COMMUNITY DETECTION IN SOCIAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KORAY ÖZTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2014

Approval of the thesis:

COMMUNITY DETECTION IN SOCIAL NETWORKS

submitted by **KORAY ÖZTÜRK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. Faruk Polat
Supervisor, **Computer Engineering Dept., METU** _____

Assist. Prof. Dr. Tansel Özyer
Co-supervisor, **Computer Eng. Dept., TOBB ETU** _____

Examining Committee Members:

Prof. Dr. Veysi İşler
Computer Engineering Department, METU _____

Prof. Dr. Faruk Polat
Computer Engineering Department, METU _____

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering Department, METU _____

Assist. Prof. Dr. Mehmet Tan
Computer Engineering Department, TOBB ETU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: KORAY ÖZTÜRK

Signature :

ABSTRACT

COMMUNITY DETECTION IN SOCIAL NETWORKS

Öztürk, Koray
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Faruk Polat
Co-Supervisor : Assist. Prof. Dr. Tansel Özyer

December 2014, 105 pages

Today, introduction of social networking applications into every area of our lives makes social network analysis an important research area. Websites and other applications on the internet provides large amounts of data and new research area to the researchers. Also, most of the other data like relationships between people and objects can be presented as social networks. In this work, detecting communities on social networks which is an important subject on social network analysis will be studied. For this, a modified Genetic Algorithm of which chromosome structure and genetic operators are modified to find communities in social networks is used. This modified Genetic Algorithm can be used without giving proposed community number at the initialization and it runs faster compared to other Genetic Algorithm methods. Additionally, we did experiments using Newman's Spectral Clustering Method as a preprocess step and it gave good results.

Keywords: Social Network, Community Detection, Genetic Algorithm, Spectral Clustering

ÖZ

SOSYAL AĞLARDAKİ TOPLULUKLARIN BULUNMASI

Öztürk, Koray
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Faruk Polat
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Tansel Özyer

Aralık 2014 , 105 sayfa

Günümüzde sosyal ağ uygulamalarının yaşamlarımızın her alanına girmesi, sosyal ağ analizini önemli bir araştırma konusu yapmaktadır. İnternet üzerindeki web siteleri ve diğer uygulamalar, araştırmacılara çok miktarda analiz edilecek veri ve yeni araştırma alanları sunmaktadır. İnsanlar ve nesneler arasındaki ilişkileri gösteren günlük hayatımızdaki verilerin çoğu da sosyal ağlar olarak modellenebilmektedirler. Bu çalışmada, sosyal ağ analizinde önemli bir konu olan, sosyal ağlardaki toplulukların bulunması üzerine çalışacağız. Bunun için kromozom yapısını ve genetik operatörlerini, sosyal ağlardaki toplulukları bulmak için özelleştirdiğimiz bir Genetik Algoritma kullanıldı. Modifiye edilmiş bu Genetik Algoritmayı elde edilmek istenen topluluk sayısını başlangıçta belirtmeden kullanabiliyoruz ve diğer Genetik Algoritma kullanan yöntemlere göre daha hızlı sonuç elde edebiliyoruz. Ayrıca, Newman'ın ağlar için Spektral Kümeleme Metodu'nu ön işleme adımı olarak kullandığımız deneyler de yaptık ve iyi sonuçlar verdiğini gördük.

Anahtar Kelimeler: Sosyal Ağ, Topluluk Bulma, Genetik Algoritma, Spektral Kümeleme

To my family...

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Faruk Polat and my cosupervisor Assist. Prof. Dr. Tansel Özyer for their constant support, guidance and friendship.

This work is supported by TÜBİTAK-BİDEB scholarship (2228).

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBRV	Abbreviation
CNM	Clauset et al.
DA	Duch and Arenas
GA	Genetic Algorithm
GGA	Grouping Genetic Algorithm
GN	Girvan-Newman Algorithm
NSA	Newman's Spectral Algorithm
Q	Modularity Score

CHAPTER 1

INTRODUCTION

The concept of social network has become popular after websites like Facebook and Google+ emerged and become a part of our everyday life. The two main properties of the social networks are entities and the relationships within these entities participating in the network. Entities might be "people" and relationships might be the "friendship" of these people like on Facebook and like most of the other social websites but they are not limited to "people" and "friendship". Entities might be entirely different e.g., organizations, websites and relationships might be something else e.g., business, trade, collaboration. Relationships can be all-or-nothing as in Facebook that you are friend with someone or not, or can be a degree as in Google+. Although social networks and their analysis has been a very popular research area in sociology [60][68] for decades, recent revolution on the internet and computer applications have made huge amount of real world data available to analyze and process for researchers. Real world networks can be very large in size, even reaching billion of vertices so there is a need for changing how to handle analyzing and processing networks and a large number new methods have been produced [56][2][58][38][42][51].

In a randomly generated network, the edge distribution is mostly homogeneous and degree of vertices are very similar[49]. However, the degree distributions of real networks are not homogeneous, edges might be denser within some group of entities and might be rarer within other group of entities [19]. This feature of real networks that edges within some specific group are denser, is called community structure [26] or clustering. The entities of a social network naturally fall into communities which the relationships within a community is dense while

the relationships between different communities are rare. This study focuses on finding communities in social networks and we propose a method for detecting communities in social networks.

1.1 Social Networks as Graphs

Graph is a convenient tool for network modeling. If we define a graph as *G=(V,E)* then V is the set of nodes and E is the set of edges that if an edge exists between the nodes V_i and V_j then we can say that nodes V_i and V_j are related to each other and the edge between them is shown as E_{ij}. While modeling social networks as graphs, an entity is modeled as a node and the relationship that connecting two entities are modeled as an edge. Undirected graphs are the best and the most natural exhibition of the social networks, so we will be using undirected graphs in this work to model social networks. In undirected graphs, E_{ij} is same as E_{ji}. Furthermore, in our work, graphs do not include loops and are non-reflexive meaning that nodes are not related to themselves. Also, multiple edges from one node to another do not exist. Order of a graph is number of nodes and size of a graph is number of edges.
We can represent a graph either visually, or with an adjacency matrix A, a V x V square matrix, where nodes are in rows and columns, and numbers in the matrix indicate the existence of edges such as if E_{ij} exists then the value of entry a_{ij} is 1 else 0. For unweighted graphs, all entries are 0 or 1; for weighted graphs the adjacency matrix contains the values of the weights. Since our graph is non-reflexive, diagonal of the A contains only zeros. In figure 1.1, an undirected sample graph with communities is shown.

1.2 Types of Social Networks

Although the most known social network type is friend networks, there are others types of networks that exhibit a good example of social networks.

- Friend Networks : Here, the nodes represent people and the edges represent

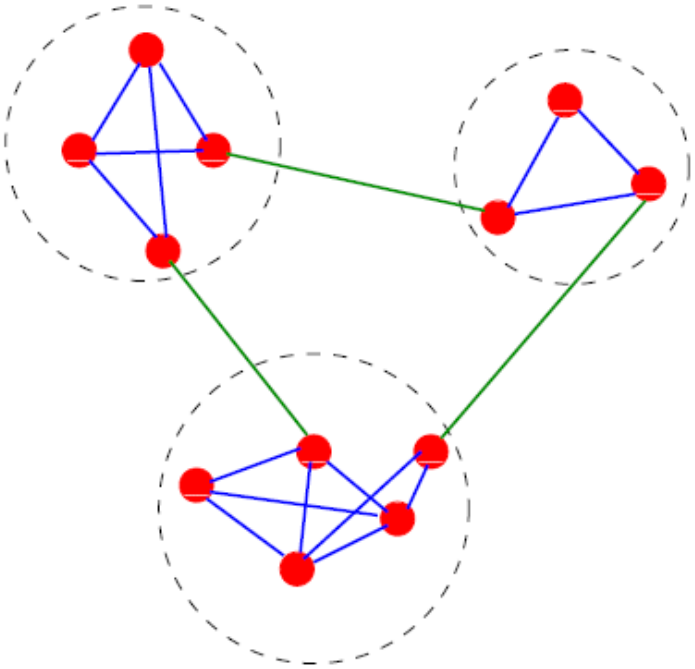

Figure 1.1: A simple graph with three communities. Reprinted figure with permission from [59]

the relationship between them. Edges in these type of networks are usually unweighted that shows they are friends or not. If the strength of the relationship is needed to be shown, weighted edges can be used.

- Telephone Networks : Communications of people over the phone can be modeled as a social network where phone numbers are considered as individuals and represented as nodes. The calls over a time period between the individuals can be modeled as edges for example, phone calls in a company within last two months.

- Email Networks: In this kind of networks, nodes represent email addresses, which are the individual entities. If an edge is formed between the two addresses then, it means that there was at least one email in at least one direction between these two entities. Another way is placing edges if there are emails only in both directions. By doing this, we can prevent from accepting spammers as "friends" with their victims. Furthermore, one approach is to label edges as weak or strong. Strong edges shows existence of email traffic in both directions, while weak edges show the traffic that the communication was in one direction only. The communities seen in

email networks come from the same sorts of groupings we mentioned in connection with telephone networks. Similar sorts of network involves people who text other people through their mobile phones and people who send instant messages to other people using instant messaging software and also people who share files with each other using file sharing software.

- Collaboration Networks: Nodes represent individuals who have done joint works like publishing research papers. Continuing from research paper example, if two individuals published papers together, an edge is established between them. Optionally, we can label edges by the number of joint publications. The communities in this network are authors working on a particular topic. Not only academic works but also relationships in a company or a club or customer-product relationships can be modeled as a collaboration network.

1.3 Properties of Social Networks

This chapter is intended to introduce basic properties and common characteristics of real world social networks. These properties and characteristics are significant because they can guide how to analyze network and how to exploit network structure for certain purposes [42].

The Small World Effect: In a complex network, even if the network has many nodes and a large size, the average distance between any two nodes within the network is short. This property is called the *small-world* effect[69]. This property was first examined in 1960s by Milgram in several experiments [65][39]. In his experiments, randomly selected people from Nebraska are wanted to send letters to a target person in Boston, a far location, who is known only by name, occupation and rough location. The paths of the letters from sources to target destination were kept. Expectation of the letter exchange number in the paths was hundreds, but at the end, average number of exchange in the paths was six. Similar experiment done by Dodds et al. [11] by using e-mails. Lately, experiments done using computer networks and instant message applications had the similar results [34].

Network Transitivity: Another property is network transitivity which is sometimes referred as clustering. This property is the extent to which my friends are friends with one another [68]. In other words, if vertices A and B are neighbors and vertices B and C are neighbors, then there is a huge probability of that vertices A and C are connected too. Transitivity is quantified by defining a clustering coefficient:

$$C = \frac{3 \times number\ of\ triangles\ in\ the\ network}{number\ of\ connected\ triples\ of\ vertices\ in\ network} \tag{1.1}$$

where *connected triples* mean three nodes connected to each other through two edges. The clustering coefficient C has a value between 0 and 1.

Another approach to network transitivity is proposed by Watts and Strogatz [69] who choose a way that uses defining local clustering coefficient values:

$$C_i = \frac{number\ of\ triangles\ connected\ to\ vertex\ i}{number\ of\ triples\ centered\ on\ vertex\ i} \tag{1.2}$$

where $0 \leq C_i \leq 1$.For the situations which denominator is zero, C_i is taken as zero. Clustering coefficient for the whole network is the average of the local clustering coefficient values:

$$C = \frac{1}{n} \sum_i (C_i)\,. \tag{1.3}$$

Since low-degree vertices have smaller denominator in equation 1.2, their addition to the equation is expected to be higher. Plus, equations 1.2 and 1.1 can give slightly different results as it seen in figure 1.2. Although eq. 1.1 seems easy for calculations, eq. 1.3 is the clustering coefficient in use, because it is more convenient for computers and has a wide use in numerical studies and data analysis[42][66]. Studies about clustering coefficient are not limited to triangles, several studies have been done for higher-ordering clustering coefficients [25][22][1][41] which we can call k-clustering coefficient, $k \geq 4$.

Degree Distributions: Degree of a vertex is the number of the edges incident with that vertex. We can formulate the degree k_i of a vertex i in terms of an adjacency matrix A, such that

$$k_i = \sum_{i \in N} a_{ij}. \tag{1.4}$$

In directed graphs, there are two kind of links called outgoing and incoming links and the total degree of the vertex is found as $k_i = k_i^{out} + k_i^{in}$. We are

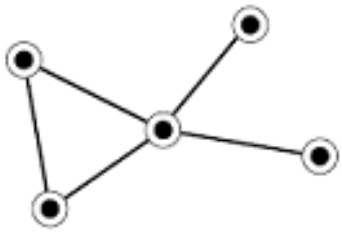

Figure 1.2: A sample network for illustration of the calculation of clustering coefficient. Reprinted figure with permission from [42]. There are one triangle and 8 triples so for eq. 1.1 C= 3 x 1/8 = 3/8. For eq. 1.2, local clustering coefficients of the nodes are 1, 1, 1/6, 0 and 0. We find C= 13/30 for eq. 1.3.

going to deal with undirected graphs so there won't be outgoing and incoming edges. The degree distribution P(k) is the probability of a degree of a randomly chosen vertex being k or the fraction of the vertices in the graph having degree k. Degree distribution of a graph gives important information about topological characterization of the graph [58]. In an undirected network, distribution of the degrees among the vertices can be found by a plot of P(k), or by the calculation of the moments of the distribution. P(k), for a specific moment of n is defined as:

$$< k^n >= \sum_k k^n P(k). \tag{1.5}$$

When the connections between vertices in a network are random, the degree distribution gives all the statistical properties of the network[38].

Community Structure: It is widely accepted that people tend to be grouped in terms of jobs, interests, education, status, vice versa and in social networks, edges between vertices within a group are denser than the edges between the vertices of different groups which is a common property that we call community structure [60][68]. Another definition of the community structure is that: Let G' be a subgraph of the graph G and if the sum of all degrees within G' is bigger than the sum of all degrees toward the rest of the graph, then we can call G' as a community [15]. Extracting the community structure of a network might be called as cluster analysis[14] in some past studies. Data clustering ,which is a method trying to find groups of data located in high dimensional data spaces[32], and network clustering might seem same, solutions for one problem can be adapted for the other one but they are different and should not be

confused with each other.

Betweenness: The shortest path in a graph is the way between two vertices that passing through the least number of edges. When we call a path AB between two vertices V_A and V_B as shortest path, there should be no other path that using lesser number of edges than the path AB. It is a very important notion in graph theory and betweenness property is based on *shortest paths* between the vertices in the graph. The betweenness of a vertex V_i or an edge E_i is the total number shortest paths between the all pair of vertices in the network that passing through the V_i or E_i. This property first introduced in sociology to define social weight of a vertex [20]. If a vertex has a larger betweenness, it has more influence. Betweenness of a vertex can be called as betweenness centrality[21].

Graph Spectra: A graph G consisting of N vertices has N eigenvalues μ_i (i = 1,2,3...) and N eigenvectors v_i (i = 1,2,3...). The spectrum of the graph will be the set of its eigenvalues. Since we use undirected graphs, it has a symmetric adjacency matrix. Therefore, it has real eigenvalues and eigenvectors of distinct eigenvalues are orthogonal. Connectivity properties of a Graph G can be extracted from the normal Matrix, $N = D^{-1}A$ where D is a diagonal matrix and A is adjacency matrix, and Laplacian Matrix, $L = D - A$. All eigenvalues of L are real values and greater than or equal to zero. Because all rows of L sum to zero, first eigenvalue $\lambda_1 = 0$ and its associated eigenvector becomes v_1=1,1,...,1. Therefore, using second smallest eigenvalue, λ_2, is better to cut graph G into pieces. Studies show that if λ_2 is larger, cutting G into pieces is harder [4]. In section 2.5, Newman's Spectral Algorithm [44] using especially graph spectra property is explained in detail.

1.4 Community Detection in Social Networks

Communities are the core structures of the network that individuals or vertices in the same community are connected more densely with each other than with individuals of other communities. Individuals are connected with each other because they just know them or they have common properties, so we can say

that if they are in the same community, they share more common and similar properties. Community detection is significant, because a community might be a small version of whole graph, which shows the very similar characteristics of it. Therefore, examining a few communities might enable us to understand the whole network. This feature is very useful especially when network is a very large real world data.

Community detection has several application areas in the real world. It is beneficial in commercial, security and academic areas. Recommending same products or services to individuals who are in the same community and using community of the individual as a feature for the recommendation systems are two very common and known application types[62]. Another example for the application areas of community detection in social networks is that, in the work of Pinheiro[53], usage of community detection in social networks to reveal the fraud events and other suspicious leakages of money is proposed by generating a network of customers using the text messages and telephone communications between the individuals and identifying community structures. It is stated that unexpected communications between individuals and their type of social structure can enable us the necessary information to find suspicious groups or individuals. For another example in the academic area, we can show that dividing citation network into communities can help researchers who are looking for a cooperation for a specialized field [9][13]. There are also studies to detect hidden criminals in the networks that we do not have any or have too little prior knowledge about individuals' identity [7][48]. In this kind of networks, since there are not much data the to characterize the individuals, the relationships between the entities become important. The relationships in criminal networks in this type of studies are built from several resources like police arrest data, crime location data, kinship or hometown data. This kind of applications make it easy to find suspicious members in the networks. Furthermore, applications to detect terrorist groups in the networks and generating automated techniques to prevent terrorist actions have gained a significant focus after 9/11 attacks at the USA [70]. Not only detecting terrorists, but also analyzing their activities and predicting their moves are important. For these purposes, there are several representations of the networks that some of them show individuals as entities

while others show terrorist activities as entities and if activities are done by the same terrorist organization, it is accepted to be a link between these entities [67].

In the studies related to social networks, the topic of community detection has been discussed largely on the context of block models which are the divisions of the networks into the basic blocks according to some criteria. If we have the block model, we can have communities [57][30] and Genetic Algorithms are one of the well-defined methods that we can use block models, generate best building blocks for the solution and preserve them throughout the solution process.

In this paper, a Genetic Algorithm which is modified for the needs of the community detection in social networks is proposed. A Genetic Algorithm is a search method to find the optimal or the nearest optimal solutions for engineering problems. They are inspired by the evolution process in the nature and Genetic Algorithms' being close to natural evolution is also one of our reasons to use them. In Genetic Algorithms, problems are encoded in a structure which they can be represented best in a computing machine. On the solution space, selection, crossover and mutation processes are applied to generate new solutions and worst ones are eliminated from the solution space [31][28]. Detailed stages of the Genetic Algorithms are explained in section 2.6. Genetic Algorithms especially performs well on combinatorial and mixed problems. They have a less chance to stuck at a local optima when compared to gradient search methods. Genetic Algorithm is one of the stochastic search methods that include methods like simulated annealing and threshold acceptance. However, most of the stochastic search methods run on a single solution for the problem, while genetic algorithms run on a population of solutions [18]. Furthermore, in recent years, the combination of increasing performance and decreasing price of the computing devices and suitability for the parallel programming makes Genetic Algorithms attractive.

CHAPTER 2

LITERATURE SURVEY

In this chapter, I will mention about the some of the state of art works which help my study. First, traditional clustering methods which consist of *Partitional Clustering* using distance measures for clustering, *Hierarchical Clustering* and *Graph Partitioning* aiming to cut graph into pieces will be mentioned. Second Girvan-Newman Algorithm(GN) [27] which introduces a new measurement methodology to evaluate the quality of community structures will be mentioned. Furthermore, algorithms using measurement methodology of GN, which are proven as well functioning for community detection will be mentioned: Duch and Arenas' Algorithm, Clauset's Algorithm and Newman's Spectral Algorithm. Finally, I will talk about genetic algorithms and their specialized use for community detection in social networks.
In the following methods, it is accepted that each individual or node of the network cannot be in more than one community at the same time, belongs to only one community.

2.1 Traditional Clustering Methodologies

2.1.1 Partitional Clustering

The first studies in computer science to find communities of similar objects are based on statistics and data mining. The most significant ones of these old studies use *partitional clustering* methodologies like k-means clustering, neural network clustering and multidimensional scaling [23].

When the edges of the graph have weights or some properties could be used as weights, these weights might be usable as a distance measure, depending on what they represented. Clustering with methods which use traditional distance measures like **K-Means** algorithm [36] are proven to be effective, especially on physical networks like cities on a map. However, when the edges do not weighted, as in a "friends" graph, there is not much we can do to define a suitable distance. Also, the number of cluster to be find must be predefined. Therefore, applying traditional distance-measured methods on social networks is not convenient. K-Means algorithm can be detailed as following: Initially number of communities is given. In first step, we distribute centroids, centers of the communities and their number is proposed community number, on network such that they are away from each other as possible as. Each of the vertices are assigned to the nearest centroid. In second step, the centers of the mass of the each cluster are recalculated and centroids are replaced to the newly found mass centers. After some iterations, locations of the centroids do not change anymore. The solution is not optimal and might depend on the initial choice of the community number and location of the centroids.

2.1.2 Hierarchical Clustering

Hierarchical Clustering is one of the most popular community detection methods for social networks. For a graph with N vertices and its similarity matrix A, hierarchical clustering is as following for general networks:

1. Assign a unique community number to all N vertices so there will be N different communities.
2. Find the closest two communities and merge them into one community.
3. Recompute the similarities between the new and old clusters.
4. Repeat the second and third steps until, all of the vertices put into the same community.
5. Resulting hierarchical tree which is also referred as dendrogram is cut horizontally and final partitions are generated. See figure 2.1.

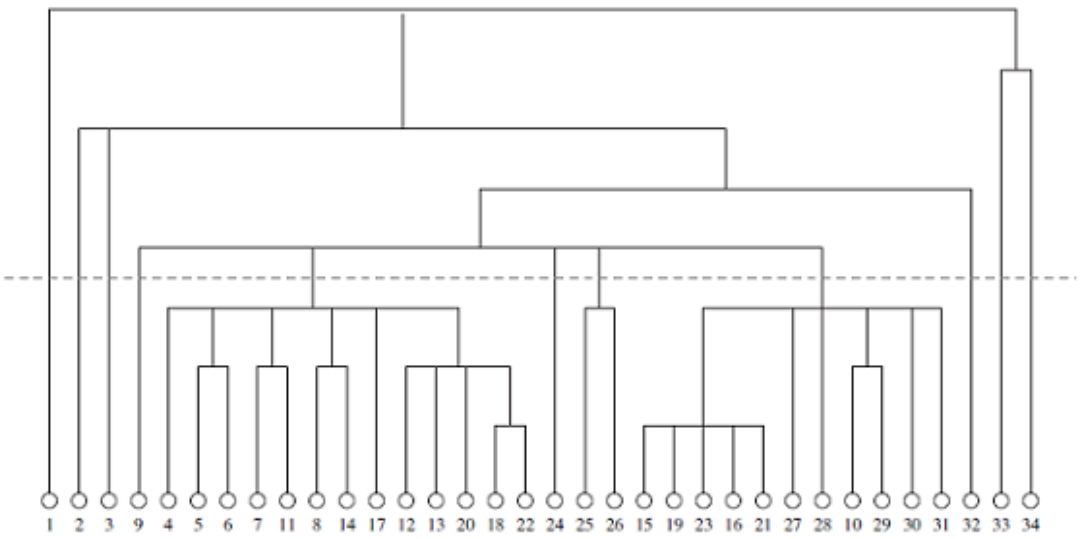

Figure 2.1: Hierarchical tree(dendrogram) of the Zachary's Karate Club.Reprinted figure with permission from [46].

Talking specifically in terms of graphs of social networks, hierarchical clustering of a social-network graph starts by joining two nodes that have an edge between them. Successively, edges that are not between two nodes of the same cluster would be chosen randomly to combine the clusters to which their two nodes belong. The choices would be random, because all distances represented by an edge are the same. Disadvantage of the hierarchical clustering of a graph like seen in Fig. 2.2 is that at after some point we have to choose to join B and D, although they are definitely in different groups. The reason we are likely to combine B and D is that D, and any cluster containing it, is as close to B and any cluster containing it, as A and C are to B. There is even a 1/9 probability

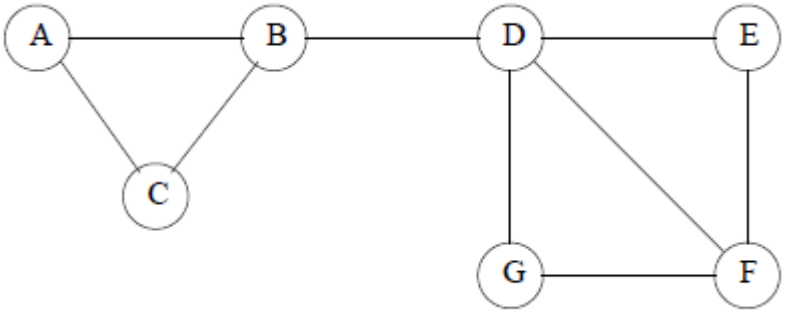

Figure 2.2: Example of a Small Social Network

that the first thing we do is to combine B and D into one cluster. There are things we can do to reduce the probability of error. We can run hierarchical clustering several times and pick the run that gives the most coherent clusters. However, whatever we do, in a large graph with many communities there is a significant probability that in the beginning stages we shall use some edges that connect two nodes that do not belong together in any large community.
To sidestep the shortcomings of the hierarchical clustering method, an alternative approach to the detection of communities is proposed by Girvan and Newman [26]. Instead of trying to construct a measure that tells which edges are most central to communities, least central edges are focused. This term is called as **betweenness**, focuses on the edges that are most "between" communities. The work of **Clauset et. al.**[8] is a variety of Hierarchical Clustering. It tries to optimize Modularity Score Q, which is proposed by [26]. It runs in O(mdlogn) time for a network with n vertices and m edges where d is the depth of the dendrogram.

2.1.3 Graph Partitioning

In computer science, Graph Partitioning is a typical process that divides the network into groups having similar size, while trying to minimize the number of the edges between these groups. Most of the methods for graph partitioning are based on dividing the graphs into two separate groups iteratively: The *spectral bisection* method [17][54] which uses Laplace Matrix of the graph and eigenvectors of it and *Kernighan–Lin algorithm* [33] which tries to optimize community structure over an initial partition of the graph in a greedy way.
Dividing the graph into two subgraphs is the main characteristic of the spectral bisection method but also it may be seen as a disadvantage. If we want to find more than two communities, we need to repeat spectral bisection iteratively on these subgraphs which is not always giving the fulfilling results. Also, deciding where to stop dividing the graphs is important. The spectral bisection method runs in $O(n^3)$ time.
The Kernighan–Lin algorithm which is a specialized approach to spectral bisection, is a greedy optimization algorithm and it tries to maximize a benefit

function. The benefit function is the sum of edges within groups minus the sum of the edges between groups. The Kernighan–Lin algorithm is as following:

1. Start with the initial partition of the graph into two groups. Size of the groups must be predefined. Vertices might be assigned to the groups randomly.

2. Consider all possible pairs of vertices which one vertex is chosen from each groups and calculate the change in the benefit function in case of we swap them.

3. The swap that maximizes the benefit function is chosen and swap is done. Step 2 and 3 are repeated until all vertices in one of the groups have been swapped once.

4. The sequence of swaps that were made are re-examined and the point during this sequence at which Q was highest is found. This is taken to be the bisection of the graph.

The main disadvantage of the Kernighan–Lin algorithm is that we have to choose the sizes of the communities at initial phase. Results highly depends on the initial size and configurations so it is inconvenient for real world datasets. Plus, it suffers the same disadvantages with other spectral bisection methods: Network is divided into two communities and division into more than two communities can be done by iterative processes but we do not where to stop for best division. Later, Kernighan–Lin algorithm is extended that when the number and sizes of communities are not specified, a single node moved to other communities at a time but it also has shortcomings that usually performs poor in time and detection of communities [3][61].

2.2 Girvan-Newman Algorithm

2.2.1 Introduction of Modularity

There are many algorithms that divide networks into communities. Most of them work well on artificially generated datasets and on real world datasets of which communities are known. However, the question of how to evaluate the quality of community structures found by the algorithms arises when we work on real world datasets and don't know the communities before. It is mainly accepted that in a well-defined community structure, edges between individuals are denser. A node must have most of its connections with the nodes which are in the same community and must have none or very few connections with the nodes which belongs to other communities.

In the work of Radicchi et al[15], a quantitative measure for evaluation of communities is proposed. However, it is also stated that quantitative measures are subjective and cannot be exactly accurate for now. Lately, *Modularity Q* concept proposed by Newman and Girvan[27] is accepted as a qualification measure for communities. This measure is based on previous work of Newman which focuses on assortative mixing [45]. Modularity calculation is shown at the following equation:

$$Q = \sum_{i} \left(e_{ii} - a_i^2 \right) \tag{2.1}$$

i is the number of communities, e_{ii} is the fraction of edges to the total number of edges in the network that has both sides inside the community and a_i is the fraction of total number of edges that has at least one side inside the community to the total number of edges in the network. When we are counting edges for a_i, if one side of the edge is inside the community i and the other side is in another community then we count that edge as 0.5. If both sides of the edge were in the community i, then we would count it as 1. After the calculation of modularity, Modularity Q takes a value between -1 and 1. Q values which are closer to 1, have better community structures.

In recent studies, it is claimed that the problem of finding maximum modularity is an NP-Hard problem [5].

2.2.2 A Divisive Algorithm

In **Girvan-Newman Algorithm(GN)**[26][27], removal of most between edges progressively from the original graph is aimed. In this algorithm, *Betweenness* property of the social networks will be used. Betweenness of an edge, is the number of the shortest paths between the vertices x_i and x_j of the graph G such that N is the number of vertices in G, x_i and $x_j \in G$, $0 < i, j <= N$ and $i \neq j$. If there are more than one shortest path between x_i and x_j, then their fraction is taken. While performing GN, first, each node is visited by Breadth First Search and shortest paths from each nodes to other nodes through edges are calculated. If we look at the Fig. 2.2 again, we see that the edge between the nodes B and D is used by the shortest paths 24 times and this edge has biggest betweenness score. To split the community into communities we remove this edge which has the highest betweenness. Then, betweenness of all edges affected by this removal is recalculated. Until no edges remaining, this process is repeated. We can summarize the steps as following:

1. For all edges in the network, find the betweenness score
2. Select the edge which has the highest betweenness and remove it from the network.
3. Recalculate betweenness for all remaining edges.
4. Repeat from step 2.

Slowness of GN is an important difficulty. It runs in $O(n^3)$ time in sparse graph, which is inefficient for networks having more than 10000 nodes. Although it is inconvenient for large networks, several successful versions of Girvan-Newman Algorithm is used in different publications by specializing for datasets used. Holme et al.[50] used it for metabolic networks and Gleiser and Danon [52] used GN to split early jazz musicians into communities that musicians who collaborated are aimed to be in the same community. Guimera et al. [55] also used GN Algorithm by using an e-mail network of a university as dataset.

2.3 Algorithm of Duch and Arenas

In the work of **Duch and Arenas(DA)**[12], a novel method to identify community structures in large complex networks is proposed. The method is based on extremal optimization of the value of modularity. Also, in this work, modularity score Q proposed by Newman and Girvan[26] accepted as measure for defining community structures. In initial step, whole graph is partitioned into two equal communities randomly. In every step, the node with the lower fitness value is transferred from one community to another and then fitness of the nodes neighbor nodes are recalculated. This process is repeated until a maximized Modularity Score Q is reached. After that, all links between two resulted communities are deleted and the same process repeated on each community until Modularity Score Q cannot be improved further.

2.4 Algorithm of Clauset et. al.

The algorithm proposed by Clauset, Newman and Moore [8] is a kind of agglomerative hierarchical clustering method. It also uses modularity score Q proposed by Girvan and Newman [26] as a measure for optimization of community structures. In the beginning of the algorithm each vertex belongs to one of the n communities solely where n is the number of the vertices in the network. Communities are joined together in pairs repeatedly, choosing in each step the join that results in the greatest increase or least decrease in Q. At the end, a dendrogram, tree that show the order of the joins, is produced and cuts through this dendrogram reveals the community structures. According to the level we cut, we can get small and many communities or get large and lesser number of communities. Place of the cut might be the chosen by looking for the maximal value of Q.

After the modifications which are done in the algorithm[8], it runs in O(mdlogn) time for a network with n vertices and m edges where d is the depth of the dendrogram which is a very good performance.

2.5 Newman's Spectral Algorithm

In this section, we examine another approach to organizing social-network graphs. We use some important tools from matrix theory ("spectral methods") to formulate the problem of partitioning a graph to minimize the number of edges that connect different components. Newman's work of "modularity and community structure in networks"[44] is a specialized version of Spectral methods for social network graphs. Given a graph, we would like to divide the nodes into two sets so that the cut, or set of edges that connect nodes in different sets is minimized. To develop the theory of how matrix algebra can help us find good graph partitions, we first need to learn about three different matrices that describe aspects of a graph. The first should be familiar: the adjacency matrix that has a 1 in row i and column j if there is an edge between nodes i and j, and 0 otherwise. We repeat our running example graph in Fig. 2.2. Its adjacency matrix appears in Fig. 2.3.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.3: Adjacency Matrix of Fig. 2.2

The second matrix we need is the degree matrix for a graph. This graph has entries only on the diagonal. The entry for row and column i is the degree of the ith node. The degree matrix for the graph of Fig. 2.2 is shown in Fig. 2.4. For instance, the entry in row 4 and column 4 is 4 because node D has edges to four other nodes. The entry in row 4 and column 5 is 0, because that entry is not on the diagonal.

Suppose our graph has adjacency matrix A and degree matrix D. Our third matrix, called the *Laplacian matrix*, is $L = D - A$, the difference between the degree matrix and the adjacency matrix. That is, the Laplacian matrix L has

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Figure 2.4: Diagonal Matrix of Fig 2.2

the same entries as D on the diagonal. Off the diagonal, at row i and column j, L has -1 if there is an edge between nodes i and j and 0 if not. The Laplacian matrix for the graph of Fig. 2.2 is shown in Fig. 2.5. Notice that each row and each column sums to zero, as must be the case for any Laplacian matrix.

$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

Figure 2.5: Laplacian Matrix of Fig 2.2

We can get a good idea of the best way to partition a graph from the eigenvalues and eigenvectors of its Laplacian matrix. By dealing with the Laplacian matrix, the smallest eigenvalues and their eigenvectors reveal the information that we need for partitioning. The smallest eigenvalue for every Laplacian matrix is 0 and its corresponding eigenvector is [1, 1, . . . , 1]. So that, we use the second-smallest eigenvalue of the Laplacian matrix. We compute the leading eigenvector of this eigenvalue. Vertices are divided into two groups according to the signs of the elements in this vector. Negative vertices are grouped together and positive vertices are grouped together. This simple matrix-based method works even though the sizes of the communities are not specified and is a good way to divide a network into two parts as described. Many networks, however,

contain more than two communities, so the method needs to be extended to find good divisions of networks into larger numbers of parts. First, we use this algorithm to divide the network into two parts, and then divide those parts into two again, repeatedly. if the leading eigenvalue is zero, which is the smallest value it can take, then the subgraph is indivisible. Absence of positive eigenvalue addresses to point where subdivision process is halted.

2.6 Genetic Algorithms

2.6.1 Traditional GA

Genetic Algorithms are a kind of optimization algorithm which imitates the genetic science and natural selection[28][31]. In real world, individuals crossover their genes in which their genetic data are hold, and generates new offsprings. Also, sometimes, a gene of a offspring can be mutated. If the crossovered and mutated chromosomes of the offspring have good genetic data to adapt the environment, then the offspring will survive. Individuals who cannot adapt the environment will be extinct[10]. In GA, instead of building high-performance samplings by trying every conceivable combination of which solution space consists, better samplings can be constructed from the best partial solutions of past samplings. This is referred as the *building block hypothesis* [31]. Genetic Algorithms adopt these approaches and try to generate optimized solutions from a given initial set of solutions by using operators of mutation, crossover and selection. A *fitness function* of which score shows how the individual fit the environment or the solution, is described. After the evaluation phase, offsprings having the best fitness score survive for next generations. Steps of the Genetic Algorithm is as following:

1. Initially, fixed number of chromosomes are generated. Number of the chromosomes are called as *population* and it is given in the beginning. Each of the chromosomes are evaluated through a cost function which is always referred as *fitness function*.

2. Genetic Algorithm produces better chromosomes. These chromosomes are replaced with worst ones because the population needs to be fixed. This step is called as *Selection*.

3. New chromosomes are generated by *crossover*.

4. If a given probability occur, some chromosomes are affected from a *mutation*.

5. If a given probability occur, some chromosomes are affected from an *inversion*.

6. Each newly generated chromosomes are evaluated by fitness function.

7. If aimed fitness score is reached then stop else go to step 2.

Selection is a step that after the fitness function is evaluated for each chromosome, predefined number of chromosomes are chosen from the solution space and these chromosomes are used for the next iteration. We can refer these chromosomes as *survivors*. There are several ways for the selection process. Selecting the chromosomes with better fitness score is one of these ways and we used this method in our study. One of the other selection methods is *Proportional Selection* [31], usually implemented using *roulette wheel* strategy. In this strategy, chromosomes are placed around a wheel and the place a chromosome cover on the wheel is related to its fitness value, as large as its fitness value's ratio over sum of fitness values of all chromosomes. In other words, the chromosomes are selected randomly but the chromosomes with larger fitness values have a higher chance for selection. Not being a guarantee for the selection of the best chromosome is the disadvantage in this strategy. Another method for the selection is *Tournament selection*, in which tournaments, comparison of the fitness values, are run between the randomly chosen a few chromosomes. Tournaments continues until proposed survivor number is reached [6].
Crossover is the core process of the GA. Two chromosomes from the solution space are chosen randomly and one crossover point or crossover points, depending on the crossover type we choose, is determined on each chromosome. In one-point crossover, one point on each chromosome is selected and chromo-

somes are divided into two pieces. Pieces with same location and size of the chromosomes are exchanged within them and new offsprings are generated. In two-point crossover, chromosomes are cut from two points and middle pieces of the chromosomes are exchanged. In uniform crossover, some genes of the one parent's chromosome are chosen which is also referred as crossover bits, and these crossover bits are exchanged with other parent's genes. See Fig. 2.6.

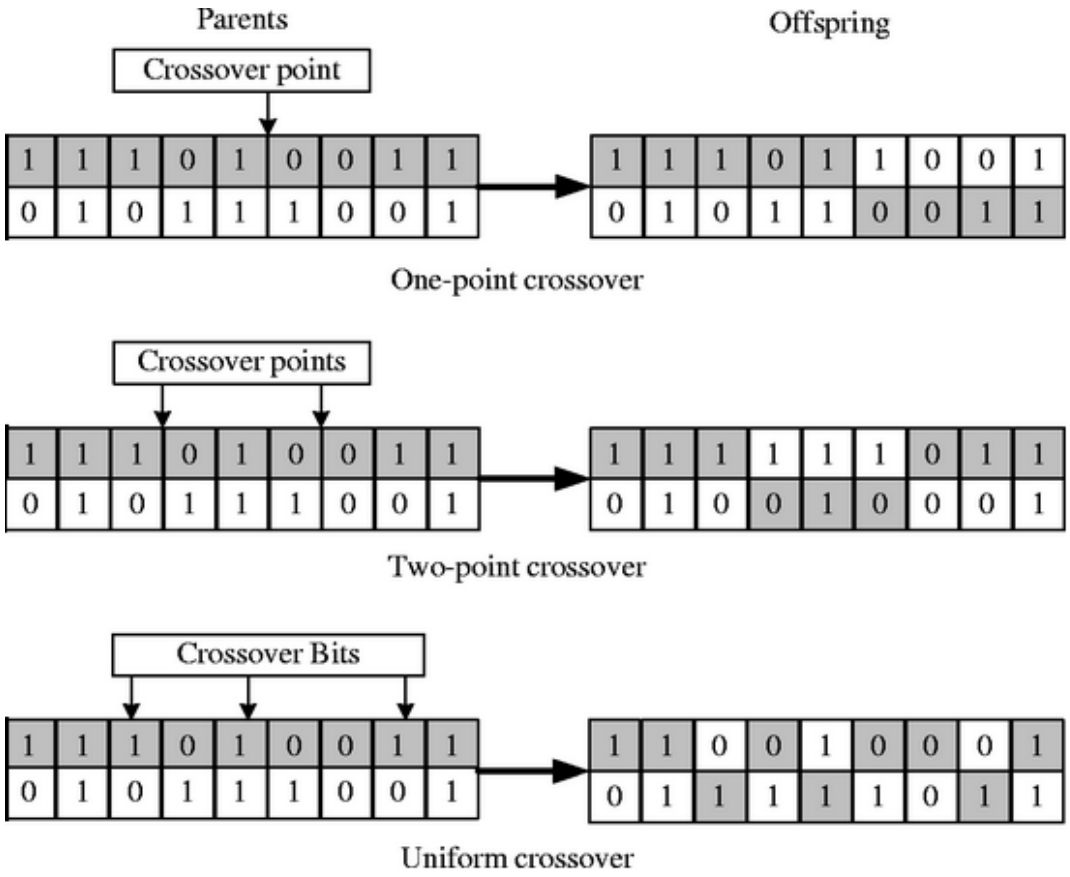

Figure 2.6: Crossover in GA

Mutation is a probabilistic operation. If a predefined probability occurs then mutation on a randomly selected chromosome occurs and values of the mutated genes changes. Mutation might be on a single random gene or might be on several random genes. In Fig. 2.7, a mutation example is shown.

Inversion operator is not always applied on real datasets. It does not change the information on the genes, it changes the presentation of the information on these genes. However, it might be important in some cases since position of the gene on different GA operations can cause different offspring information.

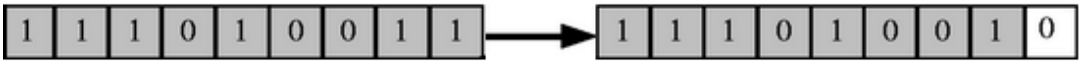

Figure 2.7: Mutation in GA

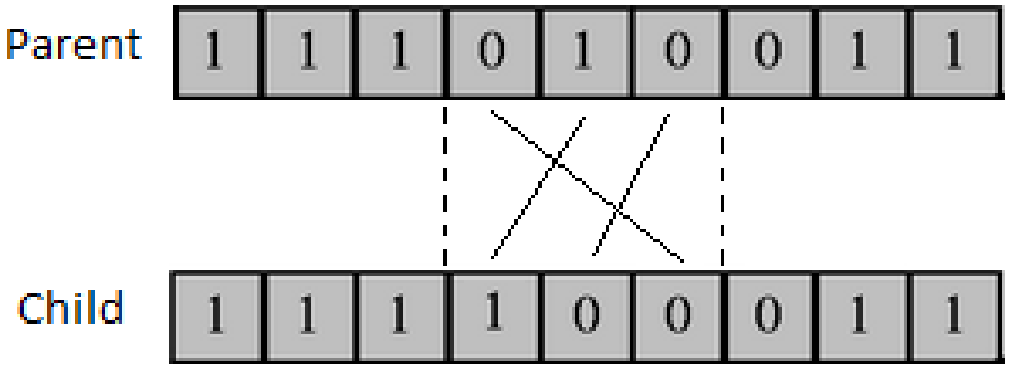

Figure 2.8: Inversion in GA

Mutation and inversion operators should not be applied on every iteration and on every chromosome. There are some ways to choose whether apply mutation or not. One of them is using probability: if a randomly generated value is bigger than the predefined mutation rate than we apply mutation operator. Another way is that if best chromosomes in a solution space do not change until a predefined threshold for number of iterations is reached. When the threshold is reached, the mutation is applied.

2.6.2 Falkanuer's Grouping Genetic Algorithm

There are more specialized techniques focusing on other grouping and clustering problems using Genetic Algorithms(GA). **Grouping genetic algorithm (GGA)**[16] is an evolution of the GA where the focus on the individual items, like in classical GAs, is shifted to groups or subgroup of items. The idea behind this GA evolution proposed by Emanuel Falkenauer is that developing solutions to some complex problems, such that clustering or partitioning problems where a set of items must be split into disjoint group of items in an optimal way, would

better be achieved by making characteristics of the groups of items equivalent to genes. Examples of these kind of problems include bin packing, line balancing, clustering according to a distance measure, equal piles, etc., on which classic genetic algorithms are proved to perform poorly. Making genes equivalent to groups implies chromosomes that are in general of variable length, and special genetic operators that manipulate whole groups of items.

2.6.3 Grouping Genetic Algorithm of Tasgin et. al.

Genetic Algorithms can be adapted for many optimization problems. Community Detection in networks is one of these problems which tries to find best community structure in a network by maximizing network modularity. In the work of Mürsel Taşgın and Haluk Bingöl [63], community detection using genetic algorithms is studied. In their work, network modularity equation proposed by Newman is used as **fitness function**, rewritten from Eq. 2.1:

$$Q = \sum_{i} \left(e_{ii} - a_i^2 \right) \tag{2.2}$$

In Eq. 2.2, i is the number of communities, e_{ii} is the fraction of edges to the total number of edges in the network that has both sides inside the community and a_is the fraction of total number of edges that has at least one side inside the community to the total number of edges in the network. In **initialization step**, as it is seen in Fig. 2.9, for the purpose of diversity, all the community IDs of every member in the population is initialized to a random number. The random number is limited with the number of nodes in the population, namely n. Theoretically, in the worst case, every node will fall into separate community and there will be n communities in the network. In initialization phase a predefined variable called randomization rate is defined. Random vertices are chosen according to this randomization rate and if the randomization is provided then the neighbors of the randomly chosen vertex are assigned the same community ID. In **Crossover**, genes are not exchanged simply, instead transferred the community identifiers of nodes in a community to nodes in the destination chromosome. As community structure is a relational property and different community identifiers in different chromosomes may mean the same community.

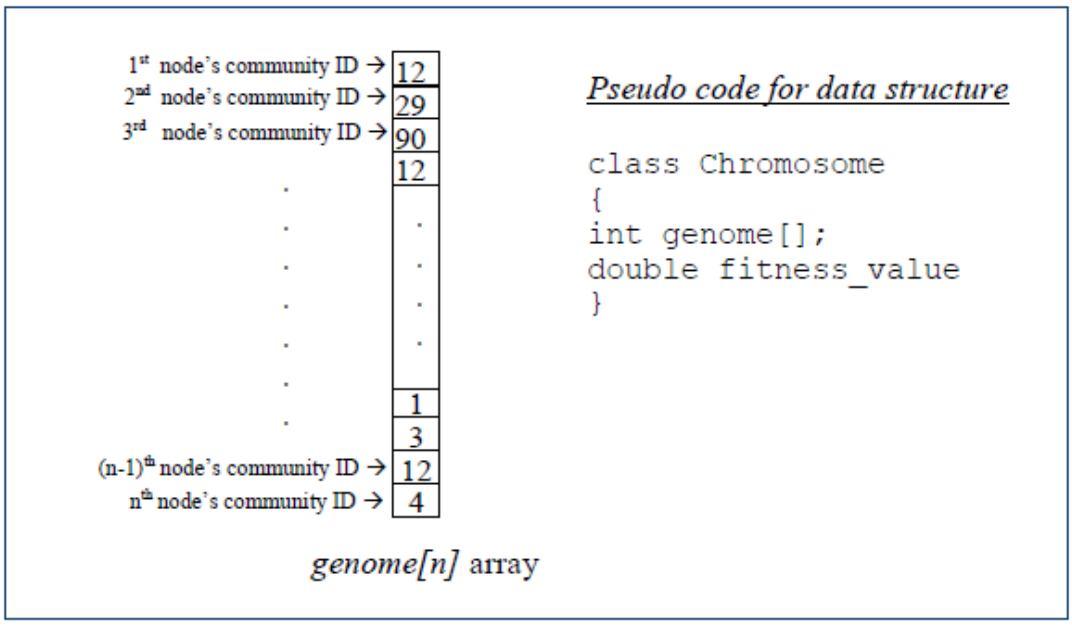

Figure 2.9: Presentation and initialization of chromosomes. Reprinted from [63]

For example, communityID=1 in solution member A and communityID=34 in solution member B have identical members, while communityID=1 in solution B has nothing to do with communityID=1 in solution A. For the reason that I have mentioned above, one way crossover is applied. Figure 2.10 explains it.
In **mutation**, node is placed into a random community in the network. This is done in this way: a random node is chosen and a new cluster number is generated by modfying a digit in its binary representation.
Also, a new mechanism called Clean-up, which is based on a new metric named community variance, aims to reduce all such misplacements, is added as a next step. It is needed to reduce the number of misplaced nodes. If the number of such misplacements is high, it is detected by the mechanisms of genetic algorithm via fitness evaluation. However, although the overall fitness value is good for a community split, there may be a small number of misplaced nodes that does not affect the overall fitness value very much. Tests of the algorithm is claimed to be giving accurate results but made only on small datasets: Zachary's Karate Club Network [71] and American College Football Network [26].

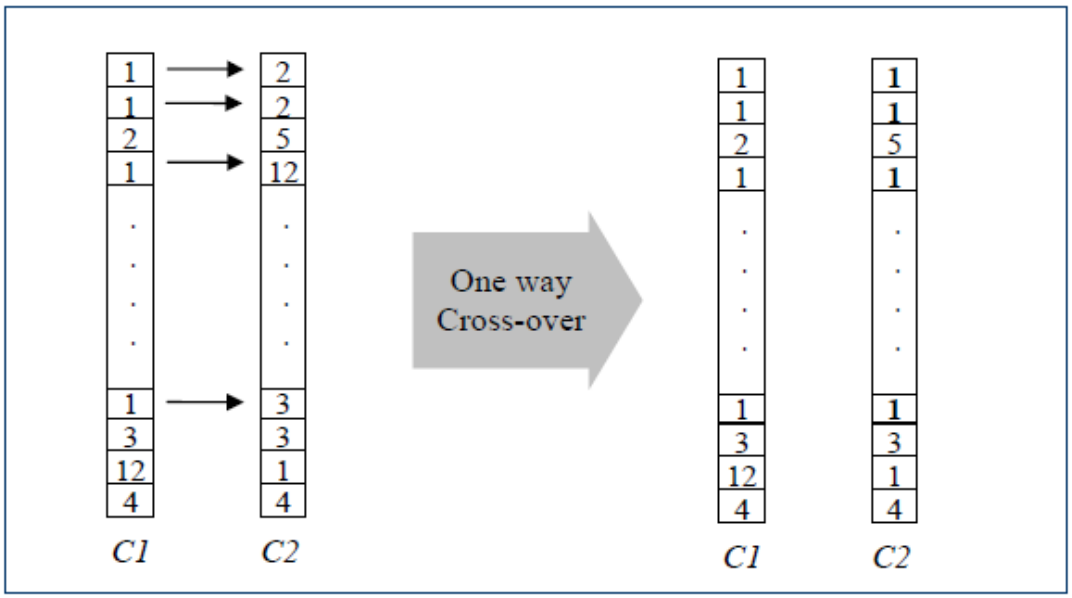

Figure 2.10: One-way crossover performed by the method of M. Tasgin et. al. Reprinted from [63]

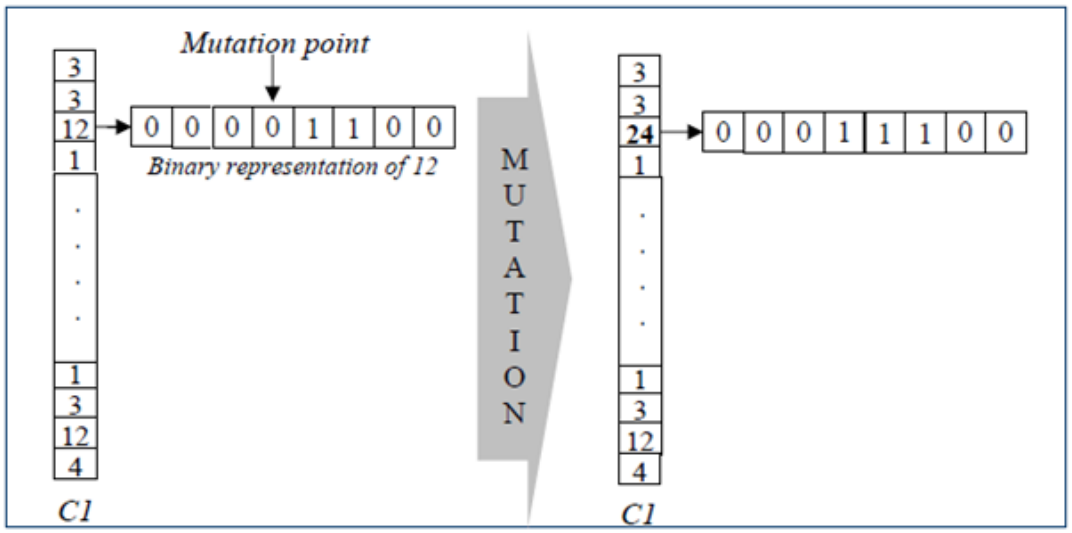

Figure 2.11: Mutation performed by the method of M. Tasgin et. al. Reprinted from [63]

CHAPTER 3

OUR METHOD

In this chapter, our method for community detection is introduced. A Genetic Algorithm which is specialized for detecting communities in social networks is used. For large real world datasets, a spectral method presented by Newman [43] is used as a preprocess for the initialization phase of the genetic algorithm. Entities in the network will be presented as vertices and relationships between the entities will be presented as edges. An adjacency matrix will be used to show nodes and edges. In our method, all vertices are accepted to be only in one community. In other words, overlapping communities [24][47] will not be considered.

As our algorithm is a genetic algorithm, it tries to find optimal solution by maximizing the fitness value. In the following sections, the steps of the algorithm, operators used in the genetic algorithm and how they are customized for the special needs of our algorithm and how all these steps are implemented will be explained.

3.1 Encoding and Initialization

Encoding of the genetic algorithm is inspired by the Grouping Genetic Algorithms (GGA) which is proposed by Falkenauer[16]. Our focus is shifted from individuals of the network to communities of the network. In our encoding, communities are represented as the genes of the chromosomes. Every gene in the chromosome represents a community. The data that is held in the genes contains vertices belonging to that community. During crossover and mutation

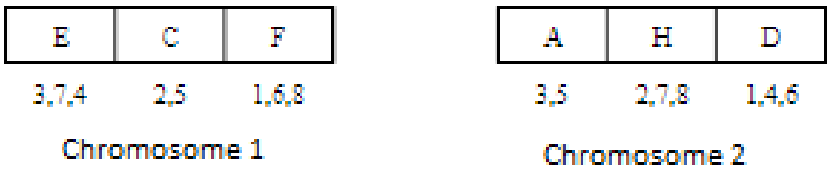

Figure 3.1: Example of Encoding

processes, vertices can change their communities. If a vertex is moved from one community from the other, data of the communities is updated. In Fig. 3.1, we see an example encoding of our specialized GGA. Chromosome 1 is consists of three communities which are shown as genes, and the data held in genes contains their vertices. In chromosome 1, vertices with numbers 3,7,4 belongs to one community shown by gene E, 2 and 4 belongs to another community shown by gene C and 1,6,8 belongs to other community shown by gene F. Chromosome 2 is also similar, gene A has individuals 3, 5 and gene H has individuals 2,7,8 and gene D has individuals 1,4,6.

In *initialization* step, communities are generated and the vertices are assigned. If we use preprocess step, communities coming from the preprocess step are generated and the vertices belonging to these communities are assigned. Vertices of which community info have not come from preprocess step will be assigned to newly generated random communities. For each vertex that does not belong to a community, a new community is generated and the vertex is assigned to that community. In the case that we do not use the preprocess step, preprocess step can be skipped when size of the network is small, n communities are generated that n is the number of vertices in the network and each sole vertex is assigned to a sole community that no more than one vertex is assigned to same community. For randomly generated communities, some vertices are chosen, of which number proportional to the number of randomly generated communities, and if they have connection with a vertex of which community also randomly -not from the preprocess step- selected then it is moved to its neighbor's community. By this way, first iterations are planned to be evolve rapidly.

Before running the program, some variables must be defined. *Population* is the number of chromosomes that must exist through the iterations. In initialization step, we define chromosomes as much as the population, which is predefined.

Mutation rate is the probability of whether a chromosome will be applied to a chromosome or not.

3.2 Fitness Function

As fitness function, the network modularity equation proposed by Newman et al. [44] is used, which is explained in section 2.2 in detail. It is shown in Eq. 3.1 that Eq. 2.1 is rewritten and also used in the work of M.Tasgin et. al. [63].

$$Q = \sum_i \left(e_{ii} - a_i^2 \right) \tag{3.1}$$

i is the number of communities, e_{ii} is the fraction of edges to the total number of edges in the network that has both sides inside the community. If we talk in terms of the community i, total number of the edges of which both sides belong to the vertices which are the members of community i is divided to the total number of edges in the network, and that gives us the e_{ii}. a_i is the fraction of total number of edges that has at least one side inside the community to the total number of edges in the network. Again, if we look at the community i, let define a variable m_i that m_i is the total number of edges that has at least one side inside the community and define a variable E that E is the total number of edges in the network. Initial value of m_i is set to zero and we calculate m_i as following: all the edges of which both sides belong to the vertices which are the members of community i increases the m_i by one and all the edges of which only one side belongs to a vertex in community i increases the m_i by 0.5. Finally, we divide m_i to E, m_i/E, and find a_i. After we find all the modularity scores for each community, their summation gives as the modularity score of the network, Q.

$$Q = \sum_i Q_i \tag{3.2}$$

In Eq. 3.2, Q_i is the modularity score of the community i and the summation of the modularity scores of each community produces the Q, the modularity score of the network.

After each iteration, fitness score of each chromosome is recalculated and the chromosomes having higher modularity score Q are advantageous for the survival for next iteration.

3.3 Crossover

Crossover is done between two randomly selected chromosomes and after the crossover, an offspring chromosome is produced. Our crossover method differentiate from traditional crossover operators: Communities will be exchanged between the chromosomes. Selecting two parents from the solution space, two new members, which we can call as children or offsprings, are generated by inserting the selected communities of the first parent into the second one, and by doing the same process in the reverse way that crossover from second parent to first one.

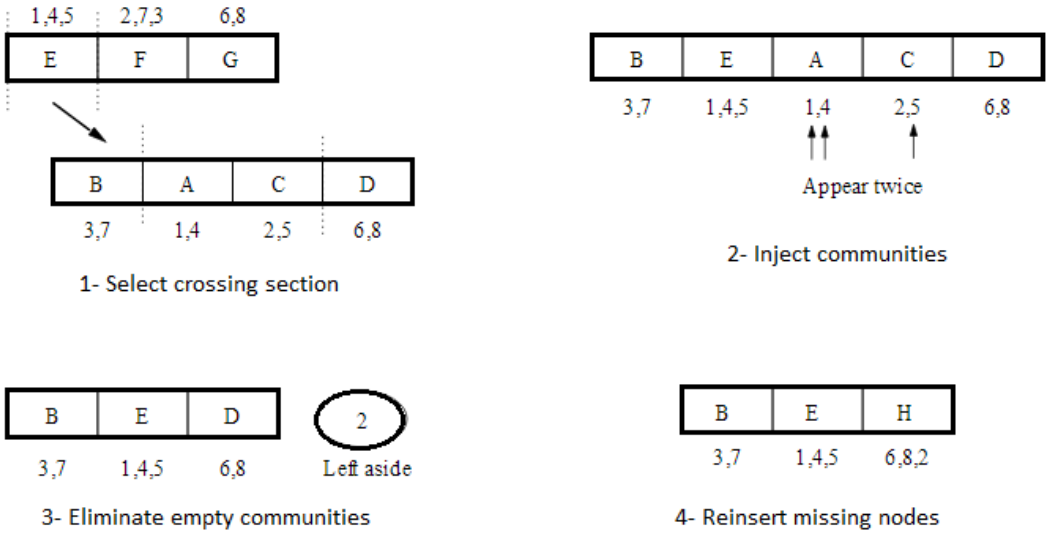

Figure 3.2: Crossover

The crossover consists of four steps:

1. Two cutting points are selected on each parent randomly and the part between these cutting points is chosen as crossing section.

2. The communities, which are the genes of the chromosomes, are chosen by the crossing section of one parent and they are put at the crossing section of the other parent. At this point, there is a possibility that some of the vertices may be seen in more than one communities of the chromosome.

3. The communities which are already existing in second parent and containing vertices that are already in the inserted communities are eliminated,

for example, some vertices are not belonging to any community anymore. If some communities become empty after this elimination process, they are deleted from the solution, too.

4. The vertices left aside are reassigned to the communities and by this way, they are reinserted into the solution. While reinserting, according to a predefined probability rate, the vertex is put in a community randomly or put in a community in which it will increase the network modularity of that community the most or decrease it least.

3.4 Mutation

Aim of the mutation operator is to add new members with new characteristics into a population to increase the search space of the Genetic Algorithm. For our proposed method, randomly a few communities are selected according to a predefined probability rate and the communities are eliminated from the chromosome. After the elimination, there will be vertices which do not belong to any community. The vertices are put in a community randomly or put in a community in which it will increase the network modularity of that community. If a predefined random value for vertex reinsertion is met then the vertex will be put in a community to increase the modularity of that community. If the predefined random value is not met or there are not any community that increase its modularity, then the vertex put in a community randomly, or put in a newly created community.

In Fig. 3.4, mutation operator of our method is shown: Genes A and D are selected randomly from the chromosome and they are eliminated. Now, since the vertices belonging to these communities do not belong any community, these vertices are reinserted to other communities. With the probability p_1, vertices put in a community in which have more neighbors and if we cannot find a community in which vertex have neighbors, we generate a new community with the probability p_2 or put in a community randomly with the probability $1-p_2$. With the probability $1-p_1$, they put in a community randomly.

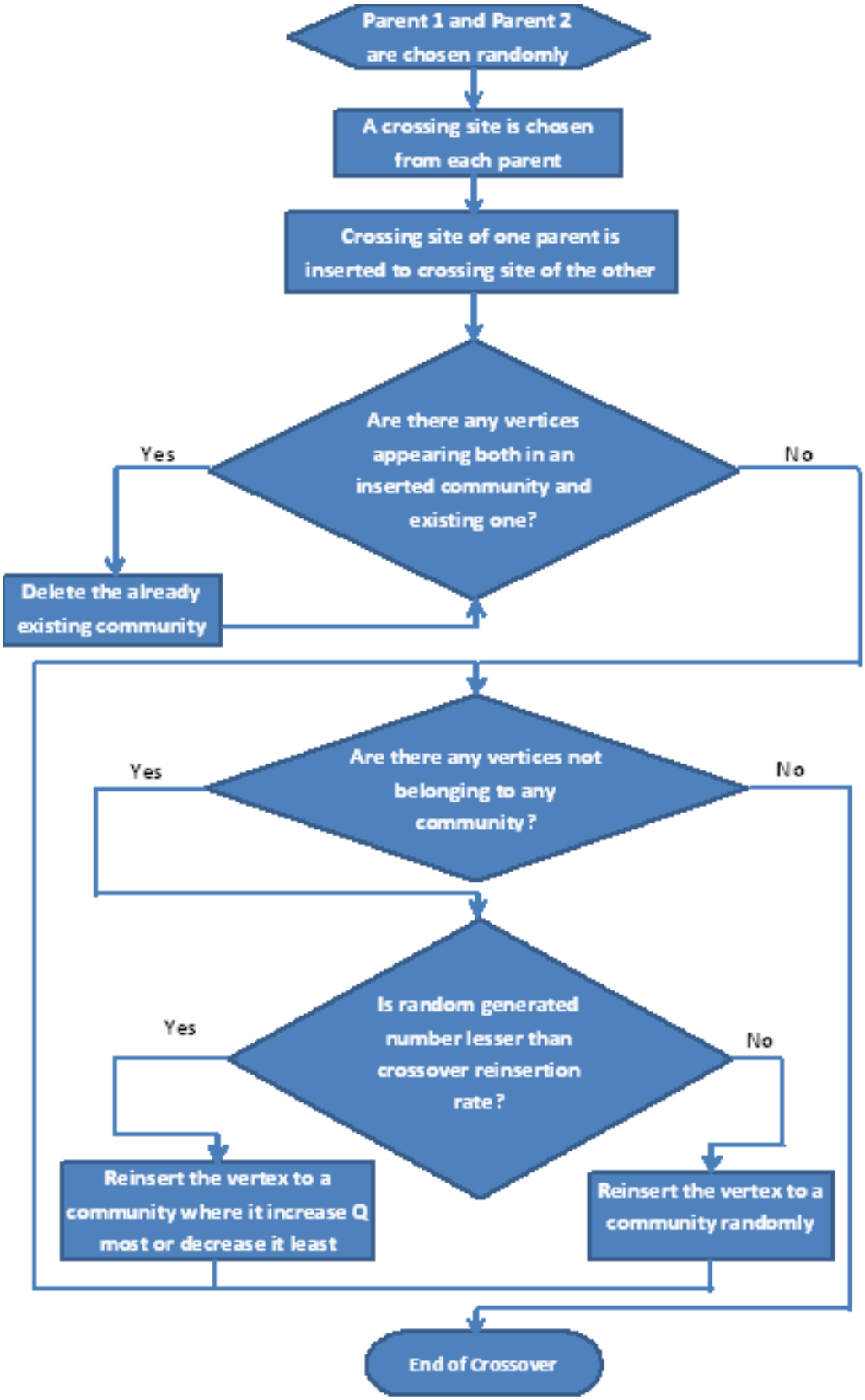

Figure 3.3: Diagram of crossover step

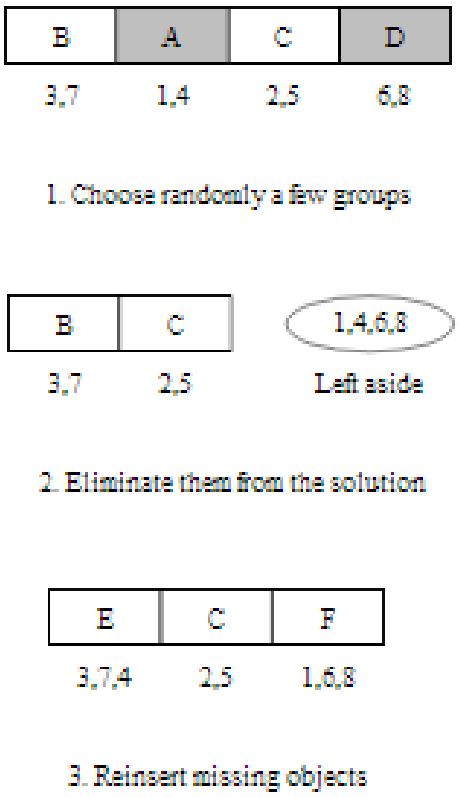

Figure 3.4: Mutation

3.5 Selection

After the generation of new offsprings which are produced by applying crossover and mutation operators to the chromosomes, and their insertion to the solution space, chromosome number exceeds the predefined population. Chromosomes must be chosen at the number of predefined population to survive for the next iteration. To decide which chromosomes will survive for next iteration, first, fitness value of all chromosomes and newly generated offsprings are evaluated, and then they are sorted in decreasing order according to their fitness values. We take the chromosomes with the best fitness values at the number of the predefined population, and use them for parents at the next iteration. Chromosomes which cannot survive after this selection process are eliminated from the solution space. This selection can be referred as *elite selection*, only selecting the chromosomes with best fitness values.

3.6 Preprocess

We expect our algorithm being satisfactory and efficient for the small datasets of which number of vertices not larger than 1000. However, for large real world dataset, definition of the communities in the initialization step can make a significant difference. Starting the algorithm with the well-defined community structures and doing operations on these well-defined structures help the evolving of the community structures of the network rapidly and accurately, as it is mentioned in the *Building Block Hypothesis* [31][30]. Therefore, we decided to use other algorithms which are proven to be giving good results, which means producing higher network modularity scores, for detecting communities in a network. We use several of the communities produced from these algorithms as building block inputs.

Spectral Algorithm for partitioning graph of a social network [44], and the Girvan-Newman Algorithm [26] are used as a preprocess step. These methods are run half-of-initial-chromosome-number times. The network modularity functions used by these algorithms and our specialized genetic algorithm are the same. Communities generated by this step are used as initial communities for the genetic algorithm. We do not use all the community structures produced by these communities that if the size of the community is bigger than one of third of the network size it is not used as an input because it can dominate the results. Also, when getting inputs, overlapping communities are not used, every vertex must be in a sole community.

3.7 The Algorithm

The algorithm we use is the combination of the preprocess step and our genetic algorithm which is specialized for the needs of the community detection in social networks. As we mentioned in section 2.2, fitness function is the network modularity score Q. Also, some predefinitions and variables need to be explained:

- N is the population.
- p_{cr} is the probability for the randomness to reinsert vertices after crossover. If randomly generated number is smaller than the p_{cr}, then idle vertices are reinserted randomly, else they inserted to a community where they have more neighbors.
- p_m is the probability for the mutation. If a randomly generated number is smaller than p_m, then mutation is applied
- p_{mr} is the probability for the randomness to reinsert vertices after mutation. If randomly generated number is smaller than the p_{mr}, then idle vertices are reinserted randomly, else they inserted to a community where they have more neighbors.
- Q_F is the planned final modularity score. If it is reached, program terminates.
- I_F is the planned final iteration count. If iteration count reaches to it, program terminates.

Now we can pass through the steps of the our algorithm. As a final Steps are in order as following:

1. Preprocess: Dataset is run in several algorithms.
2. Initialization: Community structures that are output of the preprocess step are used as input for the initialization. Vertices that are not assigned to a community after preprocess, are assigned to newly created communities.
3. Evaluation: Evaluate the network modularity score of the all chromosomes.
4. Selection: Select best N chromosomes for the next steps. Remove the others.
5. Crossover: Crossover is applied $N/2$ times to randomly selected pair of chromosomes. For the probability of p_{cr}, reinsert idle vertices to commu-

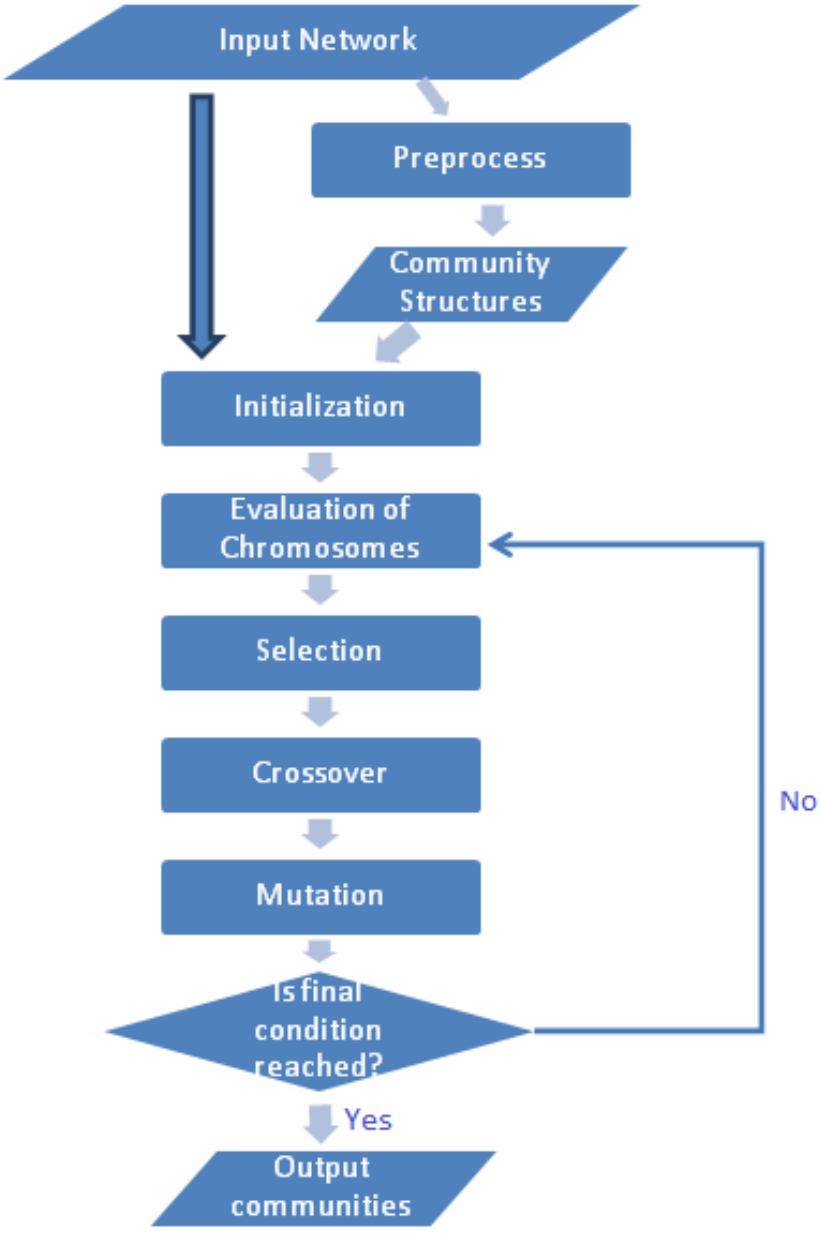

Figure 3.5: Diagram of the Algorithm

nities randomly, else reinsert them to communities where they have largest number of neighbors.

6. Mutation: Mutation is applied to each chromosome with a probability of p_m. For the probability of p_{mr}, reinsert idle vertices to communities randomly, else reinsert them to communities where they have largest number of neighbors.

7. Control: If Q is reached to Q_F or iteration count is reached to I_F then program terminates, else continue from step 3, evaluation of the modularity scores of all chromosomes.

In Figure 3.5, you see a diagram that showing the processes of our method. As you see, initialization step does not get all of inputs from the preprocess step, it also gets vertices of which communities not decided.

CHAPTER 4

EXPERIMENTS

I run the method proposed in this theses on several datasets. First, I will run the algorithm without preprocess step on small datasets to see the accuracy of the genetic algorithm proposed in this thesis. Second, I will compare results of genetic algorithm with other studies. After that, I will do experiments to larger datasets with preprocess step is involved. And then, I will do comparison with other methods again.

4.1 Datasets

In this section, the datasets that is used through the experiments are introduced and their main characteristics are explained.

Zachary's Karate Club [71] is a very popular network used by many researchers. It shows a university karate club which is divided in two communities after a conflict. The network consists of 34 vertices and 78 edges. Figure 4.1 shows the Zachary's Karate Club Network after the run of our modified Genetic Algorithm which is proposed in this thesis. Vertices belonging to the same community colored with the same color. As it seen genetic algorithm works fine on this small dataset and divides it to two communities as expected.

Collaboration in Jazz Network Dataset [52]: The data were obtained from The Red Hot Jazz Archive digital database. The data include 198 bands that performed between 1912 and 1940, with most of the bands performing in the 1920's. In this case each vertex corresponds to a band, and a link between two bands is established if they had at least one musician in common. The network consists

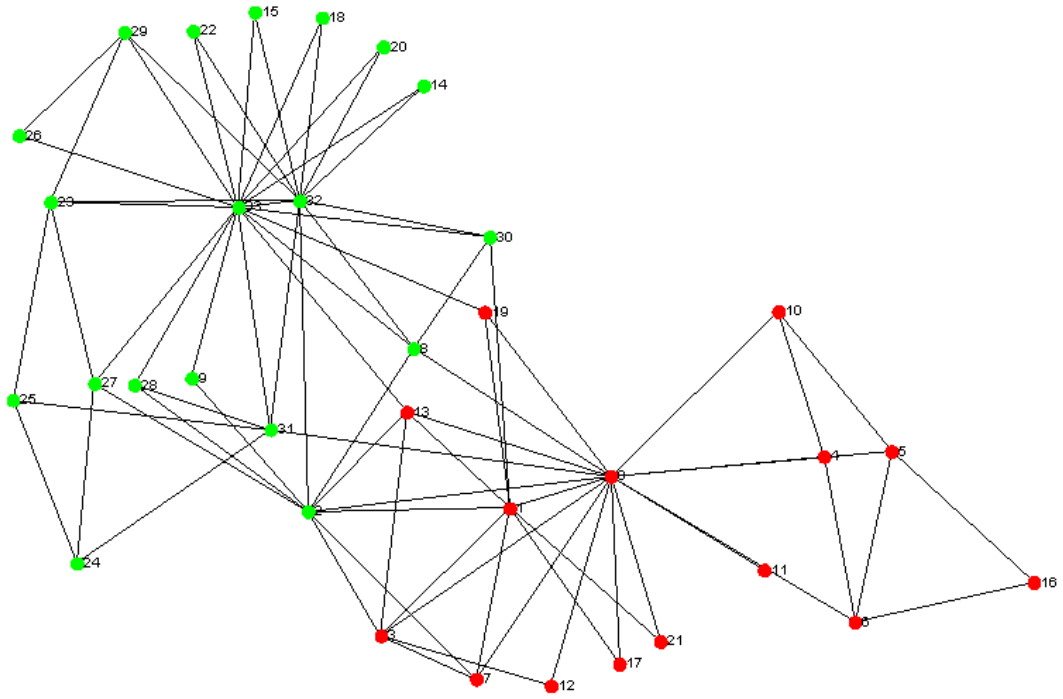

Figure 4.1: Zachary Karate Club Social Network

of 198 vertices and 2742 edges. Figure 4.2 shows how our genetic algorithm method divides this network into communities. Vertices belonging to the same community colored with the same color.

Metabolic Network Dataset [12]: This dataset is the graph that showing the metabolic pathways of a multicellular organism, C. Elegans [29]. This network consists of 453 vertices and 2032 edges.

E-mail Network Dataset [55]: This dataset is the network of e-mail interchanges between the members of the University of Rovira i Virgili in Tarragona. This network consists of 1133 vertices and 5451 edges.

Facebook(NIPS) Dataset [37]: It has the data of the Facebook users who installed the application of a Facebook Social API. This undirected networks contains Facebook user–user friendships. A node represents a user. This network consists of 2888 vertices and 2981 edges. Figure 4.3 shows the result after our modified genetic algorithm run on this dataset. Vertices belonging to the same community colored with the same color.

PGP Network Dataset [35]: This dataset is the graph of a component of a

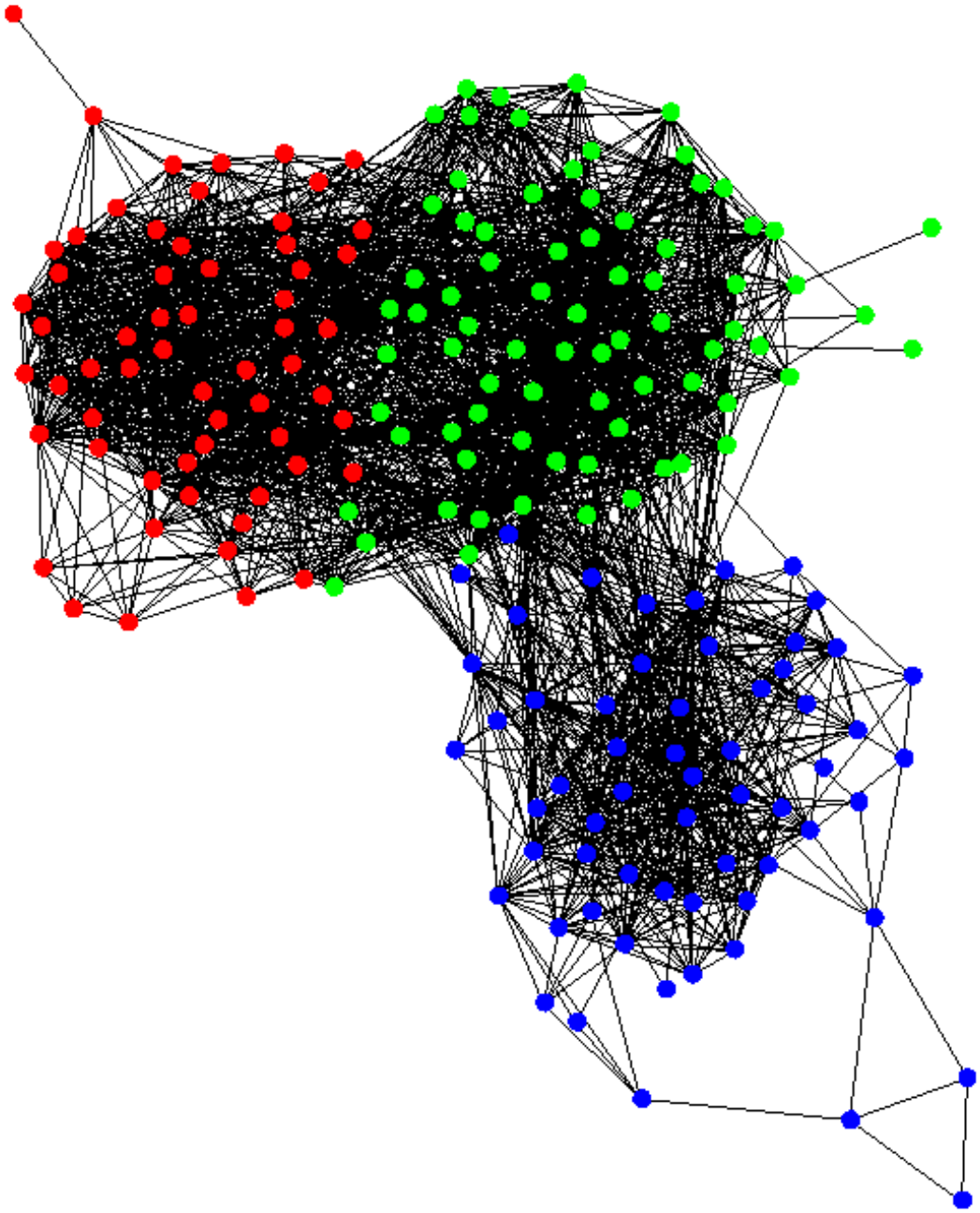

Figure 4.2: Collaboration in Jazz Social Network

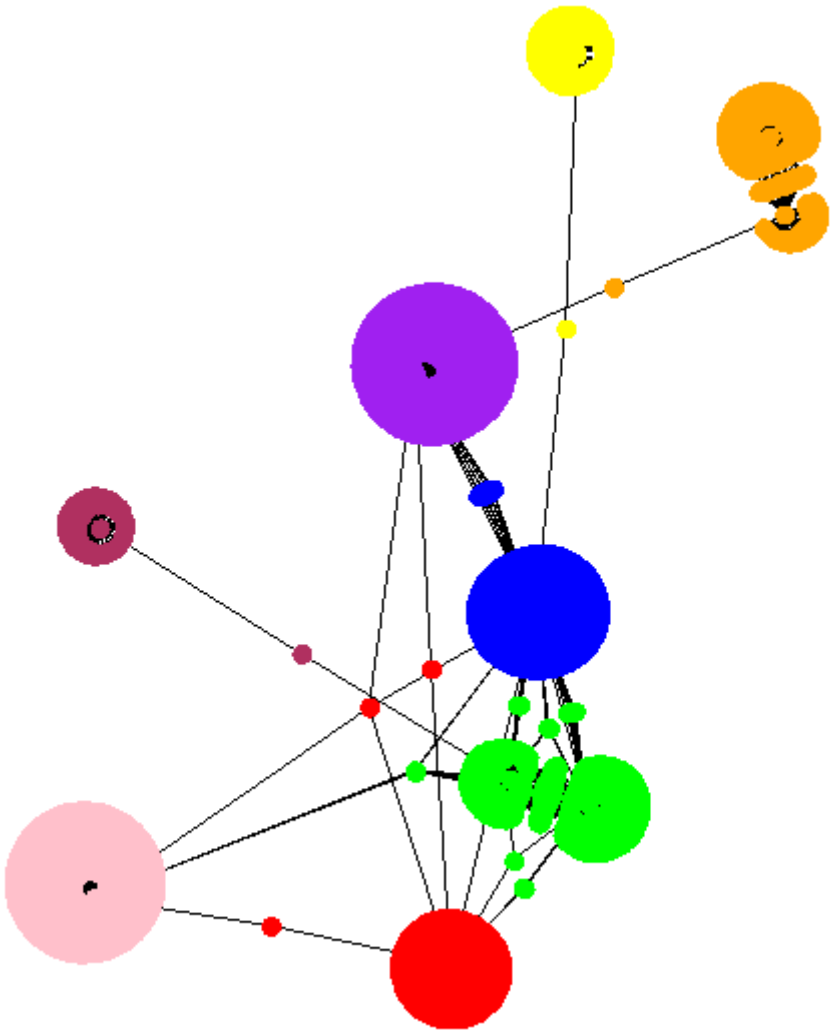

Figure 4.3: Facebook NIPS Social Network

network, consisting the users of the Pretty-Good-Privacy algorithm for secure information interchange. PGP network contains 10680 vertices and 24316 edges. *Cond-Mat Network* Dataset [40]: This network shows the relationships between the authors that shared any paper on Condense Matter. Cond-Mat Network consists of 27519 vertices.

4.2 Experimental Setup

Implementation of our algorithm is done with Java 1.6. For visualization, *Graph-Stream* (http://graphstream-project.org/), which is a java library to model and analyze dynamic graphs is used. For the experiments for Newman's Spectral

Table4.1: Datasets Used

Network	Size	Cited From
Zachary's Karate Club	34	[52]
Collaboration in Jazz	198	[52]
Metabolic	453	[12]
E-mail	1133	[55]
Facebook(NIPS) Data	2888	[37]
PGP	10680	[35]
Cond-Mat	27519	[40]

Algorithm, *Jmod* Tool (http://tschaffter.ch/projects/jmod/) which is a toolkit for community detection in networks is used. For Girvan-Newman Algorithm, A program called *Gephi*(https://gephi.github.io/) and its Girvan Newman Clustering Plugin is used.
Hardware specifications of the computer used for experiments are following: Windows 7 Operating System(64 bit), 6 GB RAM as memory of which 4000m is allocated to Java Heap Space, Intel(R) Core(TM) i5-2520M 2.50 GHz CPU as processor.

4.3 Defining Parameters for Datasets

In this section, we compare our modified Genetic Algorithm for social networks with other two Genetic Algorithms: Traditional Genetic Algorithm and the GACD proposed by M.Tasgin et. al.[63]. Zachary's Karate Club, Collaboration in Jazz, Facebook(NIPS) and E-mail networks are the datasets that all three algorithms will be run on. We will look at the three metrics for the comparison: maximum modularity score of each methods, how modularity scores of each methods evolves over iterations and how modularity scores of each methods evolve over the time. For these experiments, each methods run on the datasets 50 times and their best results are taken. Since the datasets used in these experiments are not too large, preprocess step is *not applied* for our Modified Genetic Algorithm. For the section 4.2, Constant variables and parameters taken are explained in the following:

4.3.1 Parameters for Tests on Zachary's Karate Club

For *Zachary's Karate Club*, parameters and constant variables are found specific to each of the three algorithms of which the algorithms perform the best. For the algorithm of Tasgin et. al., these values are taken from [64]. Randomization rate during the initialization phase is taken as 0.4, mutation rate is taken as 0.4, clean up rate is taken as 0.2. Population size is decided as 100. For Traditional GA, two way crossover is will be done in every iteration, so experiments are first done to find the mutation rate, p_m, taking population 100 and maximum iteration count 250 by running the algorithm 50 times for each mutation rate value. As seen in Table 4.2, maximum Modularity Q values for each mutation rate and percentage of how many times these maximum Q values reached are shown. As result, the best p_m for Traditional GA is 0.8. After we put the

Table4.2: Tests on Zachary's Karate Club to find mutation rate for Traditional GA

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.4198	0.22
0.2	0.4198	0.30
0.3	0.4198	0.38
0.4	0.4198	0.40
0.5	0.4198	0.30
0.6	0.4198	0.34
0.7	0.4198	0.38
0.8	0.4198	0.48
0.9	0.4198	0.46

mutation rate having the value of 0.8 to its place, experiments are done to find random initialization rate. On initialization phase, If the randomly generated number smaller than the random initialization rate, then the neighbors of the chosen vertex will be the member of the same community with the vertex. As it is seen in Table 4.3, this parameter is found as 0.6 for Traditional GA. After we set all the parameters for Traditional GA, we do experiments to find which initial population count performs better. In Figure 4.4, you can see how modularity score is evolving over iteration for different number of initial populations. As it is mentioned in Chapter 3, *our algorithm*, which is described in this thesis, uses

Table4.3: Tests on Zachary's Karate Club to find random initialization rate for Traditional GA

Mutation Rate	Max. Q	Accuracy
0.0	0.4172	0.02
0.1	0.4198	0.02
0.2	0.4198	0.04
0.3	0.4198	0.08
0.4	0.4188	0.04
0.5	0.4198	0.04
0.6	0.4198	0.12
0.7	0.4198	0.08
0.8	0.4198	0.10
0.9	0.4198	0.02

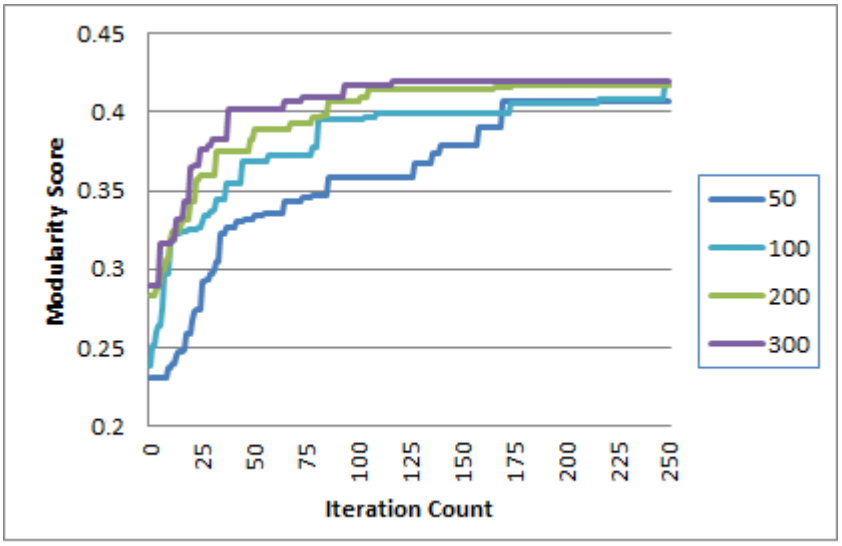

Figure 4.4: Population comparison for Traditional GA

some different parameters. Probability rate of whether the nodes which do not belong to any community after the crossover will be reinserted to a community randomly or reinserted to the community which it increases the Q is shown with the parameter p_{cr}. If the randomly produced number is larger than p_{cr}, then the vertex will be reinserted to any community randomly. Otherwise, the vertex which do not belong to any community will be reinserted to the community where the vertex increase the Modularity Q most or decrease it least.

The probability of the mutation rate is shown with p_m and the probability rate of the reinsertion of the nodes randomly after the mutation is shown with

Table4.4: Tests done on Zachary's Karate Club dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**
0.1	0.0	0.08	0.4	0.0	0.54	0.7	0.0	0.74
0.1	0.1	0.12	0.4	0.1	0.54	0.7	0.1	0.78
0.1	0.2	0.04	0.4	0.2	0.60	0.7	0.2	0.82
0.1	0.3	0.04	0.4	0.3	0.68	0.7	0.3	0.80
0.1	0.4	0.12	0.4	0.4	0.60	0.7	0.4	0.80
0.1	0.5	0.10	0.4	0.5	0.58	0.7	0.5	0.96
0.1	0.6	0.14	0.4	0.6	0.72	0.7	0.6	0.90
0.1	0.7	0.04	0.4	0.7	0.72	0.7	0.7	0.92
0.1	0.8	0.08	0.4	0.8	0.80	0.7	0.8	0.98
0.1	0.9	0.08	0.4	0.9	0.74	0.7	0.9	0.92
0.2	0.0	0.22	0.5	0.0	0.46	0.8	0.0	0.76
0.2	0.1	0.26	0.5	0.1	0.68	0.8	0.1	0.92
0.2	0.2	0.30	0.5	0.2	0.72	0.8	0.2	0.88
0.2	0.3	0.38	0.5	0.3	0.72	0.8	0.3	0.86
0.2	0.4	0.40	0.5	0.4	0.74	0.8	0.4	0.86
0.2	0.5	0.38	0.5	0.5	0.72	0.8	0.5	0.90
0.2	0.6	0.44	0.5	0.6	0.76	0.8	0.6	0.94
0.2	0.7	0.48	0.5	0.7	0.84	0.8	0.7	0.94
0.2	0.8	0.44	0.5	0.8	0.80	0.8	0.8	0.92
0.2	0.9	0.48	0.5	0.9	0.82	0.8	0.9	0.94
0.3	0.0	0.50	0.6	0.0	0.68	0.9	0.0	0.84
0.3	0.1	0.52	0.6	0.1	0.84	0.9	0.1	0.82
0.3	0.2	0.46	0.6	0.2	0.76	0.9	0.2	0.96
0.3	0.3	0.48	0.6	0.3	0.78	0.9	0.3	0.92
0.3	0.4	0.50	0.6	0.4	0.84	0.9	0.4	0.92
0.3	0.5	0.50	0.6	0.5	0.90	0.9	0.5	0.92
0.3	0.6	0.54	0.6	0.6	0.86	0.9	0.6	0.94
0.3	0.7	0.62	0.6	0.7	0.88	0.9	0.7	0.92
0.3	0.8	0.70	0.6	0.8	0.88	0.9	0.8	1.00
0.3	0.9	0.66	0.6	0.9	0.88	0.9	0.9	0.94

p_{mr}. Application of p_{mr} is the same as p_{cr}, such that the vertex, belonging no community, will be assigned to any community if randomly produced number larger than p_{mr}, otherwise reinserted to a community where its membership contribute the Modularity Q the most or decrease it least. Experiments are done to find the most accurate p_{cr}, p_m and p_{mr} taking population count 100, iteration count 250, first, the algorithm run 50 times for each pair of p_m and p_{mr}. p_{cr} is accepted as 1 such that vertices which are not member of a community will be assigned a community where it can contribute to Q most. After that, the algorithm is run again 50 times with previously found p_m and p_{mr} values placed for p_{cr}. Experiments are made and test results are shown in table 4.5 for p_{cr} and in table 4.4 for both p_m and p_{mr}. Accuracy values in the tables show the percentage of the algorithm runs which are reaching to maximum Modularity Q found in tests. We choose the most accurate parameters as optimal ones.

Table4.5: Tests done on Zachary's Karate Club dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.0	0.4197	0.52
0.1	0.4197	0.56
0.2	0.4197	0.62
0.3	0.4197	0.68
0.4	0.4197	0.72
0.5	0.4197	0.76
0.6	0.4197	0.88
0.7	0.4197	0.92
0.8	0.4197	0.94
0.9	0.4197	1.00

Later, the parameters p_m, p_{mr} and p_{cr} are replaced on the algorithm and the optimal random initialization rate for our method is calculated. It is necessary to remind again that all Modularity Q values shown on the test results tables are obtained after the algorithm is run 50 times for each specified parameter rates and the highest scores among them are taken. Accuracy values are the percentage of how many times the maximum modularity Q value is reached during these 50 runs.

After the test results, for our Modified Genetic Algorithm(MGA) presented in

Table4.6: Tests on Zachary's Karate Club to find random initialization rate for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.0	0.4197	0.80
0.1	0.4197	0.92
0.2	0.4197	0.88
0.3	0.4197	0.92
0.4	0.4197	1.00
0.5	0.4197	0.94
0.6	0.4197	1.00
0.7	0.4197	0.96
0.8	0.4197	0.92
0.9	0.4197	0.92

this thesis, p_{cr} is taken as 0.9 which means that vertices do not belonging to any community will be assigned to a community increasing the Modularity Q most or decreasing it least. p_m is taken as 0.9, p_{mr} is taken as 0.8 and the random initialization rate is taken as 0.4.

After we set all the parameters for MGA, we do experiments to find which initial population count performs better. In Figure 4.5, you can see how modularity score is evolving over iteration for different number of initial populations. All population counts reaches the same highest point. Although 50 is the least one, we used 100 for a better comparison with the solution of Tasgin et al. In Table

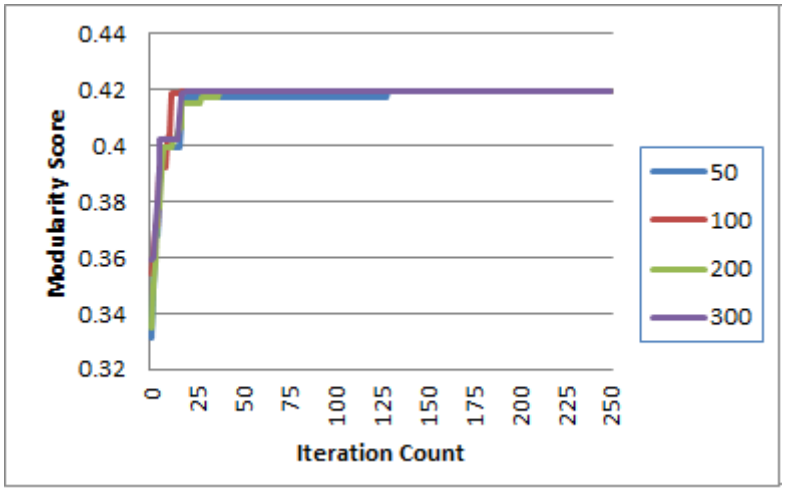

Figure 4.5: Population Comparisons of this work on Zachary's Karate Club

4.50, the maximum Modularity Score Q for the Zachary's Karate Club Network

is tried to find for each algorithm that each of them were run 500 iterations. In Fig. 4.18, how modularity scores of the algorithms are evolving over the iterations on Zachary's Karate Club data set is shown. Time comparison of the algorithms was not done since size of the network is too small and there won't be any significant difference.

4.3.2 Parameters for Tests on Collaboration in Jazz Network

For *Collaboration in Jazz Network*, experiments are done for all parameters and parameter combinations needed by each algorithms that we are comparing, like we did in the examinations of the parameters of Zachary's Karate Club dataset. Population sizes are taken as 100 and iteration counts are taken as 250 to find the parameters for each algorithm which are in fact very large numbers for this dataset.

For *Traditional GA*, examinations for the probable mutation rates are shown in Table 4.7. For each rate, the algorithm is run 50 times and the most accurate result among the records with the highest Modularity Q is chosen as the optimal rate. The most optimal value for the mutation rate is found as 0.8 because it has the highest Q. After that, 0.8 is placed as mutation rate on the algorithm and the algorithm is run 50 times for each random initialization rate. The value of 0.6 is chosen as the best random initialization rate, as seen in Table 4.8.

Table4.7: Tests done on Collaboration in Jazz Network to find mutation rate for Traditional GA

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.3579	0.02
0.2	0.3864	0.02
0.3	0.3972	0.02
0.4	0.3998	0.02
0.5	0.4047	0.02
0.6	0.4121	0.04
0.7	0.4098	0.02
0.8	0.4188	0.02
0.9	0.4019	0.02

Table4.8: Tests done on Collaboration in Jazz Network to find random initialization rate for Traditional GA

Random Init. Rate	Max. Q	Accuracy
0.0	0.4172	0.02
0.1	0.4198	0.02
0.2	0.4198	0.04
0.3	0.4198	0.08
0.4	0.4198	0.04
0.5	0.4198	0.04
0.6	0.4198	0.12
0.7	0.4198	0.08
0.8	0.4198	0.10
0.9	0.4198	0.06

In Figure 4.6, we run Traditional GA on Collaboration on Jazz Network using different initial population counts. For each initial population we run the algorithm 10 times and took the solution which have the highest modularity score. If there are more than one solutions having the same modularity score, than the one which reaches the peak point earlier is chosen. As it is seen, taking initial population 300 gives better initial scores and reaches a higher peak point at the end.

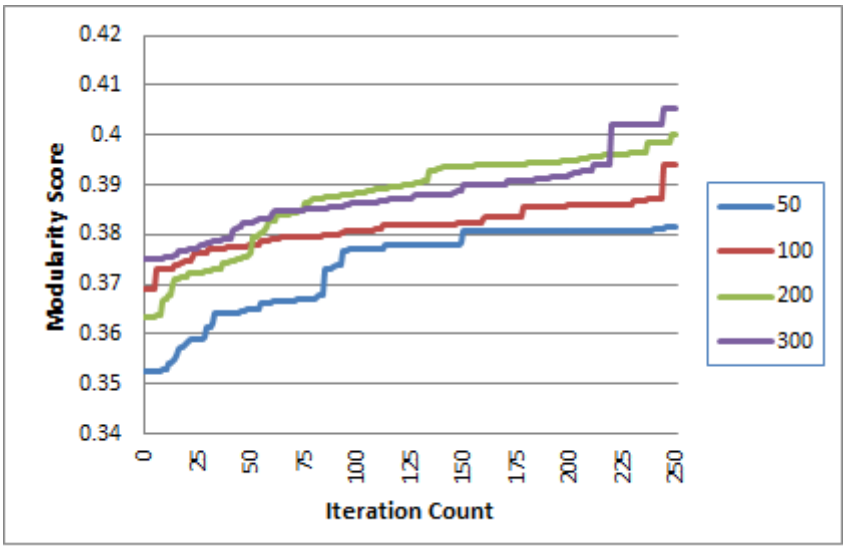

Figure 4.6: Population Comparison of Traditional GA on Collaboration on Jazz Network

The algorithm in the *work of M. Tasgin et al* had not been performed on Collaboration in Jazz Network dataset [64]. However, we are going to test it on Collaboration in Jazz Network, so we need to find the parameters with which the method is giving the best results. First, experiments are done to find mutation rate, while it is assumed that there is not random initialization -all vertices belongs to a different community at the beginning- and no clean up phase. The algorithm is run 50 times for each rate and the highest modularity score and percentage how many times it is reached in these tests are shown in Table 4.9. The optimal probability rate of the mutation operator is found as 0.9 for this part. For the other two parameters, same methodology is also applied: After the mutation rate is placed on the algorithm, experiments were done to find random initialization rate as it seen in Table 4.10 which is found as 0.7. Finally, clean up rate is found as 0.3 after we placed other two parameters, in table 4.11.

Table4.9: Tests on Collaboration in Jazz Network to find mutation rate for GACD of M. Tasgin et. al.

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.0302	0.04
0.2	0.0343	0.02
0.3	0.0441	0.04
0.4	0.0525	0.02
0.5	0.0551	0.02
0.6	0.0673	0.02
0.7	0.0736	0.02
0.8	0.0782	0.14
0.9	0.0855	0.10

In Figure 4.7, we run algorithm of Tasgin et al on Collaboration on Jazz Network using different initial populations. After we placed all the parameters, the algorithms with each of the populations 100, 200 and 300 is run 10 times. All of them reach the same peak point after 240 iterations. Although population 100 does not begin well, it evolves quicker. Since its population count 100 is lesser than 200 and 300 and it is also producing the same results at the end, we choose initial population count as 100.

For our *Modified Genetic Algorithm in this thesis*, again, parameters are found

Table4.10: Tests on Collaboration in Jazz Network to find randomization rate on initialization for GACD of M. Tasgin et. al.

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.0090	0.02
0.1	0.4259	0.02
0.2	0.4242	0.30
0.3	0.4266	0.38
0.4	0.4342	0.40
0.5	0.4310	0.30
0.6	0.4301	0.34
0.7	0.4392	0.38
0.8	0.4326	0.48
0.9	0.4340	0.46

like we have done in Zachary's Karate Club dataset. As our standard testing methodology for parameters, for each probable value of the parameter rates, the algorithm is run 50 times and the highest modularity score reached and percentage of how many times it is reached, is shown on the tables that are showing the test results. Experiments are done first for the pair of p_m and p_{mr}, as seen in the Table 4.12. The p_{cr} and random initialization rate are turned off so that in initialization phase all of the vertices will be assigned to different communities and after the crossover phase all vertices not belonging to any community will be assigned to a community randomly. By putting the values of the parameters found, p_m and p_{mr}, to their places on the algorithm, optimal value of p_{cr} is calculated, as seen in the Table 4.13. Last, the value of p_{cr} is also updated and then the random initialization rate is calculated. Results of the tests for random initialization rate can be seen in the Table 4.14.

In Figure 4.8, we run our algorithm on Collaboration on Jazz Network using different initial populations. After we placed all the parameters found, the algorithms having each of the populations 100, 200 and 300 is run 10 times and the ones with best modularity scores are shown in Figure 4.8. All of them reach the peak point before 200 iterations. Although population 100 does not begin well, it evolves quicker. Since its population count 100 is lesser than 200 and 300 and it is also producing the same results with the two others at the end,

Table4.11: Tests on Collaboration in Jazz Network to find clean up rate on initialization for GACD of M. Tasgin et. al.

Cleanup Rate	Max. Q	Accuracy
0.1	0.4392	0.40
0.2	0.4402	0.44
0.3	0.4444	0.64
0.4	0.4444	0.62
0.5	0.4444	0.56
0.6	0.4422	0.60
0.7	0.4444	0.60
0.8	0.4444	0.58
0.9	0.4410	0.56

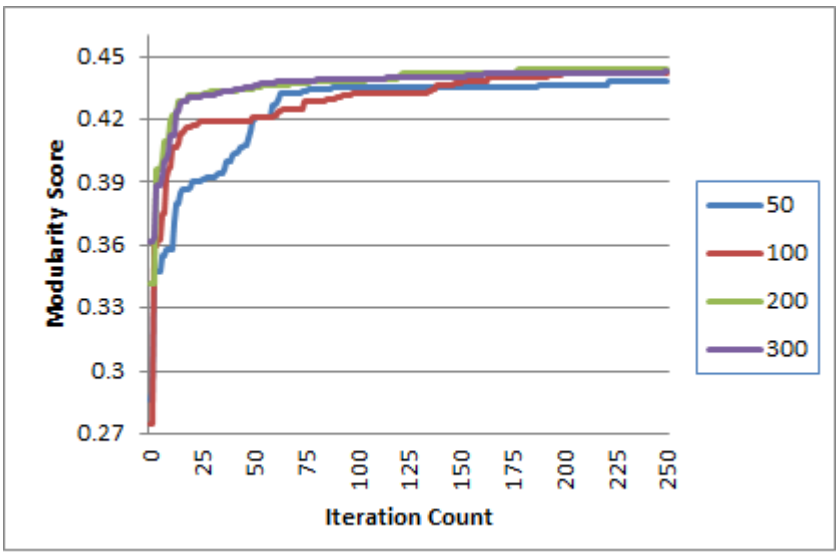

Figure 4.7: Population Comparisons of Tasgin et al on Jazz Network

we choose initial population count as 100.

At the end of the experiments, for our Modified Genetic Algortihm, p_{cr} is taken as 0.3, p_m is taken as 0.8 and p_{mr} is taken as 0.4, Random initializaton rate is taken as 0.5. In Table 4.50, the maximum Modularity Score Q for the Collaboration in Jazz Network is tried to find for each algorithm that each of them were run 500 iterations. The initial chromosome count is taken as 100. In Fig. 4.19, how modularity scores of the algorithms are evolving over the iterations on Collaboration in Jazz Network Dataset is shown. Furthermore, In Fig. 4.20, time comparison of the change on the modularity scores of the algorithms is shown.

Table4.12: Tests done on Collaboration in Jazz Network dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**
0.1	0.0	0.02	0.4	0.0	0.02	0.7	0.0	0.22
0.1	0.1	0.02	0.4	0.1	0.08	0.7	0.1	0.22
0.1	0.2	0.02	0.4	0.2	0.14	0.7	0.2	0.30
0.1	0.3	0.02	0.4	0.3	0.22	0.7	0.3	0.38
0.1	0.4	0.02	0.4	0.4	0.28	0.7	0.4	0.40
0.1	0.5	0.02	0.4	0.5	0.30	0.7	0.5	0.50
0.1	0.6	0.02	0.4	0.6	0.34	0.7	0.6	0.54
0.1	0.7	0.02	0.4	0.7	0.40	0.7	0.7	0.58
0.1	0.8	0.02	0.4	0.8	0.54	0.7	0.8	0.68
0.1	0.9	0.14	0.4	0.9	0.58	0.7	0.9	0.68
0.2	0.0	0.12	0.5	0.0	0.22	0.8	0.0	0.42
0.2	0.1	0.02	0.5	0.1	0.12	0.8	0.1	0.42
0.2	0.2	0.02	0.5	0.2	0.20	0.8	0.2	0.50
0.2	0.3	0.02	0.5	0.3	0.28	0.8	0.3	0.58
0.2	0.4	0.02	0.5	0.4	0.30	0.8	0.4	0.72
0.2	0.5	0.04	0.5	0.5	0.30	0.8	0.5	0.50
0.2	0.6	0.16	0.5	0.6	0.34	0.8	0.6	0.54
0.2	0.7	0.14	0.5	0.7	0.44	0.8	0.7	0.58
0.2	0.8	0.20	0.5	0.8	0.60	0.8	0.8	0.68
0.2	0.9	0.24	0.5	0.9	0.64	0.8	0.9	0.68
0.3	0.0	0.02	0.6	0.0	0.12	0.9	0.0	0.42
0.3	0.1	0.02	0.6	0.1	0.22	0.9	0.1	0.42
0.3	0.2	0.20	0.6	0.2	0.24	0.9	0.2	0.50
0.3	0.3	0.26	0.6	0.3	0.36	0.9	0.3	0.58
0.3	0.4	0.30	0.6	0.4	0.48	0.9	0.4	0.60
0.3	0.5	0.34	0.6	0.5	0.50	0.9	0.5	0.50
0.3	0.6	0.36	0.6	0.6	0.54	0.9	0.6	0.54
0.3	0.7	0.40	0.6	0.7	0.60	0.9	0.7	0.58
0.3	0.8	0.42	0.6	0.8	0.64	0.9	0.8	0.68
0.3	0.9	0.50	0.6	0.9	0.66	0.9	0.9	0.68

Table4.13: Tests done on Collaboration in Jazz Network dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.1	0.4202	0.14
0.2	0.4444	0.64
0.3	0.4444	0.82
0.4	0.4444	0.72
0.5	0.4444	0.74
0.6	0.4444	0.76
0.7	0.4444	0.76
0.8	0.4444	0.80
0.9	0.4444	0.78

Table4.14: Tests on Collaboration in Jazz Network to find random initialization rate for The Work of This Thesis

Random Init. Rate	Max. Q	**Accuracy**
0.0	0.3987	0.22
0.1	0.4112	0.42
0.2	0.4202	0.54
0.3	0.4444	0.70
0.4	0.4444	0.82
0.5	0.4444	0.90
0.6	0.4444	0.86
0.7	0.4444	0.74
0.8	0.4444	0.76
0.9	0.4444	0.88

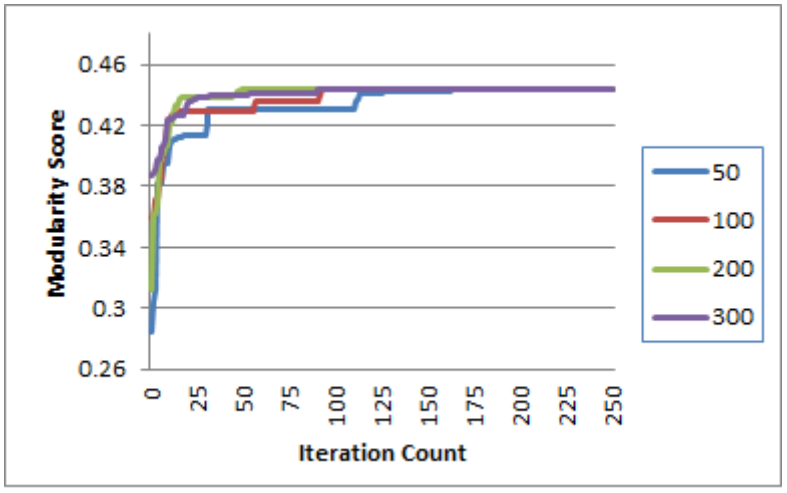

Figure 4.8: Population Comparisons of this work on Collaboration in Jazz Network

4.3.3 Parameters for Tests on Metabolic Network

For *Metabolic Network*, population sizes are taken as 100 and iteration counts are taken as 250 to find the parameters for each algorithm. For all parameter combinations seen in the tables, the algorithms run 50 times for each mutation rate value.

For the *Traditional GA*, In Table 4.15, there are Modularity Q values and their accuracy rates that showing percentage of how many times the maximum Modularity Q value is reached for each mutation rate. Later, optimal mutation rate found is placed and experiments for randomization rate at initial phase are done, which is shown in Table 4.16.

Table4.15: Tests on Metabolic Network to find mutation rate for Traditional GA

Mutation Rate	Max. Q	Accuracy
0.1	0.0131	0.02
0.2	0.0173	0.02
0.3	0.0258	0.02
0.4	0.0240	0.02
0.5	0.0287	0.02
0.6	0.0336	0.02
0.7	0.0342	0.02
0.8	0.0326	0.02
0.9	0.0407	0.02

Table4.16: Tests on Metabolic Network to find random initialization rate for Traditional GA

Random Init. Rate	Max. Q	Accuracy
0.0	0.0414	0.02
0.1	0.2334	0.02
0.2	0.2646	0.02
0.3	0.2793	0.02
0.4	0.2746	0.02
0.5	0.3193	0.02
0.6	0.2992	0.02
0.7	0.3117	0.02
0.8	0.3192	0.02
0.9	0.2892	0.02

In Figure 4.9, we run Traditional GA on Metabolic Network using different initial population counts. After all parameters are placed to its place on the algorithm, we run it 10 times for each population counts seen in Figure 4.9. As it is seen, taking initial population 200 gives better initial scores and reaches higher peak point at the end.

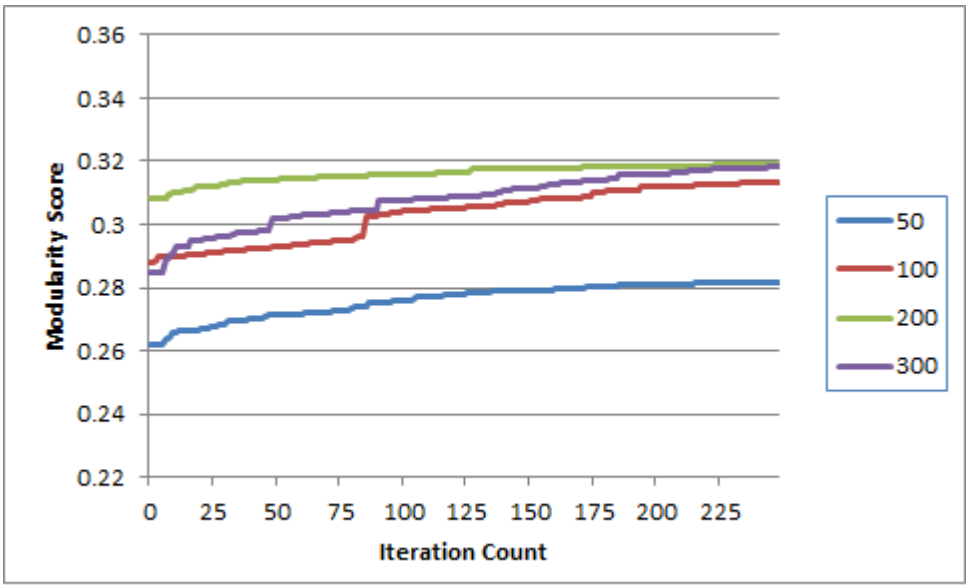

Figure 4.9: Population Comparison of Traditional GA on Metabolic Network

For the mutation rate of *GACD of Tasgin et al*, experiments are done and results can be seen in Table 4.17. Mutation rate is taken as 0.7 since it has the largest Modularity Q and accuracy values and then it placed on the algorithm

experiments are done to decide random initialization rate. As it seen in Table 4.17, the optimal random initialization rate is taken as 0.9. Last, clean up rate is found as 0.4, seen in Table 4.19. As we see in the tables, mutation rate and random initialization rate do large amount of differences while clean up rate does small amount of differences.

Table4.17: Tests on Metabolic Network dataset to find mutation rate for Tasgin et al.

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.0086	0.02
0.2	0.0086	0.02
0.3	0.0131	0.02
0.4	0.0156	0.02
0.5	0.0194	0.02
0.6	0.0231	0.02
0.7	0.0269	0.02
0.8	0.0236	0.02
0.9	0.0224	0.02

Table4.18: Tests on Metabolic Network to find random initialization rate for Tasgin et al.

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.0269	0.02
0.1	0.2248	0.02
0.2	0.2889	0.02
0.3	0.2815	0.02
0.4	0.3109	0.02
0.5	0.3071	0.02
0.6	0.3230	0.02
0.7	0.3340	0.02
0.8	0.3026	0.02
0.9	0.3349	0.02

In Figure 4.10, we run the algorithm of Tasgin et al on Metabolic Network using different initial population counts. After all parameters are placed to its place on the algorithm, we run it 10 times for each population counts seen in Figure

Table4.19: Tests on Metabolic Network to find clean-up rate for Tasgin et al.

Clean up Rate	Max. Q	Accuracy
0.1	0.3683	0.02
0.2	0.4039	0.02
0.3	0.4140	0.02
0.4	0.4170	0.02
0.5	0.3980	0.02
0.6	0.3925	0.02
0.7	0.4121	0.02
0.8	0.3992	0.02
0.9	0.4100	0.02

4.10. As it is seen, taking initial population 200 gives better results.

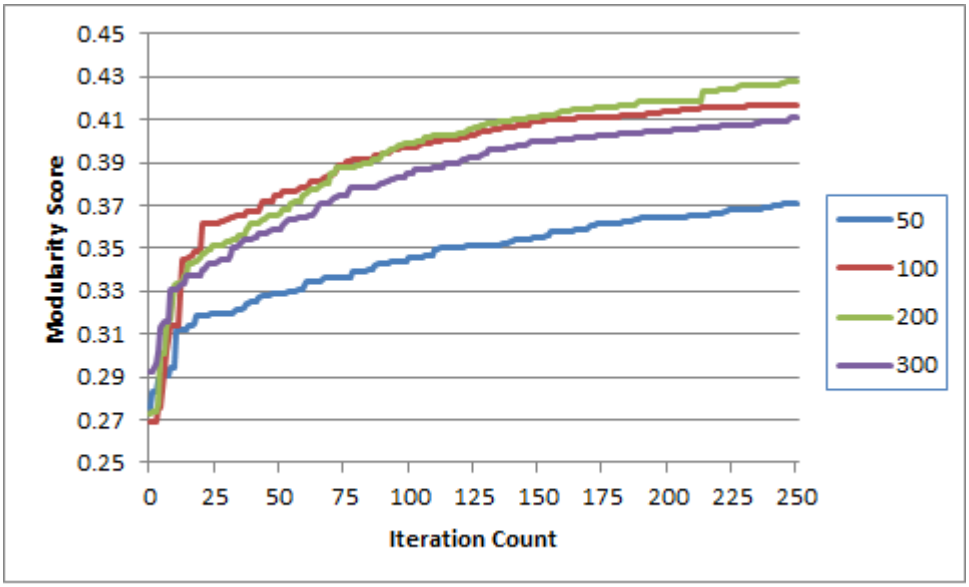

Figure 4.10: Population Comparison of Tasgin et al on Metabolic Network

Now, we look for our algorithm. First, experiments to find p_m and p_{mr} are done. Our algorithm run 50 times for each p_m and p_{mr} combination and p_{cr} is turned off during this process that the vertices do not belonging to any community after the crossover session will be assigned to a community randomly. We take the highest modularity score Q within these 50 runs. As it is seen in Table 4.20, the optimal p_m is found as 0.8 and the p_{mr} is found as 0.8 and they placed on the algorithm for our next experiments to find p_{cr}. Percentage of how many times the algorithm reached to maximum Q value is taken as the accuracy of the p_m and p_{mr} combination. If more than one result having same modularity

score, the record with the higher accuracy accepted as a better result. Again, the algorithm is run 50 times for each p_{cr} value and highest modularity score is shown on Table 4.21. The optimal p_{cr} is taken as 0.7, as seen in Table 4.21. Last, results of the experiments to find random initialization rate can be seen in Table 4.22, which is taken as 0.6.

In Figure 4.11, we run our algorithm on Metabolic Network using different initial population counts. After all parameters are placed to its place on the algorithm, we run it 10 times for each population counts seen in Figure 4.11. As it is seen, taking initial population 200 gives better results. Populations, except 100, gives better results. To compare with other algorithms clearly, we choose 200 as initial population count.

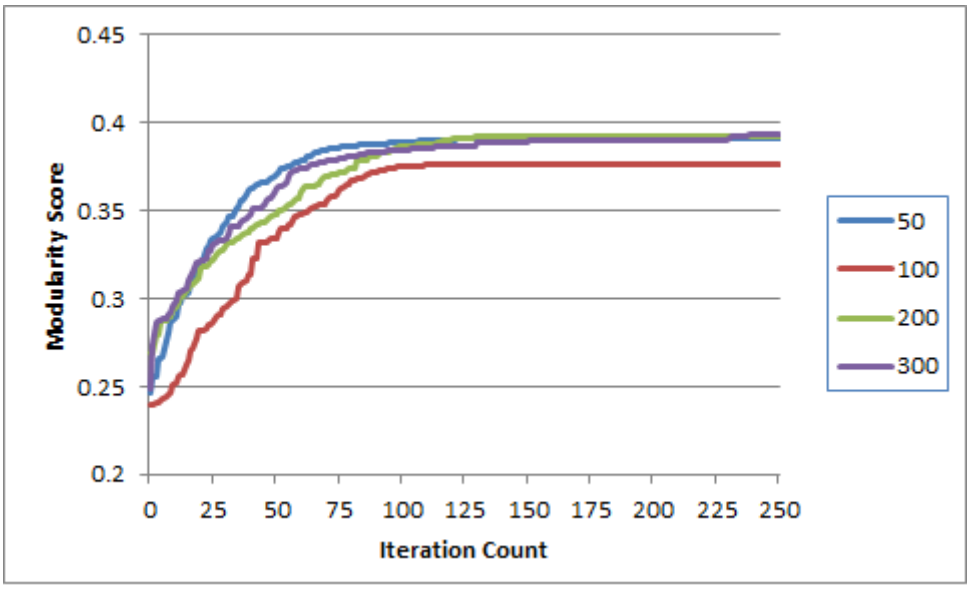

Figure 4.11: Population Comparison of this work on Metabolic Network

4.3.4 Parameters for Tests on E-mail Network

For *E-mail Network*, we apply the same methods to find optimal parameters for each of the three algorithms. Population sizes are taken as 100 and iteration counts are taken as 250 to find the parameters for each algorithm. For all parameter combinations seen in the tables, the algorithms run 50 times for each mutation rate value.

For the *Traditional GA*, In Table 4.23, there are Modularity Q values and their

Table4.20: Tests done on Metabolic Network dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Max. Q**	p_m	p_{mr}	**Max. Q**	p_m	p_{mr}	**Max. Q**
0.1	0.0	0.1002	0.4	0.0	0.1378	0.7	0.0	0.1645
0.1	0.1	0.2416	0.4	0.1	0.2735	0.7	0.1	0.2702
0.1	0.2	0.2437	0.4	0.2	0.3044	0.7	0.2	0.2877
0.1	0.3	0.2811	0.4	0.3	0.2980	0.7	0.3	0.3443
0.1	0.4	0.2836	0.4	0.4	0.3100	0.7	0.4	0.3542
0.1	0.5	0.2616	0.4	0.5	0.3326	0.7	0.5	0.3347
0.1	0.6	0.2773	0.4	0.6	0.3215	0.7	0.6	0.3415
0.1	0.7	0.2869	0.4	0.7	0.3623	0.7	0.7	0.3496
0.1	0.8	0.3061	0.4	0.8	0.3337	0.7	0.8	0.3887
0.1	0.9	0.2736	0.4	0.9	0.3622	0.7	0.9	0.3430
0.2	0.0	0.1081	0.5	0.0	0.1776	0.8	0.0	0.1888
0.2	0.1	0.2472	0.5	0.1	0.1403	0.8	0.1	0.3008
0.2	0.2	0.2737	0.5	0.2	0.2740	0.8	0.2	0.3217
0.2	0.3	0.2616	0.5	0.3	0.3026	0.8	0.3	0.3327
0.2	0.4	0.2901	0.5	0.4	0.3454	0.8	0.4	0.3296
0.2	0.5	0.3220	0.5	0.5	0.3697	0.8	0.5	0.3323
0.2	0.6	0.3182	0.5	0.6	0.3283	0.8	0.6	0.3019
0.2	0.7	0.3219	0.5	0.7	0.3068	0.8	0.7	0.3361
0.2	0.8	0.3410	0.5	0.8	0.3531	0.8	0.8	0.3439
0.2	0.9	0.3421	0.5	0.9	0.3550	0.8	0.9	0.3785
0.3	0.0	0.1127	0.6	0.0	0.1847	0.9	0.0	0.1643
0.3	0.1	0.2666	0.6	0.1	0.2889	0.9	0.1	0.3015
0.3	0.2	0.2742	0.6	0.2	0.3099	0.9	0.2	0.3217
0.3	0.3	0.2925	0.6	0.3	0.3187	0.9	0.3	0.3327
0.3	0.4	0.3226	0.6	0.4	0.2944	0.9	0.4	0.3295
0.3	0.5	0.3335	0.6	0.5	0.2615	0.9	0.5	0.3323
0.3	0.6	0.3384	0.6	0.6	0.3523	0.9	0.6	0.3019
0.3	0.7	0.3529	0.6	0.7	0.3413	0.9	0.7	0.3591
0.3	0.8	0.3480	0.6	0.8	0.3647	0.9	0.8	0.2999
0.3	0.9	0.3708	0.6	0.9	0.3588	0.9	0.9	0.3613

Table4.21: Tests done on Metabolic Network dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.1	0.3808	0.02
0.2	0.3809	0.02
0.3	0.3828	0.02
0.4	0.3849	0.02
0.5	0.3870	0.02
0.6	0.3881	0.02
0.7	0.3920	0.02
0.8	0.3952	0.02
0.9	0.4011	0.02

Table4.22: Tests on Metabolic Network to find random initialization rate for The Work of This Thesis

Random Init. Rate	Max. Q	**Accuracy**
0.0	0.4011	0.02
0.1	0.3793	0.02
0.2	0.4024	0.02
0.3	0.4044	0.02
0.4	0.3865	0.02
0.5	0.3881	0.02
0.6	0.3767	0.02
0.7	0.3644	0.02
0.8	0.3914	0.02
0.9	0.3780	0.02

accuracy rates that showing percentage of how many times the maximum Modularity Q value is reached for each mutation rate. Later, optimal mutation rate is placed and experiments for randomization rate at initial phase are done, in Table 4.24.

Table4.23: Tests on E-mail Network to find mutation rate for Traditional GA

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.0013	0.02
0.2	0.0018	0.02
0.3	0.0024	0.02
0.4	0.0045	0.02
0.5	0.0055	0.02
0.6	0.0067	0.02
0.7	0.2014	0.02
0.8	0.0074	0.02
0.9	0.0072	0.02

Table4.24: Tests on E-mail Network to find random initialization rate for Traditional GA

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.0998	0.22
0.1	0.1144	0.22
0.2	0.1344	0.30
0.3	0.1934	0.38
0.4	0.2450	0.40
0.5	0.2555	0.30
0.6	0.2546	0.34
0.7	0.2550	0.38
0.8	0.2239	0.48
0.9	0.2132	0.46

In Figure 4.12, we run Traditional GA on E-mail Network using different initial population counts. After all parameters are placed to its place on the algorithm, we run it 10 times for each population counts seen in Figure 4.12. As it is seen, taking initial population 200 gives better initial scores and reaches higher peak point at the end.

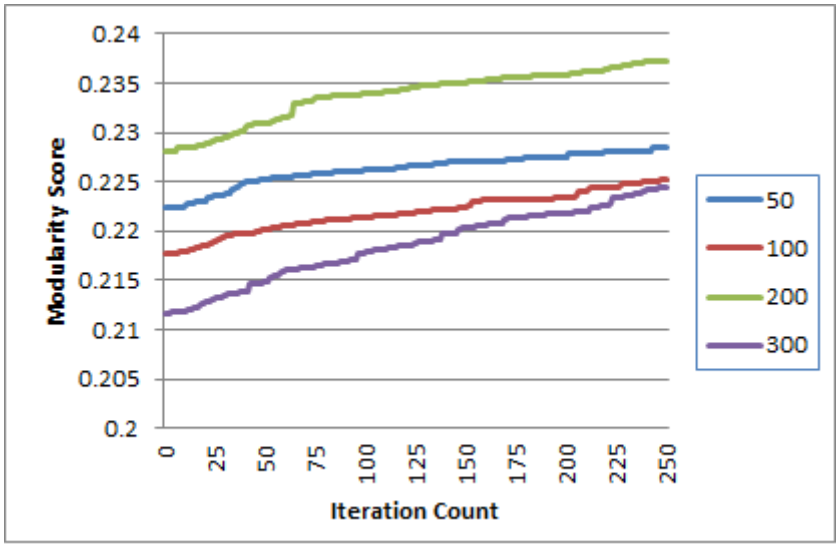

Figure 4.12: Population Comparison of Traditional GA on E-Mail Network

For the mutation rate of *GACD of Tasgin et al*, experiments are done and results can be seen in Table 4.25. Mutation rate is taken as 0.9 since it has the largest Modularity Q and accuracy values and then it placed on the algorithm experiments are done to decide random initialization rate. As it seen in Table 4.26, the optimal random initialization rate is taken as 0.4. Last, clean up rate is found as 0.3, seen in Table 4.27.

Table4.25: Tests done on E-mail Network dataset to find mutation rate for Tasgin et al.

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.0016	0.02
0.2	0.0027	0.02
0.3	0.0032	0.02
0.4	0.0045	0.02
0.5	0.0060	0.02
0.6	0.0062	0.02
0.7	0.0067	0.02
0.8	0.0075	0.02
0.9	0.0084	0.02

In Figure 4.13, we run algorithm of Tasgin et al on E-mail using different initial population counts. After all parameters are placed to its place on the algorithm, we run it 10 times for each population counts seen in Figure 4.13. We choose

Table4.26: Tests on E-mail Network to find random initialization rate for Tasgin et al.

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.0085	0.02
0.1	0.2211	0.02
0.2	0.2635	0.02
0.3	0.2780	0.02
0.4	0.3018	0.02
0.5	0.3017	0.02
0.6	0.2912	0.02
0.7	0.3048	0.02
0.8	0.3122	0.02
0.9	0.2957	0.02

Table4.27: Tests on E-mail Network to find random initialization rate for Tasgin et al.

Clean up Rate	**Max. Q**	**Accuracy**
0.1	0.3001	0.02
0.2	0.3543	0.02
0.3	0.3980	0.04
0.4	0.4170	0.64
0.5	0.4170	0.52
0.6	0.4170	0.24
0.7	0.4170	0.32
0.8	0.4122	0.16
0.9	0.3957	0.08

initial population count as 200 since it gets obviously better when compared to other population counts.

For *our algorithm* in this work, first, experiments to find p_m and p_{mr} are done. Our algorithm run 50 times for each p_m and p_{mr} combination and p_{cr} is turned off during this process. We take the highest modularity score within these 50 runs. As it is seen in Table 4.28, optimal p_m is found as 0.7 and p_{mr} is found as 0.6 and they placed on the algorithm for our next experiments to find p_{cr}. Percentage of how many times the algorithm reached to maximum Q value is taken as the accuracy of the p_m and p_{mr} combination. Again, the algorithm

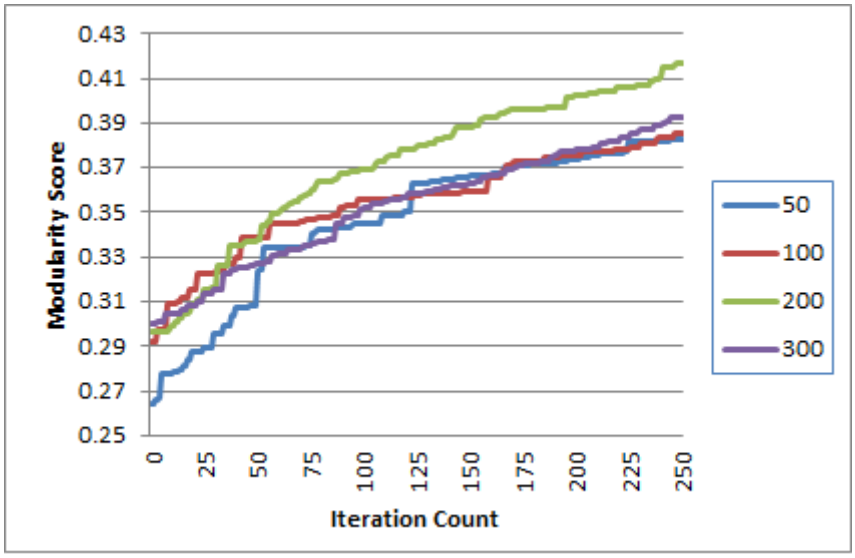

Figure 4.13: Population Comparisons of Tasgin et al on E-mail Network

run 50 times for each p_{cr} value and highest modularity score is shown on Table 4.29. The optimal p_{cr} is taken as 0.7, as seen in Table 4.29. Last, results of the experiments to find random initialization rate are seen in Table 4.30, which is taken as 0.6.

In Figure 4.14, we run algorithm of this work on E-mail Network using different initial population counts. After all parameters are placed to its place on the algorithm, we run it 10 times for each population counts seen in Figure 4.14. Although population count 300 performs better, we choose initial population count as 200 to be the same with the other two methods.

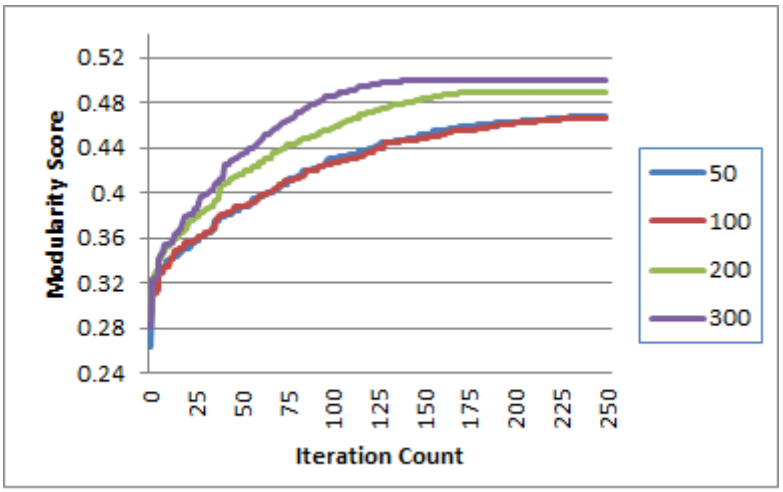

Figure 4.14: Population Comparisons of this work on E-mail Network

In Table 4.50, using the parameters found in this subsection, maximum modu-

Table4.28: Tests done on E-mail Network dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**
0.1	0.0	0.02	0.4	0.0	0.02	0.7	0.0	0.22
0.1	0.1	0.02	0.4	0.1	0.08	0.7	0.1	0.22
0.1	0.2	0.02	0.4	0.2	0.14	0.7	0.2	0.30
0.1	0.3	0.02	0.4	0.3	0.22	0.7	0.3	0.38
0.1	0.4	0.02	0.4	0.4	0.28	0.7	0.4	0.40
0.1	0.5	0.02	0.4	0.5	0.30	0.7	0.5	0.50
0.1	0.6	0.02	0.4	0.6	0.34	0.7	0.6	0.54
0.1	0.7	0.02	0.4	0.7	0.40	0.7	0.7	0.58
0.1	0.8	0.02	0.4	0.8	0.54	0.7	0.8	0.68
0.1	0.9	0.02	0.4	0.9	0.58	0.7	0.9	0.68
0.2	0.0	0.02	0.5	0.0	0.22	0.8	0.0	0.42
0.2	0.1	0.02	0.5	0.1	0.12	0.8	0.1	0.42
0.2	0.2	0.02	0.5	0.2	0.20	0.8	0.2	0.50
0.2	0.3	0.02	0.5	0.3	0.28	0.8	0.3	0.58
0.2	0.4	0.02	0.5	0.4	0.30	0.8	0.4	0.72
0.2	0.5	0.04	0.5	0.5	0.30	0.8	0.5	0.50
0.2	0.6	0.16	0.5	0.6	0.34	0.8	0.6	0.54
0.2	0.7	0.14	0.5	0.7	0.44	0.8	0.7	0.58
0.2	0.8	0.20	0.5	0.8	0.60	0.8	0.8	0.68
0.2	0.9	0.24	0.5	0.9	0.64	0.8	0.9	0.68
0.3	0.0	0.02	0.6	0.0	0.12	0.9	0.0	0.42
0.3	0.1	0.02	0.6	0.1	0.22	0.9	0.1	0.42
0.3	0.2	0.20	0.6	0.2	0.24	0.9	0.2	0.50
0.3	0.3	0.26	0.6	0.3	0.36	0.9	0.3	0.58
0.3	0.4	0.30	0.6	0.4	0.48	0.9	0.4	0.60
0.3	0.5	0.34	0.6	0.5	0.50	0.9	0.5	0.50
0.3	0.6	0.36	0.6	0.6	0.54	0.9	0.6	0.54
0.3	0.7	0.40	0.6	0.7	0.60	0.9	0.7	0.58
0.3	0.8	0.42	0.6	0.8	0.64	0.9	0.8	0.68
0.3	0.9	0.50	0.6	0.9	0.66	0.9	0.9	0.68

Table4.29: Tests done on E-mail Network dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.1	0.0043	0.02
0.2	0.0130	0.02
0.3	0.0324	0.02
0.4	0.1344	0.02
0.5	0.3023	0.04
0.6	0.3453	0.02
0.7	0.4154	0.14
0.8	0.4240	0.12
0.9	0.4234	0.18

Table4.30: Tests on E-mail Network to find random initialization rate for The Work of This Thesis

Random Init. Rate	Max. Q	**Accuracy**
0.0	0.4240	0.02
0.1	0.4543	0.12
0.2	0.4789	0.22
0.3	0.4940	0.30
0.4	0.4940	0.68
0.5	0.4940	0.76
0.6	0.4940	0.70
0.7	0.4940	0.80
0.8	0.4940	0.78
0.9	0.4940	0.68

larity score Q for the E-mail Network is tried to find for each algorithm that each of them were run 500 iterations for 50 times by getting their maximum fitness scores. In Fig. 4.23, how modularity scores of the algorithms are evolving over the iterations on E-mail Network Dataset is shown. Also, In Fig. 4.24, time comparison of the modularity scores of the algorithms is shown.

4.3.5 Parameters for Tests on Facebook(NIPS) Network

For *Facebook(NIPS) Network*, we apply the same methods to find optimal parameters for each of the three algorithms. For *Traditional GA*, the algorithm run 50 times for each mutation rate value. In Table 4.31, there are values and their accuracy rates that showing percentage of how many times the maximum Q value is reached. We take the parameter which have the highest Modularity Q and if the modularity scores are equal then the record with the highest accuracy is taken. In Table 4.31, experiments done to find the optimal modularity rate are shown and it is taken as 0.8. The random initialization rate is taken as 0.8 as seen in Table 4.32.

Table4.31: Tests on Facebook-NIPS Network to find mutation rate for Traditional GA

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.0024	0.02
0.2	0.0037	0.02
0.3	0.0084	0.02
0.4	0.0102	0.02
0.5	0.0132	0.02
0.6	0.0180	0.02
0.7	0.0243	0.02
0.8	0.0408	0.02
0.9	0.0398	0.02

In Figure 4.15, we run Traditional GA on Facebook(NIPS) Network using different initial population counts. As it is seen, taking initial population 300 gives better initial scores and reaches higher peak point at the end.

For the mutation rate of *GACD of Tasgin et al*, experiments are done and

Table4.32: Tests on Facebook-NIPS Network to find random initialization rate for Traditional GA

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.0408	0.02
0.1	0.0897	0.02
0.2	0.1243	0.02
0.3	0.2194	0.02
0.4	0.3949	0.02
0.5	0.3890	0.02
0.6	0.3940	0.02
0.7	0.4054	0.02
0.8	0.4299	0.02
0.9	0.4198	0.02

results can be seen in Table 4.33. Mutation rate is taken as 0.9 since it has the largest Modularity Q and accuracy values and then it placed on the algorithm experiments are done to decide random initialization rate. As it seen in Table 4.34, the optimal random initialization rate is taken as 0.4. Last, clean up rate is found as 0.5, seen in Table 4.35.

Table4.33: Tests on Facebook-NIPS Network to find mutation rate for Tasgin et al.

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.0001	0.02
0.2	0.0005	0.02
0.3	0.0009	0.02
0.4	0.0012	0.02
0.5	0.0019	0.02
0.6	0.0023	0.02
0.7	0.0034	0.02
0.8	0.0035	0.02
0.9	0.038	0.02

In Figure 4.15, we run Tasgin et al on Facebook(NIPS) Network using different initial population counts. As it is seen, the algorithm reaches the same maximum scores on all different populations tried before the 50th iteration. To enable a better accuracy while comparing with Traditional GA, we decided to choose

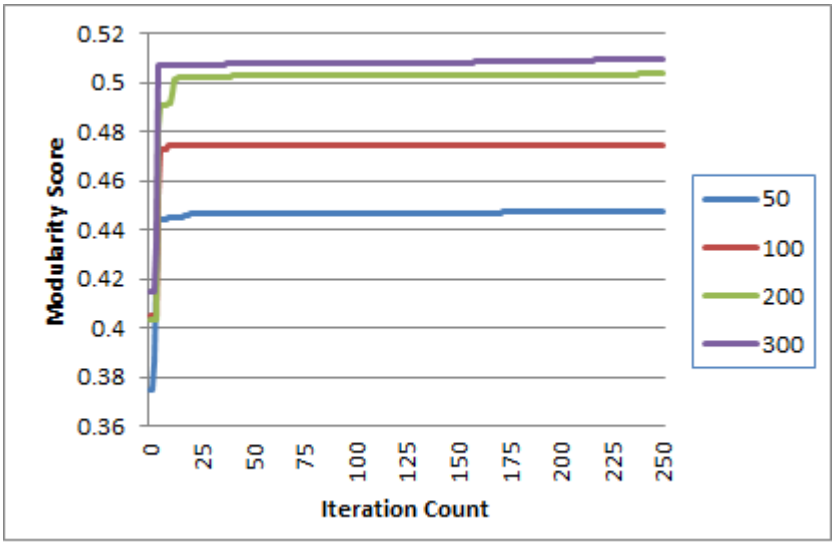

Figure 4.15: Population Comparison of Traditional GA on Facebook(NIPS) Network

initial population as 300.

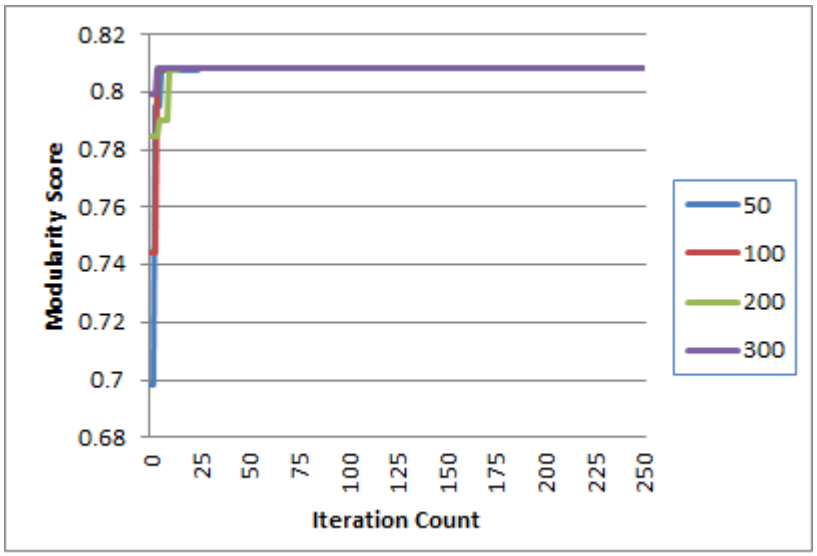

Figure 4.16: Population Comparisons of Tasgin et al on Facebook(NIPS) Network

For *our algorithm* in this work, first, experiments to find p_m and p_{mr} are done. Our algorithm run 50 times for each p_m and p_{mr} combination and p_{cr} is turned off during this process. We take the highest modularity score obtained within these 50 runs. As it is seen in Table 4.36, the optimal p_m value is found as 0.8 and p_{mr} is found as 0.4 and they placed on the algorithm for our next experiments to find p_{cr}. Percentage of how many times the algorithm reached to maximum

Table4.34: Tests on Facebook-NIPS Network to find random initialization rate for Tasgin et al.

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.0039	0.02
0.1	0.3700	0.02
0.2	0.5393	0.08
0.3	0.6369	0.08
0.4	0.6006	0.24
0.5	0.6931	0.30
0.6	0.7832	0.40
0.7	0.7417	0.42
0.8	0.7389	0.40
0.9	0.7944	0.46

Table4.35: Tests on Facebook-NIPS Network to find clean up rate for Tasgin et al.

Clean-up Rate	**Max. Q**	**Accuracy**
0.1	0.7101	0.22
0.2	0.7684	0.30
0.3	0.7546	0.38
0.4	0.7754	0.40
0.5	0.8086	0.50
0.6	0.8086	0.44
0.7	0.8086	0.48
0.8	0.8086	0.48
0.9	0.7980	0.46

Q value is taken as the accuracy of the combination of p_m and p_{mr}. Again, the algorithm run 50 times for each possible p_{cr} value and highest modularity score is shown on the Table 4.37. The optimal p_{cr} is taken as 0.7, as seen in Table 4.37. Last, values of all found parameters put on their places on the algorithm and it is run again 50 times to find random initialization rate. Results of the experiments can be seen in Table 4.38, that random initialization rate is taken as 0.9.

In Figure 4.17, we run this work on Facebook(NIPS) Network using different initial population counts: 50, 100, 200 and 300. We run the algorithm for each

Table4.36: Tests done on Facebook-NIPS Network dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**	p_m	p_{mr}	**Accuracy**
0.1	0.0	0.02	0.4	0.0	0.02	0.7	0.0	0.02
0.1	0.1	0.02	0.4	0.1	0.02	0.7	0.1	0.02
0.1	0.2	0.02	0.4	0.2	0.02	0.7	0.2	0.02
0.1	0.3	0.02	0.4	0.3	0.02	0.7	0.3	0.02
0.1	0.4	0.02	0.4	0.4	0.02	0.7	0.4	0.04
0.1	0.5	0.02	0.4	0.5	0.02	0.7	0.5	0.02
0.1	0.6	0.02	0.4	0.6	0.02	0.7	0.6	0.04
0.1	0.7	0.02	0.4	0.7	0.02	0.7	0.7	0.02
0.1	0.8	0.02	0.4	0.8	0.02	0.7	0.8	0.02
0.1	0.9	0.02	0.4	0.9	0.02	0.7	0.9	0.08
0.2	0.0	0.02	0.5	0.0	0.02	0.8	0.0	0.02
0.2	0.1	0.02	0.5	0.1	0.02	0.8	0.1	0.02
0.2	0.2	0.02	0.5	0.2	0.02	0.8	0.2	0.02
0.2	0.3	0.02	0.5	0.3	0.02	0.8	0.3	0.02
0.2	0.4	0.02	0.5	0.4	0.02	0.8	0.4	0.08
0.2	0.5	0.02	0.5	0.5	0.02	0.8	0.5	0.02
0.2	0.6	0.02	0.5	0.6	0.02	0.8	0.6	0.02
0.2	0.7	0.02	0.5	0.7	0.02	0.8	0.7	0.08
0.2	0.8	0.02	0.5	0.8	0.02	0.8	0.8	0.02
0.2	0.9	0.02	0.5	0.9	0.02	0.8	0.9	0.02
0.3	0.0	0.02	0.6	0.0	0.02	0.9	0.0	0.02
0.3	0.1	0.02	0.6	0.1	0.02	0.9	0.1	0.02
0.3	0.2	0.02	0.6	0.2	0.02	0.9	0.2	0.02
0.3	0.3	0.02	0.6	0.3	0.02	0.9	0.3	0.02
0.3	0.4	0.02	0.6	0.4	0.02	0.9	0.4	0.04
0.3	0.5	0.02	0.6	0.5	0.02	0.9	0.5	0.02
0.3	0.6	0.02	0.6	0.6	0.04	0.9	0.6	0.02
0.3	0.7	0.02	0.6	0.7	0.02	0.9	0.7	0.04
0.3	0.8	0.02	0.6	0.8	0.02	0.9	0.8	0.08
0.3	0.9	0.02	0.6	0.9	0.02	0.9	0.9	0.08

Table4.37: Tests done on Facebook-NIPS Network dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.1	0.0045	0.02
0.2	0.0063	0.02
0.3	0.0071	0.02
0.4	0.0079	0.02
0.5	0.0084	0.02
0.6	0.0090	0.02
0.7	0.0092	0.02
0.8	0.0088	0.02
0.9	0.0086	0.02

Table4.38: Tests done on Facebook-NIPS Network dataset to find random initialization rate for The Work of This Thesis

Random Init. Rate	Max. Q	**Accuracy**
0.0	0.0092	0.22
0.1	0.2608	0.22
0.2	0.2813	0.22
0.3	0.2969	0.30
0.4	0.3081	0.38
0.5	0.3141	0.40
0.6	0.3099	0.30
0.7	0.3457	0.34
0.8	0.3396	0.36
0.9	0.3474	0.34

of these population counts and taken the ones reaching the highest modularity scores. As it is seen, the algorithm performs so close on all different populations tried, although there are differences at their initial scores. Choosing any of them will not differ much. However, to enable the accuracy while doing comparison with Traditional GA and the algorithm of Tasgin et al, we decided to choose the initial population as 300.

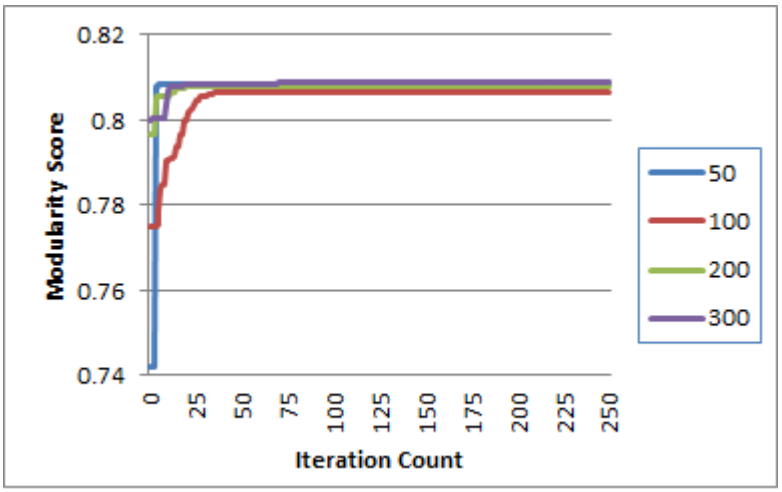

Figure 4.17: Population Comparisons of this work on Facebook(NIPS) Network

In Table 4.50, which is in the next subsection, the maximum Modularity Score Q for the Facebook(NIPS) Network is tried to find for each algorithm that each of them were run 500 iterations using the parameter values found in this subsection. Each algorithm is run for 50 times and the highest modularity scores of the algorithms are taken as their maximum Modularity Score Q for that dataset.

4.3.6 Parameters for Tests on PGP Network

For *PGP*, we apply the previous methods to find optimal parameters for each of the three algorithms. However, there is a difference: We had to take the value of number of the iterations 20 for each test, not the 50, because PGP Network is a large network of which vertex numbers are more than 10000 and it would consume so much time to run the algorithm 50 times for each of the possible parameter rates. For *Traditional GA*, the algorithm run 20 times for each mutation rate value. In Table 4.39, there are values and their accuracy rates that showing percentage of how many times the maximum Q value is

Table4.39: Tests on PGP Network to find mutation rate for Traditional GA

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.2614	0.02
0.2	0.2907	0.02
0.3	0.2994	0.02
0.4	0.3099	0.02
0.5	0.3123	0.02
0.6	0.3146	0.02
0.7	0.3166	0.02
0.8	0.3178	0.02
0.9	0.3208	0.02

Table4.40: Tests on PGP Network to find random initialization rate for Traditional GA

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.3208	0.02
0.1	0.4298	0.02
0.2	0.4994	0.02
0.3	0.5449	0.02
0.4	0.5727	0.02
0.5	0.5949	0.02
0.6	0.5539	0.02
0.7	0.5526	0.02
0.8	0.5603	0.02
0.9	0.5782	0.02

reached. We take the parameter which have the highest Modularity Q and if the modularity scores are equal then the record with the highest accuracy is taken. In Table 4.39, experiments done to find the optimal modularity rate are shown and it is taken as 0.8. The random initialization rate is taken as 0.8 as seen in Table 4.40.

For the mutation rate of *GACD of Tasgin et al*, experiments are done and results can be seen in Table 4.41. Mutation rate is taken as 0.9 since it has the largest Modularity Q and accuracy values and then it placed on the algorithm experiments are done to decide random initialization rate. As it seen in Table

Table4.41: Tests on PGP Network to find mutation rate for Tasgin et al.

Mutation Rate	**Max. Q**	**Accuracy**
0.1	0.2716	0.02
0.2	0.3607	0.02
0.3	0.3909	0.02
0.4	0.5657	0.02
0.5	0.6099	0.02
0.6	0.6504	0.02
0.7	0.6819	0.02
0.8	0.6939	0.02
0.9	0.7155	0.02

Table4.42: Tests on PGP Network to find random initialization rate for Tasgin et al.

Random Init. Rate	**Max. Q**	**Accuracy**
0.0	0.7155	0.02
0.1	0.3391	0.02
0.2	0.4355	0.02
0.3	0.7549	0.02
0.4	0.7612	0.02
0.5	0.7769	0.02
0.6	0.7926	0.02
0.7	0.7737	0.02
0.8	0.7581	0.02
0.9	0.7692	0.02

4.42, the optimal random initialization rate is taken as 0.4. Last, clean up rate is found as 0.5, seen in Table 4.43.

For our algorithm in this thesis, first, experiments to find p_m and p_{mr} are done. Our algorithm run 20 times for each p_m and p_{mr} combination. p_{cr} is turned off during this process that the vertices do not belonging to any community after the crossover session will be assigned to a community randomly. We take the highest modularity score within these 20 runs.

As it is seen in Table 4.44, which is in the next subsection, optimal p_m is found as 0.8 and p_{mr} is found as 0.6 and they placed on the algorithm for our next

Table4.43: Tests on PGP Network to find clean up rate for Tasgin et al.

Clean-up Rate	**Max. Q**	**Accuracy**
0.1	0.7947	0.02
0.2	0.8053	0.02
0.3	0.8069	0.02
0.4	0.7953	0.02
0.5	0.8012	0.02
0.6	0.8035	0.02
0.7	0.8035	0.02
0.8	0.7954	0.02
0.9	0.8000	0.02

experiments to find p_{cr}. Percentage of how many times the algorithm reached to maximum Q value within these 20 runs, is taken as the accuracy of the p_m and p_{mr} combination. If more than one result having same modularity score, the record with the higher accuracy accepted as a better result. Again, the algorithm run 20 times for each p_{cr} value and highest modularity score is shown on Table 4.45. The optimal p_{cr} is taken as 0.8, as seen in Table 4.45. At the last step, optimal value found for p_{cr} is put its place in the algorithm and the algorithm is run again 20 times to decide random initialization rate. Results of the experiments to find random initialization rate are seen in Table 4.46, which is taken as 0.7.

Initial population count is taken as 100 for the experiments of this dataset since the size of the network has more than 10000 nodes and doing experiments for other populations will consume too much time.

4.3.7 Parameters for Tests on Cond-Mat Network

Only our algorithm examined on this dataset. Therefore, this section explains only definition of parameters of our algorithm.

First, experiments to find p_m and p_{mr} are done. Our algorithm run 5 times for each p_m and p_{mr} combination and p_{cr} is turned off during this process that the vertices do not belonging to any community after the crossover session will

Table4.44: Tests done on PGP Network dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Max. Q**	p_m	p_{mr}	**Max Q.**	p_m	p_{mr}	**Max. Q**
0.1	0.0	0.1139	0.4	0.0	0.4135	0.7	0.0	0.5877
0.1	0.1	0.2094	0.4	0.1	0.4957	0.7	0.1	0.6094
0.1	0.2	0.2013	0.4	0.2	0.5389	0.7	0.2	0.6577
0.1	0.3	0.2158	0.4	0.3	0.5745	0.7	0.3	0.6602
0.1	0.4	0.3467	0.4	0.4	0.5880	0.7	0.4	0.6836
0.1	0.5	0.3711	0.4	0.5	0.5976	0.7	0.5	0.6745
0.1	0.6	0.4131	0.4	0.6	0.6188	0.7	0.6	0.6943
0.1	0.7	0.4364	0.4	0.7	0.6406	0.7	0.7	0.7036
0.1	0.8	0.4489	0.4	0.8	0.6799	0.7	0.8	0.7022
0.1	0.9	0.4692	0.4	0.9	0.6605	0.7	0.9	0.7084
0.2	0.0	0.4080	0.5	0.0	0.4705	0.8	0.0	0.6194
0.2	0.1	0.4492	0.5	0.1	0.5138	0.8	0.1	0.6757
0.2	0.2	0.4804	0.5	0.2	0.5753	0.8	0.2	0.6873
0.2	0.3	0.5063	0.5	0.3	0.6075	0.8	0.3	0.6984
0.2	0.4	0.5247	0.5	0.4	0.6103	0.8	0.4	0.7099
0.2	0.5	0.5365	0.5	0.5	0.6592	0.8	0.5	0.7002
0.2	0.6	0.5582	0.5	0.6	0.6743	0.8	0.6	0.7190
0.2	0.7	0.5945	0.5	0.7	0.6887	0.8	0.7	0.7023
0.2	0.8	0.5947	0.5	0.8	0.6854	0.8	0.8	0.7128
0.2	0.9	0.6171	0.5	0.9	0.6937	0.8	0.9	0.7139
0.3	0.0	0.4092	0.6	0.0	0.5546	0.9	0.0	0.6142
0.3	0.1	0.4733	0.6	0.1	0.6045	0.9	0.1	0.6311
0.3	0.2	0.5094	0.6	0.2	0.6174	0.9	0.2	0.6895
0.3	0.3	0.5217	0.6	0.3	0.6538	0.9	0.3	0.7032
0.3	0.4	0.5303	0.6	0.4	0.6743	0.9	0.4	0.7176
0.3	0.5	0.5561	0.6	0.5	0.6802	0.9	0.5	0.7134
0.3	0.6	0.5795	0.6	0.6	0.6854	0.9	0.6	0.7121
0.3	0.7	0.5966	0.6	0.7	0.6941	0.9	0.7	0.7188
0.3	0.8	0.5896	0.6	0.8	0.7049	0.9	0.8	0.7189
0.3	0.9	0.6008	0.6	0.9	0.6933	0.9	0.9	0.7157

Table4.45: Tests done on PGP Network dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.1	0.3983	0.02
0.2	0.4922	0.02
0.3	0.5604	0.02
0.4	0.6281	0.02
0.5	0.6687	0.02
0.6	0.7001	0.02
0.7	0.7077	0.02
0.8	0.7190	0.02
0.9	0.7187	0.48

Table4.46: Tests on PGP Network to find random initialization rate for The Work of This Thesis

Random Init. Rate	Max. Q	**Accuracy**
0.0	0.7190	0.02
0.1	0.7523	0.02
0.2	0.7602	0.02
0.3	0.7865	0.02
0.4	0.7931	0.02
0.5	0.7982	0.02
0.6	0.8123	0.02
0.7	0.8202	0.02
0.8	0.8103	0.02
0.9	0.8162	0.02

be assigned to a community randomly. We had to take the value of number of the iterations 5, lesser than our previous choices of 50 and 20, because Cond-Mat Network is a large network having more than 27000 vertices and it would consume so much time to run the algorithm for each parameter rate. We take the highest modularity score within these 5 runs. As it is seen in Table 4.47, the optimal value of p_m is found as 0.7 and the optimal value of p_{mr} is found as 0.6 and they placed on the algorithm for our next experiments to find p_{cr}. Percentage of how many times the algorithm reached to maximum Q value is taken as the accuracy of the p_m and p_{mr} combination. If more than one result

Table4.47: Tests done on Cond-Mat Network dataset to find p_m and p_{mr} for The Work done in This Thesis

p_m	p_{mr}	**Max. Q**	p_m	p_{mr}	**Max Q.**	p_m	p_{mr}	**Max. Q**
0.1	0.0	0.0023	0.4	0.0	0.0335	0.7	0.0	0.1277
0.1	0.1	0.0032	0.4	0.1	0.0957	0.7	0.1	0.1924
0.1	0.2	0.0047	0.4	0.2	0.1389	0.7	0.2	0.2735
0.1	0.3	0.0088	0.4	0.3	0.1745	0.7	0.3	0.3554
0.1	0.4	0.0112	0.4	0.4	0.2180	0.7	0.4	0.4302
0.1	0.5	0.0134	0.4	0.5	0.2376	0.7	0.5	0.4402
0.1	0.6	0.0144	0.4	0.6	0.2988	0.7	0.6	0.4965
0.1	0.7	0.0175	0.4	0.7	0.3106	0.7	0.7	0.4848
0.1	0.8	0.0188	0.4	0.8	0.3299	0.7	0.8	0.4804
0.1	0.9	0.0192	0.4	0.9	0.3376	0.7	0.9	0.4902
0.2	0.0	0.0080	0.5	0.0	0.1305	0.8	0.0	0.1994
0.2	0.1	0.0102	0.5	0.1	0.2108	0.8	0.1	0.2857
0.2	0.2	0.0174	0.5	0.2	0.2516	0.8	0.2	0.3882
0.2	0.3	0.0203	0.5	0.3	0.2988	0.8	0.3	0.3912
0.2	0.4	0.0267	0.5	0.4	0.3124	0.8	0.4	0.4263
0.2	0.5	0.0485	0.5	0.5	0.3445	0.8	0.5	0.4187
0.2	0.6	0.0787	0.5	0.6	0.3560	0.8	0.6	0.4013
0.2	0.7	0.0945	0.5	0.7	0.3574	0.8	0.7	0.4384
0.2	0.8	0.1056	0.5	0.8	0.3683	0.8	0.8	0.4416
0.2	0.9	0.1244	0.5	0.9	0.3749	0.8	0.9	0.4379
0.3	0.0	0.0357	0.6	0.0	0.1401	0.9	0.0	0.2010
0.3	0.1	0.0422	0.6	0.1	0.1832	0.9	0.1	0.2983
0.3	0.2	0.0733	0.6	0.2	0.2114	0.9	0.2	0.3942
0.3	0.3	0.1317	0.6	0.3	0.2333	0.9	0.3	0.4302
0.3	0.4	0.1903	0.6	0.4	0.2672	0.9	0.4	0.4963
0.3	0.5	0.2361	0.6	0.5	0.3003	0.9	0.5	0.4204
0.3	0.6	0.2595	0.6	0.6	0.3271	0.9	0.6	0.4598
0.3	0.7	0.2766	0.6	0.7	0.3428	0.9	0.7	0.4734
0.3	0.8	0.2896	0.6	0.8	0.3732	0.9	0.8	0.4839
0.3	0.9	0.2908	0.6	0.9	0.3830	0.9	0.9	0.4927

Table4.48: Tests done on Cond-Mat Network dataset to find p_{cr} for The Work of This Thesis

p_{cr}	Max. Q	**Accuracy**
0.1	0.4965	0.02
0.2	0.4932	0.02
0.3	0.4994	0.02
0.4	0.5092	0.02
0.5	0.5102	0.02
0.6	0.5188	0.02
0.7	0.5293	0.02
0.8	0.5193	0.02
0.9	0.4965	0.02

Table4.49: Tests on Cond-Mat Network to find random initialization rate for The Work of This Thesis

Random Init. Rate	Max. Q	**Accuracy**
0.0	0.5293	0.02
0.1	0.5334	0.02
0.2	0.5943	0.02
0.3	0.5743	0.02
0.4	0.6002	0.02
0.5	0.6154	0.02
0.6	0.6229	0.04
0.7	0.6187	0.02
0.8	0.6056	0.04
0.9	0.6086	0.02

having same modularity score, the record with the higher accuracy accepted as a better result. Again, the algorithm run 5 times for each p_{cr} value and highest modularity score is shown on Table 4.48. The optimal p_{cr} is taken as 0.7, as seen in Table 4.48. Last, results of the experiments to find random initialization rate are seen in Table 4.49, which is taken as 0.6.

Initial population count is taken as 100 for the experiments of this dataset since the size of the network is huge and doing experiments for other populations will consume too much time.

4.4 Comparison of Modularity Score and Time with Other Genetic Algorithms

After the replacement of the optimal values of the parameters we found for each algorithm, we run each of them 50 times and take their highest Modularity Q value for comparison. Each run is done for 500 iterations to let them to reach their peak points.

Table4.50: Comparison of Modularity Scores of GA Technics

Dataset	Traditional GA	Tasgin's GGA	MGA (Work in Thesis)
Zachary's Karate Club	0.4198	0.4198	0.4198
Collaboration in Jazz	0.4393	0.4444	0.4444
Metabolic	0.3341	0.4338	0.4373
E-mail	0.2555	0.4339	0.5654
Facebook(NIPS) Data	0.7976	0.8086	0.8087
PGP	0.7893	0.8071	0.8557

As you can see in Table 4.50, our Modified Genetic Algorithm for social networks gives accurate and satisfactory results when modularity scores compared to Traditional GA and GACD of Tasgin et al. Especially, you can see significant differences of maximum modularity scores reached on *E-mail* and *PGP* datasets.

In the results shown in Figure 4.18, you can see how modularity scores evolve

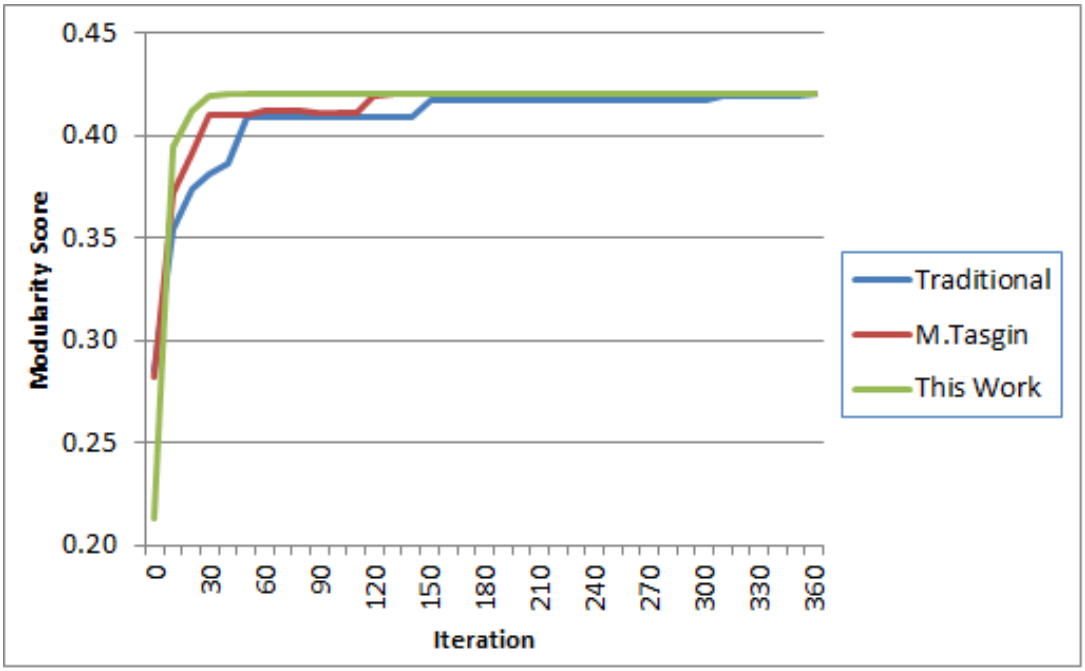

Figure 4.18: comparison on Zachary Karate Club over Iterations

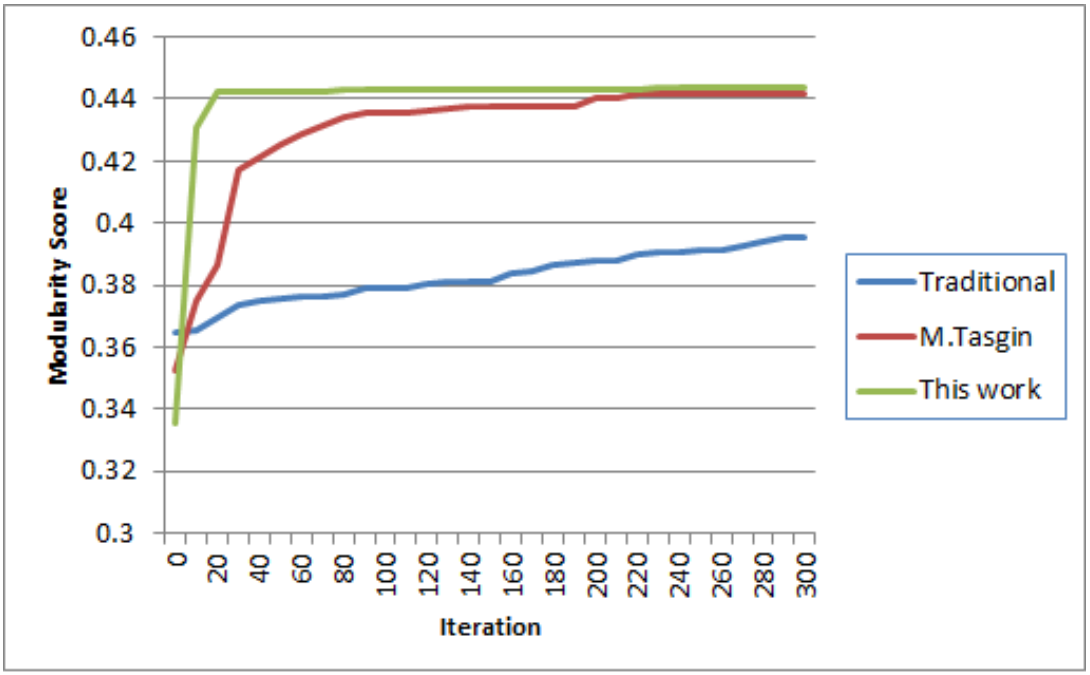

Figure 4.19: comparison on Collaboration in Jazz over Iterations

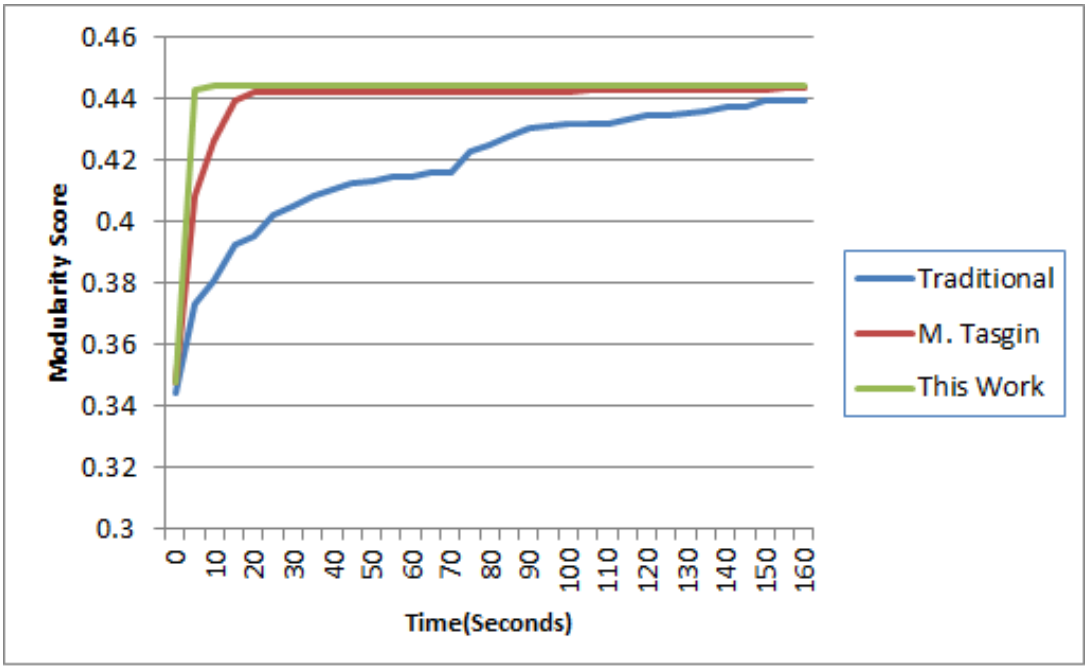

Figure 4.20: comparison on Collaboration in Jazz over Time

over each iteration while they are running on the Zachary's Karate Club dataset. The Modified Genetic Algorithm which is the work done in this thesis, GACD which is the work done by Tasgin et al and the Traditional GA is run with their optimal parameters for the Zachary's Karate Club dataset and their modularity scores are shown for every 10 turns. As you can see from the graphic, modularity scores obtained from the Modified Genetic Algorithm proposed in this work evolves better and reaches higher values quickly although the method of Tasgin et al and Traditional GA has a good start.

In figure 4.19, comparison is made on the dataset of *Collaboration in Jazz Social Network*. Modularity scores, which are obtained after the run of all three algorithms with their optimized parameters, are shown for every 10 turns. Only after 20 turns, our method reaches so close to the peak modularity score while method of Tasgin et al needs to be run more turns, around 80 turns. Also, In figure 4.20, evolution of the modularity scores of these three Genetic Algorithms is shown for every 5 seconds. Our work approaches so close to the peak point in 5 seconds and catches it up in 10 seconds, while GACD of Tasgin et al need to take around more 10 seconds and Traditional GA need to take around more 130 seconds to catch up.

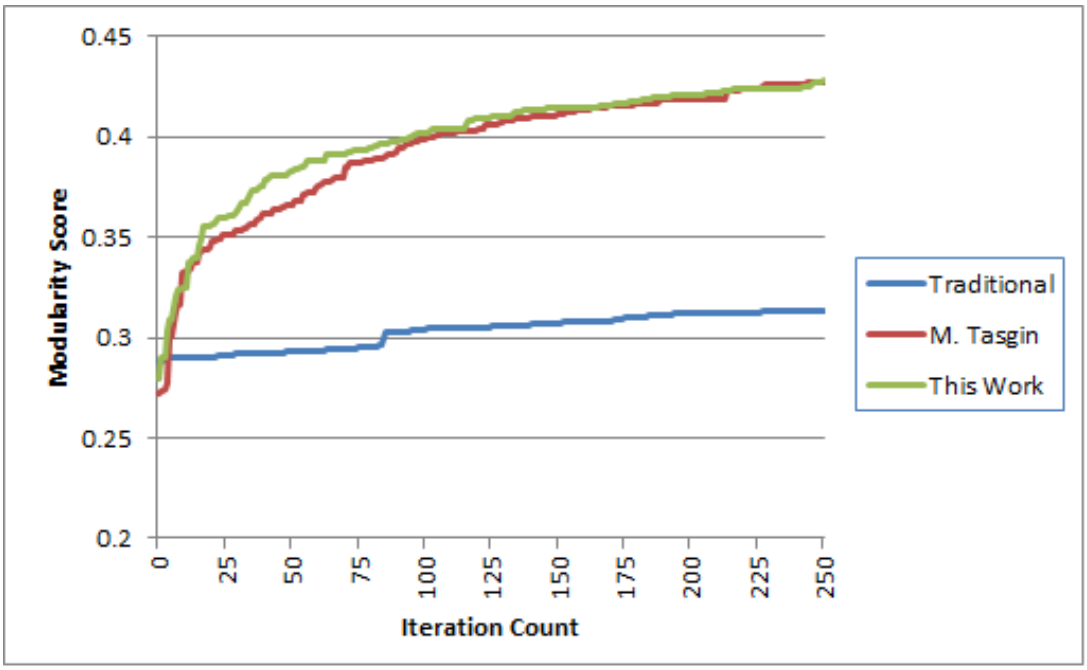

Figure 4.21: comparison on Metabolic over Iterations

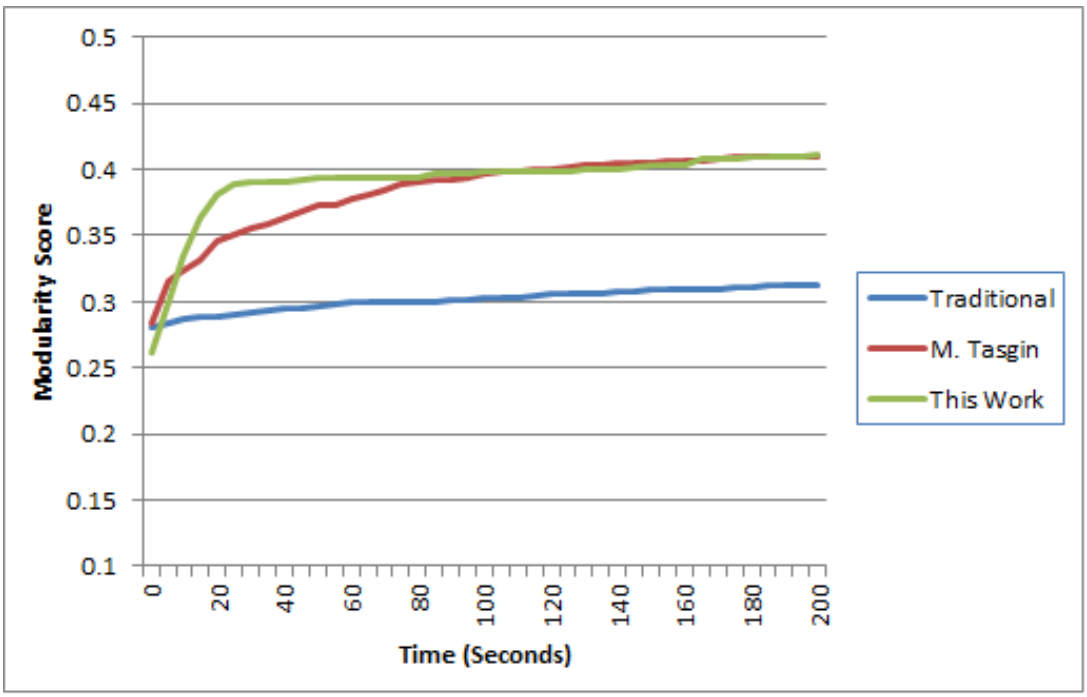

Figure 4.22: comparison on Metabolic over Time

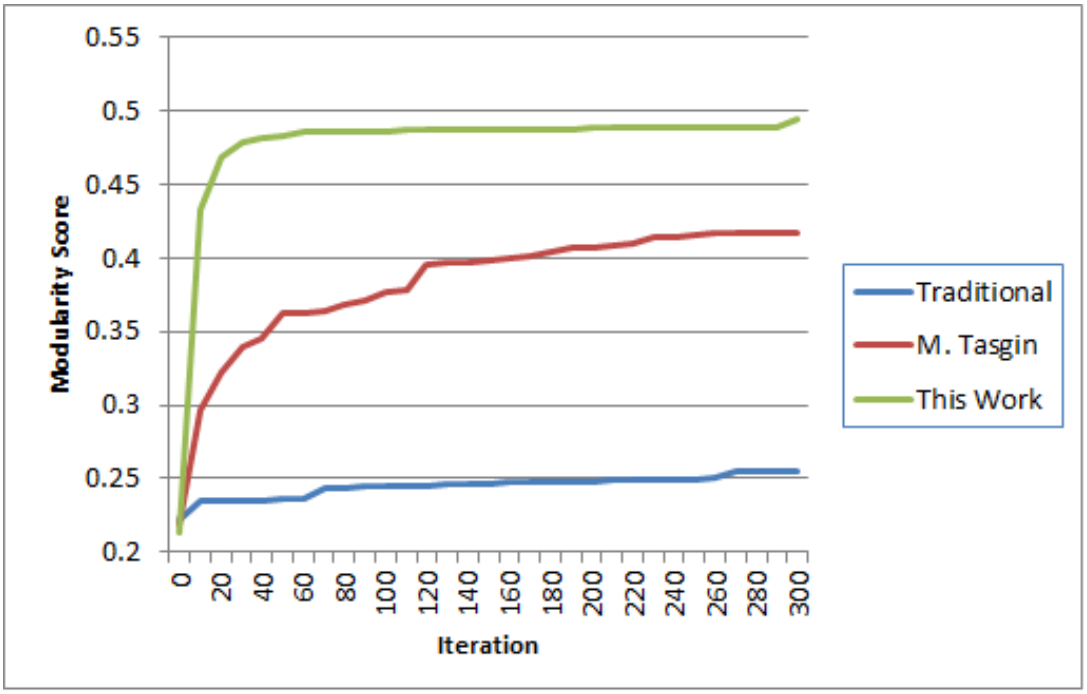

Figure 4.23: comparison on E-mail Network over Iterations

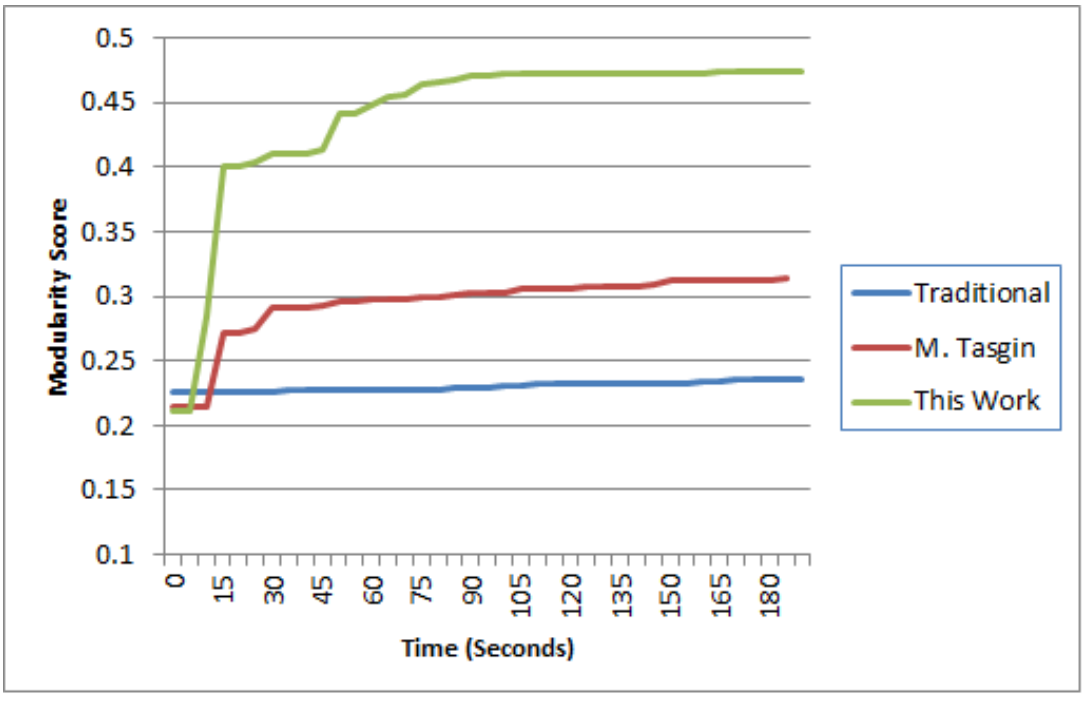

Figure 4.24: comparison on E-Mail Network over Time

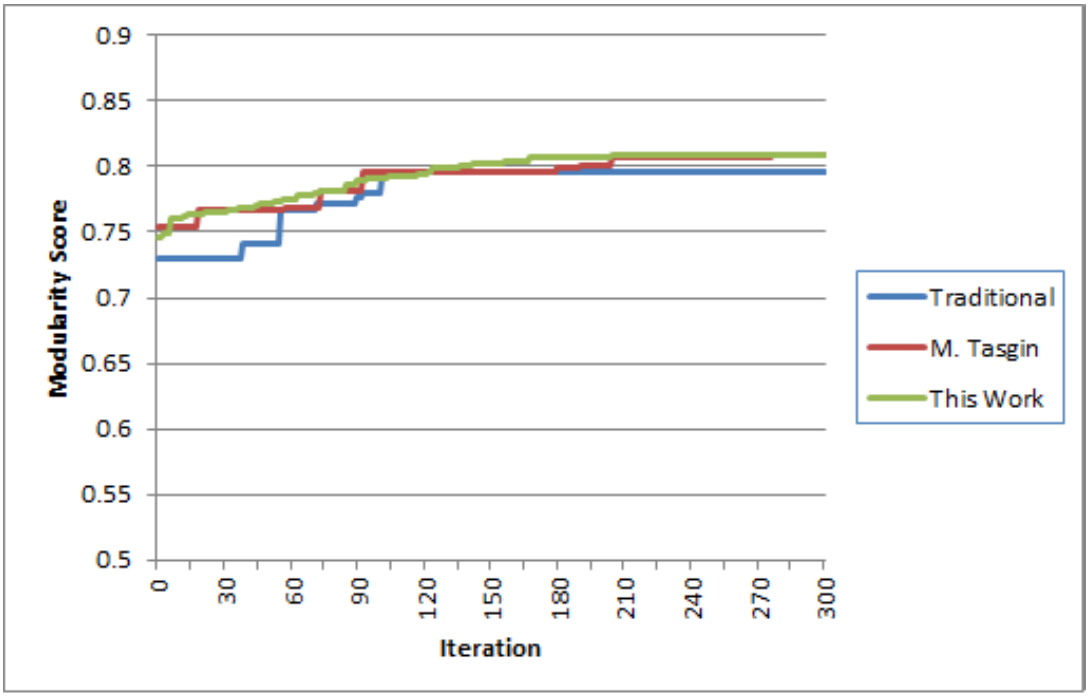

Figure 4.25: comparison on Facebook(NIPS) Dataset over Iterations

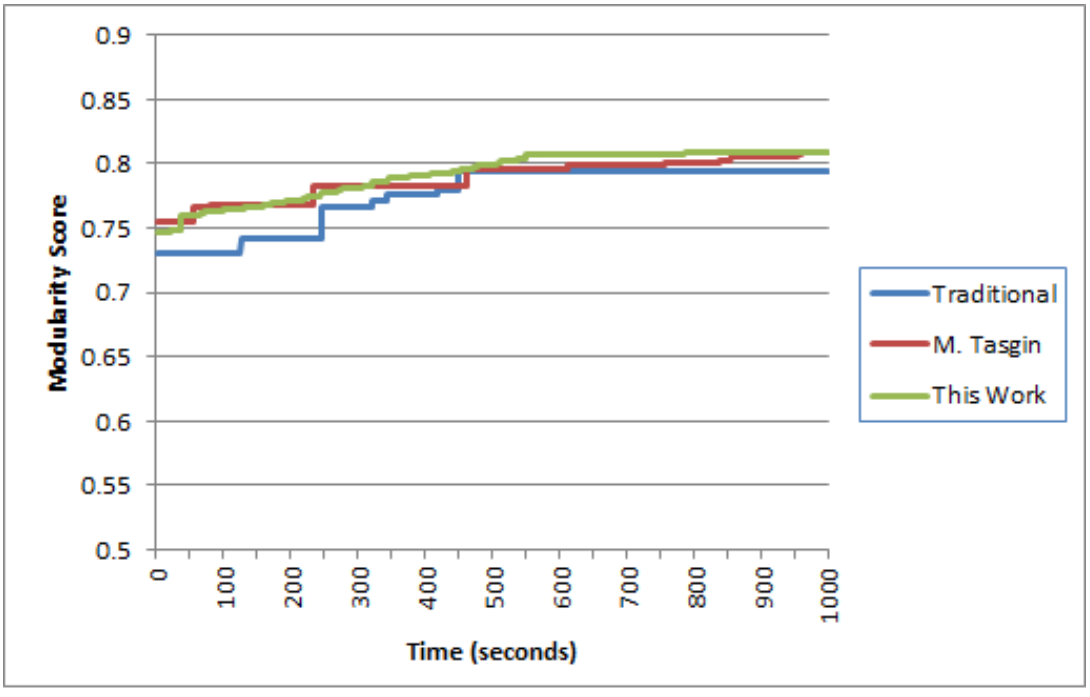

Figure 4.26: comparison on Facebook(NIPS) Dataset over Time

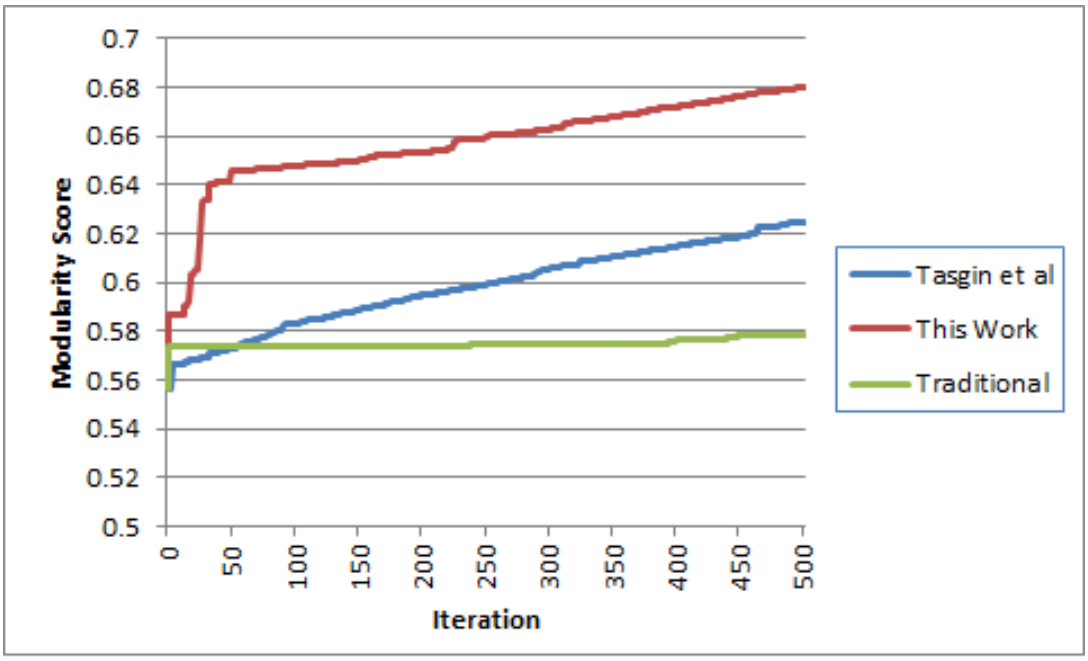

Figure 4.27: comparison on PGP Dataset over Iteration

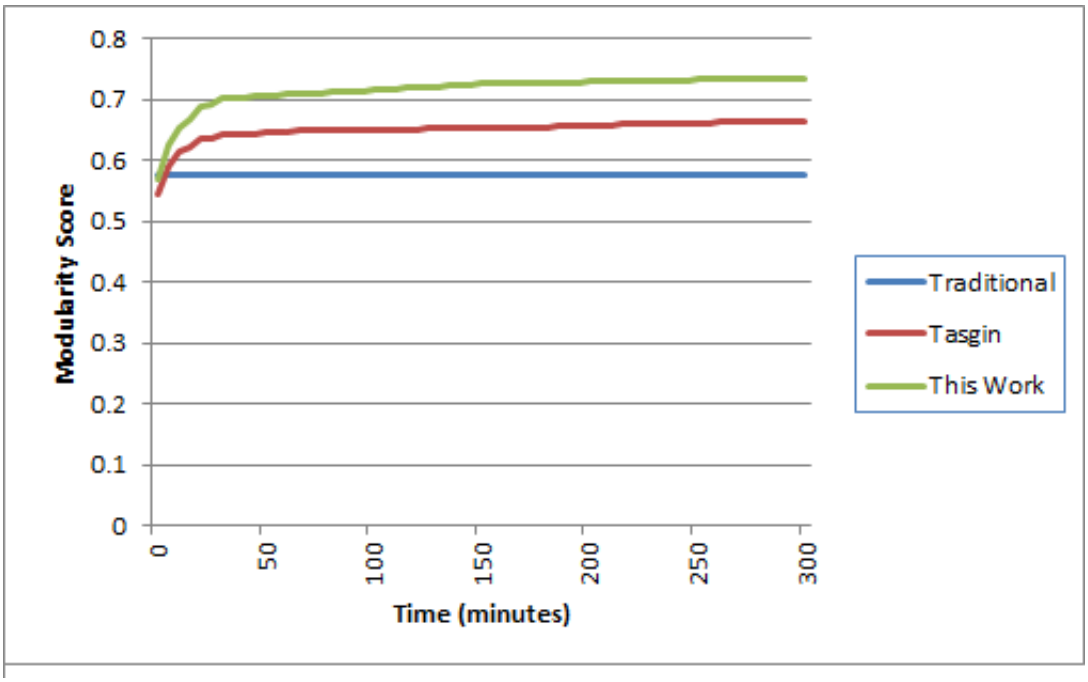

Figure 4.28: comparison on PGP Dataset over Time

In figure 4.21, comparison is made on the *Metabolic Network*. The graph shows the runs of the algorithms that reaches the maximum modularity score. Modularity score values taken from each iteration. Also, figure 4.22 shows the modularity score values taken for each seconds. As you can see, there are not any important difference between Tasgin's method and out method when considering *Metabolic Network* dataset.

In figure 4.23, comparison is made on the *E-mail Network* dataset that each of the algorithms is run with their optimal parameters. Graph shows the values of the Modularity Score Q for every 10 iterations. As you can see, all three algorithms start from a point which is very close to each one of them. Our Modified GA evolves better and approaches to peak point quicker than the both the other methods. It can also be seen from the graphic that when our algorithm reaches 0.4940 -the highest modularity Q value for this dataset- at the 300. iteration, modularity score of the Traditional GA is around 0.26 and modularity score of the work of Tasgin et al is around 0.42.In figure 4.24, this time, we run these three algorithms 50 times an use the values from their best results to compare in terms of time. In the graphic, we see the evolution of the modularity scores for every 5 seconds. As you can see, our Modified GA take a large step after fifth second while, Traditional GA and GACD of Tasgin et al performs poorly. At the 200th second, our algorithm reaches the value of 0.4940 while modularity score of the Traditional GA is around 0.24 and modularity score of the work of Tasgin et al is around 0.32.

In figures 4.25 and 4.26, you can see how modularity scores of each methods evolve over each iterations and each seconds on the *Facebook(NIPS)* dataset. As you can see, after the initialization phase, algorithms begin with high modularity scores and this dataset is not much suitable to make difference.

Furthermore, in the figure 4.27, you can see how modularity scores of each methods evolve over each iterations on the *PGP* dataset. As you can see, our algorithm performs better than other algorithms. Also, evolution of the algorithms over time on *PGP* can be seen in figure 4.28 in which the modularity scores during the first 300 minutes of the run of the algorithms are shown. These values are taken from the best ones after we run each algortihm 10 times.

4.5 Comparison with Different Community Detection Algorithms for Social Networks

We compare our Modified Genetic Algorithm with other four algorithms which are accepted as proven ways for community detection in social networks. These four algorithms are these: The betweenness based algorithm of Girvan and Newman [26], the greedy optimization algorithm of Clauset et. al.[8], the extremal optimization algorithm of Duch and Arenas[12] and the spectral method of Newman which uses eigenvectors of similarity matrices of the networks. These algorithms are run on the same datasets and their modularity scores are calculated. In Table 4.51 and Table 4.52, you can see the comparison of modularity scores of these algorithms. Datasets are, in order, Zachary's Karate Club[71], Collaboration in Jazz Network[52], Metabolic Network for the nematode C. Elegans, E-mail Network[55], PGP which is a trust network on mutual signing of cryptography keys and Cond-Mat which is a network showing the relations of the authors and the papers in Condense Matters archive. Abbreviations are used for the algorithms names:

- GN: Girvan Newman Algorithm
- CNM: Algorithm of Clauset et al.
- DA: Algorithm of Duch and Arenas
- NSA: Newman's Spectral Algorithm
- MGA: The work studied in this thesis is referred as MGA. Abbreviation is the combination of the capitals that are taken from the first letters of Modified Genetic Algorithm.

We did not need to find parameters for other algorithms since experiments are done for these algorithms on datasets we are using and their best results are claimed [26][8][12][44]. Speaking of MGA for both with preprocess and without preprocess methods, the same parameters are used. For the datasets Zachary's Karate Club, Collaboration in Jazz Network and E-mail Network, the values of

our variables and predefinitions are the same with the ones mentioned in section 4.3. However, we have not done experiments to find optimal parameters on *Metabolic Network*, *PGP Network* and *Cond-Mat Network* For *PGP Network*, in previous sections so we will do these experiments in this section.

4.5.1 Comparisons Between Algorithms

In terms of MGA, optimized parameter values and predefined constants are defined and summarized in the following. For *Zachary's Karate Club Network*, *Collaboration in Jazz Network* and *E-mail Network*, values of the parameters will be the ones defined in section 4.3. For *Metabolic Network* mutation rate is taken as 0.8, initial chromosome count is taken as 100. p_{cr} is taken as 0.7, p_{mr} is taken as 0.8. For *Cond-Mat Network*, mutation rate is taken as 0.7, initial chromosome count is taken as 100. p_{cr} is taken as 0.7, p_{mr} is taken as 0.6. For both PGP and Cond-Mat, no maximum iteration count is defined. Termination condition for PGP is modularity score's does not changing for 100 iterations after its reaching to 0.855, which is the maximum modularity value found by other methods, and for Cond-Mat is modularity score's does not changing for 200 iterations after its reaching to 0.723 which is the again the maximum modularity value found by other methods.

Table4.51: comparison of Modularity Scores with Other Community Detection Methods

Dataset	GN	CNM	DA	NSA	MGA without Preprocess
Zachary	0.401[1]	0.381[2]	0.419[2]	0.419[1]	0.420
Jazz musicians	0.405[1]	0.439[2]	0.445[2]	0.442[1]	0.445
Metabolic	0.403[1]	0.402[2]	0.434[2]	0.424[1]	0.437
Email	0.532[1]	0.494[2]	0.574[2]	0.552[1]	0.565
PGP	0.816[1]	0.733[2]	0.846[2]	0.855[1]	0.856
Cond-Mat	not applicable	0.668[2]	0.679[2]	0.723[1]	0.723

[1]This data is not found by our experiment. It is claimed in and taken from the work of Newman [44]

[2]This data is not found by our experiment. It is claimed in and taken from the work of Duch and Arenas[12]

For the algorithms of Girvan Newman and Newmans's Spectral Algorithm, libraries and tools which have been mentioned in Section 4.2 Experimental Setup, are used. The results of the algorithm of Duch and Arenas and algorithm of Clauset et al are taken from the experiments mentioned in other studies [44].
In Table 4.51, maximum network modularity scores of the algorithms when they run on the same datasets are shown. Each algorithm is run on the same datasets 40 times and their largest value of Modularity Q taken. Girvan Newman algorithm is not applied to Condense Matters dataset because of its time complexity, it takes so much time to run. The work done in this thesis referred as MGA, Modified Genetic Algorithm, and in Table 4.51, results of MGA of which preprocess is not applied is shown.

Table4.52: comparison of Modularity Scores with Other Community Detection Methods

Dataset	GN	CNM	DA	NSA	MGA with Preprocess
Zachary	0.401[1]	0.381[2]	0.419[2]	0.419[1]	not applied
Jazz musicians	0.405[1]	0.439[2]	0.445[2]	0.442[1]	not applied
Metabolic	0.403[1]	0.402[2]	0.434[2]	0.424[1]	0.438
Email	0.532[1]	0.494[2]	0.574[2]	0.552[1]	0.576
PGP	0.816[1]	0.733[2]	0.846[2]	0.855[1]	0.858
Cond-Mat	not applicable[2]	0.668[2]	0.679[2]	0.723[1]	0.729

In Table 4.52, also, maximum network modularity scores of the algorithms when they run on the same datasets are shown. Again, GN is not applied for Condense Matters dataset. For this time, MGA is run including the preprocess step. We did not run MGA on Zachary's Karate Club and Collaboration in Jazz Network datasets, because their size is too small for a preprocess step and it does not refine the results. For preprocess step, we have taken communities from the results of GN and NSA. For Metabolic and E-mail datasets, all the communities given as output after the run of GN and NSA and communities not larger than one third of the network size are used as input. For PGP and Cond-Mat datasets,

[1]This data is not found by our experiment. It is claimed in and taken from the work of Newman [44]

[2]This data is not found by our experiment. It is claimed in and taken from the work of Duch and Arenas[12]

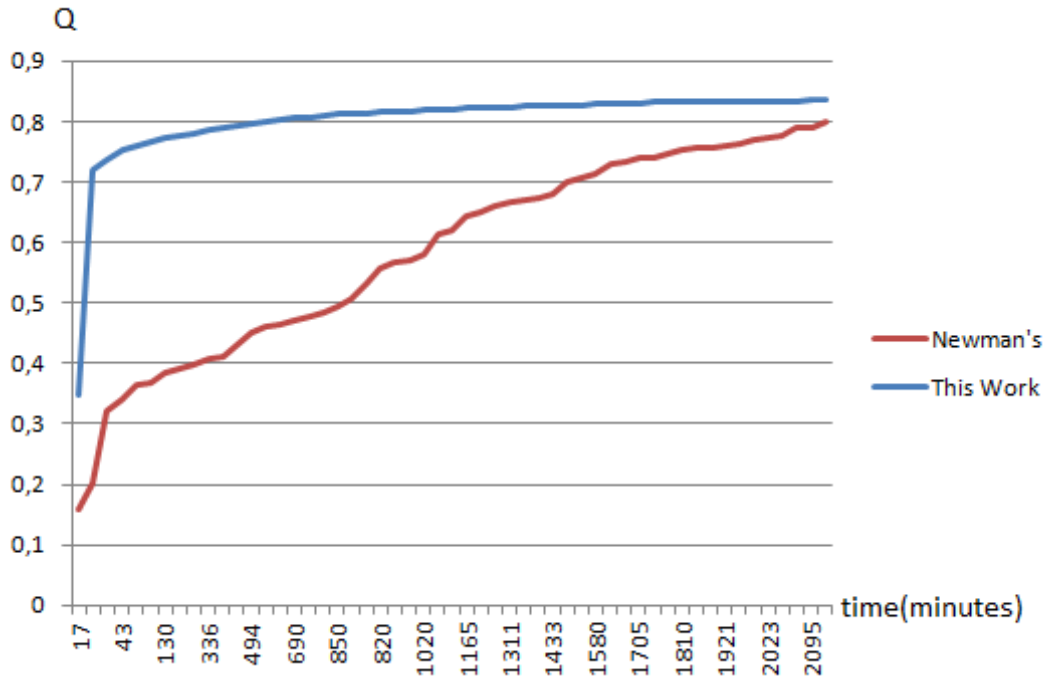

Figure 4.29: Time comparison on PGP Network

GN and NSA run for one hour and the indivisible communities found during the first one hour are used as input. Because of GN is not being applied to Condense Matters dataset, we use only results from NSA.

In the Figure 4.29, time comparison of the modularity scores between NSA and MGA with preprocess is made. MGA has taken 15 different communities randomly assigned to the communities from the community pool generated by running NSA and GACD of Tasgin et al for one hour. The snapshots of the PGP network are taken when the NSA divided it into smaller pieces. Modularity scores of the NSA and MGA with preprocess are compared at the times when the mentioned snapshots are taken. Our pool of communities consisting of 37 communities. Quality of these communities are found by dividing the edge number to the number of vertices in the community. In the Figure 4.30, it is shown how MGA with preprocess(experiments shown with numbers 1, 2, 3, 4) and MGA without preprocess step(experiment shown with number 5). The initial population for each experiments is taken as 100. In the Figure 4.30, the experiment shown by number one is taken 15 best communities, experiment with number two is taken 30 best communities, experiment with number three is

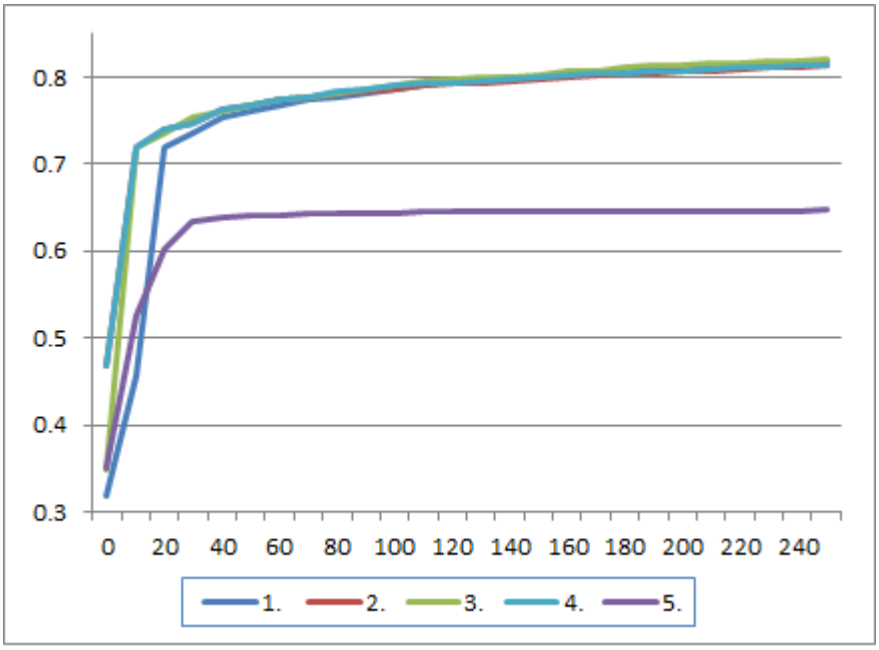

Figure 4.30: Comparison on PGP

taken 45 random communities, experiment with number five is taken 60 random communities and experiment with number five is not taken any community as input.

CHAPTER 5

CONCLUSIONS

In this thesis, a new method which is for the needs of detecting communities in social networks is studied. Several studies which focuses on the subjects like graph partitioning, clustering and community detection in networks and genetic algorithms are examined. Finally, a new method which uses a specialized version of the Genetic Algorithms which is specialized on detecting communities in social networks is proposed. Our proposed method does not need the information of how many communities are to be generated and it is not given as an input. Vertices in the networks form together and create communities to maximize the fitness score.

In this method, we inspired from Falkenauer's presentation of grouping genetic algorithms and we shifted our focus from individuals of the graph to the communities of the graph, and used an encoding that keeping communities as genes of the chromosomes and keeping individuals as the data of the genes. Additionally, for large real network data, a preprocess step which makes use of the community structure outputs produced from running of other algorithms are used. With the preprocess step, we planned to take advantage of *building blocks* for a more rapid run and accurate results. To evaluate the success of the algorithm, Modularity Q is used as fitness function.

First, we examined our algorithm with other genetic algorithms: Grouping Genetic Algorithm of Tasgin et al and Traditional Genetic Algorithm. The tests are done without using a preprocess step to focus on the performance of the genetic algorithms and to show how our way of encoding and operators perform. Results were satisfactory that our method finds community structures faster and

reaches the top modularity value sooner than other two methods. We compared how Modularity Q values changed through the iterations and changed through the time passes. Both of these comparisons show that our modified genetic algorithm is superior for community detection to GACD of Tasgin et al and the Traditional Genetic Algorithm even without using a preprocess step.

Our method's comparison with other algorithms shown in section 4.5 that all of them are using different approaches but using modularity Q in common to evaluate quality of the community structure of the network. As it is seen, our method has better results when compared to others. To improve the modularity of the network more, we used a preprocess step. Since it gives building blocks to solution space, it enhances the modularity scores of the networks. After the experiments, we see that number of the building blocks is not as effective as the quality of the building blocks for our method. As a result, our method is an effective way of using building blocks.

There are some disadvantages of our method. Since it is a genetic algorithm, it finds the optimal solution eventually, as it is expected. Also it has rapid refinements. However, its refinement gets slower when it gets closer to the peak points, that we can refer as saturation points. In future works, we will look for refinements to overcome the saturation points quicker. Plus, implementation of parallel programming to our methods can speed up the running time. Furthermore, we have an opinion that our encoding might be a suitable solution for the networks which are consisting of overlapping communities and future experiments will be done to observe our method's efficiency on these kind of networks.

REFERENCES

[1] Maciej Jedynak-Julian Sienkiewicz Agata Fronczak, Janusz A. Holyst. Higher order clustering coefficients in barabasi-albert networks. *Physica A: Statistical Mechanics and its Applications*, 316(1-4):688–694, 2002.

[2] M. Barthelemy Barrat, A. and A. Vespignani. *Dynamical processes on complex networks*. Cambridge University Press, Cambridge, UK, 2008.

[3] Vincent D Blondel, Jean loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[4] B. Bollobás. *Modern Graph Theory, Graduate Texts in Mathematics*. Springer, New York, 1998.

[5] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On finding graph clusterings with maximum modularity. In *Graph-Theoretic Concepts in Computer Science*, volume 4769 of *Lecture Notes in Computer Science*, pages 121–132. Springer Berlin Heidelberg, 2007.

[6] Martin V. Butz, Kumara Sastry, and David E. Goldberg. Tournament selection in xcs. Technical report, Proceedings Of The Fifth Genetic and Evolutionary Computation Conference (GECCO-2003, 2002.

[7] Kaiming X. Baoxin X. Guodong-L. Changjun, F. A fuzzy clustering algorithm to detect criminals without prior informations. In *Advances in Social Networks Analysis and Mining (ASONAM)*, pages 238–243. IEEE Computer Society, 2014.

[8] Moore C. Clauset A., Newman M. E. J. Finding community structure in very large networks. *Phys. Rev. E 70, 066111 (2004)*.

[9] D. Crane. *Invisible colleges: Diffusion of knowledge in scientific communities*. University of Chicago Press, Chicago, 1972.

[10] Charles Darwin. *On the Origin of Species*. John Murray, 1859.

[11] Peter Sheridan Dodds, Roby Muhamad, and Duncan J. Watts. An experimental study of search in global social networks. *Science*, 301:827–829, 2003.

[12] Arenas A. Duch J. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2), 2005.

[13] L. Egghe and R. Rousseau. *Introduction to Informetrics*. Elsevier, Amsterdam, 1990.

[14] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Wiley Publishing, 4th edition, 2009.

[15] F. Cecconi V. Loreto F. Radicchi, C. Castellano and D. Parisi. Defining and identifying communities in networks. *PNAS*, 101(9):2658–2663, 2004.

[16] Emanuel Falkenauer. *Genetic Algorithms and Grouping Problems*. John Wiley, 1998.

[17] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[18] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 5(1), January 1994.

[19] Santo Fortunato. Community detection in graphs. *Physics Reports*, (486):75–174, 2010.

[20] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[21] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, page 215, 1978.

[22] A. Vespignani G. Caldarelli, R. Pastor-Satorras. Structure of cycles and local ordering in complex networks. *The European Physical Journal B - Condensed Matter*, 38(2):183–186, 2004.

[23] C. Ma G. Gan and J. Wu. *Data clustering: Theory, algorithms, and applications*. Society for Industrial and Applied Mathematics, Philadelphia, 2007.

[24] Illés Farkas Tamás Vicsek Gergely Palla, Imre Derényi. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[25] Andrea Capocci Ginestra Bianconi. Number of loops of size h in growing scale-free networks. *Phys. Rev. Lett.*, 90, 2003.

[26] Newman M.E.J. Girvan, M. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[27] Newman M.E.J. Girvan, M. Finding and evaluating community structure in networks. *Physical Review E*, 2(69), 2004.

[28] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[29] R. Albert Z. N. Oltvai-A.-L. Barabási H. Jeong, B. Tombor. The large-scale organization of metabolic networks. *Nature*, 407:651–655, 2000.

[30] Ronald L. Breiger Harrison C. White, Scott A. Boorman. Social structure from multiple networks. i. blockmodels of roles and positions. *The American Journal of Sociology*, 81(4):730–780, 1977.

[31] John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[32] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, sep 1999.

[33] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Sys. Tech. J.*, 49(2):291–308, 1970.

[34] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network, 2008.

[35] A. Diaz-Guilera M. Boguna, R. Pastor-Satorras and A. Arenas. Pgp network. *Physical Review E*, 70, 2004.

[36] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, Berkeley, Calif., 1967.

[37] Leskovec J. McAuley J. Learning to discover social circles in ego networks. *NIPS*, 2012.

[38] J. F. F. Mendes and S. N. Dorogovtsev. *Evolution of Networks: from biological nets to the Internet and WWW.* Oxford University Press, Oxford, UK, 2003.

[39] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[40] M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Phys. Rev. E*, 64:016131, Jun 2001.

[41] M. E. J. Newman. Ego-centered networks and the ripple effect. *Social Networks*, 25:83–95, 2003.

[42] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[43] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(23):8577–8582, 2006.

[44] M. E. J. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.

[45] M.E.J. Newman. Mixing patterns in networks. *Phys. Rev. E 67, 026126 (2003)*.

[46] M.E.J. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[47] Houssem Assadi Nicolas Pissard. Detecting overlapping communities in linear time with pa algorithm, 2005.

[48] F. Ozgul, Z. Erdem, C. Bowerman, and J. Bondy. Combined detection model for criminal network detection. In *Proceedings of the 2010 Pacific Asia Conference on Intelligence and Security Informatics*, PAISI'10, pages 1–14. Springer-Verlag, 2010.

[49] A. Rényi P. Erdös. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[50] M. Huss P. Holme and H. Jeong. Subnetwork hierarchies of biochemical pathways. *Bioinformatics*, 19:532–538, 2003.

[51] R. Pastor-Satorras and A. Vespignani. *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, New York, NY, USA, 2004.

[52] P.Gleiser and L. Danon. Community structure in jazz. *Advances in Complex Systems*, 6(4):565–573, 2003.

[53] Carlos André Reis Pinheiro. Community detection to identify fraud events in telecommunications networks. In *Proceedings of the SAS® Global Forum 2012 Conference*, April 2012.

[54] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, may 1990.

[55] A. Diaz-Guilera-F. Giralt R. Guimera, L. Danon and A. Arenas. Self-similar community structure in organisations. *Physical Review E*, 68, 2003.

[56] Albert-Laszlo Barabasi Reka Albert. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(47), 2002.

[57] Phipps Arabie Ronald L. Breiger, Scott A. Boorman. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of Mathematical Psychology*, 12(3), 1975.

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

[59] Claudio Castellano Santo Fortunato. *Encyclopedia of Complexity and Systems Science*. Springer, Berlin, Germany, 2009.

[60] J. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, London, UK, 2000.

[61] P. J. Mucha T. Richardson and M. A. Porter. Spectral tripartitioning of networks. *Physical Review Letters E*, 2009.

[62] Jiliang Tang, Xia Hu, and Huan Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013.

[63] Bingol H. Tasgin, M. Community detection in complex networks using genetic algorithms. *PNAS*, 103(23):8577–8582, 2006.

[64] M. Tasgin. Community detection model using genetic algorithm in complex networks and its application in real life networks, 2006.

[65] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[66] Massimo Marchiori Vito Latora. Economic small-world behavior in weighted networks. *Euro Physics Journal B*, 32(2):249–263, 2003.

[67] L. Wangqun, K. Xiangnan, Y. Philip, S., W. Quanyuan, J. Yan, and L. Chuan. Community detection in incomplete information networks. In *WWW 2012 – Session: Community Detection in Social Networks*, pages 16–20, April 2012.

[68] S. Wasserman and K. Faust. *Social network analysis*. Cambridge University Press, Cambridge, UK, 1994.

[69] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small world networks. *Nature*, 393:440–442, 1998.

[70] C. Weinstein, W. Campbell, Brian Delaney, and G. O'Leary. Modeling and detection techniques for counter-terror social network analysis and intent recognition. In *Aerospace conference, 2009 IEEE*, pages 1–16, March 2009.

[71] W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.