py script, this loads the script into memory.

To run the algorithm type: "BfDrove()" in interpreter window.

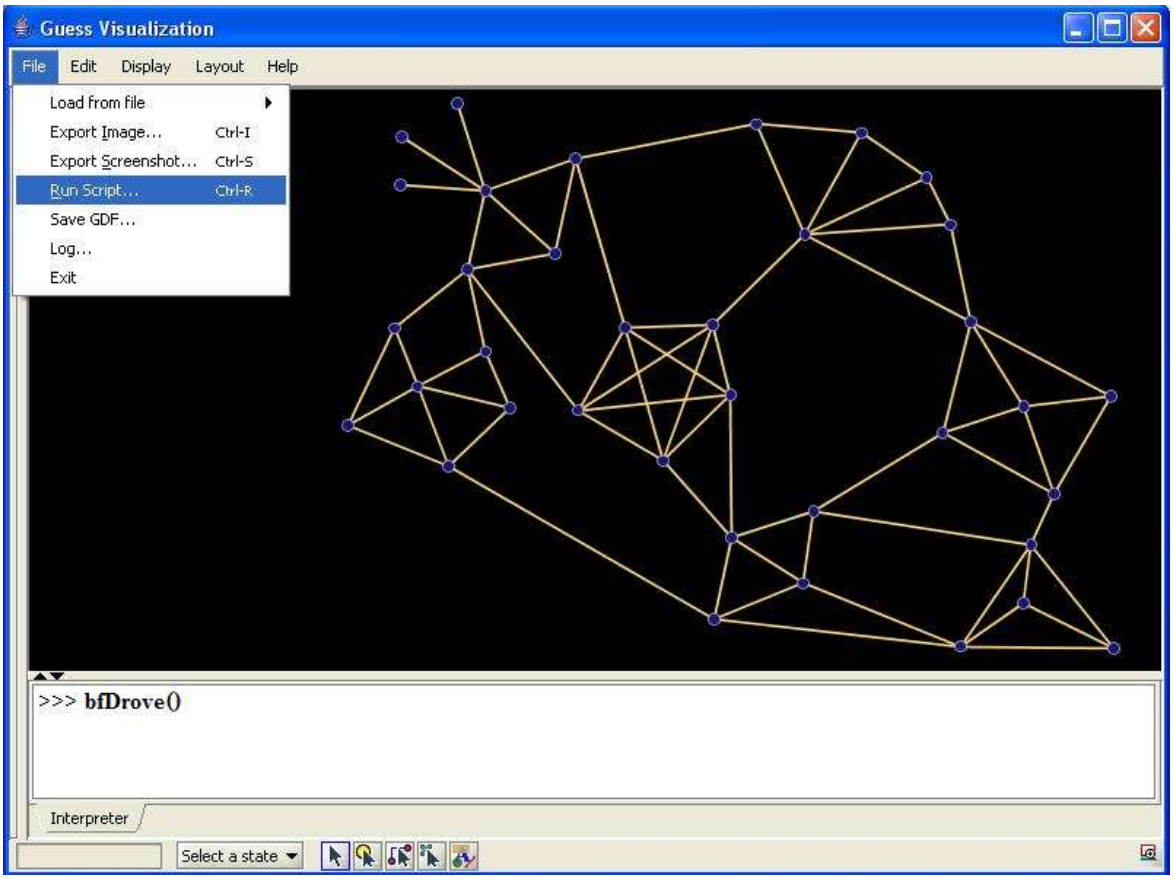


Fig. 4.2 Running Scripts

4.5 Simulation Results

We have simulated our algorithm using more datasets and got good results in linear time complexity. We have taken three network illustrations (in which communities are easily visible) and shown their outcomes to show its effectiveness.

Common illustrations used to illustrate social structure algorithms is the Zachary Karate Club, in which an internal dispute led to the schism of a karate club into the formation of three smaller clubs.

4.5.1 Network Illustrations 1 (Zachary Karate Club)

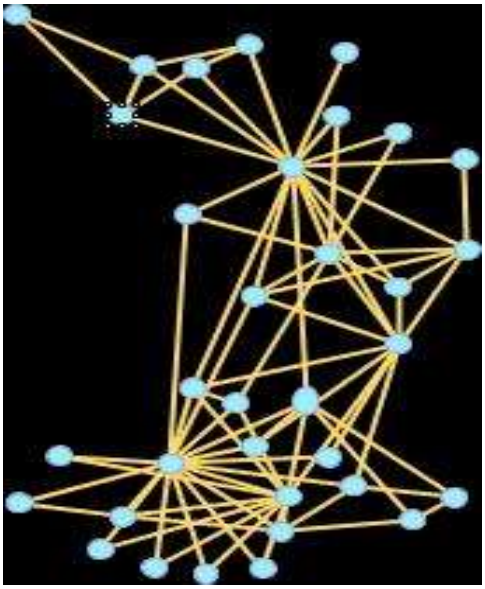


Fig. 4.3 Newman's Zachary Karate club network

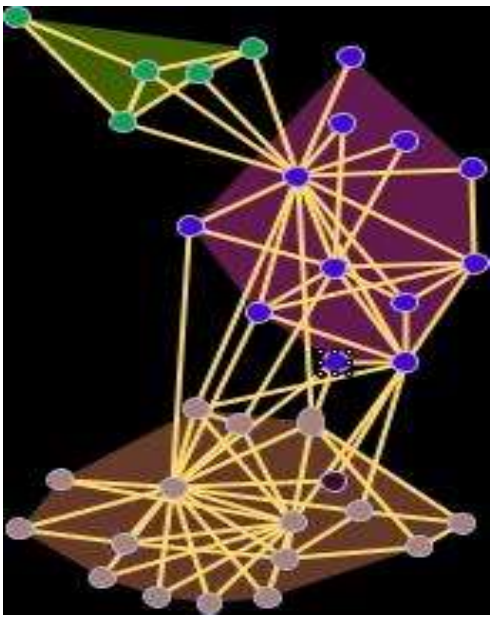


Fig. 4.4 Communities formed in Karate club.

4.5.2 Network Illustrations 2

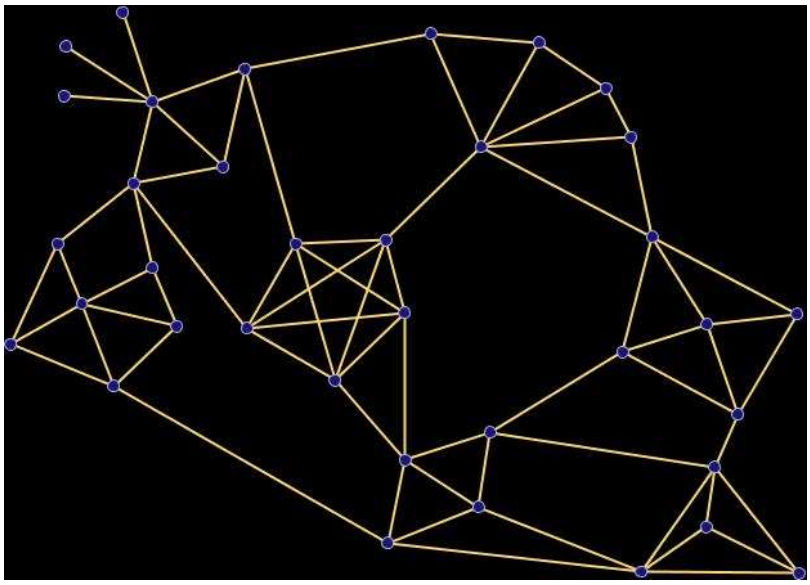


Fig. 4.5 Network Illustrations 2

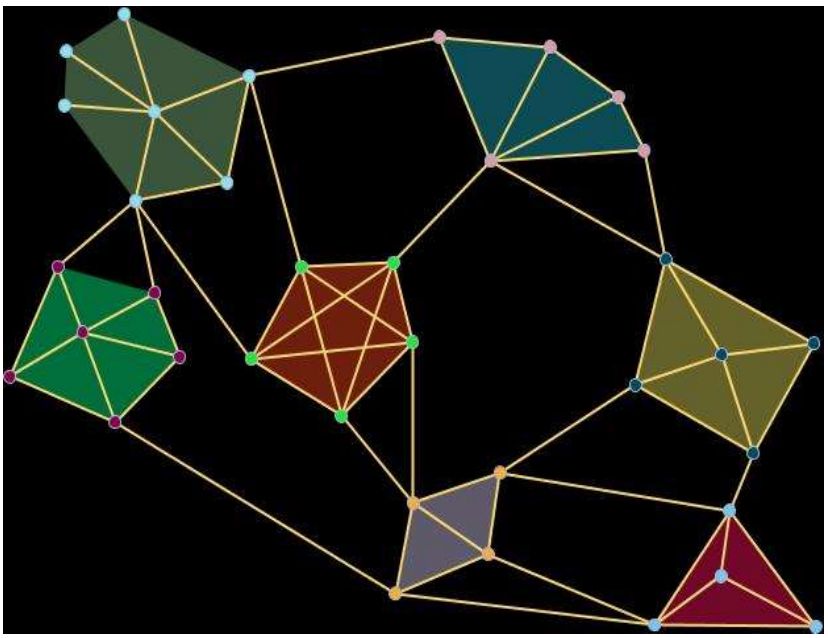


Fig. 4.6 Outcome of Illustrations 2

4.5.3 Network Illustrations 3

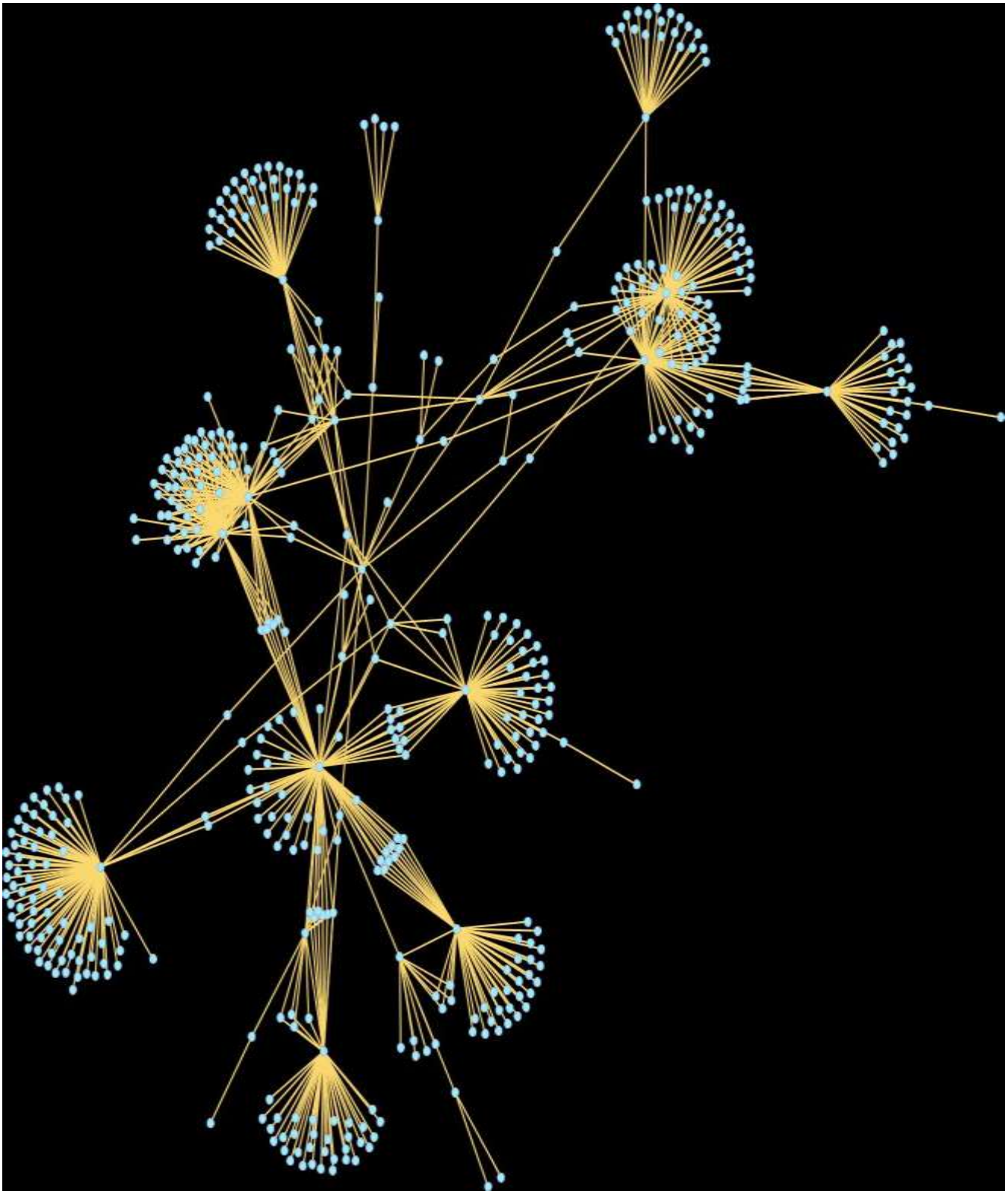


Fig. 4.7 Network Illustrations 3

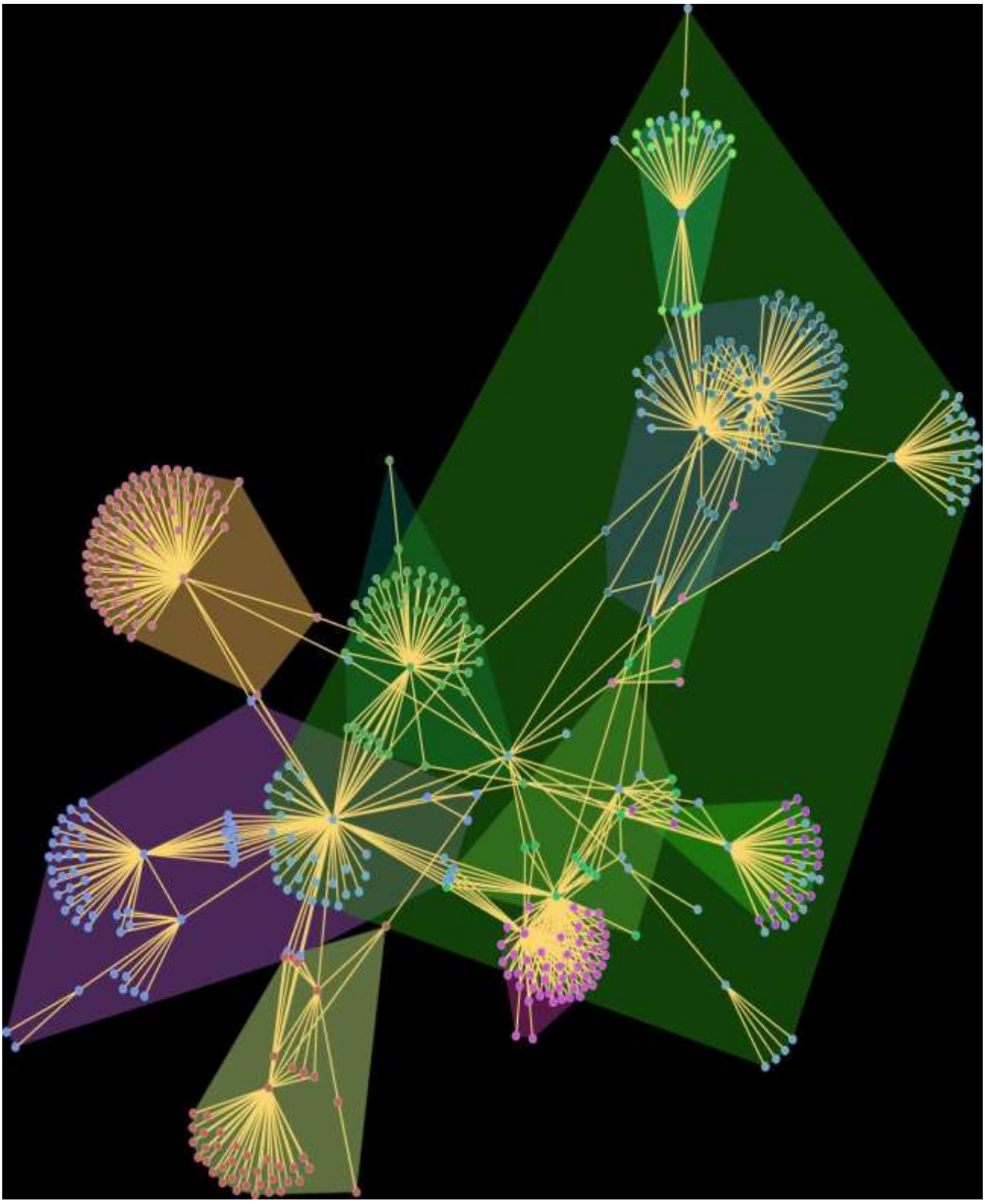


Fig. 4.8 Outcome of Illustrations 3

4.6 Performance Measure & Results

In this chapter we have disserted in brief about GUESS tool and the results we have got. We have compared our BFD algorithm execution speed with Newman's Betweenness algorithm. The speed of execution of any algorithm is one of the crucial measures to rank it. The effectiveness of algorithm is measured on the basis of results we got and what it should be.

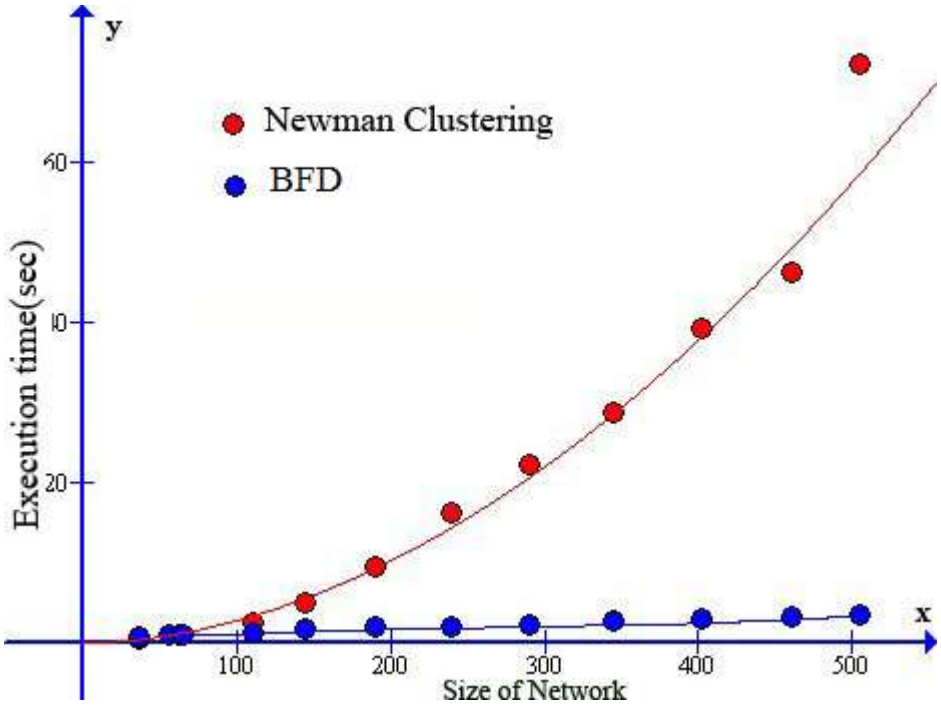


Fig. 4.9 Comparison with Newman's algorithm.

Fig. 4.9 shows graph formed by plotting points among execution speed and size of network. We can summarize the results we have got as:

- The algorithm has linear execution time complexity. Thus has fast execution speed.
- Scalable with increase in network size.
- Doesn't need any external parameter to be set.
- Has simple calculation and process.

In short we can say the algorithm outperforms to mine communities.

CHAPTER V

CONCLUSION AND SCOPE OF FUTURE WORK

In this thesis work we have disserted Social Networks from research point of view and submitted a new algorithm BFD (Breadth first droving) which uses statistical outlook for social mining in Social networks. The algorithm works in as breadth first way covering breadth of community and increasingly finds them from the Network. This algorithm can be scaled for large Social networks and it is very simple as well as fast. The effectiveness of this outlook has been validated using implementation in GUESS tool.

The time complexity of the algorithm is O(V+E) where V represent number of nodes in the vertices and E represent edges of social network. The algorithm doesn't need any parameter to be supplied for its operation like drove size or number (k) as in more other algorithms and it doesn't encompass complex iterative calculation of measures as in cut depend outlooks.

So far we have tested this algorithm with medium sized networks, in future the algorithm can be enhanced to deal with large and dynamic networks of order higher than 10^5. We haven't touched the multi-relationship view of Social Networks, so this idea can be extended to cover it. This idea can also be extended to deal with overlapping communities in social networks by using fuzzy system.

Appendix A

In this appendix we will present python code we have written for social mining, which runs in GUESS tool after loading datasets of Social Network as we disserted in Chapter 4.

Python Script for BFD Social Mining Algorithm:

```
import com
import java
from jarray import zeros, array
from Queue import *
import time
#import sets
#from sets import Set



def BFDrove():
        q=Queue()
        k=0
        udv=0
        dv=0
        max=0
        mcdn=0
        ldv=[]
        ludv=[]
        lst1=[]
        lst2=[]
        s=[]
        time.clock()
        nodes = g.getNodes()
        addNodeField("__visitcounter", Types.INTEGER, Integer(0))
        addNodeField("__visit", Types.INTEGER, Integer(0))
        addNodeField("__cls", Types.INTEGER, Integer(0))
        q.put(nodes[0])                 #TO Enqueue
        nodes[0].__visit=1
        nodecount=nodes.size()
        #__visit=zeros(nodecount,'i')
        while q.empty() == false:
                i=q.get()               #TO Dequeue
                av=i.getNeighbors()
                nav=av.size()
                #print i , ' its neighbours: ',av
                for vi in av :
                        if vi != i:
```

```
                        vi.__visitcounter+=1
                        if vi.__visit==0:
                                vi.__visit=1
                                q.put(vi)
                        if vi.__cls!=0 and vi.__visitcounter>1:
                                ldv.append(vi)
                                lst1.append(vi.__cls)  #.append is same as +=
                                lst2.append(vi.__cls)
                                dv=dv+1
                        elif vi.__visitcounter>1:
                                ludv.append(vi)
                                udv=udv+1
#print '& ldv & lst :', ldv, lst1
#print '& ludv its vCntr & nav,udv,dv',ludv,i.__visitcounter,nav,udv,dv
        if i.__visitcounter >1:
                        udvs=dv+udv
                        if nav%2!=0:
                                nav=nav+1
                        if udvs > nav/2:
                                if udv>dv:
                                #print i,": its Undroved Vertices: ", ludv
                                        ludv.append(i)
                                        k=k+1
                                        ludv.__cls=k          #new class formed
                                        #print 'New Class Formed :'
                                        #for gv in ludv:
                                        #       print gv, gv.__cls
                                elif dv>udv:
                                #print i, ': its list of Droved Vertices',ldv
                                #print ' & list of Class of these Vertices',lst1
                                        for cls in lst1:
                                                if lst1.count(cls)==1:
                                                        s.append(cls)
                                                else:
                                                        j=lst1.count(cls)
                                                        for f in range(j):
                                                                lst1.remove(cls)
                                                        s.append(cls)
                                        #print '& list of unique Class :',s
                                        for m in s:
                                                mx=lst2.count(m)
                                                if mx>max:
                                                        max=mx
                                                        mcdn=m
                                                elif mx==max:
                                                        mcdn=0
                                        for vs in ldv:
                                                if vs.__cls!=mcdn:
                                                        ldv.remove(vs)
                                        ldv.append(i)
                                        ldv.__cls=mcdn
                                        #print 'Class Appended :'
```

```
                                        #for gv in ldv:
                                        #       print gv, str(gv.__cls)
        udv=0
        dv=0
        max=0
        mcdn=0
        #print 'emptying Lists(ldv,ludv,lst,c) :'
        for i in range(ldv.size()):
                ldv.pop()
        for i in range(ludv.size()):
                ludv.pop()
        for i in range(lst1.size()):
                lst1.pop()
        for i in range(lst2.size()):
                lst2.pop()
        for i in range(s.size()):
                s.pop()
for vi in nodes:
        nbrs1=0
        if vi.__cls==0:
                if vi.outdegree!=0:
                        #print 'For node :', vi
                        nbrs1=vi.getNeighbors()
                        if nbrs1.size()!=0:
                            for nbrs2 in nbrs1:
                                if nbrs2.__cls!=0 and nbrs2.__visitcounter>1:
                                                ldv.append(nbrs2)
                                                lst1.append(nbrs2.__cls)
                                                lst2.append(nbrs2.__cls)
                        #       print ' ldv & lst', ldv, lst1
                            for cls in lst1:
                                if lst1.count(cls)==1:
                                                s.append(cls)
                                else:
                                        j=lst1.count(cls)
                                        for f in range(j):
                                                lst1.remove(cls)
                                        s.append(cls)
                        #       print '& list of unique Class :',s
                                for m in s:
                                        mx=lst2.count(m)
                                        if mx>max:
                                                max=mx
                                                mcdn=m
                                        elif mx==max:
                                                mcdn=0
                                for vs in ldv:
                                        if vs.__cls!=mcdn:
                                                ldv.remove(vs)
                                ldv.append(vi)
                                ldv.__cls=mcdn
                        #       print 'Class Appended :'
```

```
                    #       for gv in ldv:
                    #               print gv, str(gv.__cls)
                    udv=0
                    dv=0
                    max=0
                    mcdn=0
                    #print 'emptying Lists(ldv,ludv,lst,c) :'
                    for i in range(ldv.size()):
                            ldv.pop()
                    for i in range(lst1.size()):
                            lst1.pop()
                    for i in range(lst2.size()):
                            lst2.pop()
                    for i in range(s.size()):
                            s.pop()
            #print for remaining undroved vertices do operation
            print 'No. of Droves Formed :',k
            print 'Execution time of Algo. on: ', nodecount, ' nodes and ', g.getEdges().size(), '
                edges is ',time.clock(),' seconds.'
            g.gemLayout()
            clusts = groupBy(__cls)
            for clr in clusts: createConvexHull(clr,randomColor(120))
            colorize(__cls)
```

Appendix B

In this appendix we will present algorithms that we have disserted in Chapter 2.

B.1 Newman Betweenness Algorithm

There general format of their social finding algorithm contains four steps:

1. Calculate betweenness score for all edges in the network.
2. Find and remove the edge having highest score from the network.
3. Recalculate betweenness score of remaining edges after its removal.
4. Repeat from step 2.

$$E = \begin{pmatrix} e_{11} & e_{12} & \dots & e_{1m} \\ & \cdot\cdot & \cdot\cdot & \cdot \\ & \cdot\cdot & \cdot\cdot & \cdot \\ & & \cdot\cdot & e_{mm} \end{pmatrix}$$

Assuming there are m droves. e_{ij} represents edge among drove i and j.

$$Q = \sum e_{ii} - \delta_{random} = \sum e_{ii} - \sum e_{ii}^{rand}$$

E^2 is sufficiently random, therefore:

$$Q = \sum e_{ii} - \sum_i - \sum_k e_{ik} e_{ki}$$

B.2 Cut Clustering Algorithm

The cut clustering algorithm has the following steps:

B.2.1 CutClustering_Algorithm(Graph(V,E),α)

Let V'=V U t

For all nodes v ε V

    Connect t to v with edge of weight α

Let G'(V',E') be the expended graph after connecting t to v

Calculate MinCut tree T' of G'

Remove t from T'

Return all the connected components as the droves of G

B.2.2 HierarchicalCut_Clustering(Graph(V,E))

Let $G^0 = G$

For (i=0;;i++)

Set new, smaller appraises of α_i

Call CutClustering_Algorithm(G^i, α_i)

If ((Drove returned are of desired no. & size) or

(Clustering failed to form non trivial drove))

Break;

Contract Droves to form G^{i+1}

Return all droves at all levels

B.3 Improved Modularity Algorithm

1. Calculate the initial appraises of ΔQij and a_i, and populate the max-heap (H) with the largest element of each row of the matrix ΔQ.

2. Select the largest ΔQij from H, join the corresponding communities, update the matrix ΔQ, the heap H and A_i and increment Q by Qij .

3. Repeat step 2 until only one social remains.

Where

$$\Delta Qij = \begin{cases} \frac{1}{2m} - k_i k_j = (2m)^2 & if \quad i, j \quad are \quad connected \\ 0 \quad otherwise \end{cases}$$

and $a_i = \frac{k_i}{2m}$ for each i.

B.4 External Optimization Algorithm

The heuristic search introduced for modularity optimization proceeds as:

• Initially, they split the nodes of the whole graph in two random partitions having the same number of nodes each one. This splitting creates an initial communities division, where communities are understood as connected components in each partition.

• At each time step, the system self-organizes by moving the node with the lower fitness (external) from one partition to the other. In principle, each movement implies the recalculation of the fitness.

• The process is repeated until an "optimal state" with a maximum appraise of Q is reached. They then delete all the links among both partitions and proceeds recursively with every resultant connected component.

B.5 Force Depend Algorithm

Procedure ForceDepend_CM

volume of $E^3 = .75\ \pi$;

input(N); {N={V,A}, |V|=n, |A|=m}

$$l^0 = \beta.\sqrt[3]{\frac{volume\_of\_E^3}{n}}$$

t=0

loop

errors=0

for $\forall v_i \in V$ **do**

begin

{Calculating repulsive forces}

$$F_r(x_i) = \sum_{v_j \in V} m_i m_j (l^0)^2 . \frac{x_i - x_j}{\left| x_i - x_j \right|^2}$$

{Calculating attractive forces}

$$F_a(x_i) = \sum_{(V_i, v_j \in A)} m_i m_j . \frac{\left| x_i - x_j \right|^2}{l^0} . \frac{x_i - x_j}{\left| x_i - x_j \right|}$$

{Calculating sum forces}

$$F(x_i) = F_r(x_i) + F_a(x_i)$$

{calculating the displacement}

$\Delta x_i = \frac{1}{m}.\gamma.F(x_i)$

{calculating errors}

Errors=errors+ Δx_i

end

{diffusing all vertices}

for $\forall v_i \in V$ **do**

$x_i = x_i + \Delta x_i$

{checking each edge spring}

for $\forall (v_i, v_j) \in A$ **do**

if $| x_i - x_j | \geq \delta_{ij}$ **then**

begin

{broken edge spring}

$A = A - \{v_i , v_j\}$;

$\overline{A} = \overline{A} + \{v_i , v_j\}$;

end

end

for $\forall (v_i, v_j) \in \overline{A}$ **do**

if $| x_i - x_j | \geq \delta_{ij}$ **then**

begin

{reconnected edge spring}

$\overline{A} = \overline{A} - \{v_i , v_j\}$;

$A = A + \{v_i , v_j\}$;

end

end

{continue iteration}

t = t + 1;

until (t ≥ iteration or errors < ε)

output (N);

end procedure

B.6 Link Depend Clustering Algorithm

I. For *i* iterations, repeat {

1. Break the graph into connected components.
2. For each component, check to see if component is a social.
   a. If so, remove it from the graph and output it.
   b. If not, remove edges of highest betweenness, using the Link depend Clustering algorithm for large components, Continue removing edges until the social splits in two.
3. Repeat step 2 until all vertices have been removed from the graph in communities.

}

II. Aggregate the *i* structures into a final list of communities.

B.7 FEC Algorithm

Algorithm FEC(A_0, A)

A_0: Input, the initial adjacency matrix of signed network;

A: Output, the final adjacency matrix of a signed network with identified communities;

1. FC(A_0,A):

   - Calculate the l step transfer probability distribution $\varphi^{(l)}$ for each node i as:

$$\varphi^{(l)} = (P_t^l(i))_{n\times 1}$$

Where $P_t^l(i) = 1 \;.\; I_{\{i=t\}} + \sum_{<i,j>} p_{ij} \;.\; P_t^{l-1}(j) \;.I_{\{i\neq t\}}$,

Where $p_{ij}=\frac{w_{ij}}{d_i}$ where w_{ij} is weight and d_i is the degree of node i.

We can rewrite $\varphi^{(l)}$ as

$$\varphi^{(l)} =_{(t)} (B\varphi^{(l-1)}),$$

Where $B = (B_{ij})_{nxn} = (\max(0, A_{ij}) \Big/ \sum_j \max(0, A_{ij}))_{n\times n}$

- Transform the initial adjacency matrix A_0 into an output adjacency matrix A by sorting the nodes depend on their corresponding appraises in $\varphi^{(l)}$.

2. EC(A, pos):
    - Calculate the degree vector D via a top-down matrix scanning process.
    - Compute the vector λ_1 via a top-down matrix scanning process.
    - Compute the vector λ_2 via a bottom-up matrix scanning process.
    - Compute the vector S depend on λ_1 and λ_2
    - Return the position , with S-appraise being equal to 2.
3. If pos=dim(A) then return A;
4. Divide A at pos such that A={A_{11} A_{12} ; A_{21} A_{22}};
5. FEC(A_{11},A_U);
6. FEC(A_{22},A_L);
7. A=[A_U A_{12} ; A_{21} A_L] and return A.

Procedure for calculating λ_i :

```
λ1= zeros(1,n); /*initialize a zero vector*/
for r=1: n-1  /* r- row index*/
        sum_left=0; sum_right=0;
  for c=1: n-1  /*c- column index*/
        sum_left=sum_left+A_rc
        if c>=r
            sum_right = D(r) – sum_left;
            if sum_left >= sum_right
                λ1(c)  = λ1(c) + 1;
            endif
        endif
  endfor
endfor
```

Appendix C

In this appendix we will present the resources available for researching into the field of Social Networks.

C.1 Social Network Journals

1. **Journal of Social Structure (JoSS)** – JoSS is an electronic journal of the International Network for Social Network Analysis (**INSNA**). It is designed to facilitate timely dissemination of state-of-the-art results in the interdisciplinary research area of social structure, publishes theoretical and methodological articles. JoSS publishes manuscripts that are focused on social structure-on the patterning of social linkages among actors. These could be includes of different types or levels or analysis, namely animals, humans, artificial agents, groups or organizations. Visit: http://www.cmu.edu/joss/.

2. **Social Networks, Elsevier** – An international journal of structural analysis. It provides a common forum for representatives of history, anthropology, biology, social psychology, communication science, sociology, economics, political science and other disciplines who share an interest in the study of the empirical structure of social relations and associations that may be expressed in form of network.
   Visit: http://www.elsevier.com/locate/inca/505596.

3. **Journal of Mathematical Sociology** This Journal publishes articles in all areas of mathematical sociology. It also publishes papers in areas of mutual interest to sociologists and behavioral scientists, and papers which may encourage useful connections among sociology and other manners. Articles dealing with the use of mathematical models in social science, the measurement's logic, computers and applied mathematics, statistics, programming or quantitative method are welcome insofar as they make some contribution to the understanding of substantive social phenomena. Maintained by University of Pittsburgh. Visit: http://www.pitt.edu/~pitpat/jmspage1.htm.

4. **Journal of Artificial Societies & Social Simulation (JASSS)** - is an interdisciplinary journal for the exploration and understanding of social processes by means of computer simulation. Maintained by Surrey University, United Kingdom.

5. **Journal of Computational & Mathematical Organization Theory, Springerlink (CMOT)** - provides an international forum for interdisciplinary research that combines computation, organizations and society. It publishes articles in areas at the confluence of social networks, complexity, machine learning, sociology, business, economics, operations research, artificial intelligence, and political science.

C.2 Social Network Research Organization & Groups

1. **International Network for Social Network Analysis (INSNA) –** This group was founded on the premise that the behavior and lives of social entities are affected by their position in the overall social structure, started in 1977 by Barry Wellman at the University of Toronto, it now has more than 1200 members. Visit: http://www.insna.org/.
2. **Center for Computational Analysis of Social and Organizational Systems (CASOS)** - CASOS research spans multiple disciplines and technologies. Social networks, agent depend models, complex systems, entity extraction, link extraction, dynamic networks, anomaly detection, link analysis and machine learning are among the methodologies used by members of CASOS to tackle real world issues.

3. **Microsoft Research, Social Computing Group** - This group research and develop software that contributes to compelling and effective social interactions, with a focus on user's design processes and rapid prototyping combined with rigorous social science. More projects are made and made public by this group like Netscan, Snarf, Wallop, Personal map, Bridge, online social analysis, etc.

4. **IBM Research, The Collaborative User Experience Group (CUE)** - This Research group spans the Cambridge and Hawthorne research sites and accessibility leaders to focus on the social and technical underpinnings of successful and inclusive business collaboration. Currently CUE is focusing on the interrelated themes of Social Software, Collaboration Environments, Interactive Visualization, and Accessibility. There are more projects going on in this group like: Beehive, Cattail, More Eyes, Malibu Personal Productivity Assistant, History Flow, SNA, etc.

5. **UCI Social Network Research Group –** this group provide an informal setting for dissertion of current and ongoing network-related research at UCI (and elsewhere). facilitate the exchange of information regarding new ideas, tools, data sources, or research. Support graduate student training in the network field; and encourage collaboration among faculty and students on network-related topics. Visit: http://erzuli.ss.uci.edu/network/.

6. **NetLab -** at the Toronto University, studies the intersection of social, information, communication, and computing networks. Visit:http://www.chass.utoronto.ca/~wellman/.

7. **Netwiki -** A Scientific wiki page devoted to research about complex networks and usage of network science & social networks, maintained at North Carolina University. Visit:http://netwiki.amath.unc.edu/Main/HomePage.

8. **VisualComplexity.com -** Online project support research group for anyone interested in the visualization of complex networks. Main goal of the project is to leverage a critical understanding of different visualization ways, across a series of manners, as distinct as Biology, Social Networks or the World Wide Web. Visit:http://www.visualcomplexity.com/vc/index.cfm?domain=Social%20Networks.

9. **FAS.research** - network analysis and visualizations group for business and science using social network analysis. Visit : http://www.fas.at/.

C.3 Social Network Analysis tools

1. **Open Source Tools**
    a. **GUESS -** GUESS is an exploratory data analysis and visualization tool for graphs and networks. The system has a domain-specific inbuilt language called Gython (Python's extension, or Jython) which helps the operators and syntactic sugar necessary for working on graph structures in an intuitive manner. The graphs can be saved in several image formats. An interactive interpreter binds the text that you type in the interpreter to the objects being visualized for more beneficial integration.

    Visit:http://graphexploration.cond.org/.

    b. **Pajek -** SNA software and visualization tool implemented in Pascal for analysis and visualization of networks which is very large. It is available on free cost, for noncommercial use.

    Visit:http://pajek.imfm.si/doku.php?id=pajek.

    c. **InFlow -** InFlow is a software program that is able to perform both social network analysis and network graph generation. The software is designed to highlight main things in the graphs that are generated and is catered to companies looking to harness the power of or control the social networks within their organizations.

    d. **Netvis -** Netvis is an academically and business oriented social network visualization and network analysis interface that is accessed via its website. Because of this, it is fully compatible across platforms and only requires a capable platform to run it. Netvis is a useful tool that can work for both network analysis and network visualization. The software also accepts a reasonable number of file types. In addition, for academic purposes, this site is free.

2. **Others**

a. **UCINET -** A comprehensive package for the analysis of social network data by analytic technologies. This software can handle a maximum of 32,767 nodes of input network size.

Visit:https://www.analytictech.com/ucinet/ucinet.htm.

b. **AiSee -** AiSee is a website that offers network visualization in a nice software package. It is business oriented and produces a variety of graphs and visualization styles. A large number of file formats' information can be import from this software. The graphs can be stored in several image formats.

c. **NetMiner -** NetMiner software does both social network analysis and network graphing. The software is capable of generating a wide variety of very complex graphs and of exploring the network that is generated.

d. **Visone -** Visone provides both social network analysis and network graphing. The focus is mainly on academic organizations and academic uses of social network analysis for research.

C.4 Conferences on Social Network

1. IEEE Conferences:

Consumer Communications and Networking Conference (CCNC) – invites papers on Multimedia usage in social networks & Use of Semantic Web technologies in social networks..

International Conference on Semantic Computing (ICSC) – It invites papers related to semantic social networks.

International Conference on Service Operations and Logistics, and Informatics (SOLI).

Digital Ecosystem & Technologies (DEST).

International Conference on Digital Information Management (ICDIM)

International Conference on Data Mining (ICDM)

International Conference on web intelligence (WI) – Invites research papers on-

i. Social Network Mining

ii. Web Site Clustering

iii. Web 2.0

iv. Link Topology and Site Hierarchy

v. Theories of Small-World Web

vi. Virtual and Web Communities

vii. Web-Depend Cooperative Work

viii. Knowledge Social Formation and Support

ix. Ubiquitous Computing

x. Intelligent Wireless Web

xi. Ubiquitous Learning Systems

xii. Entertainment

(Above are the invite research papers on social networks.)

2. ACM conferences:

I. **International Workshop on Computational Social Networks (IWCSN) –** This workshop accepts papers on foundations of social networks as well as case studies, empirical, and other methodological works related to the computational tools for the automatic discovery of Web-depend social networks. The main points cover the design or use of various computational intelligence software and tools, simulations of social networks, description and analyze of social networks, use of semantic networks in the design and social-depend research issues namely privacy and protection, knowledge discovery, and visualization. Visit: http://www.softcomputing.net/~csn/.

II. **SIGMOD/PODS – This conference invites papers on** Data quality, semantics, integration and mining in large scale social networks. Visit: http://www.sigmod09.org/.

III. **International Conference on Advances in Geographic Information Systems.**

3. **Springer Conferences:**

   I. **International Conference on Computer Mediated Social Networking (ICCMSN) –** this conference invites paper regarding research in social networks like role of network topologies, ways to model the dynamic growth and shrinkage of online communities, privacy, security, and trust issues in on-line communities.

   Visit: http://www.business.otago.ac.nz/infosci/Conferences/Isn2008/.

   II. **International Conference on Knowledge Engineering and Knowledge Patterns (IKAW) -** this conference invites papers about methodologies, models, and tools for Social Networks and Semantic Social Networks.

   Visit: http://ekaw2008.inrialpes.fr/.

   III. **Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)** is a leading international conference in the areas of data mining and knowledge discovery. It invites research paper in data mining in Social Networks.

C.5 Books on Social Network Analysis

1. **Social Network Analysis in Program Evaluation** (by Maryann M. Durland, Kimberly A. Fredericks) This Book highlights social network analysis (SNA) methodology and its usage within program evaluation. This volume explores SNA methodology by first reviewing the foundations and development of network analysis within the social sciences and the field of evaluation. It also includes a personal account of its use by a government agency.
2. **Social Networks Analysis** (by Linton Freeman) this four-volume set is brought together by Linton Freeman. He is founder and editor of the key journal in the field for 28 years, Social Networks. He has congestive the best published work on social network analysis.
3. **Social Network Analysis in Program Evaluation** (by Maryann M. Durland, Kimberly A. Fredericks) this book highlights social network analysis (SNA) methodology and its usage within program evaluation. This volume explores SNA

methodology by first reviewing the foundations and development of network analysis within the social sciences and the field of evaluation.

4. ***Living Networks*** (Dawson, R. (2003), New Jersey, Prentice Hall.) Given the nature of our society and the nature of social networks to exist everywhere. This book, *Living Networks*, aims to reveal how the properties of social networks can assist businesses to expand and flourish modern economy. It seems to deal mainly with methods of leading companies in the modern business world and only provides the very basics of social network theory within its pages.

5. ***Social Network Analysis: A Handbook*** (Scott, J. (1991),. Newbury Park, CA, Sage Publications.) This book highlights the theory and practice of network analysis in the social sciences. It gave a neat and authoritative guidance to the general framework of network analysis, explain the concepts that is very basic, technical calculations and viewing the available computer programs. Both the theoretical basis of network analysis and the key techniques to use it as a research tool by this book's outline.

6. **Introduction to social network ways**: This on-line textbook introduces more of the basics of formal outlooks to the analysis of social networks. This book provides a very basic and brief introduction to the core ideas of social network analysis, and how these ideas are appliance in the methodologies that more social network analysts use.

7. **Six Degrees: The Science of the Connected Age** (Watts, D.J. . New York, Norton) This book is a relatively simple and basic primer for people who are interested in learning more about the rationality and reasoning behind social network analysis. *Six Degrees* attempts to assist readers understand the new and exciting field of networks and complexity. But it is more demanding than a general book *The Tipping Point*, it try to give readers a snapshot of a riveting second in science, when understanding things namely disease epidemics and the stock market seems almost within our reach.

References

[1] Bo Yang, W.K. Cheung, and Jiming Liu, Social Mining from Signed Social Networks, IEEE / KDE, VOL. 19, NO. 10, 2007.

[2] Bo Yang An Autonomy Oriented Computing (AOC) Outlook to Distributed Network Social Mining, IEEE/ SASO 2007.

[3] Ying Zhou, Joseph Davis, Discovering Web Network in the Blogspace, 40th Hawaii International Conference, 2007.

[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.U. Hwang, Complex networks: Structure and dynamics, *Physics Reports* 424, 175–308 (2006).

[5] Bo Yang, D.Y. Liu, Force Depend Incremental Algorithm for mining social structure in Dynamic Network, J. Comp. Sci. & tech., Vol 21, No.3, May 2006,

[6] Rong Qian, Wei Zhang ,Bingru Yang, Detect social structure from the Enron Email Corpus Depend on Link Mining, ISDA, 2006.

[7] Ryutaro Ichise, Hideaki Takeda, A mining way of communities keeping tacit knowledge, IEEE/ICDMW'06.

[8] Mohsen and Hassan, Different Aspects of Social Network Analysis, IEEE/ WI'06.

[9] M. E. J. Newman, Finding social structure in networks using the eigenvectors of matrices 0605087v3, 2006.

[10] Deng Cai, Zheng Shao, Mining Hidden Social in Heterogeneous Social Networks, *LinkKDD'05,* 2005 ACM.

[11] Eric D. Kelsic, Understanding complex networks with social-fnding algorithms, SURF 2005.

[12] P. Pons and M. Latapy, Computing Communities in Large Networks Using Random Walks, Proc. 20th Int'l Symp. Computer and Information Sciences (ISCIS '05), pp. 284-293, 2005.

[13] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, Uncovering the Overlapping Social Structure of Complex Networks in Nature and Society, Nature, no. 7042, pp. 812-816, 9 June 2005.

[14] D.-H. Kim and H. Jeong, Systematic Analysis of Group Identification in Markets, Physical Rev. E, vol. 72, 046133, 2005.

[15] Jordi Duch, Alex Arenas, Social detection in complex networks using Extremal Optimization, cond-mat/0501368 v1, 2005.

[16] M. E. J. Newman and M. Girvan, Fast algorithm for detecting social structure in networks. Physical Review E-69:026113, 2004.

[17] A. Clauset, M. E. J Newman & C. Moore, Finding social structure in very large networks, Physical Review E 70(066111), 2004.

[18] Andreas Noack, An energy model for visual graph clustering, *GD 2003*, pages 425.436.

[19] Gary William Flake, Kostas Tsioutsiouliklis and Robert E. Tarjan, Graph Clustering and Minimum Cut Trees, Internet Mathematics Vol. 1, Number 4: 385-408, 2004.

[20] F. Radicchi, Defining and Identifying Communities in Networks, PNAS, vol. 101, no. 9, pp. 2659-2664, 2004.

[21] M. E. J. Newman, The structure and function of complex networks, *SIAM Review* 45, 167–256 (2003).

[22] B.A. Huberman, J.R. Tyler, and D.M. Wilkinson, Email as Spectroscopy: Automated Discovery of Social Structure within Organizations, C&T '03, pp. 81-96, 2003.

[23] M. Girvan and M.E.J. Newman, Social Structure in Bilogical and Social Networks, PNAS, vol. 99, no. 12, pp. 7822-7827, 2002.

[24] G.W. Flake, Lawrence, Giles, Efficient Identification of Web Network. ACM SIGKDD, 2000.

[25] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner, Using automatic clustering to produce high-level system organizations of source code. In *Proc. IEEE International Workshop (6th)on Program Comprehension (IWPC 1998)*, pages 45.52, IEEE, 1998.

[26] Thomas H. Cormen, Ronald L. Rivest, and Charles E. Leiserson Introduction to algorithms, MIT Press, 1990.

[27] Eytan Adar, GUESS: A Language and Interface for Graph Exploration, ACM CHI, 2006.