Community Detection in Complex Networks Using Genetic Algorithms: A Comparative Analysis

Preston Engstrom

Submitted in partial fulfillment
of the requirements for the degree of

Undergraduate of Science

Department of Computer Science
Brock University
St. Catharines, Ontario

0.1 Abstract

Community detection in complex networks has been one of the most popular topics in several fields. Communities, also referred to as clusters, are typically defined as subgraphs with a higher probability of sharing edges with nodes within that subgraph that the rest of the graph. Several algorithms for identifying these structures have been developed. Genetic algorithms have long been used to tackle large scale combinatorial optimization problems, and have appeared sporadically throughout the literature for them community detection problem. For this work, four genetic approaches have been selected, using a variety of representations, fitness measures and clustering approaches. After implementing the designs as they are described, they were tested against a series of benchmarks. These include the Girvan-newmann and LFR synthetic benchmarks, random graphs, and several well studied real world networks. The obtained clusterings are compared using several measures of clustering accuracy, Variation of information, Adjusted Rand index, and normalized mutual information. The performance of the selected approaches show that GA's can be fielded as an effective approach in estimating community structures on real data.

Acknowledgements

I would like to thank Doctor Beatrice Ombuki for her continuous support, in this project and throughout my academic and professional endeavors.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Networks have long been used as a method for representing relations in data. Graph theory has its origins in Euler's 1736 paper proving a solution to the Königsberg's bridges problem. As research has been done into the properties of graphs, they have proven to be an excellent way of representing a wide variety of systems in many fields. Social, biological, communications and information can be represented as graphs, with many other fields of study making use of them as well. With the use of these representations in modern applications, their size and complexity has grown immensely. Where in the past a sociologist may have curated a dataset of social connections by hand, data can now be scraped from global social networking platforms at scales of millions to billions of nodes.

One of the main structures of interest in such networks are communities, also referred to as modules or clusters. These are densely connected groups of nodes, which indicate some inherent order in the network. They can show groups of friends or coworkers in social networks, closely related protein-protein interactions, authors of papers working in similar fields and other explainable patterns in seemingly chaotic networks. Several approaches have been implemented to tackling the task of *community detection*. Genetic algorithms have shown to be fairly efficacious at discrete optimization tasks, and have been applied to graph clustering and related problems. In this study, we implement four genetic algorithms from literature and compare their performance against real world data and synthetic benchmarks to evaluate their effectiveness.

1.1 The Problem of Community Detection in Complex Networks

Community detection in networks is analogous to clustering in vector data [21]. Given a network, the goal is to identify groups of nodes that represent tightly connected structures in a larger graph. These tightly connected components are common in real networks and are vital in the study of these systems, particularly at massive scales.

1.2 Thesis Structure

The organization of this thesis is as follows. Chapter 2 introduces the foundational terms and concepts in graph theory and complex networks, and a summary of approaches to the problem of community detection and related tasks. This is followed by a brief introduction to genetic algorithms. Chapter 3 contains a more in depth analysis of previous research done on the problem of community detection, with sections introducing each of the algorithms selected for study. The methodology used for comparing the selected algorithms, including the details regarding libraries, data sources and benchmarks is covered in chapter 4. Two types of data were used to test the algorithms, which can be classified as real world and synthetic networks. These two categories are each reported in chapters 5 and 6. Chapter 7 contains discussion regarding obtained results and exploration of ideas for future work.

Chapter 2

Background

The study of networks is common across several scientific disciplines, including computer science, mathematics, physics, biology, statistics and sociology. In order to effectively study and work with this data, a general understanding of the concepts of graph theory and complex networks is required. This chapter serves as jumping off point for the computer science student looking to become for familiar with these critical definitions and concepts. This chapter also introduces the fundamentals of genetic algorithms, and the concept of optimization through population based metaheuristics.

2.1 Elements of Graph Theory

A *Graph* G is composed of two sets, (V,E), where V is the set of *vertices*, or *nodes* and E is the set of unordered pairs of elements of V. If these pairs are ordered, the graph is *directed*. Elements of E are referred to as edges. An edge connects two nodes. Nodes associated with a particular edge are called *endpoints*.

A network can be visualized simply as nodes, drawn as points, connected by their corresponding edges. Note that graphs do not include self edges, called *loops* or more than one edge joining the same pair of vertices. However, *multigraphs* are a generalization which allow these constructs.

A *subgraph* of $G = (V, E)$ is denoted as $G' = (V', E')$ if $V' \subset V$ and $V' \subset E$. A subgraph which contains edges such that all $v \in V$ are endpoints is a *spanning subgraph* of G. A partition of a vertex set V into two subsets S and $V - S$ is called a cut. The *cut size* is the number of edges connecting members of S to $V - S$ A *path* is a finite set of edges that connect two vertices. If, for some vertices in G no path exists, G is a *disconnected* graph.

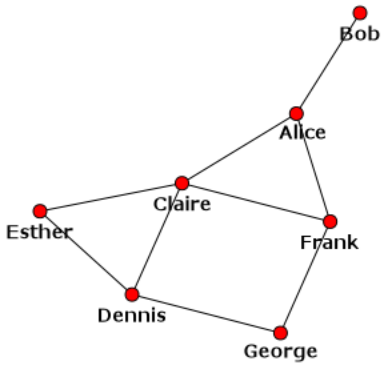


Figure 2.1: An undirected graph with 7 nodes and 9 edges.

The number of vertices and edges of a graph are indicated with n and m. n is the order of a graph, and m is the size. The maximum size of a graph if order n is $n(n-1)/2$, the number of unordered pairs of vertices. If G is of maximum size, each vertex having an edge to every other, it is a *complete graph*. A fully connected subgraph is called a *clique*. Endpoints are referred to as *neighbors*, or *adjacent nodes*. The the set of neighbors of vertex v is the *neighborhood*, indicated as $\Gamma(v)$. $|\Gamma(v)|$, the number of neighbors of v, is the degree of v, k_v. The *degree sequence* of G is the list of degrees for each vertex of G, $k_{v1}, k_{v2}, \ldots, k_{vn}$. A regular graph is one whose degree sequence contains only copies of a single integer. Directed graphs distinguish between the *indegree* and *outdegree*, the number of edges ending at v, and beginning at v respectively. While in this paper we limit experiments to unweighted graphs, it is important to understand the extension of edges from being counted as binary entities, to ones with real values. These edge values are called *weights*. The analog to degree of v in these graphs is *strength*, the sum of edge weights adjacent to v.

The *internal degree* and *external degree*, k_i^{int} and k_i^{ext}, with respect to node i being a member of subgraph $C \subset G$, are the sum of edges connecting i to nodes in C and not contained in C respectively. Another useful measure on vertices is *clustering* [27], or *transitivity* to avoid confusion when dealing with structural clusters. C_v, the transitivity of v, is the ratio of edges joining pairs of nodes in $\Gamma(v)$, to the total possible number of edges between nodes in $\Gamma(v)$. For node v, the transitivity is given as:

$$k_v(k_v - 1)/2$$

Another way of expressing the transitivity is in the number of triangles formed with v. Note that the clustering value of a graph does not measure how a graph may be

partitioned directly, as it correlates with the size and volume of the network [14]. Instead, it can be used as a normalized value to compare different networks in terms of how many triangles exist per node.

A common mathematical procedure to apply to graphs is a *random walk*. As the name implies, it is randomly generated set of steps along a path. Given a node, selected as the starting point, each step is making the transition from the current node i, to the next node j, given that $j \in \Gamma i$ with the probability $\frac{1}{k_i}$. Random walks have varied applications to problems in graph theory, and are a common feature of community detection methods [23].

2.1.1 The Adjacency Matrix

When dealing with graphs, one of the most effective ways to represent their structure, or *topology*, is an *adjacency matrix*. For a binary graph G, its adjacency matrix is an $n \times n$ matrix A. Where nodes i and j share an edge, A_{ij} is 1, and otherwise 0. This matrix is symmetric in the case of undirected graphs. The sum of the i-th row of A is the degree of node i. When The graph is weighted, it is indicated as W, where the element W_{ij} is the weight of the edge connecting adjacent nodes i and j.

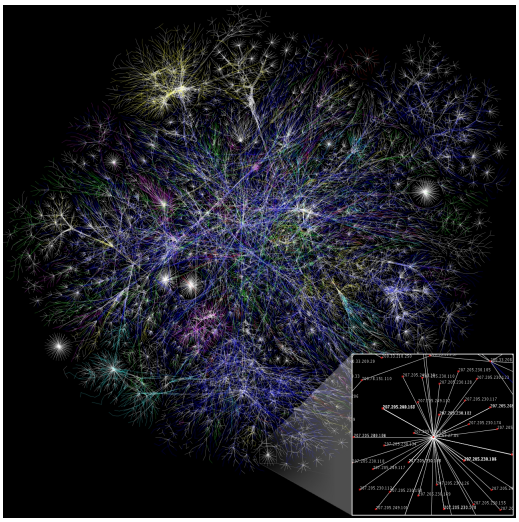


Figure 2.2: A section of the Internet, as represented by the Opte Project in 2005.

2.2 Complex Networks

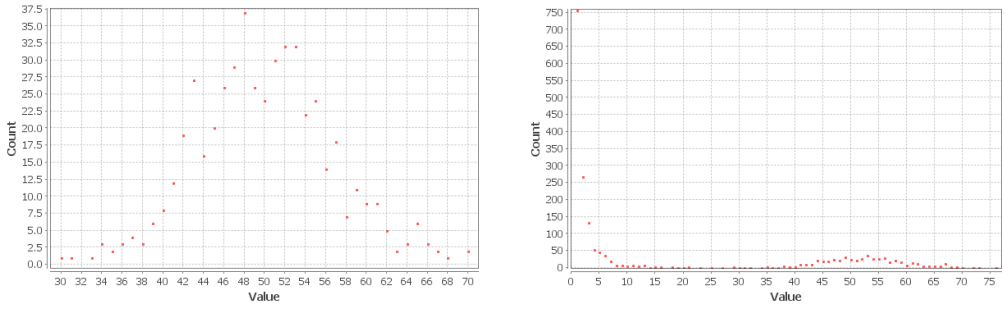
The great numbers of applications of graph theory have made it one of the core elements of several fields of study. With this abundance of interest, a shift has occurred away from understanding local properties in small graphs, to understanding the statistical properties of large systems. Graphs of natural systems do not follow the same patterns as regular or random graphs, instead exhibiting a mix of properties which show both order and noise. Its important to point out that what makes a network complex is the topologies developed as the network grows. Regular graphs are constructed following a set of conditions, however real networks are developed by the processes governing the system itself.

2.2.1 Properties of Networks

Scale Free Networks

Figure 2.3 shows the degree distributions of a random graph with 1000 nodes and a linkage probability of 0.01, and for a network of protein interactions in yeast. The average degree of the networks are 49.928 and 15.785 edges per node respectively. The distribution of the random graph is centered on the average degree, with a relatively small deviation. The yeast network demonstrates the characteristic "long tail" seen in scale free networks, extending far away with decreasing numbers of nodes having much higher degrees. Many real networks show a similar property; the average degree is low, but there is a long tail to the distribution reaching far above the average. This manifests visually as dense hubs. One of the early observations of this fact was made in studies of citations of papers. Most are cited only a few times, while seminal publications are widely cited as the basis of many branching works. Similarly in social media, where nodes are individuals, connected by their relationships (liking, following etc.), there are a small subset of users with orders of magnitude more users than the average person. When the degree distribution is plotted, it is noted to follow a curve similar to a power law distribution. While this is a good analogy, a network does not need to follow a power law precisely to be considered scale free; generally, the existence a minority highly connected nodes, and the many closer to the average is seen as the scale free property.

Scale free graphs do not occur randomly. In real systems, central components of a system will connect to new, less developed ones. A user on social media in the top percentile by followers will gain followers at a much faster rate than the average



(a) Random Network (b) Yeast

Figure 2.3: The degree distribution of a random network of 1000 nodes (a), and of a network of protein interactions in yeast

person. This idea, that as nodes are added to a network they will more likely link to nodes at a rate proportional to their degree, is called *preferential attachment*. This has been described in model graphs before to generate scale free networks, the most popular being the *BA-model*[3].

The Small World Effect

A frequent observation in networks is the fact that most nodes are not in each others neighborhood, the distance most pairs of nodes is fairly small. Formally, they are defined as a network where the average path between any two nodes is scaled by the logarithm of the number of nodes. Practically, this manifests itself as the classical "six degrees of separation" phenomenon, originally proposed in the early analysis of social networks in [11]. The observation was made that vast swaths of the population were connected by only a small number of social links. Further study into other areas of networks have shown this principal to exist in many other areas. Websites, biological networks and citation networks are some of the most common examples. It has been proposed that the small world property in biological systems was evolved as a form of redundancy against damage to the system[4]. When a network has short paths between any pair of nodes, as well as a scale free degree distrobution, it is unlikely that removing any node will disrupt the connections between the remaining nodes, as the mean-shortest path length will change very little. Random graphs, however, will show a much more dramatic change in path lengths with the removal of an arbitrary node.

2.3 Community Detection In Graphs

When visualizing real world networks, it is common to see some structure in the noise. In social graphs, these may be close friend groups. In collaboration networks, groups of researchers may interact closely to those in related fields, but rarely with those more removed from their areas of interest. These *communities* can offer a better understanding to how the network is organized, and how these modules may or may not interact. It gives researchers a way to identify smaller sections of interest in networks for closer study. It offers a method to classify nodes based on their role, or relationship to the structure.

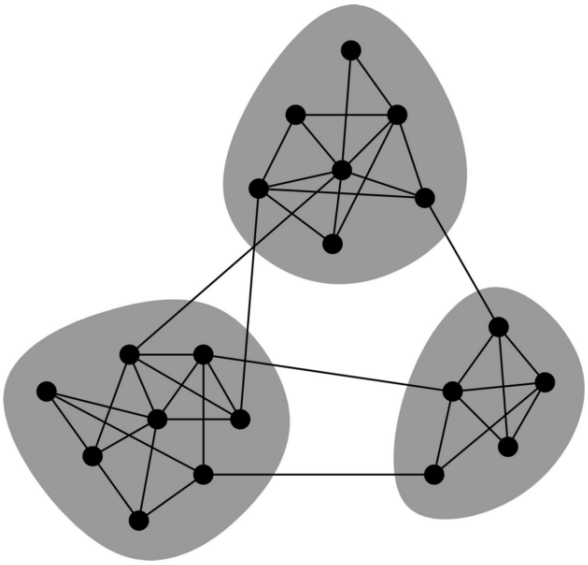


Figure 2.4: Each shaded area of the graph is a community component

The identification and study of community structure has three major roles [15]. First, it shows the organization of the graph at a courser level, as if its functional units were compressed. This understanding can allow for better insights into how the network was built, and how it may continue to evolve as a system. Secondly, it helps better explain dynamic processes as they take place. Examples like the spread of disease, or ideas over social media are considerably affected by the modular structure of the components of the system. Finally, it provides insight into relationships between entities of the system, which may not be apparent when viewing the system as a whole.

2.3.1 Defining Communities

The problem of community detection is not well defined, and no single approach is widely accepted. The main reason for this is not due to disagreement in techniques, but because it is observed that the success of a definition is sensitive to the domain of the application. A graph being used in the field of communications planning will be completely different in its properties to a network of protein-protein interactions. A primary intuition is that a community must contain more edges between members in that community, than between members and outside nodes. This is the starting point for community definitions.

One of the required conditions for a community to exist is *connectedness*. For a subgraph $\mathcal{C}$ to be a community, there must be a path connecting all pairs of nodes in $\mathcal{C}$, constructed only of edges internal to the community. When a disconnected graph is being analyzed for community structure, it is acceptable treat each connected component of the graph separately.

Given a subgraph $\mathcal{C}$ of graph G, where $|\mathcal{C}| = n_c$ and $|G| = n$. The *internal degree* and *external degree*, k_v^{int} and k_v^{ext}, with respect to node v being a member of $\mathcal{C}$, are the sum of edges connecting i to nodes in$\mathcal{C}$ and not contained in $\mathcal{C}$ respectively. If $k_v^{ext} = 0$, there are no neighbors of v outside of C. In this case, v is an internal vertex of $\mathcal{C}$. This makes $\mathcal{C}$ likely a good cluster for v. Conversely, if $k_v^{int} = 0$, v is completely disjoint to $\mathcal{C}$, and should be considered for a different cluster. In the case where $k_v^{ext} > 0$ and $k_v^{int} > 0$, v is a *boundary node* of $\mathcal{C}$. The internal degree of a subgraph $\mathcal{C}$, $k_{int}^{\mathcal{C}}$ is the sum of the external degree of every vertex in $\mathcal{C}$. Likewise, the external degree of a subgraph $\mathcal{C}$ $k_{ext}^{\mathcal{C}}$ is the sum of the external degrees of all members of $\mathcal{C}$. The total degree of $\mathcal{C}$ is the sum of the degrees of all vertices in $\mathcal{C}$, $k^{\mathcal{C}} = k_{int}^{\mathcal{C}} + k_{ext}^{\mathcal{C}}$.

The *embeddedness* of a node v, shown as ξ_v is the ratio of the internal degree to the total degree, k_v^{int}/k_v. A high embeddedness indicates a strong relationship between the vertex and its community.

2.3.2 Modularity

2.3.3 Complexity

Given the size of networks of interest in community detection applications, a careful consideration of the running time of a particular solution is vital. Graph clustering, and most problems related to it, are **NP-hard**[31, 7]. Knowing this, it is inadvisable to search iteratively for the optimal solution. Instead, non-deterministic approxima-

tion algorithms are the norm.

2.3.4 Generating Graphs with Community Structure

The standard models of random graphs are not capable of generating networks with meaningful communities. These types of graphs are instead built using a model generator called the *planted-ℓ partition* model.

2.4 Applications of Community Detection

The detection of community structure in networks has a broad range of applications. As with more traditional clustering methods, the goal is to infer some relationship between elements of a dataset, where explicit divisions of groups are not present. The focus of research is often only on the development and testing of new techniques, with applications to real cases being relatively uncommon as a topic of close investigation. This section is intended to introduce the problem in a context beyond the benchmarks and real world networks used commonly used.

2.4.1 Social

The most common application in literature is community detection in social networks. The all encompassing presence of social media comes with vast amounts of available data, at scales of millions to billions of nodes. The ability to collect data from sites like Facebook, Twitter, Youtube and other social networks drives the volume of work done, with several studies presenting applications and approaches to handling the data at scale [5, 30, 9, 19, 20]. Communities in social networks provide insight into the communities individuals immerse themselves in, and how those communities interact as larger entities. Social networks also commonly exhibit hierarchical community structure. Large communities, such as a university math and science department, typically contain smaller sub-communities of individual faculties or research groups.

2.4.2 Financial

The [16]

2.4.3 Biological

The advent of availability of data on genes, proteins and metabolic processes has spurred a great interest in computational approaches to biological problem spaces. Biological networks are massive. They also clearly exhibit modular structures, a high clustering coefficient and the small world property. It is postulated that this is an evolutionary failsafe. By having nodes tightly linked, with several redundant paths between them, a single node being removed will not destroy the systems ability to propagate information throughout. One of the primary applications of community detection on these networks, is the identification of genes relating to particular diseases.

2.5 Genetic Algorithms

The early decades of the study of computer science saw many computational strategies inspired by the natural world. In the 1950s and 1960s, a popular idea was to use the principals of Darwinian evolution to evolve a a set of candidate solutions to a problem, by performing some set of operations taken from natural processes. These ideas grew and became the field of *evolutionary computation*. Genetic algorithms, invented and developed by John Holland [12] are an abstraction of biological evolution.

Because genetic algorithms are composed of a group of individual solutions to a problem, they are an excellent candidate for parallelization and distributed compute paradigms. They have been shown to be very effective in the context of very large datasets for several applications[8, 1].

2.5.1 Motivation and Inspiration

Nontrivial problems have an expansive solution landscape, with many local optima. An exhaustive search is not possible in a reasonable amount of time, so solutions may not be required to be optimal. In most cases, an exact solution is not actually desirable, as the resources devoted to discovering it outweigh the benefit, while a "good enough" solution could be found in a fraction of the time. Genetic algorithms seek to develop a number of solutions in parallel, allowing the best components of each attempt to inform future ones. They have proven to be extremely efficacious in a number of optimization, rule induction and combinatorial problems, with a long history of academic interest and industrial applications.

As they are biologically inspired, it follows that the components of a GA are

written about using corresponding biological terms, although the constructs they refer to in the computational context are much simpler than their living counterparts.

Each and every organism is composed of cells. Every cell contains a set of genetic materials, *chromosomes*, which are the blueprint for the characteristics of the organism. A chromosome can be divided into genes, codes of DNA or RNA defining a function of a molecule. These manifest as traits such as blood type, hair and eye color, strength or intelligence. Depending on its traits, an individual may be more likely to generate offspring. The particular setting of a gene's value is called an allele. Each gene occupies a fixed position, called the locus, on the chromosome. The ability for life to reproduce itself and create future, surviving and multiplying generations is a generalization of what makes a species successful. However, in the natural world, populations are diverse. As reproduce via sexual reproduction, a mixing of genetic material of both parents takes place, described originally by Mendel's laws of genetic inheritance. Occasionally, a random external factor will change the coding of a gene or genes, resulting in a mutation.

This combination of genetic mixing and mutation, in conjunction with the fact that the fittest individuals are more likely to reproduce, providing more opportunity for their positive traits to be passed on, raising the fitness of future members of its species.

Genetic algorithms aim to apply these natural processes to develop approximal solutions to optimization problems. By transforming problem into a representation that allows the application of operators inspired by these natural processes, a framework for exploring complex search spaces to give good approximate solutions can be developed. Genetic algorithms are one such realization of this approach. While no single definition of GA exists, they are generally comprised of a few components: a population of solutions, and a set of operators that affect members of the population to combine and modify solutions.

The flow of a GA is straightforward. First, a set number of solutions are created as the initial population. This initialization method can be completely random, or guided to help aid convergence. For a set number of iterations, or until a stopping criteria is met, the population goes through several phases. First, each individual is evaluated based on some fitness function.

The representation of a chromosome is critical in describing how each individual describes the solution to the given problem. This design decision will influence what other operations can be performed, and will have a profound effect on the performance of the algorithm. Following the genetic inspiration described above, each chromosome,

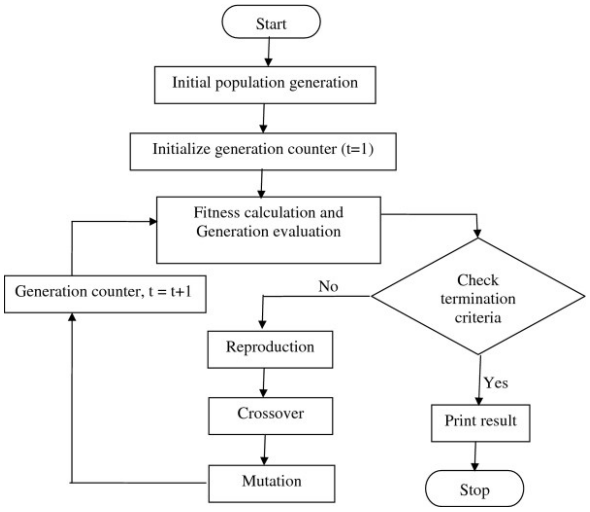


Figure 2.5: The flow of a typical genetic algorithm.

or individual consists of a sequence of genes. Each gene may hold a value contained in a genetic alphabet. This set of values was originally limited to binary digits in the original implementation by Holland, but can range from integers and strings, to more structured values such as matrices.

The algorithm runs for a set number of generations after the initial population is created. For each subsequent generation, there are several ways individuals may carry on to the next. Some portion of the best individuals may be directly copied, without any modification. This strategy is intended to carry the top solution forward without damaging it, and is called *elitism*. Another strategy is to instead, given some population of size μ, produce some number λ of new individuals and keep only the top μ of the union of the two groups. This is the $\mu + \lambda$ evolution strategy.

Selection

During the production of new next generation, parents must be selected from the existing population. The simplest solution would be to choose them at random, but this will generally result in the choice of a less than average individual. Instead, the probability of selecting an individual should be proportional to its evaluated fitness. *Roulette selection* involves evaluating each individual i's fitness f_i, and assigning a probability of selection $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$, where N is the number of individuals in the population.

Crossover

Given a pair, or some arbitrary number of selected individuals, crossover provides a mechanism to exchange components of several solutions. This is the rough approximation of biological reproduction, with parents contributing some portion of their genetic material. Two common examples are *uniform crossover*, and *one point crossover*. For uniform crossover, given a pair of parents with n genes, a random array of n bits is created. At indexes where there is a high bit, the corresponding genes in the parents will be swapped in the children. One point crossover begins by selecting an index of the parents chromosome. Every gene after the index is swapped in the new individuals. Several crossover methods exist, many specialized to the representation of a particular problem. The number of children created through this type of reproduction is controlled by the *crossover rate* variable. It dictates what fraction of the next generation should be composed of offspring, rather than direct copies of the previous generation.

Mutation

The mutation operator serves as a method to introduce a random of exploration of the search space. The *mutation rate* variable controls the chance of re-assigning a gene's value. Typically this is done by randomly assigning a new, valid allele.

2.5.2 Genetic Representations of Graphs

One of the most important aspect of any algorithmic approach, especially a GA, is how the problem is represented. A community clustering of a graph is composed of n vertices, so an individual should be able to produce a string of n integers. There are two such representations used in the literature.

The *string-of-groups* encoding is the first, representing each genome as an array of length n. For an individual S, the value of the gene $S_i = c$ indicates that node i is a member of community c. This arbitrary labeling however, leads to problems with traditional uniform and n-point crossover methods. Several approaches have been shown to work well with this representation by introducing novel crossover methods[26], but these issues spurred researchers toward developing more complex representations to reduce the need for new crossovers to be designed.

The second representation is the *locus adjacency representation*. It is a true graph based approach, as it relies on the topology of the network to enforce its structure. Each allele takes on an integer value, between 0 and n, where n is the number of

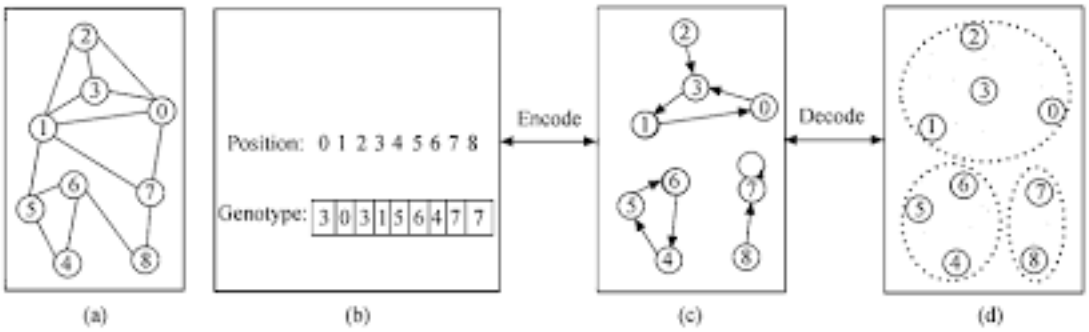


Figure 2.6: An example of the locus adjacency representation and decoded solution.

genes in the chromosome, which follows to be the number of nodes in the graph. The individual is not evaluated directly, but instead encodes a partitioning which can be extracted using a linear time decoding algorithm, described in Algorithm 1. The allele values give a path, from node to node, which produce a collection of subgraphs, each being a community. Figure 2.6 gives an example of a locus and its decoded solution. This representation comes with the condition that for the gene i of solution S, S_i to hold a value of j, nodes i and j must share and edge. A benefit of this condition is that any combination of individuals, representing the same graph, will yield a valid new individual.

Algorithm 1 Linear Time Locus Decoding‘

```
1: function DECODE(L, C)  ▷ Where L - locus array, C - cluster assignment array
2:     current_cluster = 1
3:     for each i in 1 to N do
4:         C_i = −1
5:     end for
6:     previous = Array[N]                    ▷ N is the number of nodes in the graph
7:     for each i in 1 to N do
8:         ctr = 1
9:         if C_i = −1 then
10:            C_i = current_cluster
11:            neighbor = L_i
12:            previous_ctr = i
13:            ctr = ctr + 1
14:            while C_neighbor = −1 do
15:                previous_ctr = neighbor
16:                C_neighbor = current_cluster
17:                neighbor = L_neighbor
18:                ctr = ctr + 1
19:                if C ≠ previous_ctr then
20:                    ctr = ctr − 1
21:                    while ctr ≥ 1 do
22:                        C_previous_ctr = C_neighbor
23:                        ctr = ctr − 1
24:                    end while
25:                else
26:                    C = current_cluster + 1
27:                end if
28:            end while
29:        end if
30:    end for          ▷ C now contains the the cluster assignment for each node
31: end function
```

Chapter 3

Literature Review

3.1 Methods of Community Detection

While the number of approaches for community detection and graph partitioning is vast, most applications employ one of a few popular methods. In this section we introduce other techniques beyond genetic algorithms, as well as other evolutionary methods not selected for this work.

3.1.1 Hierarchical Clustering

Hierarchical methods are the traditional method of community detection. Early implementations gained popularity by not requiring any previous knowledge of the network itself, a condition of algorithms of the related *graph partitioning* problem. In general, hierarchical clustering, applied to graph or more traditional vector data, allows the discovery of a multilevel sequence of clusters. In the case of graphs, this presents as small sub-communities composing a larger community.

3.1.2 Girvan and Newmann's Divisive Method

[10]

3.1.3 Spectral Clustering

3.1.4 Maximization Methods

3.1.5 Genetic Algortihms

[6]

3.2 Evaluating Community Detection Algorithms

Chapter 4

Methodology

Here we describe the selected algorithms and their parameters in detail. We also discuss the nature of the benchmarks and real world data, giving a summary of the range of tests to be performed. The main goal of these tests is to provide a broader scope of network data to each approach than has been presented in literature thus far, to validate the effectiveness of each method at scale.

4.1 Selected Algorithms

4.1.1 Tasgin-Bingol

One of the earliest implementations of a genetic algorithm for the network clustering problem [26], Tasgin and Bingol's approach is an example of one of the more naive approaches. Each individual is represented as an array of size n, each index corresponding to a node of the input graph. The population is initialized by assigning each gene a random integer, bounded by the number of nodes in the graph. A subset of genes, the size of which is defined by a parameter referred to as the *initialization rate* are selected, and propagate their community assignment to their neighbors. It utilizes modularity as its fitness function. Its mutation operator is simple. For the selected node i, select a member of $\Gamma(i)$, and set i to be in its community.

The authors describe a "one-way crossover", to tackle the main issue of the string of groups encoding; The same partition can be represented by multiple chromosomes. The crossover procedure is to randomly select a gene of the first parent, called the *source*. The genes of the parent in this community are then copied to the second parent, the *destination*, who lives on as the offspring. As mentioned in section 2, genetic operators on the string of groups representation can result in invalid partitions.

Parameter	Author's Default Value	Selected Default Value
Population Size	100-250	300
Generations	200-500	30
Elite Portion	Not reported	0.10
Mutation Rate	Not reported	0.10
Initialization Rate	Not reported	0.60
Cleaning Rate	Not reported	0.50
Cleaning Threshold	Not reported	0.70
Cleaning Portion	Not reported	0.50

Table 4.1: Parameters selected as the defaults for our implementation of Tasgin-Bingol's genetic algorithm for community detection, compared to the values reported by the authors

This is addressed by another operation introduced by the authors, the *cleanup* phase. In chapter 2, the concept of community variance is introduced. During the cleanup phase of the algorithm, some genes are randomly selected from a portion of individuals, and their community variance computed. If the community variance if above a parameter set by the user, that gene is assigned to the community most common to its neighbors. The algorithm was tested by the authors on limited data, with very high parameters for generations and relatively small population sizes. After achieving results similar to those reported in[26], we choose to use a much more standard set of default parameters, shown in table 4.1. Our implementation uses roulette selection, as the authors left their chosen selection method ambiguous.

4.1.2 GA-Net

One of the most widely cited examples of a GA approach to network clustering, GA-Net[22] is an approach that seeks to optimize a score other than modularity, *community score*, introduced in section 2.3. The individuals are represented using the locus adjacency representation. The initial population is generated randomly, with a guiding condition. When the gene for position i is set to value j, a test for the existence of an edge between nodes i and j is performed. Each gene's set of possible alleles is limited to the nodes in its neighborhood.

The community score fitness function introduced in the paper aims to treat the community detection problem as finding a partition of a graph into k subgraphs of maximum density, where k is not known. Given a graph G, with adjacency matrix A, let $S = (I, J)$ be a sub-matrix of A, with I being the subset of rows and J being the subset of columns of A. Let a_{iJ} be the mean value of the ith row, and a_{Ij} be the

Parameter	Selected Default Value
Population Size	300
Generations	30
Elite Portion	0.10
Crossover Rate	0.80
Mutation Rate	0.10
Power Value	1.5

Table 4.2: Parameters selected as the defaults for our implementation of GA-Net

mean of the jth column of S

$$a_{iJ} = \frac{1}{|J|}\Sigma_{j\in J}a_{ij} \quad \text{and} \quad a_{Ij} = \frac{1}{|I|}\Sigma_{i\in I}a_{ij}$$

The *volume* of a graph, recall from section 2, is the total number of edges. Thus the volume of a submatrix $S = (I, J)$ is the number of connections between nodes of S. The power mean of S of order r, $\mathbf{M}(S)$ is

$$\mathbf{M}(S) = \frac{\Sigma_{i\in I}(a_{iJ})^r}{|I|}$$

The *score* of S $Q(S)$ is defined as $\mathbf{M}(S) \times v_s$. Given a graph partitioned into communities, being a set of sub-matrices $\{S_1, \dots S_k\}$, the *community score* of the partitioning is

$$CS = \sum_{i}^{k} Q(S_i)$$

As the size of I increases, or as the number of internal links in communities decreases, the value r plays the role of scaling the value of the sub-matrix. The authors found that raising this parameter helps GA-Net perform as the definition of communities becomes weaker.

GA-Net employs a variation of uniform crossover, where two parents are chosen via roulette selection. These parents, A and B, each have a portion of their genes used to produce a single new offspring C. A random bit vector is created, with genes selected from A where the bit is 1, and genes selected from B when it is 0.

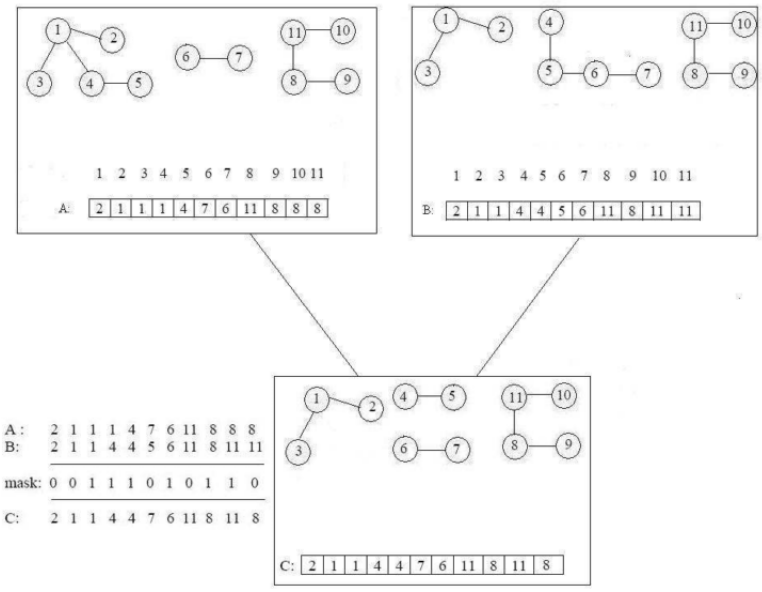


Figure 4.1: An example of uniform crossover on two locus adjacency chromosomes

4.1.3 GACD

[25]

4.1.4 GALS

GALS[**?**]

4.2 The Girvan-Newman Benchmark

Girvan and Newman introduce a benchmark which is a special case of the planted ℓ-partition model[10]. The graph is generated by creating 128 nodes, divided into 4 equally sized communities. Each node has an expected degree of 16. By setting the expected k_{in} and k_{out}, the difficulty of discerning the partition can be tuned. With an expected $k_{out} < 8$, the communities are strongly defined. It should be expected that a well defined method should discern the structure with a fair degree of accuracy.

The GN benchmarks limited size makes it an excellent model candidate for testing algorithms in early stages for the ability to detect loosely connected communities, but may not demonstrate the performance of a method when the input data is scaled up[28].

4.3 The LFR Benchmark

Parameter	Description
N	Nodes
k	Average Degree
max_k	Maximum Degree
mu	Mixing Parameter
min_c	Minimum Community Size
max_c	Maximum Community Size

Table 4.3: The parameters of the LFR benchmark generator

All synthetic networks were created using Andrea Lancichinetti's benchmark generation application.

4.4 Clustering Quality Measures

The goal of this analysis is to compare the performance of the selected algorithms on a sufficient variety of networks, with sufficient complexity. The collection of generated and real world networks has been selected to reflect other comparative studies, and to extend the observations made in the literature proposing the algorithms. These networks come with ground truth, describing the community structure as it was generated, in the case of the synthetic networks, or as it is generally accepted when using real world data.

When comparing the generated community partitions, we use several measures

4.4.1 Variation of Information

[18]

4.4.2 Normalized Mutual Information

The Normalized Mutual Information (NMI) of a community partition

$$I(A,B) = \frac{-2\Sigma_{i=1}^{c_A}\Sigma_{j=1}^{c_B}C_{ij}log(C_{ij}M/C_iC_j)}{\Sigma_{i=1}^{c_A}C_ilog(C_i/N) + \Sigma_{j=1}^{c_B}C_jlog(C_j/N)}$$

4.4.3 Rand and Adjusted Rand Index

[24]

4.5 Comparing Performance

Chapter 5

Experiments on Real Networks

Networks collected from real world data are often used as a proving ground for community detection methods. Rather than rely on communities being assigned by the model generating the network, community structure is taken from *metadata*, observed information about the network outside of its structural information. This metadata can be taken from observed interactions of individuals, membership of a particular group on a social network, or behavior patterns. It should be noted that the assumption that the metadata represents the true community structure is not always warranted. The common, small scale networks typically used in benchmarking and comparison may not give the full picture of an algorithms behaviors, so it is critical that more robust testing must be done. In these tests, the goal is to show that the algorithms do indeed extract a coherent community structure comparable to to other methods.

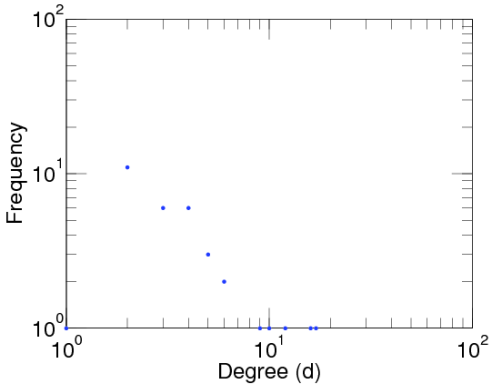
Network	**Nodes**	**Edges**	**True Communities (Metadata)**
Zachary's Karate Club	34	78	2
Dolphin	62	159	4
Football	115	613	12
Political Blogs	1493	19091	2
Political Blogs, simplified	1224	19090	2

Table 5.1: Summary of real world networks

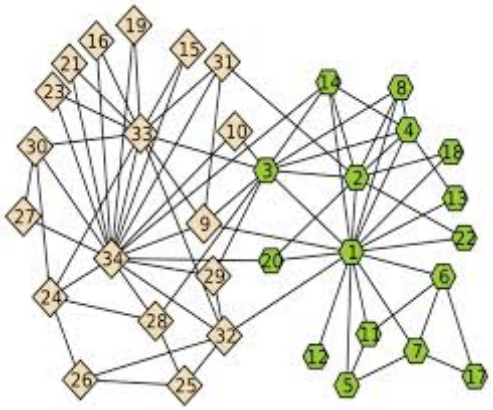
5.0.1 Zachary's Karate Club

Zachary's Karate Club [29] is one of the most widely used networks used to show an algorithm can effectively identify a set of communities. The graph shows the reported

relationships between members of a martial arts club, after a schism was formed by a conflict between the lead instructor and the owner of the club. Though small, the graph shows a degree distribution with similar properties to larger examples. Both the instructor and owner are densely connected to the club members who sided with them, with smaller sub-groups forming between members.



(a) Karate network degree distribution



(b) True communities of the karate network

Figure 5.1: Degree distribution of the karate club network[14]

5.0.2 Dolphins

[17]

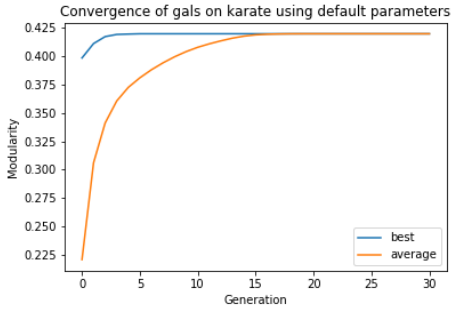
5.0.3 College Football

[10]

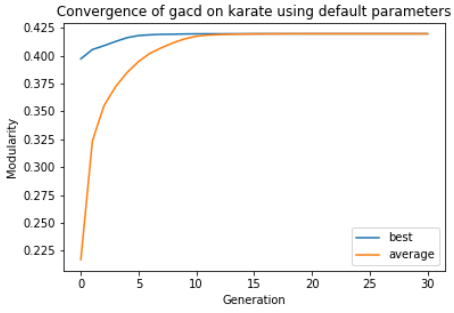
5.0.4 Political Blogs

While the experiments performed have been limited to undirected, unweighted networks, these GA approaches also apply to directed unweighted graphs with little change to the implementation. The Political blogs dataset[2] is an observed network, constructed from hyperlinks between blogs focusing on political topics. The data was collected during the course of the United States 2004 federal election campaign. Based on the content of each website, they have been hand labeled as leaning to the left or right of the political spectrum.

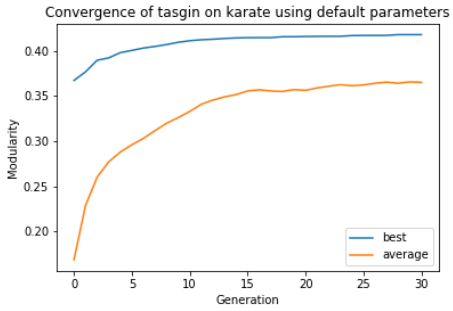
The network contains multiple disconnected components of only one node, as well as self-edges, or loops. While a generated network would not assign a single



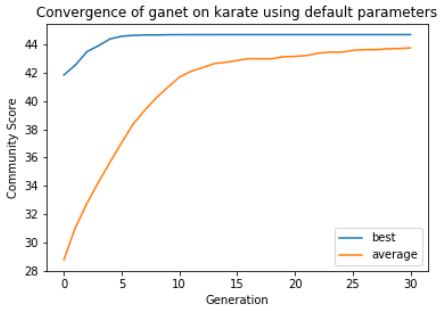
(a) GALS



(b) GACD



(c) Tasgin



(d) GA-Net

Figure 5.2: The average and best fitness, for each generation of the selected algorithms on 20 runs on the karate club network

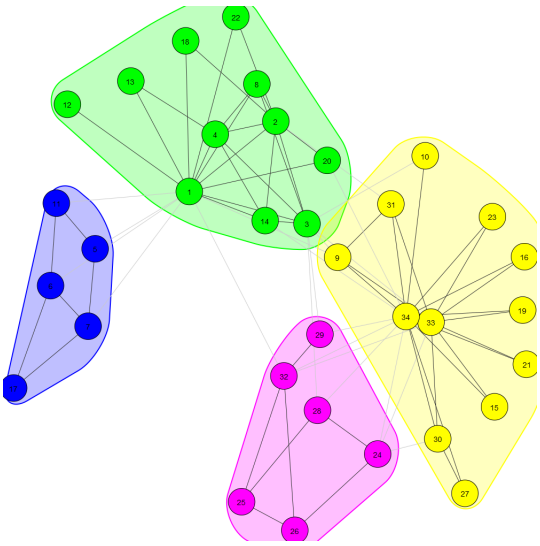


Figure 5.3: Community structure of the karate network discovered by GALS.

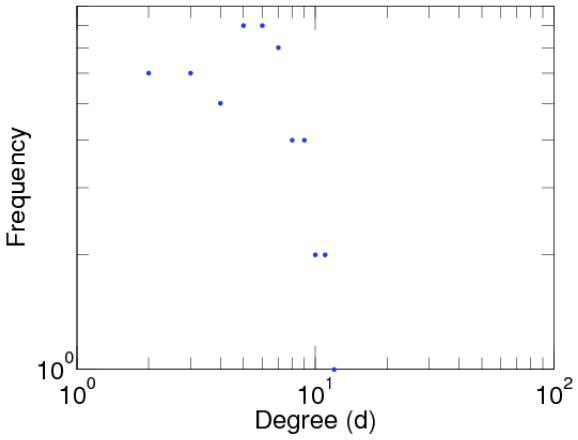


Figure 5.4: Degree distribution of the Dolphins network[14]

disconnected component a community membership that would be impossible to fulfill, the blogs with no connection to another are still labeled, and will adversely affect the results. For completeness, we present the outputs of both the original network, and a simplified version with no singleton nodes or self edges.

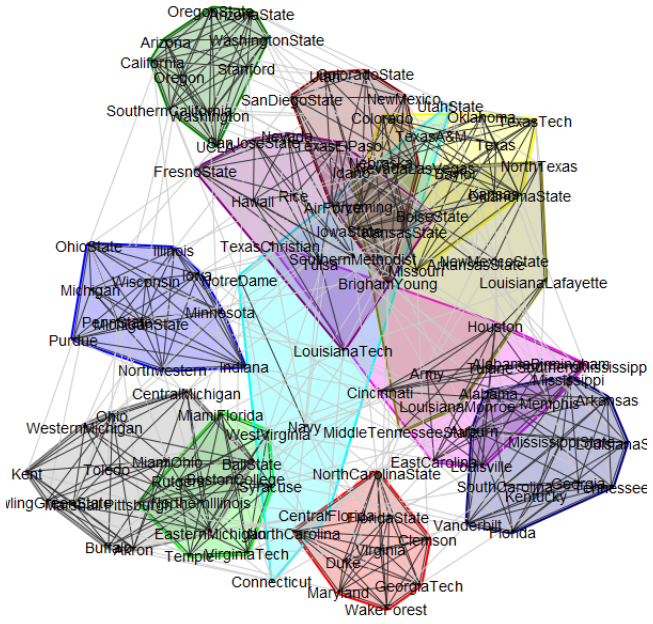
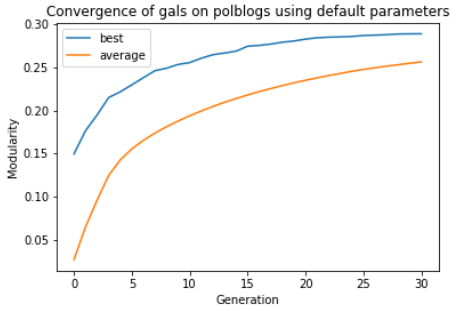
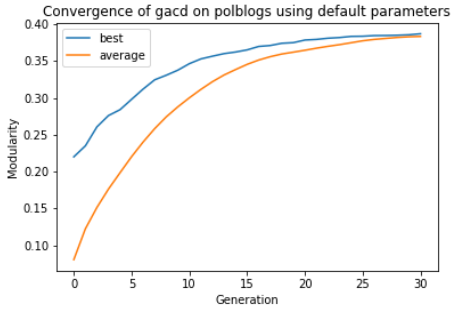


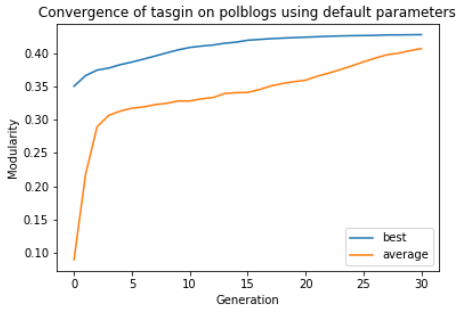
Figure 5.5: The College Football network, with conferences grouped as communities



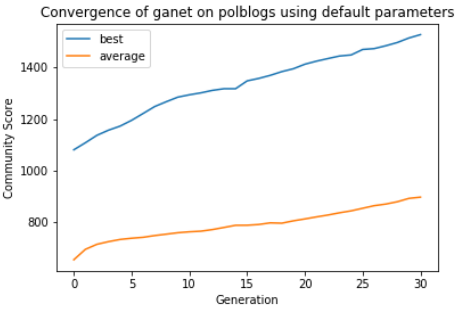
(a) GALS



(b) GACD

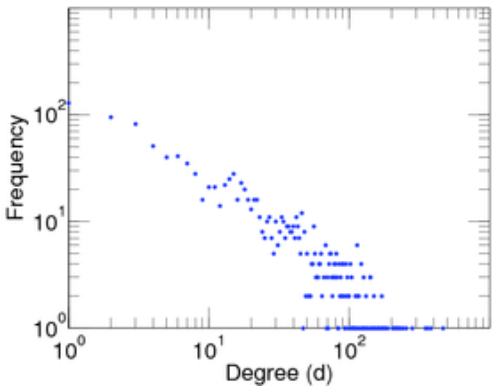


(c) Tasgin

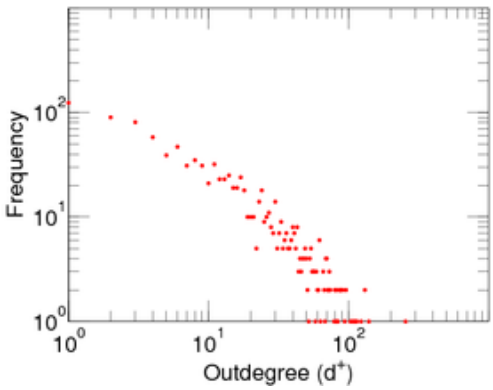


(d) GA-Net

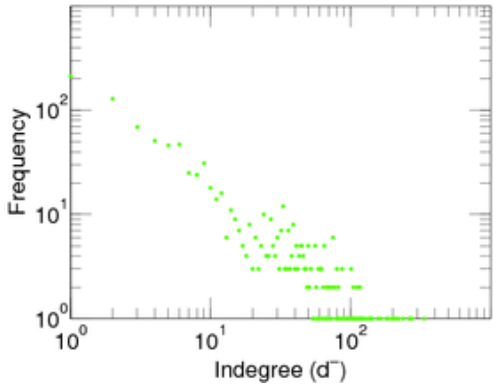
Figure 5.6: The average and best fitness, for each generation of the selected algorithms on 20 runs on the political blogs network



(a) Total Degree



(b) Outdegree



(c) Indegree

Figure 5.7: The total degree, indegree and outdegree distribution for the political blogs network[14]

Chapter 6

Experiments on Synthetic Graphs

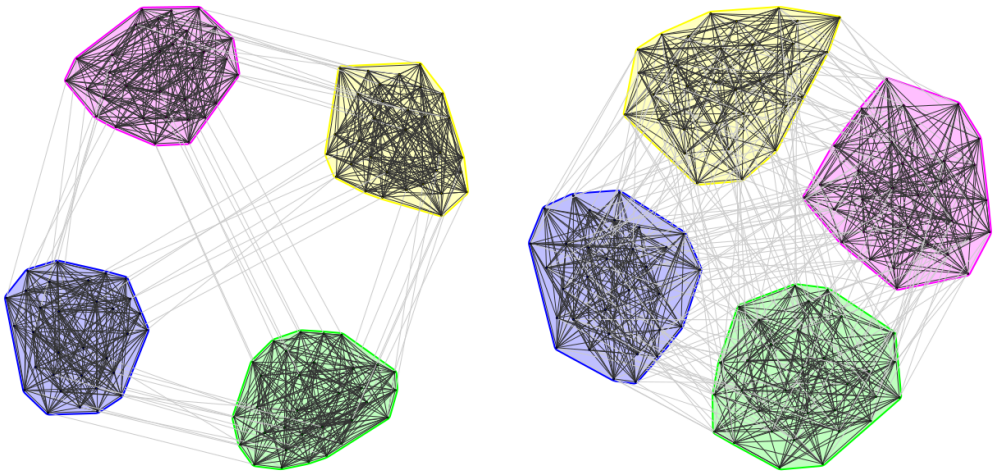
The methodology described is applied to two classes of synthetic networks. These generated benchmarks allow for the comparison to the ground truth communities, and also test how well each algorithm can extract community structures as the ratio of inter and intra edges increases, obfuscating the embedded classes. A test is also performed on networks of fixed inter/intra connection ratio, but with increasing size, to test the ability to extract communities effectively as the size of the network increases.

6.1 Girvan-Newmann Experiments

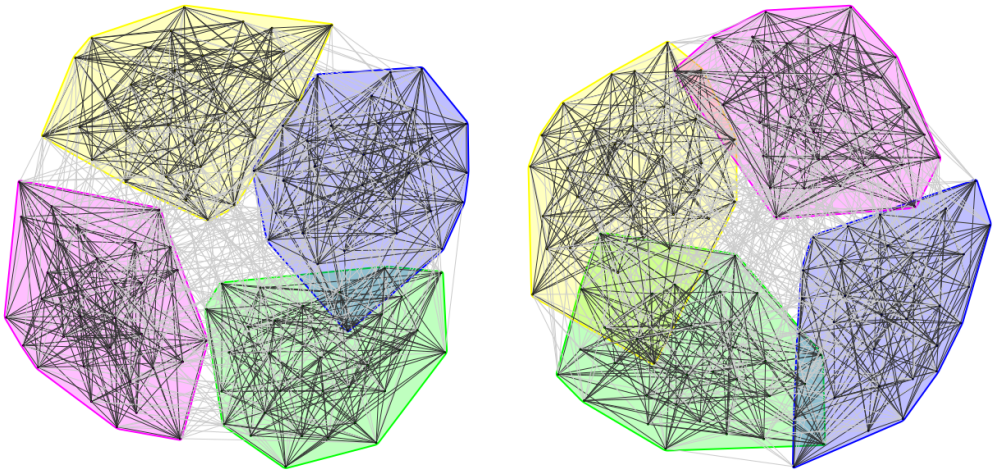
Here we present the results of each algorithm for a range of parameter sets on the Girvan-Newmann benchmark for a number of parameter sets for each of the selected algorithms. Graphs were constructed with the average external degree k_{out} of each node between 1 and 8. As the average total degree of a node in the Girvan-Newmann benchmark is 16, a value of $k_{out} \geq 8$ means that there is no strong community structure present.

6.2 LFR Benchmarks With Increasing Mixing Parameter

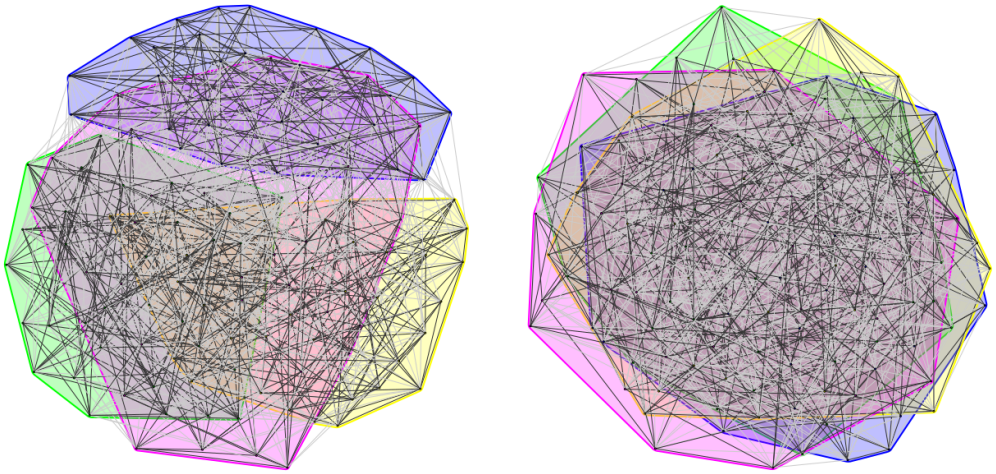
While the Girvan-Newmann benchmark allows for creating networks with increasingly difficult to identify communities, it is limited by its limits in terms of a fixed degree distribution, and no variance in community size. The LFR benchmark extends the planted ℓ-partition model to allow a degree distribution following the power law and



(a) $k_{out} = 1$ (b) $k_{out} = 3$



(c) $k_{out} = 5$ (d) $k_{out} = 6$



(c) $k_{out} = 8$ (d) $k_{out} = 10$

Figure 6.1: Visualizations of the GN benchmark with increasing k_{out} parameters.

a variety of community sizes.

Four sets of graphs were generated, with mixing parameters ranging from 0.1 to 0.6. The parameters used are summarized in table 6.1.

Group	N	k	max_k	min_c	max_c
1000_s	1000	20	50	10	50
1000_b	1000	20	50	10	100
5000_s	5000	20	50	10	50
5000_b	5000	20	50	10	100

Table 6.1: Parameters used to generate the LFR benchmarks. Each group was generated with the mixing parameter μ ranging from 0.1 to 0.6

6.3 LFR Benchmarks With Increasing Size

While an effective algorithm should detect communities even as the definition becomes more and more unclear from inter-community links, it should also be able to detect communities as the network scales in size. For this experiment, we use the parameter sets selected for each algorithm for the experiments on increasing mixing parameter for consistency.

Chapter 7

Conclusion

The continued interest in the study of community structures of complex networks has resulted in a multitude of approaches to the problem. Genetic algorithms have proved themselves to be effective in dealing with optimization problems by levering biologically inspired operators to explore a large search space. While GAs are a fairly under represented class of algorithms in the community detection literature, that by no means they are not worth further investigation.

In this work, four GAs for the community detection problem were selected and implemented as described in their respective publications. Using several parameter sets for each, as proposed in each paper and as selected for this work, their results were compared over a battery of input data. We presented two classes of well understood benchmarks, GN, and LFR. By knowing the community structure of each example graph, a direct comparison of performance is able to be made between each approach. For both cases, we show that each algorithm can detect the structure of networks with a reasonable degree of accuracy up to a point, with those algorithms using local search and guided initialization methods outperforming the others by a wide margin. In particular, the use of random mutation is often destructive. The locus-adjacency representation allows for a much greater variety of crossover strategies by ensuring community partitions are comparable. While experiments were limited to unweighted graphs, these approaches are extensible to weighted data, or even edges with metadata associated with nodes and edges by simply extending the fitness functions. Other networks may be dynamic, or contain overlapping community structures. Little exploration has been done in applying GA to these particular classes of the community detection problem.

The ubiquitousness of modularity as a fitness function has received much attention in literature, and results on the upper values of the LFR-size experiments

show the sharp decline in fitness, even for the most effective method, GALS. More work is needed to understand when fitness functions are appropriate for a particular network[13].

The methods tested, particularly Tasgin and GA-Net, rely on a fairly large set of parameters. There are also several other models of GA not explored, such as island models. These should be selected carefully for each specific network, and presenting results on a wider scope of networks can be considered for future evaluation. A close examination of running time was not carried out, however it is clear that the use of local search operators come with a large penalty to running time, but with significant performance gains. More work should be carried out to analyze and improve the marginal gene approach used by GALS, and explore other local search methods.

Bibliography

[1] D. P. Acharjya, Satchidananda Dehuri, and Sugata Sanyal. *Computational intelligence for big data analysis : frontier advances and applications*. 2015.

[2] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 U.S. election. *Proceedings of the 3rd international workshop on Link discovery - LinkKDD '05*, pages 36–43, 2005.

[3] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[4] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378, 2000.

[5] B. Amor, S. Vuik, R. Callahan, A. Darzi, S. N. Yaliraki, and M. Barahona. Community detection and role identification in directed networks: understanding the Twitter network of the care.data debate. 2015.

[6] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On Finding Graph Clusterings with Maximum Modularity. *Graph-Theoretic Concepts in Computer Science*, (001907):121–132.

[7] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on Graph Clustering Algorithms. *Lecture Notes in Computer Science*, 2832:568–579, 2003.

[8] Qin Ding, Jim Gasvoda, and a Clustering Problem. A Genetic Algorithm for Clustering on Very Large Data Sets. pages 234–239, 2007.

[9] Weining Feng, Zhiyong Zhang, Jian Wang, and Linqian Han. A novel authorization delegation scheme for multimedia social networks by using proxy re-encryption. *Multimedia Tools and Applications*, 75(21):13995–14014, 2016.

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

[11] Michael Gurevitch. *The social structure of acquaintanceship networks.* PhD thesis, Massachusetts Institute of Technology, 1961.

[12] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992.

[13] S. Kaur, S. Singh, S. Kaushal, and A. K. Sangaiah. Comparative analysis of quality metrics for community detection in social networks using genetic algorithm. *Neural Network World*, 26(6):625–641, 2016.

[14] J. Kunegis. KONECT - The koblenz network collection. *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350, 2013.

[15] Andrea Lancichinetti, Mikko Kivelä, Jari Saramäki, and Santo Fortunato. Characterizing the Community Structure of Complex Networks. *PLoS ONE*, 5(8):e11976, 2010.

[16] Juan Liu, Eric Bier, Aaron Wilson, John Alexis Guerra-Gomez, Tomonori Honda, Kumar Sricharan, Leilani Gilpin, and Daniel Davies. Graph analysis for detecting fraud, waste, and abuse in healthcare data. *AI Magazine*, 37(2):33–46, 2016.

[17] D. Lusseau. The emergent properties of a dolphin social network. *Proceedings of the Royal Society B: Biological Sciences*, 270(Suppl_2):S186–S188, 2003.

[18] M Meilă. Comparing Clusterings. *Journal of Multivariate Analysis*, 98(5):873–895, 2013.

[19] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 29–42, 2007.

[20] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media performance and application considerations. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012.

[21] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. pages 1–28, 2016.

[22] Clara Pizzuti. GA-Net: A genetic algorithm for community detection in social networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5199 LNCS:1081–1090, 2008.

[23] Pascal Pons and Matthieu Latapy. Computing Communities in Large Networks Using Random Walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[24] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[25] Chuan Shi, Yi Wang, Bin Wu, and Cha Zhong. A New Genetic Algorithm for Community Detection. *Part II LNICST*, 5:1298–1309, 2009.

[26] Mursel Tasgin and Haluk Bingol. Community Detection in Complex Networks using Genetic Algorithm. *arXiv preprint*, page 6, 2006.

[27] D J J Watts and S H H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[28] Zhao Yang, René Algesheimer, and Claudio J. Tessone. A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Scientific Reports*, 6(1):30750, 2016.

[29] Wayne W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[30] Zhiyong Zhang and Kanliang Wang. A formal analytic approach to credible potential path and mining algorithms for multimedia social networks. *Computer Journal*, 58(4):668–678, 2015.

[31] Sjoerd D. Zwart and Maarten Franssen. An impossibility theorem for clustering. *Synthese*, 158(1):75–92, 2007.

Appendix A

GALS Synthetic Results By Parameter Set

Below are the mean NMI scores, by experiment group and parameter set for all four algorithms tested.