NETWORK WITH SPECIFIC MOTIFS

By

IBRAHIM AQEEL

MSc. Information Technology Business

August 2010

"I Ibrahim Aqeel confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the words of other authors in any form e.g., ideas, equations, figures, text, tables, programs etc are properly acknowledged. A list of references employed is included."
ACKNOWLEDGMENTS

I would like to thank my project supervisor, PierLuigi Frisco for his guidance and support throughout the project.

I would like to thank my friends Muhnad Zorer, Murad Shami, Hassan Qusadi, Muftah Elhummidi, Maruai Alshrahi, Yahya Alajlani and Alwi Bamhdi for their advice and help for keeping me motivated.

Lastly I owe my deepest gratitude to my parents, Ibrahim Hakamy, Jamal Basmd, my aunt, my wife, my son, my sisters, Prof. Saleh Cabli, Ahmed Shami, Eng. Zilay Zorer and Prof. AbdulGhaffar Bazuhair for always supporting me throughout my education.
ABSTRACT

The field of systems biology and network motifs has rapidly emerged as a major research area in recent times. In all these networks, there are repeating patterns known as Network Motifs which occur at random intervals. The study of these motifs plays an important role in the understanding of complex biological networks and their functions. The aim of this paper is to study the theory behind the formation of network motifs in complex networks. This can be done by simulating the creation of these motifs in real time with the help of a software program. In this research, a program developed in Java is used to simulate the creation of network motifs. This program creates virtual undirected networks with specific motif makeup depending upon certain input parameters. When the input parameters are changed, the program responds accordingly and the resultant network structure, size, number of motifs etc also change. This research aims to study the variations in the network motif creation with the help of this simulation program by conducting several test runs on it, each with a set of different input parameters. The test runs will be conducted on Windows and Linux operating systems. The resultant data related to network motifs will be recorded and analyzed to arrive at various observations regarding the formation of network motifs. By doing so, the research aims to provide insights into the subject of recurring network motifs in biological networks.
# Table of Contents

ACKNOWLEDGMENTS .................................................................................................................. 1
ABSTRACT .................................................................................................................................... 2
INTRODUCTION ........................................................................................................................... 6
   Aim of Research ......................................................................................................................... 6
   Stages of the Project ................................................................................................................ 8
   Scope and Data Interpretation ............................................................................................... 10
   Methodology .......................................................................................................................... 10
   Structure of Paper .................................................................................................................. 10
REVIEW OF LITERATURE ........................................................................................................ 12
   Overview .................................................................................................................................. 13
   Biological Networks ................................................................................................................ 14
   History of biological networks ............................................................................................ 15
   Graph theory .......................................................................................................................... 18
   Properties of biological networks ..................................................................................... 21
   Prototype Models of Complex Networks ............................................................................. 25
      The Erdős–Rényi Model ..................................................................................................... 25
      The Watts–Strogatz Model ................................................................................................. 27
      The Barabási–Albert Model ............................................................................................... 29
   Where are Network Motifs Found? ....................................................................................... 31
   Uses of network motifs ......................................................................................................... 32
   Identifying Network Motifs .................................................................................................. 32
   Methods and Tools for analysis of Network Motifs .............................................................. 34
      Pajek ...................................................................................................................................... 34
      MFinder ............................................................................................................................... 34
      MAVisto ............................................................................................................................... 35
      FANMOD ............................................................................................................................. 36
   Creation of new model .......................................................................................................... 36
   Algorithms to create Networks ............................................................................................ 38
      Graph Traversal Algorithms ............................................................................................. 38
      Shortest Path Algorithms .................................................................................................. 39
INTRODUCTION

In the field of Biometrics, complex network analysis is playing a more important role than ever before and several studies are being conducted to understand the intricacies of biological networks and to gain more knowledge about the functioning of cellular systems. In the study of any network, presence of recurring patterns known as network motifs aid in the understanding of the functionalities and behavior to a very large extent. At the very basic level, these network motifs can be termed as the simplest blocks of the overall network. It is possible to simulate the behavior of networks and generate motifs in response to different input parameters in order to gauge the behavior of these networks in real time. Such controlled simulations are extremely useful in understanding the overall functioning of networks.

Aim of Research

The purpose of the project is to study the creation of network motifs and the formation of motifs using a simulation program developed in Java. This simulation program mimics the creation of network motifs depending on various parameters in real time. The study is conducted by running several tests with varying input parameters and collecting the results after each test using this program. By analyzing the data collected during these test runs, the research will aim to understand the way network motifs are created and changed. Overall around 1900 test runs have been conducted on the simulation program to gather sufficient data in order to form observations and conclusions. The simulation program is studied in detail in chapter 3. Overall, the main aims of this research can be summarized as follows:
• To understand the theory behind network motifs and their occurrence in complex networks.

• To simulate the creation of networks and network motifs in real time with the help of a simulation program.

• To study the way network motif formations change in response to various input parameters and to gather and analyze the data related to these variations to form reasonable conclusions.

Requirements of Research

The main requirements of the research were finalized with the help of supervisor. The outlines of the requirement were established based loosely on MoSCoW prioritization as follows:

Must Have.

• The research must determine the way creation of networks and motifs are influenced by the changes in input parameters.

• The data collected must be analyzed to form reasonable conclusions and Well documented for future references.

Should Have.

• The research should run as many tests as possible and try to gain optimum results in each case.

• The research should be able to shed light on definite values and factors contributing towards the creation of well formed networks.
Could Have.

- The tests could be run on Windows and Linux platforms for any many runs as possible in order to ensure cross platform compliance of results.

Won’t Have.

- Tests which have failed due to memory issues will not be rerun.
- The research will not deal with the creation of directed networks.

Stages of the Project

The project has been divided into the following stages:

1. Initial Research
2. Literature Review
3. Study of Simulation program
4. Running Tests in stages and Data Collection
5. Data Analysis and Observations

Each of these stages has been explored in detail in the coming sections. Accordingly the initial and actual project plans can be visualized as follows:
The execution of the project was carried out within a duration of 7 months. Most of the literature review was conducted with the help of university library and online resources. The university lab was used for running the tests. The tests were run using multiple machines on Windows and Linux platforms. The data gathering and analysis stages overlapped each other to an extent. Several meetings were held with supervisor after and during each stage of the project for guidance as well as to validate the correctness of the progress.
**Scope and Data Interpretation**

As mentioned earlier, the simulation program deals only with creation of undirected networks. Directed networks will not be dealt as part of the study. The tests will be run on windows and Linux. The data collected will be in summary format and will include the results of all the test runs. After analyzing the data conclusions about the behavior as well as details about the input parameters leading to the creation of well formed networks will be drawn.

**Methodology**

The methodology used for the execution of this project resembles a typical waterfall cycle with each stage following the previous one in a logical manner. The initial stage was devoted to research and system study which was followed by the code analysis, code changes, implementation and testing.

**Structure of Paper**

The remaining portion of this paper is structured in the following way:

Chapter 2 – REVIEW OF LITERATURE: This chapter presents an overview of networks with special emphasis on motifs.

Chapter 3- STUDY OF SIMULATION PROGRAM AND TESTS: This chapter will present a detailed explanation of the simulation program and different parameters involved.

Chapter 4 – DATA COLLECTION AND ANALYSIS: This chapter will capture the data from the various tests and analyze those using graphs and excels.
Chapter 5 – OBSERVATIONS AND FINDINGS: This chapter will draw observations and conclusions based on the data gathered and analyzed in the earlier stages.

Chapter 6 – CONCLUSION: This chapter will present an overview of the conclusions drawn in the previous stages through data analysis. It will also highlight the issues faced and the future works.
REVIEW OF LITERATURE

The focus of this literature review is to explore biological networks in general along with a study of network motifs. The remaining portion of this paper is dedicated to the following areas:

[2] History of biological networks - This section outlines the research conducted in the area of biological networks.
[3] Different type of biological networks and their uses - This section describes the different kinds of biological networks and their applicability.
[4] Graph theory - This section deals with the different kinds of graphs and their corresponding properties.
[7] Prototype Models of Complex Networks - This section explores the 3 prototype models.
[8] Network Motifs - This section provides a brief introduction to network motifs.
[9] Where are Network Motifs Found? - This section describes the location of motifs in real life.
[10] Uses of network motifs - This section describes the applicability of motifs.
[11] Methods and Tools for analysis of Network Motifs- This section explores various tools like Pajek, MFinder, MAVisto, FANMOD etc.
[12] Creation of new model - This section outlines the aim of this project.
Network motifs are basically sub-graphs or patterns of local interconnection that occur within a network at a regularity that is more than random. The basic idea behind motifs is that big complicated networks are ultimately made up of simple and sub-graphs or patterns which are linked to each other. These patterns are used to represent important parts of the network and though it is difficult to accurately predict the cause of a particular motif recurring frequently within a network, these nevertheless make it easier to understand complex networks by breaking them down into often repeated structures [15]. Network motifs are basically the atomic units of complex networks and usually are commonly occurring phenomenon in technological networks, biological networks etc. The networks by themselves are just static blocks of data and to understand the dynamics behind the functions provided by this data, scientists rely heavily on patterns [19]. Towards this end, network motifs provide important information about the functioning of biological circuits. Before dwelling deep into the concept of motifs and their functions, it is helpful to understand the basics of the encompassing biological networks.
**Biological Networks**

Molecular biologists have long believed that knowing the structure of a biological entity is critical for understanding how it works. However, studying the components of a structure in isolation is usually not very beneficial unless the relationships and the connections between those components are understood [16]. A biological network is a complex network made up of relationships between various elements wherein the elements form the vertices and the relationships form the edges [16].

The following figures show a few typical biological networks [23]:

**FIGURE 2 - THE GENE REGULATORY NETWORK OF E.COLI**
Apart from the above, several other type of networks are known like signal transduction networks, metabolic networks, neural networks etc. The study of each of these yields a different kind of information and together they help in understanding the overall functioning of the cellular systems.

History of biological networks

The study of Biochemical networks is a relatively old area of research. The field is known to have originated way back in 1969[6]. However, till recently, the scope of the study in this field was limited to the components and elements of the network as against the design of the network and the connections between the elements.
However, in recent times, the availability of new technology has made it possible to perform an in-depth study and mapping of networks which has enabled scientists to construct models of networks and patterns. Also, recent works have proven that it is possible to reconstruct big portions of prokaryotic regulatory networks from collections of genetic data sequences [13].

However, despite the continuous research and progress made in the previous decade, there is still a lot of scope for research in this area. Going ahead, modeling and parameterizing biological networks are being hailed as the way forward [4].

**Different type of biological networks and their uses**

There are several different kinds of biological networks known till date out of which some are microscopic and some are macroscopic in nature [23]. Some of the well-known microscopic biological networks are as follows:

- **Gene Regulation Networks** – These are also known as transcriptional regulation networks. They control the gene expressions in cells. A gene can be an activator or an inhibitor (repressor) and the expression of one gene can be controlled by another gene. In a gene regulatory network, genes represent the vertices and controls represent the edges. This network provides information about the rates at which genes in the network are converted into mRNA and about cellular processes.

- **Signal Transduction Networks** – These are responsible for intercellular signal transductions and provide important information for understanding the regulation of various cellular events like cell proliferation, differentiation, stress responses
etc. Quantitative modelling techniques have been used to find various properties like integration of signals across multiple time zones within these networks.

- **Protein Interaction Networks** – A protein interaction network is a complex network made up of nodes which represent cellular proteins. The edges represent interactions between these proteins and can be physical, biochemical, metabolic or genetic in nature or a combination of any or all of these. The study of protein networks and interactions is extremely helpful in understanding cells and diseases at a broader level.

- **Metabolic Networks** - Metabolic networks are made up of metabolites. The enzymes work upon these metabolites to convert them into others. Biochemical experiments are helpful in identifying these networks. They are useful in studying the metabolism of organisms and also provide information about the molecular mechanisms of the organisms.

- **Neural Networks** – These networks are made up of interconnected elements similar to the neurons of the brain and the interconnection of these elements provides information about how the activation of one element acts as an input to another element or unit. These networks provide important information about the way human brains work and are used in several artificial intelligence systems.

Apart from these microscopic networks, biological networks are also found on a much larger scale. Some examples of macroscopic biological networks are ecological networks which describe the interaction between different organisms, food webs which symbolize the relationship between what is traditionally known as predator and its prey,
Phylogenetic networks which describe the way organisms have evolved and the relationships between them etc [23].

Also, there are special types of networks like correlation networks studies of which are still in preliminary stages. These networks are not the result of experiments conducted in laboratory. They are usually determined by accumulating huge amounts of data and then mathematically calculating the correlation between them.

**Graph theory**

Networks are often modeled as graphs to facilitate ease of working. A graph is made up of vertices and edges representing elements and nodes respectively. Vertices are also known as nodes or points and edges are also known as arcs or links. Several attributes like values, coordinates etc can be associated with a graph [17].

Biological networks are also often represented as graphs. A biological network can be termed as an undirected graph where the edge between the vertices is represented by an unordered vertex pair or as a directed graph in which ordered vertex pairs are used to signify the edges between the vertices [14]. They can also be mixed graphs which is a combination of undirected and directed edges. The different types of graphs are shown diagrammatically as below [14]:

**FIGURE 4 – UNDIRECTED, DIRECTED AND MIXED GRAPHS**
Apart from the above mentioned graphs, there are also special type of graphs like hypergraphs, bipartite graphs and trees which are used to represent certain special cases of networks.

Hypergraphs are graphs used to represent networks where an interaction can be between more than 2 items. A hypergraph is made up of a set of vertices and hypredges where the hyperedge is a non-empty subset of the set of vertices. A hypergraph is represented as follows [10]:

![Hypergraph Diagram](image)

**FIGURE 5 - HYPERGRAPH**

A bipartite graph is one in which the vertices are separated into several sets which are disjoint in such a way that each vertex in the first set is connected to a vertex in the second set through an edge. A bipartite graph is shown below [10]:

![Bipartite Graph Diagram](image)
A tree is a specific kind of graph where the relationship between elements in the network is visualized like a tree. A tree is usually an undirected, connected, acyclic graph except for rooted trees which are considered to be directed graphs. A tree can be represented in many ways. Some of the common ways are shown diagrammatically as follows [15]:

Graphs can be represented programmatically using an adjacency matrix or an adjacency list and graph algorithms are used to solve many of the challenges related to biological networks [14].
Properties of biological networks

As discussed earlier, there are several kinds of biological networks with different features and origins [8]. However, studies have found that these networks often share similar characteristics and properties at a broader level [5]. Some of the common or global properties of biological networks are as follows:

- **Shortest Length** – In a network where there are \( n \) number of vertices, the shortest path between a pair of vertices is defined as the minimum edges that need traversing in order to reach from vertex \( x \) to vertex \( y \). For directed networks, distance between a pair of vertices is usually asymmetric. Similarly, for directed networks and disconnected networks, it is not mandatory to have a direct path from \( x \) to \( y \) and in such cases distance is considered to be infinite. There are several algorithms like Dijkstra and Floyd-Warshall algorithms which are used to determine the shortest distance between 2 vertices [5].

- **Small World** – For most of the networks, irrespective of their size, the average length of the path is usually minute and measurable.

- **The Degree Distribution** – The degree of a vertex of a network is determined by the total number of adjacent edges of the vertex. In a network which has neither self loops not multiple links, the degree is same as the total number of neighbours of the vertex.

- **Clustering Coefficient** – This is basically a unit used to calculate the probability of a pair of vertices with one common neighbour being connected to each other. The clustering coefficient of most biological and emphirical networks is high.
which means that vertices have a tendency to form clusters.

• The Matching Index – It is not necessary that 2 vertices which are functionally similar need to be connected. This index is a unit to measure the similarity between a pair of vertices based on common neighbours that they share. This can also be measured across multiple vertices.

• Network Centralities – Network centrality is a way of identifying every vertex or edge within a network with relation to its position. There are several measures of centrality like closeness centrality, betweenness centrality etc [11].

• Spectral Properties – The spectral properties of random graphs are one of the oldest and most widely used characteristics of biological networks. The spectral properties and eigenvalues are used widely in the field of coupled oscillators and to determine the stability and local dynamic characteristics of complex networks.

• Structural Robustness and Random Failure Tolerance – Most networks are capable of maintaining their functionalities and withstanding random failures of individual nodes. A failure of one or more node does not cause overall system failure. Also, most networks are able to maintain properties like average path length etc when one or more vertex is removed [1-3].

• Modularity, Community Structure and Hierarchy – In most of the biological networks, the nodes form tightly coupled modules or communities with loose interconnections between the communities [9]. The concept of communities and clusters are closely related to each other. Modular networks exhibit several advantages due to their ability to adopt and incorporate new functionalities.
Recurring elements – Biological networks display a pattern of recurring elements where the same connecting patterns are repeated throughout the network and any part of a network holds some similarity to other parts. These patterns which are highly useful in understanding the structure and function of a network are known as network motifs [15].

Out of the common properties mentioned above, the study of recurring elements or network motifs is considered to be the most significant aspect of biological network analysis [8]. Hence, study of biological models and patterns within those models will be the focus of the rest of the chapter.

**Modeling of biological networks**

Network models at a very high level can be described as a way to identify and segregate the different layers in a network and to agree upon a set of protocols for the communication between those layers. Accordingly, a biological network layer model specifies abstraction boundaries, interfaces, and standards which enable individual components to come together to form a biological system [3-14].

The layers of a biological network model can be roughly represented as follows [4]:

<table>
<thead>
<tr>
<th>LAYER NUMBER</th>
<th>LAYER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAYER 7</td>
<td>APPLICATION</td>
</tr>
<tr>
<td>LAYER 6</td>
<td>PACKAGING</td>
</tr>
</tbody>
</table>
There are several reasons for adopting network models and creating new models. Some of the main reasons are outlined below:

- Network models are useful to present a synthetic view of the knowledge which is already available. It also helps in designing the structure of the network in such a way that relevant properties which can otherwise remain hidden are brought forth.

- Network models help in testing whether the maps are correct and whether there are any critical interactions missing by analyzing the models and the expected behaviour from them. They are helpful in generating new hypothesis which can be tested.

- To understand and predict the result of changes to a particular network, it is important to know and understand the origin of those changes. This can be
accomplished with the help of network models and layers as the changes usually originate and propagate across layers.

- Modeling of networks helps in getting information about how the structure results in a particular behaviour [16].

Therefore, to understand and represent complex biological networks, modeling is a useful technique [4]. The basics of biological network models can be understood with the help of prototype models.

**Prototype Models of Complex Networks**

There are three basic prototype models which are also known as models of complex networks. The main purpose of these models is to provide a minimal structure which helps in understanding the generic features of complex networks. They also provide information about how the construction rules of the models give rise to certain functionalities [4].

The three prototype models are – (a) The Erdős–Rényi Model (b) The Watts–Strogatz Model (c) The Barabási–Albert Model [17].

**The Erdős–Rényi Model**

The Erdős–Rényi model is a basic and important model given by Erdős–Rényi (ER) network. This model is used to prove the existence of graphs satisfying several properties. It is based on a simple assumption that each pair of vertices is chosen to be an edge independently with some probability p for some fixed p.
In the ER model, given a set of \( Nv \) vertices, there are about \( Nv (Nv-1)/2 \) edges which are possible and the number of edges which interconnect these vertices is given by \( Ne \). The edges are taken in a random fashion from the possible sets of all edges. The ER model can also be said to be made up of \( Nv \) vertices where the connection probability \( p \) for each and every pair of vertex \( \leq 1 \). Therefore, the probability of two random vertices being connected to each other is given by \( p = 2Ne/Nv (NV - 1) \).

Based on the above equation the number of edges can then be tabulated as \( \{Ne\} = pNv (Nv - 1)/2 \).

The ER model is quite efficient in extracting the behavior of various graph properties and generating random graphs. A sample random graph for calculating the average search length using the Erdős–Rényi model is shown below [9]:

![Graph showing average search length using the Erdős–Rényi model](image)

**FIGURE 8 – AVERAGE SEARCH LENGTH USING ERDŐS–RÉNYI MODEL**
The clustering coefficient of this network is given by $C = p = \langle k \rangle / N_v$, which means that the probability of a pair of vertices with a common neighbor being connected is same as the probability of any two randomly chosen vertices being connected to each other. The ER model does not exhibit local cohesiveness or degree correlations. The main property that can be demonstrated by using this model is the small world property discussed previously.

**The Watts–Strogatz Model**

The need for Watts-Strogatz stems from the fact that while the ER model is efficient enough in demonstrating the small world property, the clustering coefficient calculated using it is usually inaccurate and far lesser than the value obtained through experimental data. To overcome this issue, the Watts-Strogatz model is used. This model produces graphs with short average path lengths and higher amounts of clustering.

In this model, the vertices are arranged as a one dimensional ring around a circle and each vertex is connected to $n/2$ number of its nearest neighbors. This means that each vertex is aware only of its immediate neighbors. This means that there is a strong bonding inside the model which means a higher clustering coefficient, but as each element knows only its immediate neighbors, the information traverses and spreads at a much slower rate, and as the size of the system increases, the time taken and the average value of path length also increases parallely.

The model can be diagrammatically represented as below [8]:
This model introduces shortcuts between vertices such that whenever a particular link undergoes rewiring, its one end is removed from its original vertex and attached to some random vertex and the probability of this is given by $P_{rew}$. As more and more links are rewired, the probability $P_{rew}$ edges closer to 1 and when it finally approaches 1 ($p \rightarrow 1$), this model becomes the same as the ER model.

An important aspect of the WS model is that it highlights the differences between local properties and global properties of a network. For e.g. the clustering coefficient is a local property and it remains unaffected by the introduction of new shortcuts into the network whereas the average path length is a global property and is inversely proportional to the number of shortcuts.

However, other than capturing the high clustering coefficient and low average path length, the WS model provides no specific features.
The Barabási–Albert Model

The Barabási–Albert Model introduced by Barabási and Albert attempts to overcome the limitations of the earlier models. This model provides a scale-free degree distribution and acts as a base for most of the other models.

There are two main features in the Barabási–Albert model (BA) model:

Growth: In the actual world scenario, numbers of vertices keep growing with time. Keeping this in mind, the BA model does not assume a fixed number of vertices within the network as done by the ER and WS models. This model allows the introduction of new vertices and consequent growth of the network as time progresses.

Preferential attachment: In a real network, the probability of a particular vertex getting a new neighbor is dependent on its current degree and the BA model remains consistent with this behavior.

A diagrammatic representation of the BA model is shown below [8]:

![Diagram of the Barabási–Albert Model]

**FIGURE 10 – The Barabási–Albert Model**

The diagram above depicts that the BA model is a small network made up of $N_0$ unconnected vertices to begin with. New vertices get added regularly and the number of
edges of the new vertex given by m is always less than or equal to N0. The above figure depicts this case with where m is 2.

At any point, the probability of a new node being connected to node i is calculated as:

$$p_i = \frac{k_i}{\sum_j k_{ij}}$$

The main advantage of the BA model is that it explains the scale free distribution observed in several networks. The results obtained from this model also indicate that large complex networks are often self organizing.

**Network Motifs**

The idea of *network motifs*, introduced by Alon and co-workers [15], has turned out to be a rapidly evolving area of interest and a useful tool in the analysis of complex networks. A network motif is a pattern that occurs repeatedly at some frequency in a true network like a sub-graph. This repeated occurrence of some motifs in the networks has been linked to the different functional requirements expected from different networks [19]. The smaller functional units inside the motifs in the network work together to control the behavior of cells as a unit and the number of vertices of a motif dictate the size of it.

There are various different kinds of network motifs:

The feed-forward loop motif - This is made up of a pair of transcription factors with one factor regulating the other and both together control a target gene [12]
The Bifan motif - This is made up of 2 source nodes each of which cross regulates a target node.

The Single Input Module motif – Here, the expression of many different structural genes is controlled by a sole transcription factor.

The Multiple Input Module motif – Here, a layer of overlapping interactions enable different inputs to regulate different outputs [2].

The following figure represents the different kinds of network motifs diagrammatically [8].

![Network Motifs Diagram](image)

**FIGURE 11 – Network Motifs**

The above figure depicts a Feed-forward loop motif(a), Bifan motif(b), Single-input motif(c) and Multi-input motif(d).

Where are Network Motifs Found?

Though most of the work concerning the study of network motifs is directed towards the area of gene regulation, motifs are also a regular occurrence in many different types of biological networks like protein interaction networks and neural networks [23].
Apart from biological networks, motifs are also common occurrences in other types of networks like the World Wide Web, information processing networks, engineering networks etc. There are several open source softwares available in the market which can detect network motifs given a network as an input [19].

**Uses of network motifs**

As mentioned before, the basic idea behind motifs is that a huge complex network is comprised of many smaller units like interlinked sub-graphs or patterns and breaking down complex structures into simpler sub parts makes it easier to understand the structures [16]. Scientists rely heavily on recurring patterns inside a network to understand the functioning dynamics behind masses of static data provided by the network [19].

Network motifs can be considered as the design patterns behind complicated networks and many motifs play an important role functionally within networks [9].

Network motifs provide important information about several biological concepts like gene regulations, protein interactions, metabolism in organisms, response to stimuli, cell structures and diseases, gene mutations etc [2].

Some of the common usages of network motifs are in the field of general medical research, molecular cell biology, cancer research, microbiology, genetics, healthcare etc.

**Identifying Network Motifs**

The significance of a motif in a network is measured statistically by computing its Z-score using the concept of frequency. The Z-score is defined as “the difference in
frequencies of a motif m within a network and its average frequency across a selection of random networks, divided by the standard deviation of the frequency values of the randomized networks” [15].

The simplest way of identifying network motifs is given as below [15]:

- Find n-node Subgraphs in the real graph.
- Find all n-node Subgraphs in a set of randomized graphs with the same distribution of incoming and outgoing arrows.
- Calculate and assign Z-score for each subgraph.

Subgraphs with high Z-scores are marked as Network Motifs.

A sample network with the presence of 5 motifs is as shown [15]:

![Figure 12 - A Sample Network with 5 Motifs](image)

The Z-Score can be represented in the form of an equation as follows:
\[ Z(m) = \frac{F_1(m) - \bar{F}_{1,r}(m)}{\sigma_r(m)} \]

In the above equation, \( F_1 \) is the motif frequency within the target network and \( F_1, r \) is the mean frequency of the network in a large set of randomized networks and \( \sigma_r \) is the standard deviation of the frequency values.

**Methods and Tools for analysis of Network Motifs**

Various methods, tools and specialized techniques are developed to analyze the network motifs. Some important tools which are used for analysis of networks and motif detection are Pajek, Mfinder, MAVisto, and FANMOD [24] each of which is described below:

**Pajek**

Pajek is an open source software program for the analyzing large networks and also helps in visualizing these networks and it is available as a free download for non-commercial use. This program allows the calculation of frequencies for certain subgraphs like triads and particular tetrads. Triads can either be connected or unconnected and Pajek calculates the total no of triads of a network along with the values of the frequencies expected. Also network topologies can be represented by Pajek files.

**MFinder**

The Mfinder is an open source software tool for analysis of networks and detection of network motifs in both directed as well as undirected networks. It uses 2 methods for detecting network motifs – Full enumeration of subgraph method and the Sampling of subgraph method [24]. In the first method, all the connected subgraphs are
estimated and an enumeration of all the unique subgraph matches is executed. These enumerations then serve as the inputs for indexing and filtering out motifs. In the second method, subgraphs are randomly sampled to detect patterns [10]. This tool computes the frequency of motif in the target network along with a uniqueness value. The statistical significance is determined based on frequency of motif occurrence in randomized networks and is depicted by a P-value and Z-score. Z-score as explained previously is a measure of standard deviation and helps in deciding whether to accept or reject a hypothesis. The P-value is a measure of the probability that a particular motif m appears n or more times in a randomized network, n being the motif frequency in the target network [15]. To generate random networks, the MFinder uses methods like switching where the edges in the graphs are reconnected or switched randomly to form new subgraphs along with several specialized algorithms.

MAVisto

MAVisto is an open source tool for the detection and analysis of motifs in networks with a motif search algorithm which is quite flexible. It also provides different views for analyzing and visualizing of network motifs. Its motif search algorithm focuses on discovering motifs of a given size based on the number of edges or vertices it contains. All motifs which are of this size undergo analysis after which the frequencies are determined for all the different frequency concepts along with the P-values and Z-scores explained previously. MAVisto works with both vertex-labeled and / or edge-labeled networks. MAVisto can be used with Java webstart.
FANMOD

FANMOD is an open source tool for detection of motifs in both directed and undirected networks. It is found to be superior in performance compared to many existing tools [21]. FANMOD basically detects motifs that are induced sub graphs. It works with vertex-labeled and edge-labeled networks. The results of this tool can be viewed as either text or html files. FANMOD calculates the P-values and Z-scores explained previously. In addition, it also calculates the relative frequencies of subgraphs. FANMOD provides a sampling method to approximate the motif numbers rapidly where the motifs are detected by sampling a small number of subgraphs [21].

For more information about downloading each of the above mentioned tools, refer to Appendix A.

Creation of new model

The purpose of this project is to test the behavior of networks and also analyze the generation of motifs using a given simulation program. The simulation program deals only with undirected networks and various tests will be conducted to determine the changes in behavior in response to varying input parameters. The program generates networks containing n nodes connected to each other in a specific way. The model will try to emulate the network motif with varying number of initial nodes.

There are a few other programs which are meant for network analysis [24]. The existing tools for network analysis which are available as open source downloads are described below:

- BioNetBuilder – This tool allows creation of biological networks integrated from
a database and it is available as an open source download. The tool is available in the form of a plugin with a user interface which allows visual creation of networks.

- **VisANT** – This is a web based tool which is useful for analyzing and visualizing biological networks. It is possible to save, modify and share these views of networks with other users using this tool.

- **FCModeler** - FCModeler creates visual models of metabolic and regulatory networks. These sub-networks can then be analyzed using fuzzy cognitive maps.

However, most of them are not flexible enough to accommodate changes to the network and are usually not user friendly. Some of the tools only allow visualization of data and do not allow modification [10-23].

The aim of this project is to test a platform independent, user friendly application which creates networks based on a set of input parameters fed through plain text files. Also, the networks generated have been found to have specific motif makeup like 80% of motif 1, 10% of motif 2, 10% of motif 3 and so on based on the recorded outputs.

The simulation program uses a simple text based approach to change the input parameters so that changes can be made without touching the underlying implementation thereby eliminating the need for recompilation. Results will be recorded in excel format and subsequent analysis will be done using graphs and charts.
**Algorithms to create Networks**

Network modeling can result in the generation of either directed or undirected graphs. Given two nodes A and B, an undirected graph is a graph in which if there is an edge from A to B, it is implied implicitly that an edge from B to A also exists whereas in a directed network, an edge from A to B does not necessarily mean an edge from B to A and vice versa [8]. There are several network models available which produce both kinds of graphs with the help of various algorithms. The basic features of a few algorithms to achieve various such functionalities are explored below:

**Graph Traversal Algorithms**

These algorithms visit each and every vertex in a graph and take some action on each of these visits. The two important graph traversal algorithms are Depth First Search algorithm and Breadth First Search algorithm [8].

The Depth First Search (DFS) algorithm visits each vertex, marks it as visited and then recursively visits each of its neighbors marking each as it goes along. The cycle continues till all the vertices are marked as visited. The Breadth First Search (BFS) algorithm visits all the neighbors of a vertex before it visits the neighbors of neighbors. It keeps track of the visited vertices by putting them in a queue to begin with and then marking them as visited and moving them to the end of queue as and when they are visited [14].
Shortest Path Algorithms

Shortest path between a pair of vertices is basically the shortest distance that has to be traversed to reach from one vertex to another. The most common algorithms for finding the shortest path are the Dijkstra and the Floyd–Warshall algorithms [8].

Dijkstra algorithm calculates only one shortest path. It starts at the vertex identified as source and starts spanning across all the vertices reachable from the source. It creates a queue to which the vertices are added in the order of their distance from the source. Whenever a new vertex with a shorter distance is discovered, it replaces the earlier shortest distance vertex in the queue. The queue can be implemented as a linear array or a heap [22]. After all the vertices are visited, the shortest distance can be computed using the elements of this queue. The Floyd–Warshall algorithm finds the shortest path between vertices in a graph and returns a matrix containing the distances. It uses dynamic programming to compare all the possible paths between two vertices till the most optimal path is found [22].

Transitive Closure Algorithms

Transitive closure property is used to find all the nodes which can reach each other through a non-null path. This is very important in the reachability analysis of directed graphs. The most common choice for this is the Warshall algorithm which produces the transitive closure of a relation in the form of an adjacency matrix. It uses a simple logic to determine if there exists an edge between Vertex A and B and tabulates the Boolean results in the form of an adjacency matrix [10].
There are several more algorithms available to analyze the graphs along with several variations for each. A combination of one or more of these algorithms can be adopted to create directed graphs and find the relevant properties.

**Choice of language and platform**

Before proceeding to the development stage, decisions regarding software and approach need to be taken for testing the model mentioned above. There are several languages and platforms to choose from and it is important to choose a language which is well documented and suits the requirements. An object oriented language will be suitable for this task as the networks and its components can be visualized in terms of real-life objects and classes. Also an object oriented language is beneficial considering future enhancements as the individual modules need not be modified when new objects are added.

Based on the above factors, a simulation program developed in Java was chosen as the choice of language for this task as it provides an extensive API well suited for complex programming tasks. It has specific libraries and engines like Java object oriented Neural Engine, Java Universal Graph Framework etc which provide several precompiled functions thus making the programming job easier [20]. Also, the java compiler compiles java code to byte code rather than machine code. Machine code is processor specific (e.g. Intel, ARM) as well as the operating system specific, whereas byte code is a format executed by a virtual machine and hence it will provide more portability for the application. Since java is platform independent, it will be easier to run the code on different machines irrespective of the underlying hardware and hence it provides an ideal
solution for this program. Also, the automation program used to record data from text files into the output summary file was also developed in Java.

**Hypothesis**

Based on the research conducted so far, it is predicted that a tool can be developed to programmatically create a new network model with specific network motif makeup. A simulation program developed using Java will be used to test this hypothesis. The project will be termed as successful once it is demonstrated that the tool is able to create and modify network models with specific motifs and also the effects of changing input parameters on the resultant networks are studied.

**Professional, Legal, Ethical and Social Issues**

According to supervisor PierLuigi Frisco there are "no Professional, Legal, Ethical and Social Issues arise from this project."
The simulation program developed in Java is the core of this project. This program generates undirected networks in real time with specific network motif make up. Before running the test cases to gather data related to the formation of motifs, it is important to understand the functioning of the simulation program.

**Overview of simulation program**

The simulation program has been developed in Java using netbeans as the IDE. There are 2 aspects of this program – The java files which provide the functionality and the input file which controls the generation of motifs. The input file is a plain text file called Properties.txt which contains various input parameters in Key-Value format. The program reads this file and gets the input parameters from it to use in the generation of motifs. The program generates the details of the motifs formed in plain text format inside the output directories.

**Important Definitions**

Before understanding the functioning of the simulation program, it is important to be familiar with a few definitions as given below:

*Initial Node*: This is just a string of alphabets representing the structure of the node with which the network creation begins.

*Unit Distance*: This is an integer which influences the calculation of distance between 2 nodes by comparing them symbol-wise. This influences the calculation of Type Distance which in turn influences the way the edges are added in between nodes. If the unit distance is set to 1, when comparing two nodes, each of the symbols of the first node is compared to each symbol in the second node. If the unit distance is set to 2, every
pair of symbols in the first node is compared to each pair of symbol in the second node. Two pairs of symbols are considered equal if they contain the same alphabets irrespective of the order. Similarly, if unit distance is set to 3, symbols are compared with each other in groups of 3 and so on.

**Hamming Distance:** This is a kind of Type distance which is used if the number of symbols in the nodes being compared is the same. It is depends on the value of unit distance. The hamming distance between two nodes is determined by calculating the number of symbols which are different between the two where the difference is determined depending on the value of unit distance.

**Max Distance:** This is an integer value which indicates the maximum value of hamming distance at which a pair of nodes has a common edge between them.

**Frequency Save:** The simulation program saves the networks created inside different log files which are mentioned in the properties file. The frequency save parameter indicates the frequency at which the data has to be saved inside log files. For e.g., if the number of nodes is set as 900 and frequency save is set as 300, the data gets saved after the creation of 300, 600 and 900 nodes.

**Prob_to_mutate:** This indicates the probability of mutation within a node structure. Mutation indicates the changes in sequence of the structure.

**Prob_to_add:** This is a float value indicating the probability of symbols getting added within a node structure.
**Prob_To_Delete:** This is a float value indicating the probability of symbols getting deleted within a node structure.

**Prob_To_Duplicate:** This is a float value indicating the probability of symbols getting duplicated within a node structure.

**Match Files:** A match file is basically a text file which indicates the symbols which are considered equal. This is useful in the calculation of hamming distance. The name of this text file is indicated by the parameter File_Matches inside the properties file. Inside this text file, if it is mentioned that AB = BA = BB, the simulation program will effectively consider the three substrings AB, BA and BB to be equal and calculate the distance between nodes accordingly.

**Modular structure of the program**

A brief modular overview of the program is given below:

**Constants.java:** This module contains all the constant values which are needed for the functioning of the program.

**NodeStructure.java:** This module determines and defines the structure of the nodes within the network in question. This contains methods like insert, delete etc which facilitate the creation of node structure.

**Node.java:** This is the template for individual nodes containing the NodeStructure along with variables to store to and from edges, probabilities etc. It can convert a string of characters or a character array into a corresponding NodeStructure.
Network.java: This file has variables for holding the nodes and edges and hence acts as a template for the network. This module has methods to add/remove the nodes, edges etc.

ManagesIO.java: The main function of this module is to manage the input and output transactions. Also, this module reads and checks the match files for consistency and repetitions.

Statistics.java: It prints the output networks according to what is specified in the property file.

Functionality of the program

The starting point of the program is provided by Main.java which begins by creating a blank network instance to which nodes and edges are added progressively. A network instance in this case is an object of the Network class which contains a collection of nodes and edges. A node is an instance of the Node class which contains a string variable to contain its structure. The structure of a node is the string of alphabets which it is made of (E.g. ATCATCAT). Edges are denoted by plain integers.

The program initially reads the properties.txt file with the help of Input/output functions provided by ManagesIO class. After reading the properties.txt file, all the input parameters are checked to determine if they are in valid format, if not, the programs throws an error and exits. For e.g., all the probability variables like Prob_to_add, Prob_to_delete etc should be less than 1. If all the input parameters are found to be in valid format, the input variable FILE_MATCHES is read to see if a match file has been specified and if yes, the match file is read. Each and every match specified in this file is added to a network instance in the form of nodes and links. If there are several matches
defined for a particular node, they are all separated using the SEPARATOR specified in the input file and edges are added for each of the node. This file or structure is then used by the functions during the creation of networks as the program progresses.

There are several modes in which the program can run. This is determined from the variable RUNNING_MODE in the input file. If it is set to “Single”, the network is created and printed just once. If it is set to “Enumerate”, the node structure is replicated based on the Initial Node and alphabet length and accordingly added to network. While creating the network structure with several nodes for the Enumeration mode, the creation begins with the Initial Nodes. After that the structure of the nodes further added is determined using the Initial Node as well as the length of the Alphabet specified in the input file. The new node is created with the same length as that of the initial node and the alphabets specified in the input file are used in different positions to create the structure.

If the running mode is set to “Tests”, a series of runs are conducted for each of the random seeds that are specified in the program. This paper concentrates on the Testing mode.

**Network Creation**

After the initial gauging of property files etc, the program initiates the creation of network by calling the runNetwork method with the property file, match file and the counter as the input parameters. If the Network size is not mentioned as INCREMENTAL, the program exits after printing an error. If the network size is specified as INCREMENTAL, The INITIAL_NODE property is read to check if the structure of the initial node has been specified or not. If it has been specified, there are two scenarios:
1. Initial Node is Barabasi-Albert (BA) - In this case, the property
NUM_BA_INITIAL_NODES is read and accordingly, that many numbers of
nodes are added to the network along with edges. In this case, the position at
which the edge needs to be added is determined using the Barabasi-Albert
algorithm.

2. Initial Node is not Barabasi-Albert (BA) - In this case, if the Initial Node
contains only one node, it is added to the network. Otherwise, if it contains
more than one node, they are separated into individual nodes using the
Separator specified in the property file and all of them are added along with
edges to the network. To add the edges in this case, the input variable
TYPE_DISTANCE specified in the properties file is also considered. If the
Type distance is HAMMING, the hamming distance is calculated according
to the Unit Distance given as explained previously. After calculating the
hamming distance, if it is less than the MAX_DISTANCE specified in the
input file, an edge is added between the last node in the network and the
remaining nodes. If the Type of distance is set to PARIKH, the edges are
added if the Parikh distance between 2 nodes is less than the value mentioned
in the input file using the variable DISCEPANCY_PARIKH. The Parikh
distance is calculated by computing the probability of the alphabets specified
in the input file existing in the structure of the node. (For e.g., if node
structure is ABABC and the alphabet is A, the probability is 2/5). If the Type
of distance is set as HAMMING_PARIKH, an edge is added between 2
nodes only if they meet the criteria for both Hamming and Parikh distances
mentioned above.

After this, the Number of nodes for which the tests need to be run is checked by reading the property NUMBER_RUNS_EACH_NETWORK. The number of edges desired is specified using REMOVE_MAX_NODES. If the number of edges of the network created in the previous step is more than this number, the last element is removed and the edges are renumbered. If the number of edges is less than the desired number and the frequency at which the node structure needs to be saved is specified, the size of the network is divided by the frequency to determine the number of nodes that need to be saved in each turn. After this, all the nodes having less than the minimum desired number of edges are removed, the edges are rewired and finally a duplicate of the network is created at the frequency at which it needs to be saved. Similarly networks are duplicated at each of the interval derived using the Frequency save variable. After duplicating, the networks are printed using the printStats function. This entire process is repeated till the number of nodes reaches the number specified by the variable NUMBER_RUNS_EACH_NETWORK.

**Effect of various parameters.**

The running of the program is dependent on various input parameters as mentioned in the earlier section. Some of the parameters which were used during this research are mentioned below:

- **RUNNING_MODE** –. To run a series of tests with different random seeds, this has been set to “TESTS”.

- **TYPE_DISTANCES** –. Since this was set to Hamming, all the edges were added according to the Hamming distance during the tests.
MATCH_FILES – Several match files have been used for the tests with different matches specified in each. The variations in network motifs depending on the introduction and variation of match files has been explored in the coming sections.

Base_dir – Since this is set to TESTS_, all the output directories are created with a prefix of TESTS_.

INITIAL_NODE – Various custom values have been used for this variable during the tests. The output node structures generated for these different values begin with the initial node specified in this. For e.g., if this is set to ATCATCAT, the output node always has the first node structure as ATCATCAT.

Remove_min_nodes, Remove_max_nodes – These specify the minimum and maximum values of edges that the nodes can have. Any nodes having less than the minimum value and more than maximum value of edges are removed. This was set to 0 to specify that a motif can have 0 edges. This can be seen in the output data gathered.

Num_runs_each_network, Frequency_Save - Since this has been set to 900 and 300 respectively, the output data was saved for 300, 600 and 900 nodes for each run.

Unit and Max Distance - The tests have been run for various values of these parameters and are explained in the next section.

Output of the program

When the program is run in the testing mode, it runs the tests and generates the output files to reside in the same directory. Depending upon the value of a randomly chosen number known as Random seed, the program generates output directories with one directory per seed. The name of the directories is decided by the value of base directory and the counter variables in the property file. If the base directory is given as TESTS_.,
counter is given as 100 and random seeds from 1 to 5, then 5 directories called TESTS_100, TESTS_101, TESTS_102, TESTS_103 and TESTS_104 are generated.

Inside each of the output directories, two types of log files are created. The first one called network.log stores the details of the edges and nodes of the networks created. The second one called outputMotifs.log stores the details of the percentages of motifs with no edges, 1 edge, 2 edges and so on in separate lines. Depending upon the value of Frequency Save, these log files are stored at regular intervals as separate files. For e.g., if the number of nodes is 900 and frequency save is 300, then the network logs are saved as network.log1, network.log2 and network.log3 storing the details at 300, 600 and 900 nodes respectively. Similarly, the motif logs are generated. The name of these files is mentioned in the properties file in the File Network and File Motifs parameters. Other details like number of edges, probability of path lengths etc can also be outputted by giving the values of the log files to store these details in the properties file.

**Process Flow Diagram**

The functionality of the entire program can be represented with the help of a process flow diagram as follows:
FIGURE 13 – PROCESS FLOW DIAGRAM
Test Plan

After the study and understanding of the simulation program, a series of tests were run over a period of several days over several machines parallel and the resultant data was recorded in a predetermined format. A total of about 1900 tests were run. Overall, the variance in the creation of motifs with respect to varying number of nodes, unit distance, maximum distance, probabilities etc were studied. A Java program was developed for automating data collection. Then the data analysis was carried out to draw the conclusions. For more details about the summary of the data collected, please refer to Appendix B. The testing which forms the core of the project was also carried out in 8 stages. There were 4 stages without match file and 4 stages with match files. The details of the tests without match files are given in the following table:

<table>
<thead>
<tr>
<th>Match File = None, Number of Nodes = 900, Frequency Save = 300, Alphabet = AT, ATC, ATCG</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Nodes</td>
<td>4,6,8,10,12,16,21,26,31</td>
<td>4,6,8,10,12,16,21,26,31</td>
<td>4,6,8,10,12,16,21,26,31</td>
<td>4,6,8,10,12,16,21,26,31</td>
</tr>
<tr>
<td>Unit Distance</td>
<td>1 to 4</td>
<td>1 to 4</td>
<td>1 to 4</td>
<td>1 to 4</td>
</tr>
<tr>
<td>Maximum Distance</td>
<td>1 to 3</td>
<td>1 to 3</td>
<td>1 to 3</td>
<td>1 to 3</td>
</tr>
<tr>
<td>Prob_to_mutation,Prob_to_add,Prob_To_Delete,Prob_To_Duplicate</td>
<td>0.2, 0.4, 0.2, 0.2</td>
<td>0.0, 0.6, 0.4, 0.0</td>
<td>0.2, 0.8, 0.0, 0.0</td>
<td>0.0, 0.6, 0.4</td>
</tr>
</tbody>
</table>

**TABLE 1 – TEST STAGES WITHOUT MATCH FILES**
Similarly, the details of the tests with Match files are given in the following table:

<table>
<thead>
<tr>
<th>Match File = geneticCodePlus_1, geneticCodePlus_2, geneticCodePlus_3, geneticCodePlus_4 and geneticCodePlus_5, Number of Nodes = 900, Frequency Save = 300, Alphabet = ATCG</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Nodes</td>
<td>4,6,8,10,12,16,21,26,31</td>
<td>4,6,8,10,12,16,21,26,31</td>
<td>4,6,8,10,12,16,21,26,31</td>
<td>4,6,8,10,12,16,21,26,31</td>
</tr>
<tr>
<td>Unit Distance</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Maximum Distance</td>
<td>1 to 3</td>
<td>1 to 3</td>
<td>1 to 3</td>
<td>1 to 3</td>
</tr>
<tr>
<td>Prob_to_murate, Prob_to_add, Prob_to_delete, Prob_to_duplicate</td>
<td>0.2, 0.4, 0.2</td>
<td>0.0, 0.6, 0.4, 0.0</td>
<td>0.2, 0.8, 0.0, 0.0</td>
<td>0.0, 0.0, 0.6, 0.4</td>
</tr>
</tbody>
</table>

**TABLE 2 – TEST STAGES WITH MATCH FILES**

At each stage, the networks were generated for 5 random seeds.

**Automated Data Gathering**

To record the data collected into an output summary file, an automated program was developed in Java. This development activity was conducted along with the running of the tests. This program is called Parser.java and it takes the output data folders as the input and generates the summary excel file as the output. The purpose of this program is to save time by automating the process of manual data entry. The main files of this program are as follows:

*Parser.java:* This is the file which reads through the input folders and extracts the data. It
also writes the data into the output excel file.

*Params.java:* This is used by the main Parser.java class to pass parameters.

*Alphabets.java:* This contains the enumerations which helps in converting integer columns into excel columns.

*DebugLevel.java:* This is used for debugging purposes.

*ParserApp.java:* This contains the main method which starts the application execution.

*ParserView.java:* This provides the user interface for the frontend of the application.

There are other supporting files and forms which help in providing the overall functionality. The entire source code of this application is available in Appendix D.
DATA COLLECTION AND ANALYSIS

The following section will establish the data collected through the various tests and analyze them with the help of graphs. A total of about 1900 tests were carried out in a series of 8 stages as mentioned in the previous section wherein 4 stages were carried out without match files and the rest with a variety of match files. The data gathered as well as the 5 match files used for testing are available in Appendix B.

The aim of the section is to understand how the outcome changes according to the changes in input values. This includes looking at the data gathered through various stages of testing and analyzing how the outcome varies depending upon the input values. This will be done by creating various graphs depicting the changes when a particular input changes keeping the rest of the input values same.

Summary Data

The data has been collected mainly in terms of the variations in the percentage of motifs. All the data collected for various tests has been consolidated into 8 summary sheets each representing a stage of tests using an automation program developed in java. The summary sheet contains 3 parts – one for 300 nodes, one for 600 and one for 900 nodes. This has been achieved by setting the number of nodes as 900 and frequency saves as 300. The data has been captured in a in terms of variations in Unit Distance and Max Distance followed by variations in the initial nodes and number of nodes in a sequential manner.
Data variations

As mentioned earlier, the purpose of this section is to analyze the data changes in response to a particular input parameter. Following is a brief outline of the data captured:

- Well-defined networks with almost 100% edges intact for all the motifs have been achieved in some of the tests.
- At the same time, some of the tests have yielded motifs without any edges at all. In such cases, the number of motifs with 0 edges is almost 100%.
- Some of the tests could not be completed due to memory issues and those entries have been left blank.

The analysis was done in a step by step manner by analyzing the results when each of the input parameters was changed for random seeds from 1 to 5. Accordingly the data analysis has been divided into various stages:

Variations Observed with changes in Unit Distance:

Tests were conducted by changing the Unit Distance alone keeping Initial Nodes, Maximum Distance, Number of nodes and the probabilities constant. The Unit distance was varied between 1 and 3 for tests without match files. The below graph depicts the variations in the percentage of motifs when Initial Node was set to 8 characters i.e. ATCATCAT, Maximum Distance was set to 1, Alphabet was set to A,T and the Unit Distance was changed from 1 to 2. The results have been taken when the number of nodes was 300 with No match files. The probabilities have been set as 0.2, 0.4, 0.2 and
0.2 for Prob_to_mutate, Prob_to_add, Prob_To_Delete and Prob_To_Duplicate respectively.

**GRAPH 1** - Comparision between Avg, Min and Max values when UD=1 & UD=2

The above graph compares the Minimum, Maximum and Average values of the Percentage of Motifs with 0, 1, 2 and 3 edges when the Unit Distance is changed from 1 to 2. The X-axis represents the no of edges and Y axis represents the % of Motifs.

It is seen from the above graph that the all the number of motifs with 0 edges is high when Unit Distance is low. This is depicted by the fact that the average, minimum and maximum values for UD=1 are higher than the ones for UD=2 for 0 edges. This means that the number of motifs without any edges is high when the Unit distance is low.
As the Unit distance increases, the number of motifs with edges seems to increase. This is evident by the fact that the number of motifs with 1, 2 and 3 edges is higher when UD is increased to 2. However, it is also observed that the number of motifs with 1 or more edges is far lesser as compared to the number of motifs with no edges in either case. It is observed that a minimum of 72% of motifs have no edges whereas the number of motifs with 3 edges is less than 1% always. The increase in UD seems to cause a slight increase in the percentage of motifs with 2 and 3 edges even though the value is still very low. These facts are further confirmed with the help of below graph which compare the values when UD is further increased to 3:

**GRAPH 2-Comparision between Avg, Min and Max values when UD=2 & UD=3**
The above graph confirms the theory that the percentage of motifs with 1 or more edges increases as the value of UD increases. This is evident by the fact that the minimum, maximum and average values of motifs with 0 edges is higher when UD=2 and lower when UD=3. Also, the percentage of motifs with one or more edges increases when UD increases thereby substantiating the theory that an increase in UD leads to more nodes with edges. When UD=3, the number of motifs with 3 edges becomes more than 1%. The below graph compares the average values of the percentage of motifs when UD=1, 2, 3 and 4:

![Graph 3 - Comparision between Avg values of UD=1, 2, 3 and 4](image)

The graph shows that the number of motifs with no edges goes in decreasing progressively as UD increases and the nodes with one or more edges increase with
increasing values of UD. The average percentage of motifs with no edges goes down from almost 90% to 60% as UD is increased from 1 to 4.

For more graphs comparing the varying values of UD, please refer to Appendix C. None of the graphs created for UD variations exhibited substantially unexpected variations. Though there were slight deviations, there were no major deviations and the values generally followed the expected path.

**Variations Observed with changes in Maximum Distance:**

The tests for observing variations due to change in Maximum distance were conducted by varying the Maximum Distance in the input property files between 1 to 3 for stages without match files and with match files. The below graph depicts the variations in the percentage of motifs when Initial Node was set to 8 characters i.e. ATCATCAT, Unit Distance was set to 1, Alphabet was set to A,T and The Maximum Distance was changed from 1 to 2. The results have been taken when the number of nodes was 300 with No match files.

Also, the probabilities have been set as 0.2, 0.4, 0.2 and 0.2 for Prob_to_mutate, Prob_to_add, Prob_To_Delete and Prob_To_Duplicate respectively.
The above graph compares the Minimum, Maximum and Average values of the Percentage of Motifs with 0, 1, 2 and 3 edges when the Maximum Distance is changed from 1 to 2. The X-axis represents the number of edges and the Y-axis represents the percentage of Motifs.

The variations observed in the above graph is similar to the one observed in the graphs for Unit distance variations. It is seen from the above graph that the all the number of motifs with 0 edges is highest when Maximum Distance =1. This is depicted by the fact that the average, minimum and maximum values for MD=1 are higher than the ones for MD=2 for 0 edges. This means that the number of motifs without any edges is high.
when the Maximum distance is low. As the Maximum distance increases, the number of motifs with edges seems to increase. This is evident by the fact that the number of motifs with 1, 2 and 3 edges is higher when MD=2 as compared to MD=1. However, it is also observed that the number of motifs with 1 or more edges is far lesser as compared to the number of motifs with no edges in either case. While the number of motifs with no edges is always more than 80%, the number of motifs with 3 edges is hardly 1% every time. The number of motifs with 3 edges is almost minimal. The increase in MD seems to cause a slight increase in the percentage of motifs with 2 and 3 edges even though the value is still very low. However, when MD is increased to 3, the number of nodes with more than 1 edge seems to increase at a substantially higher rate as compared to the values when UD is increased.

These facts are further confirmed with the help of below graph which compare the values when MD is further increased to 3 with the rest of the input parameters being the same:
The above graph confirms the theory that the percentage of motifs with 1 or more edges increases as the value of MD increases. This is evident by the fact that the minimum, maximum and average values of motifs with 0 edges is higher when MD=2 and lower for the rest. Also, the percentage of motifs with one or more edges is higher when MD=3 thereby substantiating the theory that an increase in MD leads to more motifs with edges. For more graphs comparing the values of MD, please refer to Appendix C. None of the graphs created exhibited substantially unexpected results. The number of motifs with 3 edges increased to an average value of 9% when MD was increased to 3.
Variations Observed with changes in Initial Nodes:

Initial Node is the structure of the node with which the network creation begins. As mentioned earlier, the tests were run for several values of Initial Nodes varying between 4 and 31. Several tests were run for different values of Initial Nodes with varying unit and maximum distances and probabilities. The below graph compares the variations in results when the Initial Node was set to “ATCAAC” i.e. 6 characters and “ATCATCAT” i.e. 8 characters. All the other parameters like Unit Distance, Maximum Distance, Number of Nodes etc was kept constant.

GRAPH 6-Comparision between Avg, Min and Max values when IN=6 & IN=8
The above graph shows that the response to variations in the Initial Node is opposite to that of the response to increase in Unit and Maximum distances. In case of variations in Initial Nodes, it is seen that the number of motifs with 0 edges increases as the number of characters in the Initial Node increases. Subsequently, the number of motifs with one or more edges decreases with the increase in the number of characters in Initial node. In the above graph, when IN had 6 characters, the percentage of motifs with 0 edges was less than 80%. When IN was increased to contain 8 characters, the number of motifs with 0 edges almost reached 90%. Consequently, the number of motifs with one or more edges kept decreasing for increasing values of IN and reached negligible levels for nodes with 3 edges. The below graph shows a comparison of average percentage values of motifs with 0,1,2, and 3 edges for various increasing values of IN:

GRAPH 7-Comparison between Avg percentage values for different IN
The graph goes on to prove that an increase in the number of characters leads to a decrease in the number of nodes with edges and consequently an increase in the number of edgeless nodes. When IN=4, the number of motifs with 0 edges is around 65 whereas it reached almost 95% when IN is increased upto 31 characters.

However, there is a brief period in between the 8th and 10th Initial Node where there was a slight deviation from the expected result and the graph showed contrary behavior. The number of motifs with 0 edges decreased from around 89% to 88% when the initial node was changed from 8 to 10 characters. However, the difference was slight to bear an impact on the overall result.

**Variations Observed with changes in number of nodes:**

All the tests were run to create networks with 900 nodes. However, the frequency of saving the networks was set to 300 which mean that the values were saved for 300, 600 and 900 nodes. The probabilities have been set as 0.2, 0.4, 0.2 and 0.2 for Prob_to_mutate, Prob_to_add, Prob_To_Delete and Prob_To_Duplicate respectively. Each of the output folders had different log files which captured the data at a frequency of 300 as more and more nodes were added to the network. The resultant data was analyzed to study the variations when the rest of the input parameters were same. The variations in output network as the number of nodes increases from 300 to 600 is depicted by the following graph where the Unit and Maximum distances are set to 1, alphabet is set to A,T and Initial Node is set to 8 characters:
The results seen from the above graph are similar to the results observed with the graphs for Initial Nodes. As more and more nodes are added, the number of edges seems to decrease. When the number of nodes increases from 300 to 600, the average number of motifs with no edges increases from 88 to 93%. Consequently, the number of motifs with one or more edges seems to decrease. When the number of nodes reaches 600, the percentage of motifs with 3 edges becomes almost negligible.

The below graph compares the average percentage values of nodes with 0, 1, 2 and 3 edges when the number of nodes reaches 300, 600 and 900:
GRAPH 9-Comparison between Avg percentage values for different Number of Nodes

From the above graph it is seen that when the number of nodes is 300, the average percentage of motifs without edges is around 88 whereas it increases to around 95 when the number of edges increases to 900. Similarly, the number of 1, 2 and 3 edges goes on decreasing with increasing number of nodes.

Variations Observed with changes in alphabets:

Overall, the tests were conducted with 3 alphabet combinations – AT, ATC and ATCG. All the nodes including the Initial Nodes were made up of various combinations of these alphabets. Each stage of the tests was conducted for all these 3 combinations. The following graph depicts the variations in the results when the alphabets are changed.
from A, T to A, T, C. The Unit and Maximum distances are set to 1 and the results have been taken for 300 nodes without match files. The probabilities have been set as 0.2, 0.4, 0.2 and 0.2 for Prob_to_mutate, Prob_to_add, Prob_To_Delete and Prob_To_Duplicate respectively.

GRAPH 10-Comparision between Avg, Min and Max values when Alphabet = AT & Alphabet = ATC

It is seen from the above graph that as the number of alphabets increases, the number of motifs without edges also increases. When the alphabet is set as AT, the average number of motifs without edges is around 89 whereas for ATC it increases to 91.
Consequently, the number of motifs with one or more edges decreases progressively. The same holds good for minimum and maximum values as well.

This trend is also evident from the below graph comparing the average values for AT, ATC and ATCG together.

**GRAPH 11-Comparison between Avg percentage values for different Values of Alphabets**

It is evident from the above graph that the number of motifs without edges increases as the alphabet size increases. However, it is also observed that the variations in this case are very less as compared to other input variables and the graphs almost follow the same path. For more similar graphs depicting the alphabet variations, please refer to Appendix C.
Variations Observed when the probabilities were changed:

All the tests have been conducted with 4 probability variables. The values of these variables have been taken as a set and have been used twice during the course of the tests, once without match files and once with match files. The following 4 set of probability values have been used to run 4 stages of tests:

First Stage - Prob_to_add:0.2, Prob_to_delete:0.4, Prob_to_mutate:0.2, Prob_to_duplicate:0.2

Second Stage - Prob_to_add:0.0, Prob_to_delete:0.6, Prob_to_mutate:0.4, Prob_to_duplicate:0.0

Third Stage - Prob_to_add:0.2, Prob_to_delete:0.8, Prob_to_mutate:0.0, Prob_to_duplicate:0.0

Fourth Stage - Prob_to_add:0.0, Prob_to_delete:0.0, Prob_to_mutate:0.6, Prob_to_duplicate:0.4

The variations in the average values of the percentage of motifs for these 4 stages have been depicted below graphically for with and without match files. The other input variables have been kept same. Unit Distance has been set to 3, Maximum Distance to 1, Number of Nodes to 300 and Initial Node has been set to ATCATCAT (IN=6).
GRAPH 12-Comparision of average values between stages without match file

GRAPH 13-Comparision of average values between stages with match file

**geneticCodePlus_1**
It is seen from the above graphs that the number of motifs without edges decreases for second stage, increases in the third stage and again decreases in the fourth stage. Similarly, the value of number of motifs with 1 or more edges increases in the second stage from the first stage, decreases again and then increases again in the fourth stage. This shows that the number of motifs without edges is higher in the first and third stages as compared to second and fourth. Consequently, the number of motifs with one or more edges is more in the second and fourth stages. This holds good for tests with and without match files. However, it has been observed that the variations are less severe when the match file is involved and the values increase or decrease only by slight amounts as compared to the tests without match files. Since the Prob_to_add is 0.2 in the first and third stages and 0 in the remaining stages, it goes to show that increasing this probability variable causes an increase in the number of motifs without edges.

**Variations Observed when Match files were introduced:**

After 4 stages of tests, Match files were introduced and the same tests were repeated in the same number of stages for 5 match files. The match files basically define the strings which are to be considered equal during the creation of networks and are plain text files. There were a total of 5 match files which were used each with different matches defined. The 5 match files are geneticCodePlus_1, geneticCodePlus_2, geneticCodePlus_3, geneticCodePlus_4 and geneticCodePlus_5.

The following graph compares the data variations for the same set of input parameters with and without match files. The data has been captured for Unit Distance = 3, Maximum Distance = 1, Nodes = 300, Alphabet = A, T, C, G and Initial Node = ATCATCAT (IN = 8). In case of tests with match files the unit distance was kept
constant at 3 for all the tests. The probabilities have been set as 0.2, 0.4, 0.2 and 0.2 for Prob_to_mutate, Prob_to_add, Prob_To_Delete and Prob_To_Duplicate respectively for both the tests. The match file used is geneticCodePlus_1. This file defines several matches for various strings like TTA = TTG, CTT, CTC, CTA etc. The actual match files used can be found in Appendix B:

![Graph](image)

**GRAPH 14-Comparison between Avg, Min and Max values With and Without match files.**

It can be seen from the above graph that the introduction of the match file is actually causing a slight increase in the number of motifs with edges. Without the match file, the average number of motifs with no edges was around 82% which decreased to
around 80% after the introduction of match files which consequently caused a increase in
the number of motifs with edges though the increase was slight.

Variations Observed when Match file was changed:

Within each stage of test, the tests were run for the 5 different match files and the
results were summarized in the same sheet. The data was found to vary whenever the
match file was changed. The below graph represents the variation in data when the match
file was changed from geneticCodePlus_1 to geneticCodePlus_2. All the remaining input
parameters were the same for both the tests. The Unit Distance was set at 3, Maximum
Distance at 1, Number of Nodes at 300, Alphabet at A,T and Initial Node as ATCATCAT
(IN = 8). The probabilities Prob_to_mutate, Prob_to_add, Prob_To_Delete and
Prob_To_Duplicate were set as 0.2, 0.4, 0.2 and 0.2 respectively for both the tests:
It is seen from the above graph that the average number of nodes without edges is higher for geneticCodePlus_2 as compared to geneticCodePlus_1. The difference between the two match files is only in terms of the matching strings that they define. geneticCodePlus_2 defines more number of matching strings. For e.g., in geneticCodePlus_1, there are 5 strings defined which have a self match whereas in geneticCodePlus_2, there are 16 strings which have a self match. Similarly, the number of strings with a self match only is increased progressively in the match files. Self match is indicated by leaving the right hand side of the match as blank (ABC=). This goes to
show that the increasing the number of self matching strings leads to an increase in the number of edgeless nodes and a decrease in the number of edges. This is seen in a 3D graph format as below:

**GRAPH 16-Comparision between different Match Files**

The test for the 4th match file could not complete due to the memory getting full. Also, a slight variation from the expected results was observed in case of match file 5 wherein the number of motifs with no edges decreased slightly and the number of motifs with edges increased even though the number of self matching strings was highest in this match file. This can be observed by the slightly uneven curve near the surface for match file 5 in the graph.
As mentioned before, the tests were run for 5 match files. Each of them defines a different set of matches. If a particular string has nothing on the right hand side, it is considered to be a self-matching string or a single match string with no other matches (e.g. AAA=AAA). If a particular string has one match defined, it is considered as a string with 2 matches (self match and the defined match). The various match files have different strings defined with several matches starting from 1 to 5. Each file has a total of 64 strings defined each of them having a minimum of 1 match (self) and a maximum of 16 matches. The frequency of the occurrence of these matches has been depicted graphically below for the 5 match files.

GRAPH 17-Frequency of Number of matches for Match File geneticCodePlus_1
GRAPH 18-Frequency of Number of matches for Match File geneticCodePlus_2

GRAPH 19-Frequency of Number of matches for Match File geneticCodePlus_3
From the above graphs it is clear that the number of self matching strings is increasing in the consecutive match files. This can also be visualised overall in the form of the below graph:
As discussed above, the increase in number of self-matches seems to be directly proportional to the number of nodes without edges except for some minor deviations.

All the graphs depicted above the above section have been picked up randomly from the overall data outputs captured. For the complete set of data captured and more graphs, please refer to Appendix B and C. The observations made by analyzing these graphs and the data will be discussed in more detail in the next section.
OBSERVATIONS AND FINDINGS

The data gathered through the testing phase has been analyzed to put the findings in perspective. Based on the findings of the research and evaluation and interpretation of the same, a few observations were drawn out. These observations were also documented for future references along with the rest of the data. Some of these observations and analysis are discussed further.

An increase in the value of Unit Distance causes a decrease in the number of motifs with no edges and a subsequent increase in the number of nodes with one or more edges (Refer to Graph 1 in Appendix C). This is because the program uses Type of distance as “Hamming” in which case, when the unit distance is 1, the program adds edges only when the hamming distance is less than the max distance. Hence when the Unit Distance increases, the hamming distance becomes less and the probability of hamming distance being less than maximum distance increases. Hence, more edges are added.

An increase in the value of Maximum Distance causes a decrease in the number of motifs with no edges and a subsequent increase in the number of motifs with one or more edges (Refer to Graph 2 in Appendix C). This is because the program uses Type of distance as “Hamming” in which case, the program adds edges only when the hamming distance is less than the max distance. Hence when the Maximum Distance increases, the probability of hamming distance being less than maximum distance increases. Hence, more edges are added.

Increasing the number of nodes in Initial characters causes an increase in the number of motifs without edges and a decrease in the number of nodes with edges (Refer to Graph 3
in Appendix C). This goes to show that the complexity of the initial node is a contributing factor towards the number of edges. This is because when the structure of nodes becomes complex, the hamming distance increases and lesser number of edges are added.

Adding more nodes to the networks causes an increase in number of motifs without edges (Refer to Graph 4 in Appendix C). This goes to show that nodes are added at a higher rate than edges.

Changing the alphabets from AT to ATC and ATCG causes an increase in the number of motifs without edges and a corresponding decrease in the number of motifs with edges (Refer to Graph 5 in Appendix C). However, the variations observed in this case are relatively minor even though the number of motifs with 3 edges is almost negligible. The variations can be attributed to the fact that if the node contains more number of alphabets, the difference in structure and consequently hamming distance will be more.

Increasing the Prob_to_add variable causes an increase in the number of edgeless nodes (Refer to Graph 6 in Appendix C). This is because when the probability to add is high, more symbols or nodes get added to the node structure. This holds good with or without match files. However, the variations are less dramatic when match files are involved.

Introduction of match files causes an increase in the number of motifs with edges (Refer to Graph 7 in Appendix C). However, when the match files were changed by changing the value of “File_matches” in the properties file to indicate a different match file, the number of motifs without edges increased. For e.g., in geneticCodePlus_1, the number of self matches is defined as 5 and the remaining nodes have more than 1 match whereas in
geneticCodePlus_2, the number of self matches is 16 which means the number of nodes with more than 1 match is lesser. This leads to a higher value of hamming distance thereby decreasing the number of edges.

The edges are lowest when the Unit distance and Max distance are both 1 (Refer to Graph 8 in Appendix C). In this case, the percentage of nodes with more than two edges is very low. As the Max Distance is increased, the number of motifs with no edges decreases. The same holds good when the Unit Distance increases, though the variance in the percentage of nodes with no edges is slightly lesser in this case.

As the initial node becomes more complex and more number of nodes are added, increasing Unit and Maximum distances seem to have little impact on the addition of edges.

When the initial node is fairly simple like ATCA and Unit and Maximum Distances are set to their maximum values (4 and 3 respectively), the number of motifs with 3 edges reaches almost 100%. This seems to hold good for all the alphabets when the tests are run without match files.

However, when match files are used, irrespective of the other input parameters, the number of motifs with 3 edges never seems to go beyond 70%.
Findings

Based on the above observations, the following conclusions have been drawn:

- To generate well-defined networks with more edges, the Unit and Maximum distance have to be kept high.

- If the node structure is smaller, for e.g., if the initial node is ATAC then to get a 300 node network with all edges intact, approximately a unit distance and max distance of 3 will suffice.

- As seen above, as the number of nodes increases, the number of edges decreases. In order to get a 300 node network with all edges intact and the initial node TTACCCTTATCG, the unit and max distance will have to be set to be around 4.

- As the initial node becomes more complex say when IN =16, even to get a node with 50% edges intact, the unit and maximum distances have to be set to 4 and 3 respectively.

- When match files are involved, to ensure a well formed network with proper edges, the matches have to be defined carefully in such a way as to minimize the hamming distance.

- When the initial node is sufficiently small like ATAC, with a well-defined match file, around 70% of network with intact edges has been achieved. To create a well formed network with all the edges intact, probably either the distances have to be increased or the matches redefined.
**Anomalies**

Also, some variations have been observed in places where the actual data deviates from the expected data. Some of the deviations observed are as follows:

- It has been observed that the number of motifs with 0 edges increases when the initial node becomes more complex. However, when the initial node was increased from 8 to 10 characters, the number of motifs without edges dropped from 89 to 88 percent.

- It has been observed that the number of motifs without edges increases when match files with more self matches are introduced. However, when match file 5(geneticCodePlus_5) which had the highest number of self matches among all the match files was introduced, the number of motifs without edges decreased from 90 to 84 percent.

However, none of these variations were significant enough to be considered more than random. Slight random variations can probably be attributed towards normal software margins.
CONCLUSION

The project intended was to study the theory and behavior of network motifs in general and to analyze the creation of networks motifs with the help of a simulation program in particular. The overall aim of this research was to gather findings and observations with respect to the motif formation in order to understand the ways to create well-defined networks. The aim of the project has been met successfully. An extensive study was conducted on the theory behind network motifs in the literature review stage. In the testing stage, several tests with different input parameters were run in order to gather data for analysis. The gathered data was analyzed extensively to draw observations and conclusions. Several observations and findings have been drawn out consequently which will help in better understanding of the way networks and motifs are created and the requirements for creating a good well-formed network.

Problems encountered

Overall, the execution of the project went fairly smoothly. However, some of the tests could not be completed due to memory problems. Also, the sheer number of tests which needed to be run to form reasonable opinions was time consuming. The summarizing of the data was also slightly time consuming due to which an automated program was developed to reduce effort.

Future Works

The project has been successful in finding the requirements to form well defined networks with 100% edges intact in all the motifs for smaller nodes and networks. However, a maximum value of 70% motifs with well-formed edges could be achieved for very high values of nodes (900 nodes) with complex node structure and match files.
Further tests will need to be run for higher values of distance and more match files to determine the thresholds at which these types of nodes can achieve 100% well formed motifs. This could be an area of future research.

Also, the current work explores only the formation of undirected networks. Further research can be conducted to apply the same principles to the creation of directed networks and corresponding motifs.

Finally, the gathering of data into summary sheets manually was a time consuming task due to which an automated program was developed. However, this program cannot read .gzip files at the moment due to which the data had to be unzipped manually before running the program. The program can be further improved in order to read different formats of files automatically so that it will ease future research.
REFERENCES


http://biogrid. engr. uconn. edu/REU/Reports_09/Final_Reports/Michael_Zuba. pdf.
Appendix A

Download locations for Network Motif Detection Tools

**Pajek** – The latest version of Pajek can be downloaded for free from the following location:

[http://pajek.imfm.si/doku.php?id=download](http://pajek.imfm.si/doku.php?id=download)

**Mfinder** – The latest version of Mfinder can be downloaded for free from the following location:


**MAVisto** – The latest version of MAVisto can be downloaded for free from the following location:

[http://mavisto.ipk-gatersleben.de/](http://mavisto.ipk-gatersleben.de/)

**FANMOD** – The latest version of FANMOD can be downloaded for free from the following location:

[http://theinf1.informatik.uni-jena.de/~wernicke/motifs/index](http://theinf1.informatik.uni-jena.de/~wernicke/motifs/index).
Appendix B

Data and match files:

Data for tests without match files:

Data for tests with match files:

Match files used:

Sample Summary Data:

Summary data collected when Node = AT, Initial Node = 8, Unit Distance and Maximum Distance = 1 and Nodes = 300.

<table>
<thead>
<tr>
<th>random seed</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>average</th>
<th>variance</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg edge=0</td>
<td>0.906147</td>
<td>0.909562</td>
<td>0.869423</td>
<td>0.895629</td>
<td>0.867981</td>
<td>0.889749</td>
<td>0.000396</td>
<td>0.867981</td>
<td>0.909562</td>
</tr>
<tr>
<td>avg edge=1</td>
<td>0.088857</td>
<td>0.073085</td>
<td>0.110764</td>
<td>0.08769</td>
<td>0.118614</td>
<td>0.095802</td>
<td>0.000344</td>
<td>0.073085</td>
<td>0.118614</td>
</tr>
<tr>
<td>avg edge=2</td>
<td>0.004313</td>
<td>0.016447</td>
<td>0.018095</td>
<td>0.01566</td>
<td>0.011872</td>
<td>0.013278</td>
<td>3.03E-05</td>
<td>0.004313</td>
<td>0.018095</td>
</tr>
<tr>
<td>avg edge=3</td>
<td>6.83E-04</td>
<td>9.05E-04</td>
<td>0.001717</td>
<td>0.001022</td>
<td>0.001533</td>
<td>0.001172</td>
<td>1.9E-07</td>
<td>0.000683</td>
<td>0.001717</td>
</tr>
</tbody>
</table>
Appendix C

Summary Graphs:

Graphs for Various Input variable Variations:

Referred Graphs:

**Graph 1**

UD=1, IN=4, Alphabet=A, T, Nodes=300

<table>
<thead>
<tr>
<th>Number of Edges</th>
<th>Percentage of Motifs</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg edge= 0</td>
<td>0.6585, 0.2622, 0.0676, 0.0118</td>
</tr>
<tr>
<td>avg edge= 1</td>
<td>0.3788, 0.4050, 0.1630, 0.0532</td>
</tr>
<tr>
<td>avg edge= 2</td>
<td>0.1130, 0.3230, 0.3388, 0.2252</td>
</tr>
<tr>
<td>avg edge= 3</td>
<td>0.0000, 0.0000, 0.0000, 0.0000</td>
</tr>
</tbody>
</table>

**Graph 2**

UD=1, IN=4, Alphabet=A, T, Nodes=300

<table>
<thead>
<tr>
<th>Number of Edges</th>
<th>Percentage of Motifs</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg edge= 0</td>
<td>0.6585, 0.2622, 0.0676, 0.0118</td>
</tr>
<tr>
<td>avg edge= 1</td>
<td>0.3788, 0.4050, 0.1630, 0.0532</td>
</tr>
<tr>
<td>avg edge= 2</td>
<td>0.1130, 0.3230, 0.3388, 0.2252</td>
</tr>
<tr>
<td>avg edge= 3</td>
<td>0.0000, 0.0000, 0.0000, 0.0000</td>
</tr>
</tbody>
</table>
**Graph 7**

UD=3, MD=1, IN=8, Nodes=300

<table>
<thead>
<tr>
<th>Match File 1</th>
<th>Match File 2</th>
<th>Match File 3</th>
<th>Match File 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8023</td>
<td>0.8312</td>
<td>0.9047</td>
<td>0.8461</td>
</tr>
<tr>
<td>0.1617</td>
<td>0.1441</td>
<td>0.0825</td>
<td>0.1294</td>
</tr>
<tr>
<td>0.0394</td>
<td>0.0214</td>
<td>0.0116</td>
<td>0.0211</td>
</tr>
<tr>
<td>0.0056</td>
<td>0.0033</td>
<td>0.0012</td>
<td>0.0034</td>
</tr>
</tbody>
</table>

**Graph 8**

MD=1, IN=6, Nodes=300, Alphabet=AT

<table>
<thead>
<tr>
<th>UD 1</th>
<th>UD 2</th>
<th>UD 3</th>
<th>UD 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7719</td>
<td>0.5214</td>
<td>0.3794</td>
<td>0.2476</td>
</tr>
<tr>
<td>0.1936</td>
<td>0.3473</td>
<td>0.4110</td>
<td>0.4134</td>
</tr>
<tr>
<td>0.0301</td>
<td>0.1006</td>
<td>0.1507</td>
<td>0.2207</td>
</tr>
<tr>
<td>0.0044</td>
<td>0.0307</td>
<td>0.0589</td>
<td>0.1183</td>
</tr>
</tbody>
</table>
Appendix D

Source code for Parser program which automates the data entry process:

Parser.zip

Input Data  Folder Structure for this program:

Data.rar