ANALYSIS OF PROKARYOTIC METABOLIC NETWORKS

by

Caroline Urquhart

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
March 2011

DALHOUSIE UNIVERSITY

DEPARTMENT OF MATHEMATICS AND STATISTICS

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "ANALYSIS OF PROKARYOTIC METABOLIC NETWORKS" by Caroline Urquhart in partial fulfillment of the requirements for the degree of Master of Science.

Dated: March 30, 2011

Supervisors: ______________________

____

Reader: ______________________

DALHOUSIE UNIVERSITY

DATE: March 30, 2011

AUTHOR: Caroline Urquhart

TITLE: ANALYSIS OF PROKARYOTIC METABOLIC NETWORKS

DEPARTMENT OR SCHOOL: Department of Mathematics and Statistics

DEGREE: M.Sc. CONVOCATION: May YEAR: 2011

Signature of Author

I would like to dedicate this thesis to my parents, Donald and Diane Urquhart, who taught me the value of education. I am deeply indebted to them for their continued support and unwavering faith in me.

Table of Contents

List of Tables

List of Figures

Abstract

Establishing group structure in complex networks is potentially very useful since nodes belonging to the same module can often be related by commonalities in their biological function. However, module detection in complex networks poses a challenging problem and has sparked a great deal of interest in various disciplines in recent years [5]. In real networks, which can be quite complex, we have no idea about the true number of modules that exist. Furthermore, the structure of the modules may be hierarchical meaning they may be further divided into sub-modules and so forth. Many attempts have been made to deal with these problems and because the involved methods vary considerably they have been difficult to compare [5]. The objectives of this thesis are (i) to create and implement a new algorithm that will identify modules in complex networks and reconstruct the network in such a way so as to maximize modularity, (ii) to evaluate the performance of a new method, and compare it to a popular method based on a simulated annealing algorithm, and (iii) to apply the new method, and a comparator method, to analyze the metabolic network of the bacterial genus *Listeria*, an important pathogen in both agricultural and human clinical settings.

Acknowledgements

I wish to thank Dr. Hong Gu and Dr. Joseph Bielawski, without whose guidance and encouragement, this thesis would not have been possible. I would also like to thank Dr. Katherine Dunn who has offered me technical assistance many times throughout.

Chapter 1

Introduction

1.1 The Cartography of Complex Networks

When studying the structure of complex networks, knowing the manner in which the components of the system are connected is crucial [8]. It is practical to think in terms of network structure when studying complex networks since the topology of these networks defines the interactions, which in turn, provides insight into the function and evolution of the network's components. However, due to the large sizes and complexities of these networks, such insight is often difficult to discern [8]. Examples of complex networks include the internet, food webs, social networks, and metabolic networks to name a few [19].

To extract relevant information from the structure of large complex networks, it is useful to depict these networks as graphs with a set of nodes (vertices) and links (edges) between couples of nodes. To demonstrate this point, consider forming a network of all of the cities and towns within a country and all the roads that connect them. The cities and towns are represented by nodes and the roads by links. It is impractical to have a map that represents the cities and towns by circles of a fixed size and the roads by lines of a fixed width. Instead, real maps highlight capitals and main lines of communication so that one can obtain scale-specific information at a glimpse. Likewise, attempting to extract information from a network comprised of hundreds or even thousands of nodes and links is almost impossible unless the information about nodes and links is displayed in a scale-specific context [8].

Social network analysis and statistical modeling are common analytical methods used for modeling real-world networks. A great deal of the terminology of social network analysis, such as path lengths, cliques, and connected components, to mention a few, were either directly taken from or adapted from the field of graph theory, to deal with questions pertaining to status, influence, cohesiveness, social roles, and identities in social networks. Thus, over and above the role graph theory plays in describing abstract models, graph theory became a functional tool for analyzing empirical data [18]. Traditional tools of social network analysis are more useful in cases where networks are poor candidates for statistical modeling. An example of such a network is the network of contacts between terrorists studied by Krebs [14]. It is a poor candidate for statistical modeling because the questions of interest in this network are not statistical in nature [18]. Nevertheless, statistical models can be very successful and informative when applied to real world networks [18]. Statistical models and techniques that are commonly used in network analysis are the small-world model which is motivated by clustering algorithms, random graph models which include Poisson random graphs and generalized random graphs, degree distributions, and clustering coefficients to name a few (for a review, see references [25] [11] [18]). Other complex approaches exist but are beyond the scope of this thesis, as they are not applicable to metabolic networks.

Whether networks are biological or artificial in nature, the science of measuring the structural features of a network, constructing a map of its structural and functional components, and measuring how they are connected (i.e., the topology of the network) is referred to as "cartography".

1.2 Some Basic Concepts on the Measures of Complex Networks

Many measures of complex networks have been proposed in recent years. Among the most significant are (i) average path length, $\bar{l}$, (ii) average node degree, <k>, (iii) clustering coefficient, C_i, and (iv) degree distribution (gamma value). Below I review each one, and provide an interpretation within the context of metabolic networks, as these are the focus of this thesis.

Average path[1] length is defined as the shortest distance averaged over all pairs of nodes in a network, where distance is defined as the minimum number of edges between nodes [25]. For instance, shorter average path length, in a metabolic network, means that metabolic end-products can be produced in a shorter average number of steps, thus, using a smaller amount of energetic resources [16]. For a random network (i.e., obtained from starting with a set of n isolated nodes and adding edges between randomly chosen pairs), the average expected path length is given by:

$$\bar{l}_{rand} = \frac{ln(n)}{ln < k >} \tag{1.1}$$

where n is the number of nodes and <k> is the average degree[2] per node in the network. This equation is used to compare the average path length of a real or

[1]A path is defined as any route along the edges of a graph

[2]The degree of a node i is defined as the total number of its connections

simulated network to a random network with the same n and <k> [24].

The average node degree, <k>, is the average number of links of a node in a network. For a metabolic network, low average node degree implies that fewer cellular resources are devoted to the processing of each metabolite[3] [16].

The clustering coefficient C_i for a node i is defined by Watts and Strogatz [26] as the proportion of links between the nodes within its neighbourhood divided by the number of links that could possibly exist between them. The clustering coefficient is given by:

$$C_i = \frac{2E_i}{k_i(k_i - 1)} \tag{1.2}$$

where E_i is the number of links among all neighbours[4] of node i and k_i is the degree of node i. If node i has a clustering coefficient of 1, each of its neighbour nodes are directly connected to each other. In the context of metabolic networks, node i therefore increases redundancy in the network by providing alternate pathways between neighbouring metabolites [16]. The clustering coefficient for the whole system is given as the average of C_i over all n nodes:

$$\overline{C} = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{1.3}$$

The average clustering coefficient only ever equals 1 when the network forms a complete graph. That is, when every node connects with every other node.

The spread of node degrees across a network is referred to as the degree distribution. The degree distribution is characterized by a distribution function $P(k)$,

[3]An organic compound that is a starting material, an intermediate in, or an end product of metabolism [17]

[4]If two nodes are connected by a link, they are called neighbours

which is the probability that a node, chosen at random, has precisely k edges [25]. The degree distribution for many networks can be described by a power law[5], of the form $P(k) \sim k^{-\gamma}$, where γ represents the rate of the decay. High values of γ indicate a higher rate of decay (i.e., a smaller fraction of high-degree nodes). Networks that follow a power-law degree distribution are referred to as scale-free networks [25].

1.3 Metabolism & Metabolic Networks

Metabolic processes are the basis of life, allowing cells to grow and reproduce, maintain their structures, and respond to their environments. The process of metabolism involves breaking down certain materials to provide energy for vital cellular processes, as well as producing other materials that are necessary for the maintenance of life [11].

A metabolic network is an abstract representation of cellular metabolism, which depicts all the interactions occurring between metabolites and biochemical reactions within a living cell. Studying cellular metabolism through the analysis of metabolic networks, can help us understand and make use of cellular metabolic processes in order to promote development in the medical industry [11].

In metabolic networks, nodes represent metabolites and two nodes i and j are connected by a link if there is a chemical reaction in which i is a substrate[6] and j a product[7], or vice versa. Figure 1.1 shows a cartographic model of the metabolic network of the bacterium *Psuedomonas*.

[5]The probability P(k) that a node in the network connects with k other nodes is roughly proportional to $k^{-\gamma}$

[6]a molecule which is acted upon by an enzyme

[7]a substance found at the end of a chemical reaction

It is conceivable that the nodes in a modular[8] metabolic network are connected based on the biological role they fulfil [7]. Hence, nodes with the same biological role could have similar topological roles. The participation coefficient and the within-module degree of a node can determine its topological role (or "node" role) to a great extent. Nodes with similar roles are expected to have similar relative within-module connectivity. If k_i is the number of links of node i to other nodes in its module s_i, $\overline{k}_{s_i}$ is the average of k over all the nodes in s_i and $\sigma_{k_{s_i}}$ is the standard deviation of k in s_i, then:

$$Z_i = \frac{k_i - \overline{k}_{s_i}}{\sigma_{k_{s_i}}} \tag{1.4}$$

is referred to as the Z-score [7]. The within-module degree Z-score is a measure of how well node i connects with other nodes within its module. High values of Z_i, more specifically $Z \geq 2.5$, indicate high within-module degrees and are classified as hub-nodes. Lower values of Z_i (i.e., $Z < 2.5$) are classified as non-hub nodes.

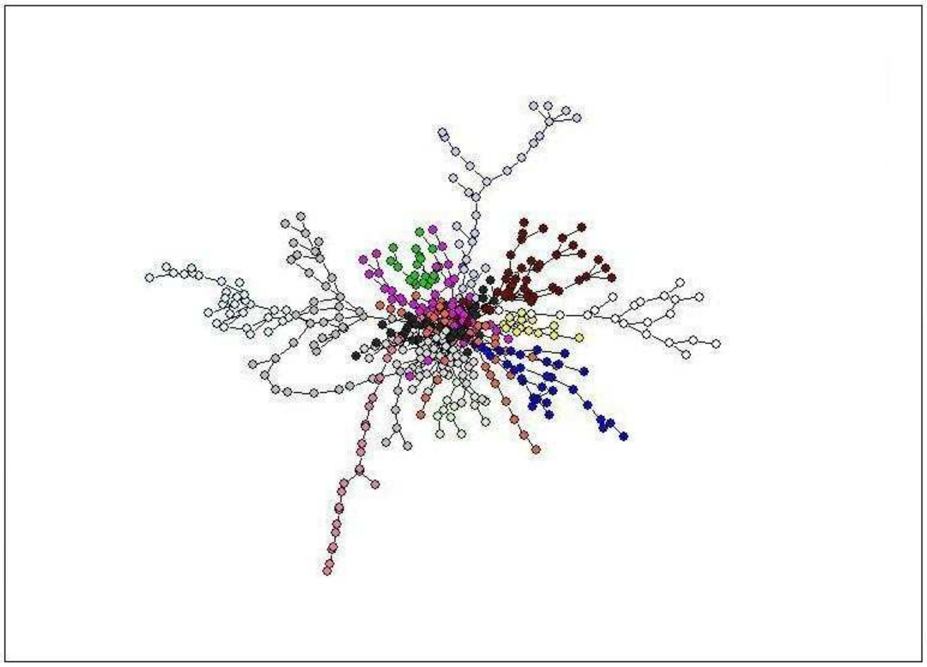
The participation coefficient P_i measures how uniformly the links of node i are distributed among different modules. The participation coefficient P_i of a node i is defined by Guimerà and Amaral [7] as:

$$P_i = 1 - \sum_{s=1}^{N_m} (\frac{k_{is}}{k_i})^2 \tag{1.5}$$

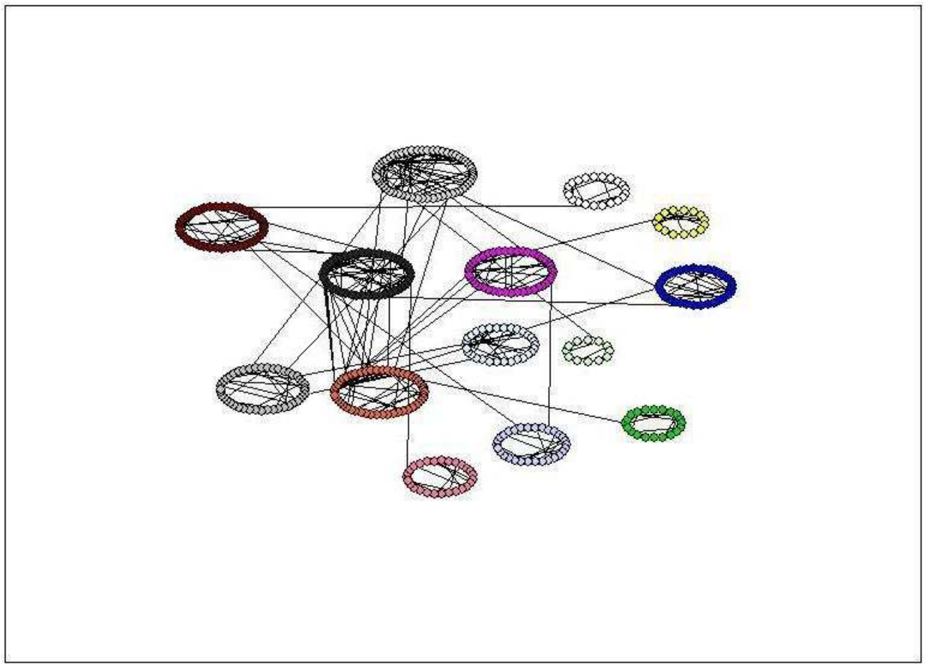
where N_m is the number of modules in the network, k_{is} is the number of links of node i to nodes in module s, and k_i is the total degree of node i. The participation coefficient of a node is therefore close to 1 if its links are evenly distributed among all the modules and 0 if all its links are within its own module.

[8]A modular representation of a network depicts groups of nodes, which are presumed to be related in their functionality, as circular modules

Guimerà and Amaral [7] define seven different role classifications, three of which are assigned to hub-nodes and the remaining four are attributed to non-hub nodes. Each role is defined by a region in the Z-P parameter space (see Figure 1.2). Hub-nodes can be classified as (i) provincial hubs - hubs that have most of their links within their module ($P \leq 0.30$)); (ii) connector hubs - hubs with many of their links going to most of the other modules ($0.30 < P \leq 0.75$); and (iii) kinless hubs - hubs with links distributed uniformly among all modules ($P > 0.75$). Non-hub nodes can be classified as (i) ultra-peripheral nodes - nodes with all their links within their module ($P \leq 0.05$); (ii) peripheral nodes - nodes with the vast majority of their links within their module ($0.05 < P \leq 0.62$); (iii) non-hub connector nodes - nodes with many links to other modules ($0.62 < P \leq 0.80$); and (iv) non-hub kinless nodes - nodes with links distributed uniformly among all modules $P > 0.80$).



(a) A cartographic model of the *Pseudomonas* metabolic network



(b) A modular model of the *Pseudomonas* metabolic network

Figure 1.1: *The metabolic network of the bacterium Pseudomonas. (a) A cartographic model of the* Pseudomonas *metabolic network. The nodes represent metabolites (i.e., products and substrates) and the links represent genes (enzymes). Node colour represents module assignment. (b) A modular representation of (a). Each circle in the network represents a module, which is a set of nodes that are grouped together based on similarities in their connectivity patterns [7]. The modules in this network are computationally defined to aid in module identification.*

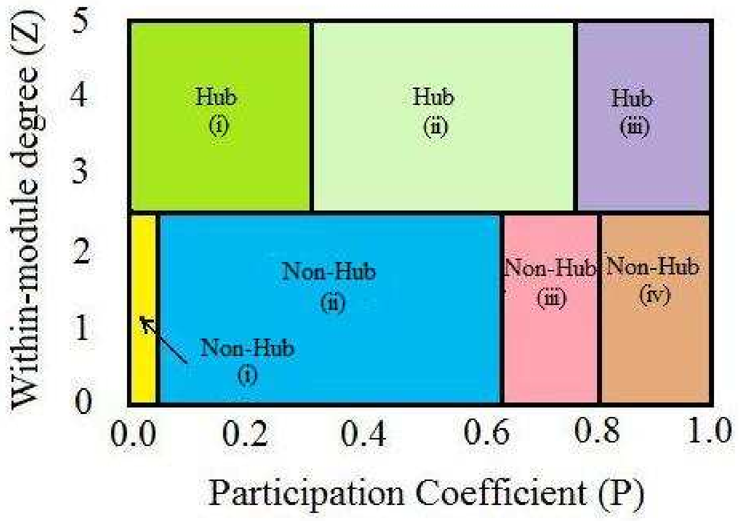


Figure 1.2: *Role-specific regions in the Z-P parameter space. Node roles are determined by the within-module degree, Z, and the participation coefficient, P. Nodes with $Z \geq 2.5$ are classified as module hubs and nodes with $Z < 2.5$ as non-hubs. Non-hub nodes can be classified into four different roles:(i) ultra-peripheral nodes, (ii) peripheral nodes, (iii) non-hub connector nodes, and (iv) non-hub kinless nodes. Hub-nodes can be classified as (i) provincial hubs, (ii) connector hubs, and (iii) kinless hubs [7].*

1.4 Biological Modularity, Module Structure & Module Identification Methods

Biological networks appear to exhibit modularity in topological structure. In the field of network biology, nodes and edges are defined according to the type of network being examined. As mentioned earlier, metabolic networks contain metabolite nodes and edges represent the enzyme-catalyzing reactions that connect them. Modularity in biological networks is believed to allow network sub-groups to function semi-autonomously. Hence, modularity could give a certain degree of evolvability to a network by enabling sub-groups to undergo changes without substantially altering the functionality of the whole system [22]. This would suggest that the metabolic pathways between-modules are relatively more constrained than those within-modules.

The modular structure of complex networks is believed to play a crucial role in their functionality. It is therefore necessary to develop algorithms to identify modules accurately [8]. The goal of module determining algorithms is to maximize a given modularity function, thus uncovering the configuration that maximizes modularity in the network. A good network partition consists of many within-module links and as few as possible between-module links. However, by strictly minimizing the number of between-module links or equivalently maximizing the number of within-module links, the most favourable partition of the network will be composed of a single module and no between-module links [8].

Guimerà and Amaral [7] identify modules by maximizing the network's modularity function. For a given partition of the nodes of a network into modules, the modularity M of this partition is

$$M = \sum_{s=1}^{N_m} [\frac{l_s}{L} - (\frac{d_s}{2L})^2] \tag{1.6}$$

where N_m is the number of modules in the network, L is the total number of links in the network, l_s is the number of links between nodes in module s, and d_s $(\sum k_i)$ is the sum of the degree of the nodes in module s. The quantity $\frac{l_s}{L}$ is the proportion of links within module s and the quantity $(\frac{d_s}{2L})^2$ is the estimated proportion of links one would expect within module s from chance alone. The logic behind this modularity definition is that, in a modular network, links are not uniformly distributed, thus a partition with a high modularity is such that the density of within-module links is significantly higher than the random expectation of that density [10].
They identify modular structure of metabolic networks through the use of simulated annealing [7], which is a probabilistic algorithm designed to locate a good approximation to the global optimum of a given function in a search space. Simulated annealing will be discussed in further detail in chapter 2.

Newman [19] defines modularity as the number of links falling within-groups minus the expected number in the equivalent network with links placed at random. This modularity can take on a positive or negative value with positive values indicating the possible presence of modular structure. Newman used this framework to search for modular structure within ecological networks by looking for divisions of a network that have positive, and preferably large values of modularity. A review of Newman's modular concept and a discussion about it is given below.

For a network with n nodes and a particular division of the network into two groups, the modularity function, Q, is given by:

$$Q = \frac{1}{4L} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2L})(s_i s_j) \tag{1.7}$$

where $s_i = 1$ if node i belongs to group one and $s_i = -1$ if node i belongs to group two. Let the number of links between nodes i and j be A_{ij} (these are the elements of the adjacency matrix), which normally takes values of 0 or 1, although larger values are possible for networks having multiple links between nodes. The expected number of links between nodes i and j, if links are placed at random, is $\frac{k_i k_j}{2L}$ where k_i and k_j are the degrees of the nodes and $L = \frac{1}{2}\sum_i k_i$ is the total number of links in the network. The fraction $\frac{1}{4L}$ is a multiplicative constant included for compatibility with the previous modularity definition [20].

It is noteworthy that the quantity $\frac{k_i k_j}{2L}$ can be larger than 1, thus when A_{ij} takes values of 0 or 1, this quantity cannot be the expected number of links between nodes i and j, as possibly intended by the definition originally. Thus, the above function may not be properly defined. The effect of maximizing the sum of the first term, A_{ij}, in the function Q, is to maximize the difference between the total number of links within-groups and the total number of links between-groups. If we let x denote the total number of links in group one, let y denote the total number of links in group two, and let z denote the total number of links between the two groups, then the sum of the first term in (1.7) can be re-written as:

$$\sum_{ij} A_{ij}(s_i s_j) = 2(x + y - z) \tag{1.8}$$

The sum of the second term in (1.7) is equal to $\sum_i \frac{(k_i s_i)^2}{2L}$, which can be interpreted as:

$$\begin{aligned}
\frac{(\sum_i k_i s_i)^2}{2L} &= \frac{1}{2L}\Big(\sum_{i\in grp1} k_i - \sum_{i\in grp2} k_i\Big)^2 \\
&= \frac{1}{2L}(2x-2y)^2 \\
&= \frac{4(x-y)^2}{2L}
\end{aligned}$$

Thus, (1.7) can be re-written as:

$$Q = \frac{1}{4L}\left[2(x+y-z) - \frac{4(x-y)^2}{2L}\right] \tag{1.9}$$

Since L in equations (1.7) and (1.9) denotes the total number of links in the network, $L = x + y + z$. Substituting this quantity into 1.9 yields:

$$Q = \frac{4xy - z^2}{2L^2} \tag{1.10}$$

Thus, maximizing Q is equivalent to maximizing 4 times the product of the number of within-group links minus the square of the total number of links between-groups. This is not a bad function to maximize, but it is not necessarily the best. This is an equivalent function to maximize for the two module case in (1.6), in the sense that both functions will identify the same modules. The relationship between functions Q and M in the two module case is as follows. If the term d_s in (1.6) is the sum of the degree of all nodes in group s (for $s \in 1:2$) and if the term $k_i s_i$ in (1.7) is the product of the degree of node i and its respective group membership ($s_i = \pm 1$), then the relationship between d_s and $k_i s_i$ is:

$$\Big|\sum_i k_i s_i\Big| = \begin{cases} d_1 & \text{if } s_i = 1 \\ d_2 & \text{if } s_i = -1 \end{cases}$$

At the same time, if the term l_s in (1.6) is the total number of links in group s (for $s \in 1:2$) and the term $\sum_{ij} A_{ij}(s_i s_j)$ in (1.7) is the difference between the total number of within-module links, $x + y$, and the total number of between-module links, z, then the relationship between l_s and $\sum_{ij} A_{ij}(s_i s_j)$ is:

$$\sum_{ij} A_{ij}(s_i s_j) = \begin{cases} l_1 + l_2 & \text{if } s_i s_j = 1 \\ -z & \text{if } s_i s_j = -1 \end{cases}$$

It is noteworthy that Newman's method may be extended to networks with more than two groups. The standard approach is the repeated division of the groups into two. Newman proposes that modularity can be expressed in terms of eigenvalues and eigenvectors of a characteristic matrix for the network, called a modularity matrix and that this expression leads to a spectral algorithm for community detection. For a more detailed review of this algorithm, refer to [19].

1.5 Thesis Overview

The preceding sections of this chapter are intended to provide the reader with a basic understanding of complex networks, biological modularity, and cellular metabolism. Also included are reviews of two ubiquitous modularity functions commonly used for maximizing modularity in complex networks. In Chapter 2, I revisit the concept of biological modularity and discuss why simulated annealing can be used to maximize this quantity in metabolic networks. I then introduce a new method for detecting group structure in complex networks. In Chapter 3, I compare the new method for detecting biological modularity with a method using simulated annealing, as implemented by Guimerà and Amaral [7]. Simulated annealing was chosen for comparison because it is a widely used method for calculating biological modularity and identifying group structure. The evaluation is done by comparing results for both simulated data and the metabolic network of the bacterium *Listeria monocytogenes*, an important pathogen in both agricultural and clinical settings. In Chapter 4, I employ regression analysis to decipher the relationship between a purely cartographic definition of modularity and the functional features of a biological network. Chapter 5 concludes this thesis and discusses plans for future work.

Chapter 2

Methods & Computing

In this chapter, I will briefly revisit the concept of biological modularity and discuss the importance of developing algorithms that give insight into the complex relationships that exist in biological networks. I then provide an overview of simulated annealing, a popular module identification method that strives to maximize the modularity of complex networks. I then introduce and provide a detailed overview of a method for uncovering modular structure and maximizing modularity in complex networks. The chapter concludes with a comparison of the two methods.

2.1 Biological Modularity Revisited

As discussed in §1.2, biological networks appear to exhibit modularity in topological structure. Studying the modular topology of complex networks can provide insight about the important relationship that exists between network structure and biological function. Therefore, it is important to develop and implement algorithms to uncover a network's modularity with accuracy[1] and computational efficiency. Many methods have been proposed in recent years to uncover modular structure in complex networks but many have failed due to high computational costs and lack of accuracy [5].

[1]By accuracy we mean a program's ability to correctly classify nodes into modules.

In the previous chapter, I introduced the method of Guimerà and Amaral [7] which identifies modules by maximizing the network's modularity through the use of simulated annealing. Simulated annealing is currently a widely used method for module identification in complex networks and will be discussed in the next section. For a more detailed review of simulated annealing, refer to Kirkpatrick et al. [13] or Johnson et al., [12].

2.2 Simulated Annealing (SA)

Simulated annealing (SA) is an approach, developed by Kirkpatrick et al. [13], used to approximate the solution to complex combinatorial optimization problems [12]. The SA algorithm is designed to find the global maximum of an objective (cost) function [13] which, in this case, is the maximization of the modularity function (1.6) described in Chapter 1.

The name and inspiration comes from the physical annealing process, which is best described in terms of the physics of crystal growth [12]. The technique for growing a crystal involves melting and controlled cooling of the raw materials used to make the crystal. The high temperatures enable the atoms to become unstuck from their initial positions allowing them to drift randomly through states of high internal energy. The gradual cooling gives the atoms opportunities to find arrangements with lower internal energy than the initial state of the system. If the temperature is lowered too quickly (this is often referred to as *rapid quenching*), the resulting crystal will have many defects and the trapped energy level will be much higher than in a perfectly structured crystal [12].

By algorithmic analogy to this physical process, the *rapid quenching* is analogous to the physical system getting stuck in local optima. The states of the physical system relate to the solutions of the optimization problem, and the internal energy of the state relates to the cost of the solution (i.e., the cost function). Lastly, when the physical system is in its ground state (i.e., in a state of minimum energy), this corresponds to the optimal (global) solution [12].

2.2.1 The Basic Iteration

The basic iteration of the SA algorithm is that at each step, it considers some neighbour *s'* of the current state *s*, and probabilistically decides whether the configuration of the current state is retained or not [13]. The probabilities are chosen such that the system ultimately moves to lower states of energy (i.e., it makes downhill moves). However, to avoid entrapment in poor local optima, the occasional uphill move (i.e., moving to a state of higher internal energy) is allowed [12].

2.2.2 Acceptance Probabilities

The probability of transitioning from the current state *s*, to the neighbouring state, *s'*, is a function $P(\triangle E, T)$ of the change in the energy ($\triangle E$) of the system (i.e., the change in energy between *s* and *s'*) and the temperature (T). The random part of the SA algorithm is implemented by randomly generating numbers from the uniform [0,1] distribution. One such number is selected and compared with $P(\triangle E, T)$. If it is less than $P(\triangle E, T)$, the new state, *s'* is accepted; if not the original state, *s* is used to initiate the next step [13]. This means that the system may move to a new state even if it costs more energy than the current state. It is this feature that prevents the method from becoming stuck in local minima.

2.2.3 Computational Temperature

Initially, temperature is set to a high value and as the simulation proceeds, the temperature is gradually reduced at each step in the annealing schedule until it reaches a final temperature of zero. As the temperature goes to zero, the transition probability tends to zero if $\triangle E > 0$ and is a positive value if $\triangle E \leq 0$. That way, for sufficiently small values of T, the system will increasingly favour lower energy values and avoid higher energy values. When *T=0*, the procedure will only make a move if it is at low cost.

For the particular simulated annealing program (modulesSA) provided to us by Roger Guimerà, when identifying modules, the cost associated with moving the system from state s to state *s'* is $C = -M$, where M is the modularity as defined in (1.6). At each temperature, T, these random updates are accepted with probability:

$$P = \begin{cases} 1 & \text{if } C_f \leq C_i \\ \exp\left(-\frac{C_f - C_i}{T}\right) & \text{if } C_f > C_i \end{cases}$$

where C_f is the cost after the update and C_i is the cost before the update. Guimerà and Amaral [7] propose at each T, $N_i = fn^2$ individual node movements are made from one module to another, where n is the number of nodes in the network. They also propose $N_c = fn$ collective movements, which entails either the merging of two modules or the splitting of a module. The number f is the iteration factor which has a recommended range of $f \in [0.1, 1]$. However, larger values of f (1 or larger) will generally give better results. After the movements are evaluated at a certain T, the system is cooled down to a new temperature *T'* $= cT$, where c is the cooling factor. This cooling factor must be strictly larger than 0 and strictly smaller than 1. In general, values closer to 1 will result in better results and longer execution times. The recommended values for cooling factor are $c \in [0.990, 0.999]$. This step is repeated until the system reaches a satisfactory state, or until the number of iterations has been exhausted.

Although SA is one of the more accurate methods, it is computationally expensive and thus its use is not recommended for larger networks (in the order of 10^5) [5]. In the next chapter, I describe a new method for module identification, which I hope will yield comparable results in a computationally efficient manner.

2.3 A Schematic Searching (SS) Algorithm

2.3.1 The Inspiration Behind the SS Algorithm

The inspiration for the SS algorithm comes from a data mining paper entitled Fully Automatic Cross-Associations [4]. This method takes a binary matrix, which represents associations between row and column objects, and it simultaneously groups the row and column objects so that similar objects are grouped together into rectangular regions called *cross-associations*. These cross-associations reveal the fundamental structure or patterns associated with the binary matrix. Essentially, this approach takes as input a binary matrix and automatically determines a good number of row groups, k, and column groups, l, and rearranges the rows and columns to uncover the hidden structure of the matrix. A method of this sort is ideal for uncovering structure in complex networks and thus the SS algorithm is strongly motivated by the methods used in [4]. The SS algorithm is described in detail in the sections that follow.

2.3.2 Methodology

For a metabolic network consisting of n metabolites, let $A = [a_{i,j}]$ be a $n \times n$ binary symmetric matrix that contains information about the associations between all metabolites in the network, where

$$a_{ij} = \begin{cases} 1 & \text{if } \exists \text{ a link between i and j} \\ 0 & \text{if } \nexists \text{ a link between i and j} \end{cases}$$

It is noteworthy that $a_{i,j} = 0$ when $i = j$. That is, all diagonal entries of A are zeros as a node cannot be linked to itself. Now, let us index the rows and columns of A as $1, 2, \ldots, n$. To gain further information about the modular structure, let us rearrange the underlying matrix such that all nodes corresponding to group one are listed first, followed by nodes in group two, and so on. Since the matrix is symmetric, the new row and column index of A will be identical. Such a rearrangement subdivides A into smaller sub-matrices called modules that, in an ideal situation, contain more within module links and few between module links.

In real complex networks, the true number of modules, m, is unknown. However, in general the number of modules that exist in a network are substantially fewer than the total number of nodes, n, that it possesses. This algorithm requires that one specifies the largest possible number of modules *a priori*. Since we are unsure of the true number of modules, we start off with $m = n$. In other words, a *starting index*[2] is created in which each node is assigned its own module. We find that after the first iteration, the number of modules decreases substantially thus allowing for the algorithm to progress at a faster rate.

[2]The starting index for an $n \times n$ network matrix is of length n and states which module each node belongs to.

2.3.3 The Basic Iteration

To calculate the starting modularity score for a given network, the modularity function (refer to Appendix A.1) takes as input the network matrix, the starting index for the network - which is defined by specifying the largest possible number of modules, and the total number of network links; it calculates its current modularity score and records it. I use the modularity definition from Guimerà and Amaral [7] to calculate the modularity of networks. Since each node is within its own module, one should expect to see a modularity score hovering around zero. The implementation of the basic iteration algorithm is comprised of a modularity function and two other important functions which will be described in detail below.

The first function takes the starting index for the network and changes the group membership of the i^{th} node (for $i \in 1 : n$) to the j^{th} group (for $j \in 1 : m$). Each time the group membership of node i changes, a new modularity score is calculated for the 'new' index and is recorded into a vector of length m. When a node has exhausted all possible groupings, that node will change its group membership to the group that yields the highest modularity score. The program then moves on to the next node and this entire process is repeated again. An iteration of this function is complete once every node in the index has changed its group membership to the group that yields the highest modularity score (including the possibility of keeping its original group membership). This process of individual node movements continues until either the index or the modularity score remains unchanged during an entire iteration.

The second function in the algorithm takes the converged index resulting from the first function and collectively moves a complete set of nodes from one group into another group. There are $\binom{m}{2}$ possible group merges in total. Each time a set of nodes in group j merges with nodes in group k, for $j \in 1 : (m-1)$ and $k \in (j+1) : m$, the modularity score for the changed index is recorded into a $\binom{m}{2} + 1$ column vector, along with its corresponding index recorded in an $n \times (\binom{m}{2} + 1)$ matrix. It should be noted that a one is added to hold the modularity score of the input network matrix and the starting index, respectively. Once all possible group merges have been made, the iteration is complete and the index that yields the best modularity score is output and then taken as input to the first function. This process of iterating between the two functions described above continues until either the index or modularity score remains unchanged. The end result of the program is the converged modularity score and the converged index which places each node in the network into the best possible module.

This is a hill-climbing algorithm. A change in the group membership of a node will only be accepted if its corresponding modularity score is larger than the modularity score preceding the change. It is this feature that allows us to find a good partition of the network into modules. There are drawbacks, however, to hill climbing algorithms. Hill-climbing algorithms can be susceptible to finding sub-optimal peaks, meaning that the algorithm may only be finding the local best solution instead of the global best solution. Further discussion of hill-climbing algorithms will take place in §2.4.

2.3.4 Pseudo-code for the SS Algorithm

The following pseudo-code implements the SS algorithm as described above starting with a given network and its starting index. The main function in the program is called ‘mod.opt’. It consists of functions for individual node movements (inm) and for collective node movements (cnm). The function ‘modularity’ was created to calculate the modularity score for a network, and is called within the ‘inm’ and ‘cnm’ functions. Refer to Appendices A.1-A.4 for the R scripts for all of the aforementioned functions.

```
 1: Read the network matrix and the initial assignment of nodes to groups
 2: for each of $n$ nodes in the network do
 3:     for each of $k$ groups do
 4:         Change group membership of node $i$ to group $k$
 5:         Compute modularity score
 6:         Save modularity score
 7:     end for
 8:     Change group membership of node $i$ to group that yields highest modularity
        score
 9: end for
10: for each group of nodes do
11:     for each remaining group of nodes do
12:         Merge groups
13:         Compute modularity score for each merge
14:         Save modularity score and node assignments to groups
15:     end for
```

```
16:     Merge the pair of groups that yields the largest increase in modularity score
17:     Save modularity score and node assignments to groups
18: end for
19: repeat
20:     2 through 18
21: until either (i) modularity score remains unchanged, or (ii) the assignment of
    nodes to groups does not change
```

2.4 The SS Algorithm Versus SA Algorithm

The SA and the SS algorithms are both iterative improvement algorithms. According to Russell and Norvig [23], SA falls in a class all of its own, whereas the SS algorithm falls under the class of hill-climbing algorithms (specifically, steepest-ascent version). The differences between SA and the SS algorithm are as follows.

SA *randomly* moves a node from one group to another and allows the search to accept lower modularity scores in order to avoid entrapment at a sub-optimal score. If the move actually improves the modularity score, it is always executed. Otherwise, the algorithm makes the move with some probability less than 1 (refer to section 2.2.3). When the temperature is high, “bad” moves are more likely to be allowed. As the temperature tends to zero, they become more and more unlikely, until the algorithm behaves more or less like a hill-climbing algorithm. The benefit to using the SA algorithm is that if given a long enough cooling schedule, SA will find a global optimal solution [23]. The downfalls of SA are (i) to apply the algorithm to a real dataset, the user is faced with the problem of having to choose the values of the parameters, such as the iteration factor, cooling factor, final temperature and number of iterations, and (ii) depending on the annealing schedule (i.e., the rate at which the temperature is lowered), execution times will be much longer [23] [12].

The SS algorithm does not explore the effect of a random move. Nodes are systematically moved (in either singletons or collectively) into each possible group and a change in the group membership of a node (or group of nodes) will only occur if it yields a higher modularity score. In other words, we don't waste time making random jumps to groups. Instead, we cover all possible jumps for each node or each group of nodes. The benefit to using a hill-climbing algorithm is that it often converges to a solution much faster as it is usually quite easy to improve a bad state [23]. The drawback is the risk of getting stuck at a sub-optimal solution. One solution to this problem is to restart the search from a different starting point [23]. In this case, this means restarting with a new set of starting indices. By doing so, a reasonably good solution can often be found after a small number of restarts. It is noteworthy that the SS algorithm does not require a lot of parameters to execute the program. The only choice the user is faced with is the choice of a starting set of indices.

In the next chapter, I discuss the results from the SS algorithm and SA under both simulated networks and the real world network of *Listeria monocytogenes*. I then compare the efficiency and accuracy of both approaches in these examples.

Chapter 3

Application of the SS Algorithm

The first section of this chapter discusses the criteria used to construct the simulated networks, followed by a discussion of the results for these networks using both the SS and SA algorithms. The second section of this chapter discusses the results of both algorithms when applied to the real world network of the *L. monocytogenes* bacterium. Cartographic representations of all networks discussed in §3.1.1 and §3.2.1 are available in Appendix C. The network visualization software, Pajek, was used to draw all networks.

3.1 Simulated Data

To test the performance of the SS algorithm on different types of networks, I simulated eleven networks under each of three classes (denoted "dense", "fuzzy", and "sparse") of binary symmetric networks. The networks under each class contain the same number of nodes, but one network has known modular structure and the remaining ten have random structure with about the same connectivity. The random networks are generated as follows: each link is included in the network with fixed probability p and with the presence or absence of any two links in the network being independent. The fixed probability p, is chosen such that the connectivity of the random network is about the same as that of the structured network. The rationale for generating ten randomizations under each class is to ensure that if any differences exist in the modularity between the structured network and its random counterparts, they aren't just occurring by chance. The first class of networks has very "dense" within-module connections. The network with known structure contains a high percentage (85-90%) of within-module connections and a very low percentage (1%) of between-module connections. The second class of networks is called a "fuzzy" network because the division of the network into distinct modules is not as clear. For the network with known modular structure, there is a high percentage (85-90%) of within-module connections and a moderate percentage (25%) of between-module connections. The third class of network is "sparse", having within-module connection rates of about 8% and between-module connection rates of about 0.2% for the network of known structure. Note that the percentage of within- and between-module connections should not add up to 100%. The percentage of within-module links is defined as the fraction of links within the module over of all possible within-module links that can exist (i.e., $\binom{n}{2}$); the percentage of between-module links is similarly defined. It is also noteworthy that sparsely connected networks usually contain many singleton nodes. The modulesSA program only takes into consideration the links between nodes, subsequently classifying singleton nodes into their own module. Hence, for a fair comparison, I developed an R script (refer to Appendix A.5) to remove all zero rows and columns (i.e., singleton nodes) from our sparse network matrices. This way each singleton node can be later counted as a module for both methods. The simulated networks were initially constructed in matrix format suitable for the SS algorithm and I developed an R script (refer to Appendices A.6 and A.7) for converting data between the format of the SS algorithm and the format suitable for the modulesSA program provided by Guimera and Amaral.

To analyze the networks using the SS algorithm, the program requires the user to enter the network matrix and a starting index for the network in the modularity optimization function. Recall that the starting index of a network states the initial assignment of nodes to groups. As mentioned in §2.4, because the SS algorithm is a hill-climbing algorithm, it is wise to restart the algorithm with different starting indexes to avoid finishing with a sub-optimal partition of the network. This can be done through the use of batch jobs, with each job starting the search with a different starting index. The effect of restarting the search with a new starting index could lead to the same partition in differing amounts of time or it could lead to a different partition of the network altogether. Since the true number of groups in real complex networks is unknown, when confronted with analyzing a network for the first time, it is recommended that the starting index is set such that the number of groups in the network is equivalent to the number of nodes in the network (i.e., m=n). Any subsequent starting indexes may be generated by taking a random sample of groups ranging from m=*2* up to m=n.

The simulated networks are also analyzed using the modulesSA program. The program prompts the user for the following parameters: (i) A random generator seed which must be a positive integer. Since the algorithm is stochastic, different runs will generally give slightly different partitions of the network. Two runs with the same seed should give the same results. (ii) The network file. (iii) The iteration factor, *f*, which at each temperature of the simulated annealing, the program performs fn^2 individual node updates and fn collective updates. Large values of f (1 or larger) will generally give better results, albeit longer execution times. I chose an iteration factor of 3 as this was recommended by the authors of the program. (iv) The cooling factor, c, which is the rate at which the system is cooled, must be strictly larger than 0 and strictly smaller than 1. In general, values close to 1 will result in better partitions of the network and longer execution times. I chose a cooling factor of $c = 0.999$ for all networks. (v) The number of randomizations gives the option to calculate the value of the modularity in a random network with the same connectivity as the original network. Calculation of the modularity for a random network will take approximately the same amount of time as for the original network. I did not run any randomizations therefore this parameter was always set to zero.

3.1.1 Results for Simulated Data

Table 3.1 shows selected results (i.e., the highest modularity scores achieved) from the simulated data for the SS and SA algorithms. For the structured network with "dense" connections within-modules and sparse connections between-modules, both algorithms converged to the same result. For the SS algorithm, initiating each search with a different starting index did not result in different partitions of the network, although, it did affect the run-time of the algorithm in that decreasing the starting number of modules resulted in shorter run-times. Even when the search was initiated with each node as a module of its own, the SS algorithm ran substantially faster than the SA algorithm. For a random network of the same size and connectivity as its densely connected and structured counterpart, both algorithms converged to very low modularity scores, which is to be expected for a network with no inherent structure. Such results indicate that the modularity of the structured network is significant, although to be certain, I calculate summary statistics (i.e., minimum, maximum, mean, and standard deviation) and ranges (i.e., mean $\pm$ 1σ, 2σ, or 3σ) for the ten random networks in this class to see how much the maximum modularity of the structured network deviates from the mean modularity of the random networks. I also plot a histogram of the modularity scores of the ten random networks and superimpose the maximum modularity score of the structured network on this plot (refer to plot A in Appendix B). I then check whether the maximum modularity score achieved for the structured network (refer to Table 3.1) falls outside of the computed ranges. Since the maximum modularity score for the structured network falls far more than three standard deviations above the mean modularity score for the ten random networks, this indicates that the modularity of the structured network is significant. For visualizations of the structured and a random densely connected networks (analyzed by the SS algorithm) refer to Appendix C.1.

For the "fuzzy" structured-network with dense connections within-modules and moderate connections between-modules, the modularity scores resulting from both algorithms are quite low, indicating that extracting discernible modular structure in such networks is difficult. It is noteworthy that for the SS algorithm, differences in the starting index did result in different partitions of the network and also in different run-times. Table 3.1 shows the highest modularity score attained by each method. SA did yield slightly better results, but the run-time was substantially slower. For a random network of same size and connectivity as its "fuzzy" connected and structured counterpart, both algorithms converged to lower modularity scores than those of the structured network. It is not as clear here whether the modularity scores for the random and structured networks are significantly different, since the modularity scores for both network types are quite low. I want to see how much the maximum modularity of the structured network deviates from the mean modularity of the random networks. The test is carried out in the same manner as mentioned above and I find that the maximum modularity for the structured network (refer to Table 3.1) falls more than three standard deviations above the mean modularity for the ten random networks (refer to Table 3.2 and plot B in Appendix B). This gives strong evidence that the modularity of the structured network is significant. For visualizations of the structured and a random densely connected networks (analyzed by the SS algorithm) refer to Appendix C.2.

For the structured network with "sparse" connections within- and between-modules, both algorithms converged to very similar results. The modularity score from the SA algorithm was marginally higher than that of the SS algorithm and the run-time for the SS algorithm was considerably longer. In theory, since the SS algorithm is a hill-climbing algorithm, it should converge to a solution much faster than the SA algorithm, yet in this case the SS algorithm failed to do so. This suggests that the SS algorithm is not as efficient at analyzing "sparse" networks as it is at analyzing networks with "dense" or "fuzzy" connections between nodes. The longer run-time could be due to inefficiencies in the program's code. Nonetheless, the SS algorithm is promising in that it gives comparable results to SA. In fact, more than half of the modules identified by SS are identical to modules identified by SA and many of the remaining modules have a high percentage of nodes in common. Again, it is noteworthy that for the SS algorithm, differences in the starting index did result in different partitions of the network and also in different run times. Table 3.1 shows the highest modularity score attained by each method. For a random network of same size and similar connectivity as its sparsely connected and structured counterpart, I found that the modularity scores attained from both methods were very similar to the modularity scores for the structured version. Since the modularity scores are very similar, I want to see how much the maximum modularity of the structured network deviates from the mean modularity of the random networks. The test is carried out in the same manner as mentioned in the two previous paragraphs and I find that the maximum modularity for the structured network (refer to Table 3.1) barely falls three standard deviations outside of the mean modularity for the ten random networks (refer to Table 3.2 and plot C in Appendix B). This gives only marginal evidence that the modularity of the structured network is significant. In a situation such as this, where the difference in modularity between the structured and random networks is not so clear, this could imply that either the current definition is not appropriate for calculating modularity in sparsely connected networks, or that the randomizations for the sparse networks are indeed highly structured. Guimerà and Amaral [9] demonstrate both numerically and analytically that, due to fluctuations in the establishment of links, random graphs have high modularity. For visualizations of the structured and a random densely connected networks (analyzed by the SS algorithm) refer to Appendix C.2.

In the next section, I test the performance of the SS algorithm and the SA algorithm on the metabolic network for *L. monocytogenes*.

Table 3.1: Selected Results from SS and SA Algorithms for Simulated Data

Network	Class	"True" Mod Score	Program Used	Starting # Groups	Converged Mod Score	Estimated # Groups	Run Time
nodes: 120							
groups: 3	"dense"	0.641328	SS	120	0.641328	3	25.69 sec
within: 85-90%			SA	n/a	0.641328	3	29.37 min
between: 1%							
total links: 2102							
nodes: 120							
random structure	"dense"	unknown	SS	120	0.1185924	6	35.15 sec
connection rate: 30%			SA	n/a	0.127432	5	41.57 min
total links: 2102							
nodes: 200							
groups: 25	"fuzzy"	0.07740604	SS	25	0.102588	7	61.69 sec
within: 85-90%			SA	n/a	0.108549	5	4.02 hrs
between: 25%							
total links: 5514							
nodes: 200							
random structure	"fuzzy"	unknown	SS	200	0.09342316	7	3.54 min
connection rate: 27.5%			SA	n/a	0.103105	5	2.68 hrs
total links: 5514							
nodes: 600							
groups: 90	"sparse"	0.2634662	SS	400	0.8308528	34	14.36 hrs
within: 8%			SA	n/a	0.838178	33	4.54 hrs
between: 0.23%							
total links: 565							
nodes: 600							
random structure	"sparse"	unknown	SS	517	0.8216082	41	16.83 hrs
connection rate: 0.30%			SA	n/a	0.831251	39	3.76 hrs
total links: 567							

Table 3.2: Summary Statistics for Modularity Scores of Random Networks

Summary Statistics	“Dense” Random	“Fuzzy” Random	“Sparse” Random
Min	0.1104	0.09056	0.8014
Max	0.1186	0.09618	0.8216
Mean	0.1148	0.09311	0.8121
SD	0.002479	0.001745	0.007178
Mean $\pm$ SD	(0.1123, 0.1172)	(0.09136,0.9485)	(0.8049, 0.8193)
Mean $\pm$ 2*SD	(0.1098, 0.1197)	(0.08962, 0.09660)	(0.7977, 0.8264)
Mean $\pm$ 3*SD	(0.1073, 0.1222)	(0.08787, 0.09834)	(0.7906, 0.8336)

3.2 Real Genomic Data

To test the performance the SS algorithm on a real genomic network, I chose the bacterial network of *L. monocytogenes* which is a bacterium commonly found in water and soil and is the cause of listeriosis in humans - a serious infection that is fatal if left untreated. The structural model of the *Listeria* metabolic network was derived from the substrate, product, and enzyme code information within the Ma and Zeng [15] database [6]. This network was found to contain 657 metabolite nodes and 697 genes (which encode enzymes that are represented by links), making it a sparse network. Note that not all biochemical reactions between metabolite nodes in the *Listeria* network are reversible, unlike our simulated networks. However, in this application I will treat the directed links as undirected under both methods of analysis (SS and SA) because the modularity formula given in (1.6), which is employed in both the SS and SA algorithms, treats networks as symmetric. The *L. monocytogenes* network will be discussed in more detail in Chapter 4.

3.2.1 Results for Real Genomic Data

Table 3.3 shows the best partition achieved by each method for the *L. monocytogenes* network. As can be seen from the table below, the SS and SA algorithms partitioned the *Listeria* network into 95 and 92 modules, respectively. The SS algorithm resulted in a slightly smaller modularity score as compared to the modularity score resulting from the SA algorithm and the run-time for the SS algorithm was substantially longer. The latter was not surprising; I deduced from the results in §3.1.1 that the SS algorithm does not perform as efficiently when analyzing sparsely connected networks. Since the modularity scores resulting from the two methods are relatively close, it is interesting to know what similarities and what differences exist between the two partitions of the network. To determine the similarity of the results from the two methods, a two-way table of the modules as determined by each method was constructed. The table, unfortunately too large to be displayed in this thesis, will be described in detail here. The table is arranged such that the main diagonal entries show the number of metabolite nodes that the modules from each method have in common. The majority of the off-diagonal entries are zeros indicating that module j from the SS algorithm has no nodes in common with module k from the SA algorithm (where $j \neq k$). Ten non-zero off-diagonal entries were found; they represent nodes that were classified into one module by one method and split into two or more modules by the other method. I found that the network partitions from the two methods share 82 identical modules. The smallest modules are of size two (there are 52 of these) and the largest shared module contains 47 nodes. Recall that SA and SS partitioned the *Listeria* network into 92 and 95 structural modules, respectively, leaving ten SA modules and 13 SS modules that differ from one another. Examples of these differences are as follows: of the ten SA modules, four of them are subsets of SS modules and the remaining six SA modules have their nodes split between one or more SS modules. Of the 13 SS modules, six of them are subsets of SA modules and the remaining seven have their nodes split between one or more SA modules. It is important to note that between the two methods, all but two modules (one module from each method) have between 59-94% of their nodes in common. This, along with the results from the simulated data, indicates that the SS method gives comparable results to SA and that the SS method has merit, albeit it is less efficient at analyzing sparse complex networks. Fortunately, the efficiency of the code can be improved, although this task falls under future work. For visualizations of the *Listeria* network, as partitioned by both methods, refer to Appendix C.2.

In summary, I have tested the performance of the SS algorithm on both simulated and real world networks as well as to test its performance against that of SA in this chapter. In Chapter 2, I established that the SS algorithm is a hill-climbing algorithm and that these types of algorithms often make rapid progress toward a solution since improving a "bad" state is usually quite easy. I also established, however, that hill-climbing algorithms are susceptible to getting "stuck" in local maxima. One resolution to this problem is to restart the search using a new set of starting indices. By doing so, a reasonably good partition of the network can often be found after a small number of restarts. That being said, we expected the SS algorithm to always converge to a good solution more quickly than SA, but that the result may or may not be the global optimal solution. The tests confirmed the aforementioned expectation to be partially true; SS did find reasonably good partitions of the networks, however, SA resulted in slightly better partitions in some cases. Nonetheless, I have concluded that the SS algorithm yields comparable results to SA since the differences in modularity scores are marginal and because the network partitions are quite structurally similar. The SS algorithm proved to be more efficient than SA at analyzing "dense" and "fuzzy" networks, but not at analyzing "sparse" networks. It is interesting that the modularity scores of the structured and random networks are very similar under the sparse class of networks. In a random network one would expect that there is no obvious modularity score or structure to converge to. The fact that the modularity score for the structured network is so similar to that of the random network implies that there is not an obvious point for the sparse structured network to converge to either. This is quite

Table 3.3: Selected Results from SS and SA Algorithms for the *Listeria monocytogenes* Metabolic Network

Network	"True"	Program	Starting #	Converged	Estimated	Run Time
	Mod Score	Used	Groups	Mod Score	# Groups	
L. monocytogenes						
nodes: 657	unknown	SS	657	0.8899461	95	2.02 days
structure: unknown		SA	n/a	0.890524	92	5.72 hrs
total links: 678						

possibly the reason why the SS algorithm is not as efficient at analyzing sparse networks. I do believe, however, this problem can be rectified through the use of a more efficient program. In the next chapter, I employ statistical analyses to explore the relationship between a purely cartographic definition of modularity and the functional components of a real biological network.

Chapter 4

The Relationship Between Metabolic Topology & Genome Evolution in Species of the Pathogenic Bacterium *Listeria monocytogenes*

Thus far, this thesis has focused on the topological features of metabolic networks and the use of module determining algorithms, such as SA and the newly introduced SS method, to identify modular structure in metabolic networks. We have seen that it is reasonable, when studying metabolic networks, to think in terms of their topology as this helps us to identify sets, or modules, of interactions. Such information can provide us with insight into the function and evolutionary history of the components of the system. The question remains, however, do the structural features of metabolic networks have any biological relevance? More specifically, is there a connection between the evolutionary (selective) pressures exerted by the environment on an organism and the topology of a metabolic network? In this chapter, I will use regression analysis to explore possible associations between the metabolic structure of the *L. monocytogenes* network and the evolutionary features of the genes.

The structure of Chapter 4 is as follows: the first section of this chapter describes the *L. monocytogenes* bacterium in greater detail than previously discussed. The second section presents an overview of evolutionary genomics. The third and fourth sections introduce the data and methods, respectively, and in the final section I discuss the results.

4.1 Ecology & Pathogenicity of *L. monocytogenes*

L. monocytogenes is a food-borne pathogen that causes a highly invasive disease called listeriosis in both humans and animals. Persons most susceptible to listerial infections are pregnant women, newborns, and individuals with weakened immune systems. In humans, listeriosis is often clinically manifested as meningitis, encephalitis, late-term spontaneous abortion, and septicemia. In animals, this pathogen has been linked to invasive diseases in more than 40 species of mammals and birds. *L. monocytogenes* has the ability to survive and multiply outside of mammalian hosts for long periods of time and can withstand stressful environmental conditions that would otherwise kill many other food-borne bacterial pathogens [21].

There are many different strains of the *L. monocytogenes* bacterium, and several different strain-typing methods (more commonly referred to as sub-typing methods) may be used to identify the various strains within a bacterial species or subspecies. The two main types of methods are conventional (or phenotypic) sub-typing methods and DNA-based sub-typing methods, with the latter being the more superior of the two, as it allows for more sensitive discrimination of different strains [27]. Through the use of DNA-based sub-typing methods, it has been shown that *L.*

monocytogenes can be grouped into two major lineages, referred to as lineages I and II [27]. Night et al. [21] state that "Lineage I isolates appear to have significantly greater pathogenic potential...while lineage II strains may represent an environmentally adapted lineage", although lineage II is still capable of invasive disease. These two lineages represent ecologically divergent pathogens, meaning they "differ in their evolutionary history and population structure" [6]. It is noteworthy that *L. innocua*, a closely related species which is non-pathogenic, represents an even larger level of ecological divergence from lineages I and II of *L. monocytogenes* [6]. A comparative genomic analysis of *L. innocua* and lineages I and II of the *L. monocytogenes* bacterium (hereafter referred to as *L.m-L1*, and *L.m-L2*, respectively) is presented in the next section.

4.2 Evolutionary Genomics

Recall that metabolic networks are abstract representations of metabolism. In §3.2, we established that the metabolic network of *L. monocytogenes* is a product-reactant centred network. That is, the metabolites are represented by nodes and the genes encode enzymes that are represented by the links. Since we are interested in how ecology-specific selection pressure relates to metabolism and metabolic structure, a gene-specific measure of the strength and direction of selection pressure called the d_N/d_S ratio (ω) was computed for each member of the *Listeria* **core genome**[1]. Here, *L. innocua* and lineages I and II of *L. monocytogenes* are ecologically divergent, so the d_N/d_S ratio was independently estimated for each of these groups.

A non-synonymous substitution is a change between codons[2] that leads to a change

[1]The core genome is defined here as the set of genes found across all strains of a species.

[2]A codon is a combination of three nucleotides in a row, that can take one of four states (i.e., one for each of the four nucleotide bases A, C, G, or T occurring in DNA) at each position producing

in amino acid sequence, whereas a synonymous substitution is a change between codons that does not. The rate of non-synonymous substitution within a given gene sequence is denoted by d_N, and the rate of synonymous substitution is denoted by d_S. The ratio d_N/d_S (also denoted as ω) is employed as a measure of the strength and direction of selection pressure. It is used to infer whether natural selection is acting to promote the fixation of advantageous mutations (i.e., positive selection) or to remove deleterious mutations (i.e. purifying selection) [2]. In general:

$\omega < 1$: negative (purifying) selection
$\omega = 1$: neutral selection
$\omega > 1$: positive (diversifying) selection

That is, when synonymous substitutions occur at a faster rate than non-synonymous substitutions, the ratio of non-synonymous to synonymous substitutions (i.e., ω value) is less than one. When synonymous substitutions occur at about the same rate as non-synonymous substitutions, the ratio of non-synonymous to synonymous substitutions (i.e., ω value) is approximately one. When non-synonymous substitutions occur at a faster rate than synonymous substitutions, the ratio of non-synonymous to synonymous substitutions (i.e., ω value) is greater than one.

4.3 Materials & Methods

4.3.1 Genomic Data

Dunn et al. [6] examined one genome of *L. innocua* and four genomes of *L. monocytogenes*, having complete genome sizes ranging from 2,821-3,111 genes. These data are the subject of the analyses presented in this chapter. Only those genes that were present in all five genomes and that were greater than or equal to 100 codons (300 nucleotides) in length were considered for further analysis. In total, there were

a total of $4^3 = 64$ possible combinations.

1,905 genes that met this criterion and maximum likelihood (ML) estimates of ω were obtained for each gene. As mentioned in §3.2, the metabolic relationships for *L. monocytogenes* was extracted from the Ma & Zeng database [15], a publicly available database for metabolomic data [6]. The undirected version of the *Listeria* metabolic network, which was analyzed in Chapter 3, contains 657 metabolite nodes and 678 genes. For the purposes of the statistical analysis in this chapter, I will use the 92 structural modules that were identified by SA in Chapter 3, since this method achieved a slightly better result. Furthermore, I will only consider the 388 genes that comprise the giant strong component (GSC) as it is the most complex and core part of the network. The GSC is shown in Figure 4.1. Each enzyme in the network is also classified according to a traditional biochemical pathway as specified by the Kyoto encyclopedia of genes and genomes (KEGG)[3] database. A total of 70 traditional biochemical pathways were found to be associated with the 388 genes of the GSC.

A genome-scale phylogeny (Figure 4.2) was estimated via ML from the combined set of 1,905 genes. ML also was employed to estimate a tree topology individually for each gene. Of the 1,905 genes, 1,309 showed a topology identical to the genome-scale phylogeny.

The strength and direction of natural selection pressure (i.e., values of ω) was estimated for each member of the *Listeria* core genome using a Markov model of codon evolution. Several hypotheses (denoted H_1 to H_3) for divergent selection pressure were tested. The null hypothesis (H_0), assumed uniform selection pressure across

[3]The KEGG database is a collection of online databases for understanding and simulating higher order biological functions of the cell or the organism from its genome information [3].

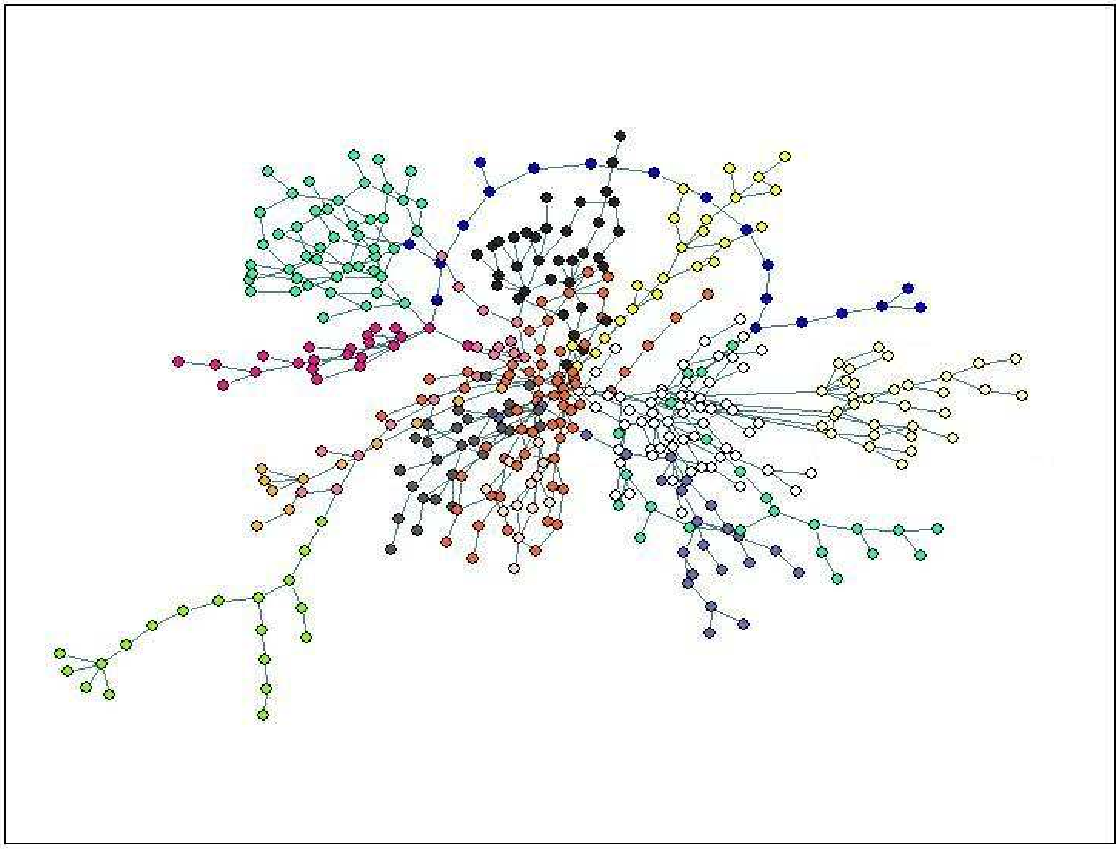


Figure 4.1: *A cartographic representation of the GSC of the L. monocytogenes metabolic network. The GSC represents the most interconnected part of the network. The nodes represent the metabolites and their colour represents module assignment. The lines represent the genes that encode the enzyme-catalyzing reactions.*

the entire evolutionary history, with one ω parameter for all branches in the genome tree. H_1 assumed divergent selection pressure between pathogens (*L.m-L1* and *L.m-L2*) and non-pathogen *L. innocua*, with each having an independent ω parameter. H_2 assumed that *L.m-L1* was subject to different selection pressure than the other lineages and H_3 assumed that *L.m-L2* was subject to different selection pressure than the other lineages. A depiction of H_1 to H_3 models of selection pressure are presented in Figure 4.2 below.

Likelihood ratio tests (LRTs) were used to determine if H_1, H_2, or H_3 models of

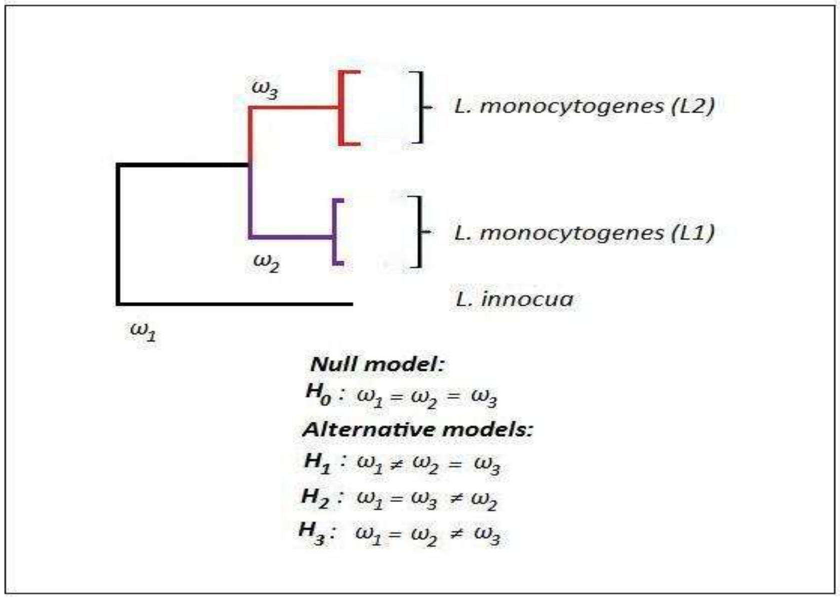

ω_3
L. monocytogenes (L2)
L. monocytogenes (L1)
ω_2
L. innocua
ω_1
Null model:
$H_0 : \omega_1 = \omega_2 = \omega_3$
Alternative models:
$H_1 : \omega_1 \neq \omega_2 = \omega_3$
$H_2 : \omega_1 = \omega_3 \neq \omega_2$
$H_3 : \omega_1 = \omega_2 \neq \omega_3$

Figure 4.2: *A genome-scale estimate of the Listeria phylogeny. The null model assumes that selection pressure is homogeneous across all branches. H_1 is based on the idea of divergent selection pressure between pathogens (L.monocytogenes: $\omega_2 = \omega_3$) and non-pathogens (L.innocua: ω_1). H_2 is based on the idea that only L.m-L2 (ω_2) was subject to divergent selection; therefore, L.m-L1 maintains the ancestral level of selection pressure, with $\omega_1 = \omega_3$. H_3 is based on the idea that only L.m-L1 (ω_3) was subject to divergent selection; therefore, L.m-L2 maintains the ancestral level of selection pressure, with $\omega_1 = \omega_2$ [6]*

selection pressure fit the data better than H_0 for each of the 1,309 genes. Through the use of the LRTs, 351 of the 1,309 genes were identified as having a signal for divergent selection pressure (i.e., different levels of selection intensity between ecologically divergent lineages of *Listeria*) and of those 351 genes, 249 were present in the GSC.

4.3.2 Is Divergent Selection Pressure Related to Metabolic Structure?

The purpose of the analysis is to decipher how ecologically divergent selection pressure relates to metabolic structure. In order to formulate this into a statistical analysis problem, we must first form the variables to label types of selection pressure, the structural modules and the traditional biochemical pathways of the metabolic network for each gene. The type of selection pressure that a gene is subject to is labelled by a discrete (nominal) variable taking on the values of 0, 1, 2, or 3. It gives information about the best hypothesis found, where a 0 implies that the null hypothesis best describes type of selection pressure acting on a particular gene and a 1, 2, or 3 implies that H_1, H_2, or H_3 hypothesis best describes type of selection pressure acting on a particular gene, respectively. The structural modules obtained through the use of SA and the traditional biochemical pathways obtained from the KEGG database are also denoted by discrete (nominal) variables; a gene is assigned a binary variable (0 or 1) depending on whether it belongs to a structural module of the metabolic network or a traditional biochemical pathway. The problem is then formulated by describing the type of divergent selection pressure (H_1, H_2, H_3) in terms of the structural modules and traditional biochemical pathways of the metabolic network. Since divergent selection pressure is a multinomial variable, logistic regression for multinomial responses is used to describe the relationship between biological function and network topology.

4.3.3 Logistic Regression for Multinomial Responses: Relating Best Hypothesis for Diversifying Selection to Network Structure

4.3.3.1 Review of logistic regression for multinomial responses

Multicategory responses use multinomial GLMs. For the purposes of this thesis we will focus on logistic regression for nominal response variables. Let Y be a categorical response with J categories. Multicategory logit models for nominal response variables simultaneously describe log odds for all $\binom{J}{2}$ pairs of categories.

4.3.3.2 Baseline-category Logits

Let $\pi_j(\mathbf{x}) = P(Y = j \mid \mathbf{x})$ for the given explanatory variables, $\mathbf{x}$, with $\sum_j \pi_j(\mathbf{x}) = 1$. The counts of the J categories of Y is assumed to follow a multinomial distribution having probabilities $\pi_1(\mathbf{x}), \ldots, \pi_J(\mathbf{x})$. The baseline-category logit models contrast each response category with a baseline-category, often the last one or the most common one. The model:

$$log\frac{\pi_j(\mathbf{x})}{\pi_J(\mathbf{x})} = \alpha_j + \boldsymbol{\beta}_j^{\boldsymbol{T}}\mathbf{x}, \qquad j = 1, \ldots, J-1 \tag{4.1}$$

simultaneously describes the effects of $\mathbf{x}$ on these J-1 logits. These J-1 equations determine parameters for logits with other pairs of response categories, since

$$log\frac{\pi_a(\mathbf{x})}{\pi_b(\mathbf{x})} = \frac{\pi_a(\mathbf{x})}{\pi_J(\mathbf{x})} - \frac{\pi_b(\mathbf{x})}{\pi_J(\mathbf{x})} \tag{4.2}$$

With categorical predictors, χ^2 or G^2 goodness-of-fit statistics provide a model check when data are not sparse. When predictors are continuous or the data are sparse, such statistics are suitable for comparing nested models differing by relatively few terms [1].

4.3.4 Results & Discussion

A multinomial regression analysis was performed using R statistical software. The dependent variable in this analysis (model of divergent selection pressure) was taken as a discrete variable (H_0, H_1, H_2, H_3) and the independent variables were the structural modules of the metabolic network and the KEGG biochemical pathways. Each gene in the GSC was assigned a binary variable (0 or 1) that indicated whether a gene belonged to one of 76 different structural modules and 70 different biochemical pathways. Data for divergent selection pressure are available only for the subset of genes in the GSC that have a gene tree topology corresponding with the genome tree

(249 genes). With the removal of those genes in the GSC that did not show signal for divergent selection pressure, singularities arose in the filtered dataset. Singularities arise when the data matrix contains variables which are not linearly independent. This indicates that one or more of the independent variables are redundant with one another and should be removed from the data set as they do not add any predictive power. I identified 20 singularities in the dataset; with the removal of these singularities and any modules and pathways containing none of the 249 genes, we are left with 46 modules and 57 pathways for a total of 103 predictor variables left for analysis.

It is not practical to fit a statistical model containing 103 independent variables. Identifying and omitting those independent variables that were not associated with the response variable was the first step in the analyses. In the case of a multinomial response variable and binary independent variables, this can be achieved through the use of a standard χ^2 text for independence; thus, in the first step I used Pearsons χ^2 tests to find those structural modules and biochemical pathways that are significantly associated with the best hypothesis for divergent selection pressure. Results from the χ^2 tests showed that the model of divergent selection pressure was significantly associated with the following seven independent variables: module 16, module 84, module 86, KEGG pathway 220, KEGG pathway 230, KEGG pathway 710 and KEGG pathway 3030.

I then fit a baseline-category logit model for nominal responses. The initial step in the multinomial regression was to define the regression model; in the case at hand, the model included the above seven independent variables plus all possible interactions among these variables. This model is over-saturated (i.e., it serves as a baseline for comparison with other model fits) because there are more regression coefficients (33) than there are distinct data patterns. I want to fit the model which provides the maximum explanatory power by using as few parameters as possible. To accomplish this I performed a parameter elimination procedure based on the deviance between the fit of a simplified model relative to the baseline (or saturated) model. The deviance is a likelihood-ratio statistic, which is two times the difference in the likelihood of the saturated and simplified models. It is typically denoted by G^2. G^2 approximately follows a χ^2 distribution with $df = N - p$ where N is the number of observations and p is the number of model parameters. Therefore, G^2 was used to identify the most parsimonious regression model while still maintaining the same explanatory power as the saturated model. I removed those variables where I observed a large p-value for G^2 ($p \geq 0.05$), because in those cases the simplified model fit the data just as well/almost as well as the saturated model.

Box 4.1: Details of the Multinomial Regression Analysis (Includes Portion of R Summary Output)

The simplified model:

$$\log \frac{\pi(\mathrm{H_i})}{\pi(\mathrm{H_0})} \sim \text{module16} + \text{module84} + \text{module86} + \text{pathway220} + \text{pathway230} + \text{pathway3030}$$

where $\pi(\mathrm{H_i})$ represents the probability that hypothesis $\mathrm{H_i}$ is the best model.

Portion of R summary output:

```
Coefficients:
     (Intercept)      x16       x84       x86     pw220      pw230     pw3030
    1   -1.5324   1.53238   -4.8414 -0.17719    2.2255 -26.11201   -0.45435
    2   -2.3102   0.70084   -2.7715  1.10078 -16.5406   -0.85470 -11.11059
    3   -2.0545 -23.34415   42.5549  0.33603 -18.0752   -1.05154    3.51132


Std. Errors:
  (Intercept)       x16          x84      x86        pw220        pw230       pw3030
1   0.23508 6.7473e-001 1.0867e-013 0.50190 8.9737e-001 2.9073e-010 5.5847e-010
2   0.32829 1.1435e+000 2.9790e-014 0.48494 1.2427e-007 1.0673e+000 8.8991e+002
3   0.29341 3.1194e-011 1.3802e-018 0.53004 2.7182e-009 1.0600e+000 1.3710e+000


Residual Deviance: 447.5932
AIC: 489.5932
```

The fitted model is :

$$\log(\pi_{H1}/\pi_{H0}) = -1.532361 + 1.5323820\text{module16} - 4.841401\text{module84} - 0.1771939\text{module86} + 2.225524\text{pathway220} - 26.1120139\text{pathway230} - 0.4543482\text{pathway3030}$$

Equations for $\log(\pi_{H2}/\pi_{H0})$ and $\log(\pi_{H3}/\pi_{H0})$ are similar to above.

Fitted probabilities for best hypothesis as determined by the best multinomial regression model. Probabilities were obtained by setting each predictor variable to a value of 1 and all other predictors to a value of 0:

Best	Module	Module	Pathway	Pathway	Module	Pathway
Hypothesis	16	86	220	230	84	3030
H0	0.4545	0.6029	0.3333	0.9199	0.0000	0.4000
H1	0.4545	0.1091	0.6667	0.0000	0.0000	0.0000
H2	0.09092	0.1799	0.0000	0.03882	0.0000	0.0000
H3	0.0000	0.1081	0.0000	0.04132	1.0000	0.6000

Predicted odds ratios:

	Module	Module	Module	Pathway	Pathway	Pathway
Odds(Y=H_a)	16	84	86	220	230	3030
Odds(π_{H1})	1.00	1.71e-3	0.181	2.00	9.87e-13	0.137
Odds(π_{H2})	0.200	6.21e-3	0.272	6.20e-9	0.0422	1.48e-6
Odds(π_{H3})	9.32e-12	3.88e17	0.179	1.81e-9	0.0448	4.29
Odds(π_{H1})/Odds(π_{H2})	5.00	0.275	0.666	3.07e8	2.34e-11	9.24e4
Odds(π_{H1})/Odds(π_{H3})	1.07e11	4.39e-21	1.00	1.10e9	2.20e-11	0.0320
Odds(π_{H2})/Odds(π_{H3})	2.15e10	1.60e-20	1.52	3.59	0.943	3.46e-7

Based on this analysis, I identify three structural modules (16, 84, and 86) and three traditional biochemical pathways (KEGG pathway numbers 220, 230, and 3030) in the *L. monocytogenes* bacterium that are significantly associated with a given model of divergent selection pressure (H_1, H_2, H_3). For example, in structural module 16 (refer to Figure 4.3), the odds of observing divergent selection pressure between pathogens (*L.m-L1* and *L.m-L2*) and non-pathogen *L. innocua* (i.e., model H_1) is five times higher than the odds of observing unique selection pressure within ecotype *L.m-L1*, while ecotype *L.m-L2* maintains the ancestral level of selection pressure (i.e., model H_2). The odds ratios for the remaining five predictor variables are interpreted in a similar fashion.

Admittedly, the regression model is a highly idealized portrayal of a reality that is quite complex. However, the results from this analysis show that a connection exists between gene-level evolution and complex metabolic phenotypes (i.e., metabolic network topology) of the *L. monocytogenes* bacterium. Although this result needs to be tested on other complex biological networks, this is a very important discovery from a biological standpoint, as these results provide evidence, albeit indirect, that natural selection is acting on these higher-level complex phenotypes.

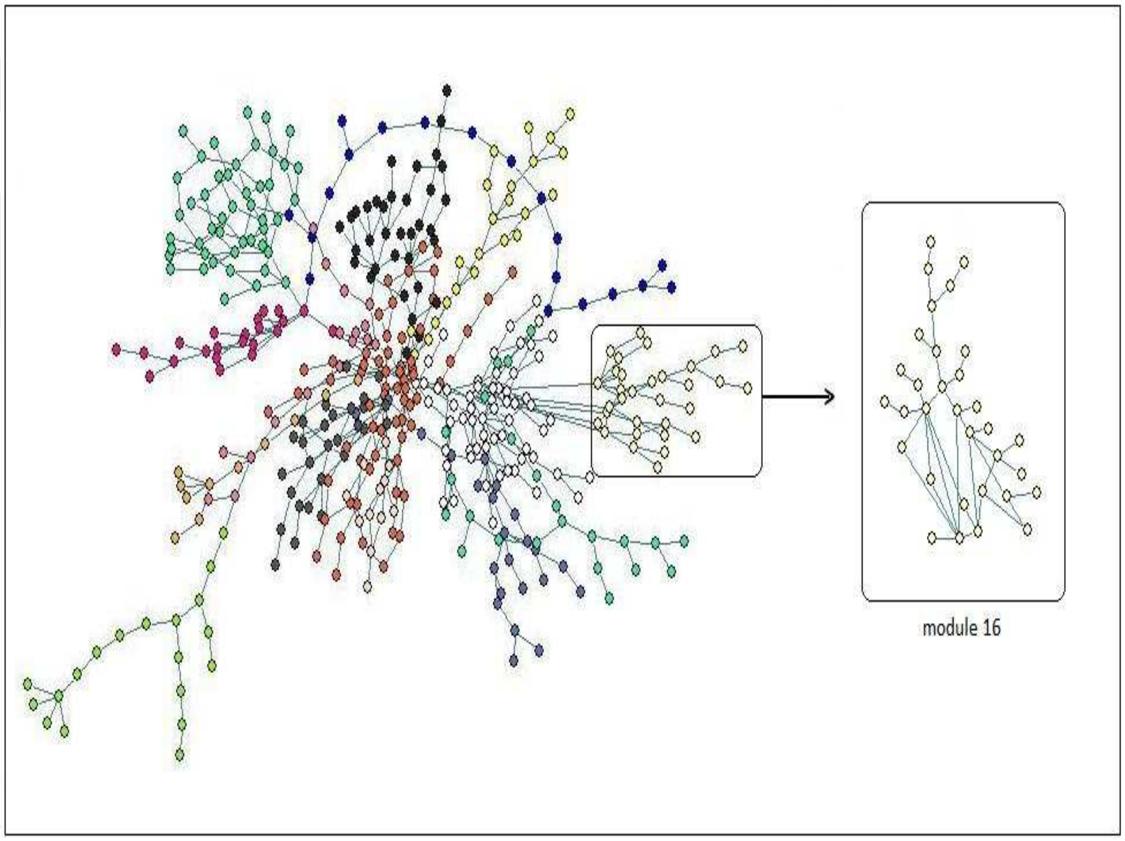


Figure 4.3: *A cartographic representation of the GSC of the L. monocytogenes metabolic network with an expansion of structural module 16. The GSC represents the most interconnected part of the network. The nodes represent metabolites and their colour represents module assignment. The links represent the genes that encode the enzyme-catalyzing reactions.*

Chapter 5

Thesis Conclusion

The problem of module detection in complex metabolic networks is challenging and has been the subject of much discussion in systems biology in recent years. The existing literature on complex networks and the use of module identification methods demonstrates that many complex networks are found to divide naturally into communities or modules. It also demonstrates that studying the topology of complex metabolic networks is potentially very useful since nodes belonging to modules can often be related by commonalities in their biological function. Therefore, it is necessary to develop algorithms that uncover a network's modularity with accuracy and computational efficiency. One of the main goals of this research was to create a new algorithm that identifies modules in complex networks and reconstructs networks in such a way so as to maximize modularity. Other goals were to evaluate the performance of the new method, and compare it to a popular method based on a simulated annealing algorithm, and to apply the module determining methods in conjunction with a statistical analysis to decipher a relationship between a purely cartographic definition of modularity and the functional and evolutionary features of a metabolic network.

The SA method is a widely used method for detecting community structure in complex networks. Dannon et al. [5] demonstrate that SA has a computational cost of order $O(m^2n)$, where m is the number of links and n is the number of nodes in the network. A method of this order is considered to be computationally expensive; however, it generally produces accurate partitions of a complex network. It was my goal to develop an algorithm that produces comparative results to SA, but in a fraction of the time. I decided that, like the SA algorithm, the SS algorithm would be an iterative improvement algorithm, but that it would be a strictly hill-climbing algorithm, as an algorithm of this sort often converges to a reasonably good solution much faster. The statistical software program, *R*, was used to program the SS algorithm. The SA algorithm was provided to us by Guimerã and Amaral. I tested the performance of the SS algorithm on simulated networks and on the genomic network for *L. monocytogenes*. Additionally, I tested its performance against that of SA. I expected that because the SS algorithm is a hill-climbing algorithm, it would converge to a good solution more quickly than SA, but that the result may or may not be the global optimal solution. I found that SS did find reasonably good partitions of the networks; however, SA resulted in slightly better partitions in some cases. Nonetheless, I concluded that the SS algorithm yields comparable results to SA since the differences in modularity scores were marginal and because the network partitions were structurally similar. The SS algorithm proved to be more efficient than SA at analyzing densely and fuzzy connected networks, but not at analyzing sparsely connected networks.

I was also interested in using statistical analysis in conjunction with the results from the module identification methods to explore possible relationships between the topology of the *L. monocytogenes* metabolic network and the evolutionary (selective) pressures exerted on the *Listeria* core genome. In Chapter 3, the structural model of the metabolic network for *L. monocytogenes* was partitioned into modules by using the SS and SA algorithms. For the purposes of the statistical analysis in Chapter 4, I used the structural modules that were identified by SA in Chapter 3, since this method achieved a slightly better result in terms of modularity score. Data for divergent selection pressure are available only for those genes in the *Listeria* core genome that have a gene tree topology congruent with the genome tree. Therefore, those genes identified as having non-genome trees were removed from the data, leaving 249 genes for regression analysis on divergent selection pressure. The dependant variable (model of divergent selection pressure) in this analysis was taken as a discrete variable (H_0, H_1, H_2 or H_3). The independent variables in the regression were the structural modules of the metabolic network and the KEGG biochemical pathways. Accordingly, I carried out a multinomial regression analysis using *R* statistical software. I found three structural modules and three biochemical pathways that were significantly enriched or deficient with respect to a given model of divergent selection pressure. This implies that a connection exists between gene-level evolution and complex metabolic phenotypes (i.e., metabolic network topology) of the *L. monocytogenes* bacterium. This result needs to be tested on other complex biological networks, however, this is a very important discovery from a biological standpoint, as these results provide evidence that natural selection is acting on these higher-level complex phenotypes.

The results indicate that it is reasonable to use statistical analysis in combination with network theory to investigate the relationship between the topology of metabolic networks and the biological and evolutionary functioning of an organism. However, current modularity definitions employed in module determining algorithms seem to only focus on the purely topological features of complex networks. This may be sufficient for investigating community structure of networks in other disciplines, but as far as systems biology is concerned, in order to gain more insight into the function and evolution of these very complex metabolic networks, a modularity definition must be defined with biological criteria in mind. The creation of such a definition could lead to major advancements in the field of systems biology.

I have made comparisons between the SS and SA algorithms but the reader should know that these algorithms are not qualitatively comparative. The SS algorithm is coded in *R* programming language whereas the SA algorithm is coded in *C* programming language. The expectation is that an algorithm coded in *R* is slower in its implementation than an algorithm coded in *C*. The fact that the SS algorithm is substantially faster at analyzing “dense” and “fuzzy” networks implies that qualitatively the SS algorithm is an improvement on the SA algorithm.

Future work on this project could look into: (i) Developing a more efficient program for the SS algorithm, as we have seen that this method has merit but it is not efficient at analyzing sparsely connected networks. (ii) Developing a modularity definition that incorporates one or more biological covariates, as studying purely topological features alone offers limited insight into the functional and evolutionary components of metabolic networks.

Appendix A

R Scripts

A.1 R Script for Calculating the Modularity of a Network

```
modularity <-

# Description:

# This function calculates the modularity for a network


# Arguments:

# x: A network matrix which is binary and symmetric and must have zeros

# along the main diagonal.

# index: Identifies the group membership of each node

# total.links: Total number of links in the network.


function(x, index, total.links){

index.unique <- unique(index)

ls <- ds <- NULL

for(i in index.unique){

ls <- c(ls, sum(x[index==i, index==i])/2)

ds <- c(ds, sum(x[index==i,]))
}

modularity <- sum((ls/total.links) - (ds/(2*total.links))^2)

return(modularity)}
```

A.2 R Script for Performing Individual Node Movements

```
inm <-
# Description:
# inm makes individual node movements from one group to another.
# Each time the group membership of a node changes, a modularity
# score is calculated and recorded. The group membership of a
# node is changed to the group that yields the highest modularity score.
# default epsilon = 0.001

# Arguments:
# x: A network matrix which is binary and symmetric and must have zeros
# along the main diagonal.
# index: Identifies the group membership of each node

function(x, index, epsilon = 0.001){
now <- proc.time()[1:2]
total.links <- sum(x)/2
n <- length(index)
M.new <- modularity(x, index, total.links)
M.old <- 0
index.old <- rep(0,n)
no.ite <- 0
mod.vec <- NULL
current.index <- index
while(abs(M.new-M.old) > epsilon | abs(sum(index-index.old)) != 0){

index.old <- index # assign index.old the most current index

M.old <- M.new # assign M.old the most current M.new


for(j in 1:n){

index.unique <- unique(index)

no.grps <- length(index.unique)

mod.vec <- NULL # 'clean' mod.vec

current.index <- index


for(i in index.unique){

current.index[j] <- i

M.new <- modularity(x, current.index, total.links)

mod.vec <- c(mod.vec, M.new)

}

index[j]<- index.unique[(1:no.grps)][mod.vec==max(mod.vec)][1]

M.new <- max(mod.vec)

}

no.ite <- no.ite + 1

}

speed <- proc.time()[1:2]

run.time.in.sec <- speed[1]

retval <- list(M.new = M.new, index = index, no.ite = no.ite,

run.time.in.sec = run.time.in.sec)

return(retval)}
```

A.3 R Script for Performing Collective Node Movements

```
cnm <-

# Description:

# cnm is used to move a collection of nodes from one module into

# another module. Each time two modules merge, the modularity score is

# calculated and recorded. The grouping that yields the highest modularity

# score is kept.


# Arguments:

# x: A network matrix which is binary and symmetric and must have zeros

# along the main diagonal.

# index: Identifies the group membership of each node


function(x, index){

total.links <- sum(x)/2

index.unique <- unique(index)

no.grps <- length(index.unique)

n <- length(index)

new.indexes.mat <- matrix(0, n, choose(no.grps, 2) + 1)

new.index <- index

new.mods.mat <- matrix(0, choose(no.grps, 2) + 1, 1)

new.mods.mat[1,] <- modularity(x, index, total.links)

new.indexes.mat[,1] <- index

counter <- 2
for(j in 1:(no.grps - 1)){

for(k in (j + 1):no.grps){

index <- new.index

index[index == index.unique[j]] <- index.unique[k]

M.new <- modularity(x, index, total.links)

new.mods.mat[counter, ] <- M.new

new.indexes.mat[ ,counter] <- index

counter <- counter + 1

}

M.new <- max(new.mods.mat)

best.col.index <-

c(1:(choose(no.grps, 2) + 1))[new.mods.mat==max(new.mods.mat)][1]

index <- new.indexes.mat[, best.col.index]

}

retval <- list(M.new = M.new, index = index, best.col.index = best.col.index)

return(retval)}
```

A.4 R Script for SS Algorithm

```
mod.opt <-

#Description:

# mod.opt re-arranges a network such that it optimizes the networks

# modularity.

# default epsilon: 0.001


# Arguments:

# x: A network matrix which is binary and symmetric and must have zeros

# along the main diagonal.

# index: Identifies the group membership of each node


function(x, index, epsilon=0.001){

now <- proc.time()[1:2]

total.links <- sum(x)/2

init.results.inm <- inm(x, index)

index.old <- index.current <- init.results.inm$index

M.old <- init.results.inm$M.new

no.ite <- 0

best.col.index <- 2 #initialize


while(best.col.index > 1){

index.old <- index.current

init.results.cnm <- cnm(x, index.old)

index.current <- init.results.cnm$index
M.new <- init.results.cnm$M.new
best.col.index <- init.results.cnm$best.col.index
no.ite <- no.ite + 1
}
index <- index.current

while(M.new-M.old > epsilon | abs(sum(index-index.old)) != 0){
results.inm <- inm(x, index)
index.old <- results.inm$index
M.old <- results.inm$M.new

results.cnm <- cnm(x, index.old)
index <- results.cnm$index
M.new <- results.cnm$M.new
}

group.dim <- as.vector(table(index))
no.grps <- length(group.dim)
speed <- proc.time()[1:2] - now
run.time.in.sec <- speed[1]
retval <- list(M.new = M.new, index = index, no.grps = no.grps,
group.dim = group.dim, no.ite = no.ite, run.time.in.sec = run.time.in.sec)
return(retval)}
```

A.5 R Script for Removing Singleton Nodes from Sparse Networks

```
# This script removes all zero rows/columns from an nxn symmetric matrix

# leaving only rows/columns that contain links.

rowcol.rem <- NULL

for(i in 1:(dim(x)[1])){

if(sum(x[i,])==0) rowcol.rem <- c(rowcol.rem, -i)

}

x.new <- x[rowcol.rem, rowcol.rem]
```

A.6 R Script for Converting Networks from SS Format to SA Format

```
link2nodes.vec <- NULL

for(i in 1:(dim(x)[1]-1)){

for(j in (i + 1):dim(x)[1]){

if(x[i,j]==1) link2nodes.vec <-c(link2nodes.vec, i)

}

}

link2nodes.vec1 <- NULL

for(i in 1:(dim(x)[1]-1)){

for(j in (i + 1):dim(x)[1]){

if(x[i,j]==1) link2nodes.vec1 <-c(link2nodes.vec1, j)

}

}

SA.prod.react <- matrix(c(link2nodes.vec, link2nodes.vec1), ncol=2)

write.table(SA.prod.react, "filename.txt", row.names=F, col.names=F, sep="\t")
```

A.7 R Script for Converting Networks from SA Format to SS Format

```
prod.react.list <- read.table("filename.txt", sep="\t" )

SS.network <- matrix(0, n, n) # n is the number of nodes in the network

a <- prod.react.list[,1]

b <- prod.react.list[,2]

for(i in 1:length(a)){

SS.network[a[i],b[i]]<-1

}

SS.sym.network <- SS.network + t(SS.network) # ensures symmetry

sum(SS.sym.network) - sum(t(SS.sym.network))

SS.sym.network[SS.sym.network==2]<-1

diag(SS.sym.network) <- c(rep(0, n))

write(SS.sym.network, "filename.txt", ncol=n, sep="\t")
```

Appendix B

Histograms of Modularity Scores for Random-Networks

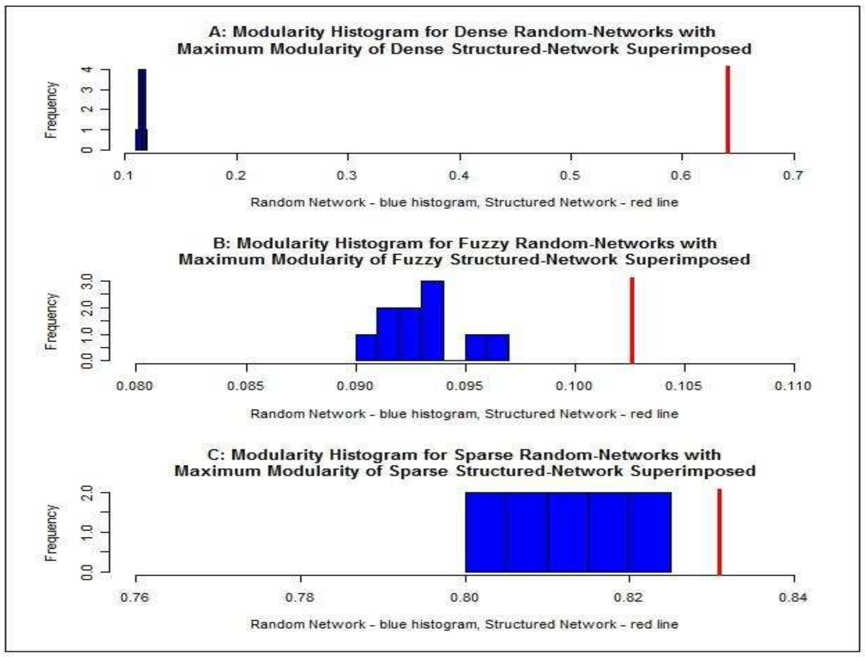


Figure B.1: *The blue histogram in each plot (**A**, **B**, and **C**) represents the modularity scores of the ten "dense", "fuzzy", and "structured" networks, respectively. The red line in each plot represents the maximum modularity score of the corresponding structured network. **A**: Since the red line is far away from the histogram, this indicates that the modularity of the structured network is significant. **B**: Since the red line is a reasonable distance away from the histogram, this indicates that the modularity of the structured network is significant. **C**: Since the red line is relatively close to the histogram, this implies that the modularity of the structured network is only marginally significant.*

Appendix C

Modular Models of the Simulated & *Listeria Monocytogenes* Networks

C.1 Simulated Networks

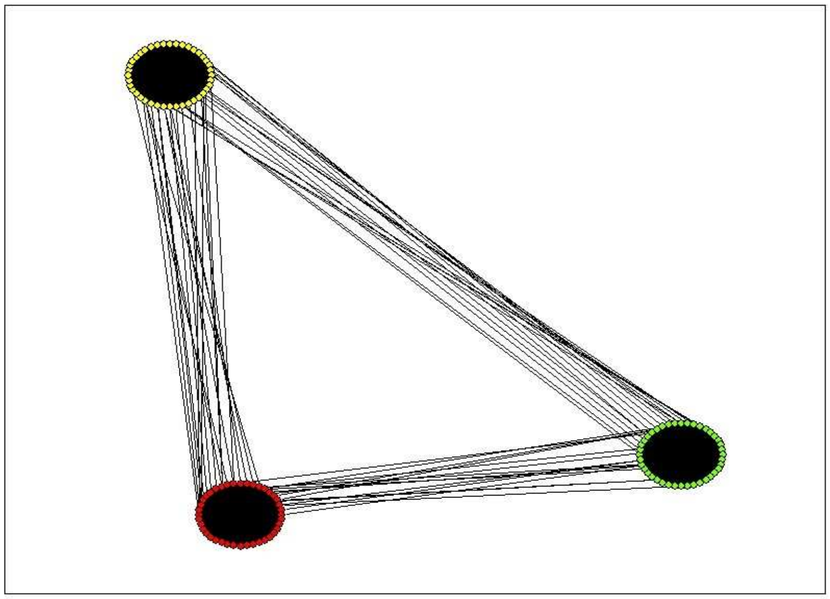


Figure C.1: *A modular model of a "dense" structured-network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into three modules and the modularity score for this network is $M = 0.641328$.*

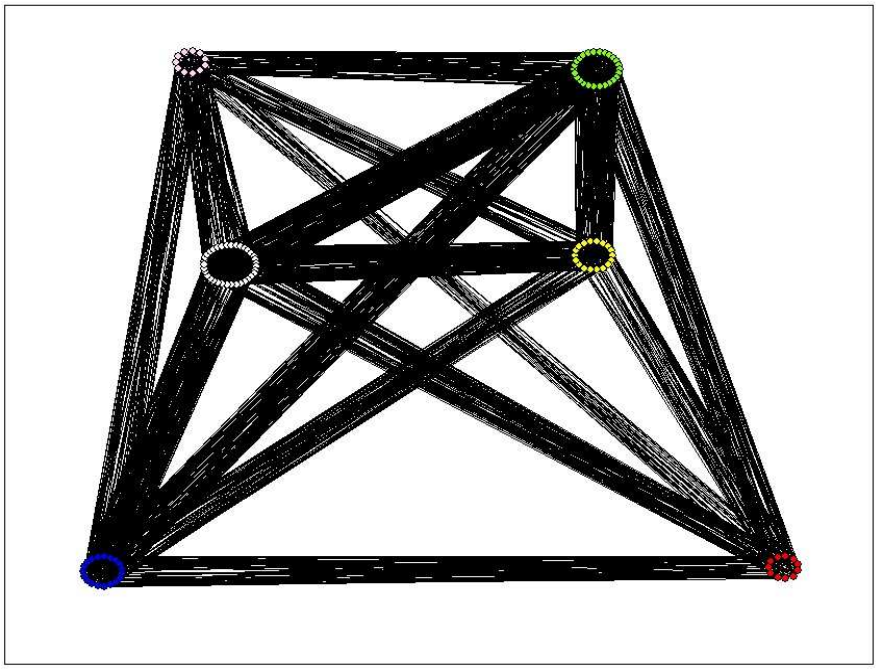


Figure C.2: *A modular model of a "dense" random-network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into six modules and the modularity score for this network is M = 0.1185924.*

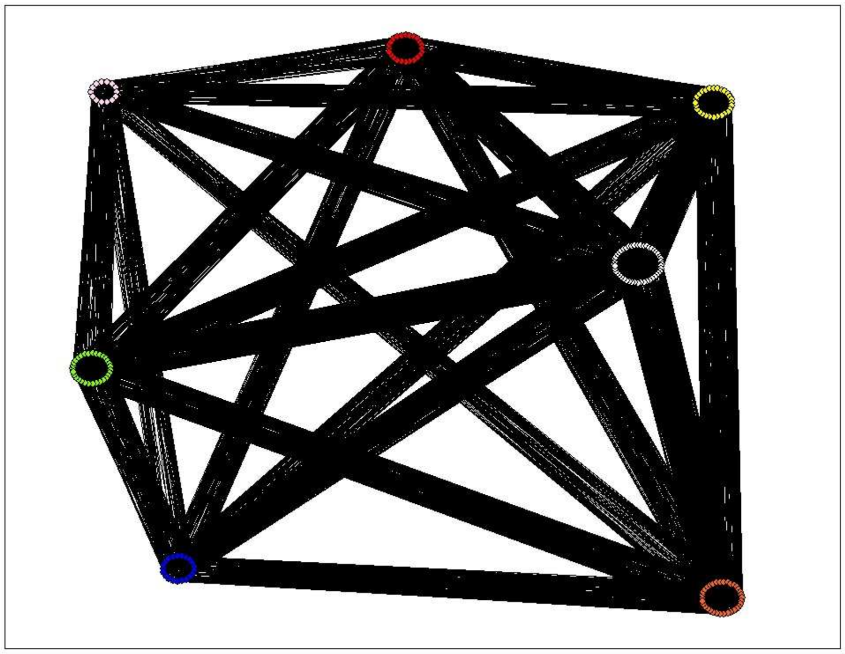


Figure C.3: *A modular model of a "fuzzy" structured-network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into seven modules and the modularity score for this network is $M = 0.102588$.*

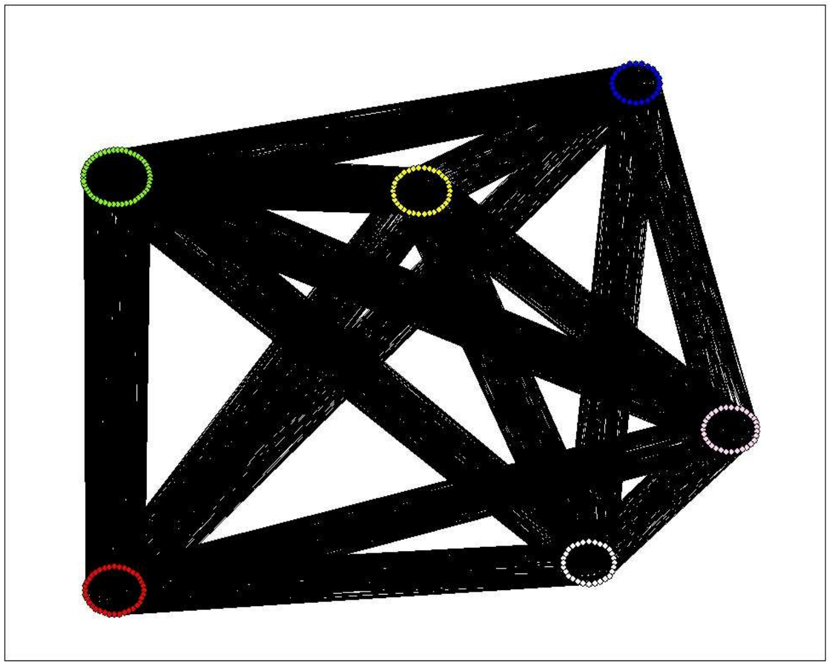


Figure C.4: *A modular model of a "fuzzy" random-network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into six modules and the modularity score for this network is M = 0.09617799.*

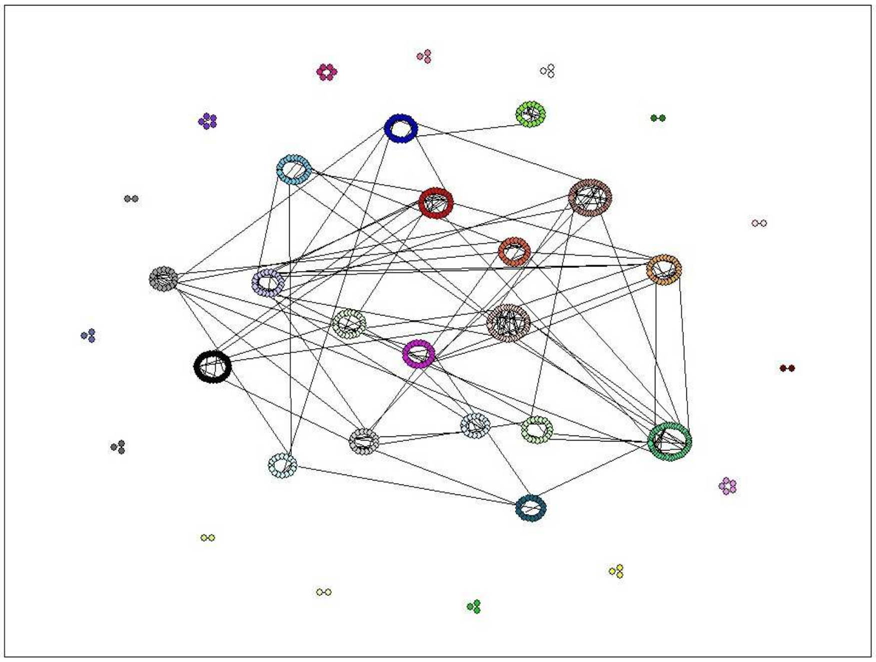


Figure C.5: *A modular model of a "sparse" structured-network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into 34 modules and the modularity score for this network is M = 0.8308528. This network is not fully connected; it is comprised of a giant strong component (GSC), which is the most interconnected part (i.e., the core) of the network, and 15 smaller modules around the outside that are isolated from the GSC.*

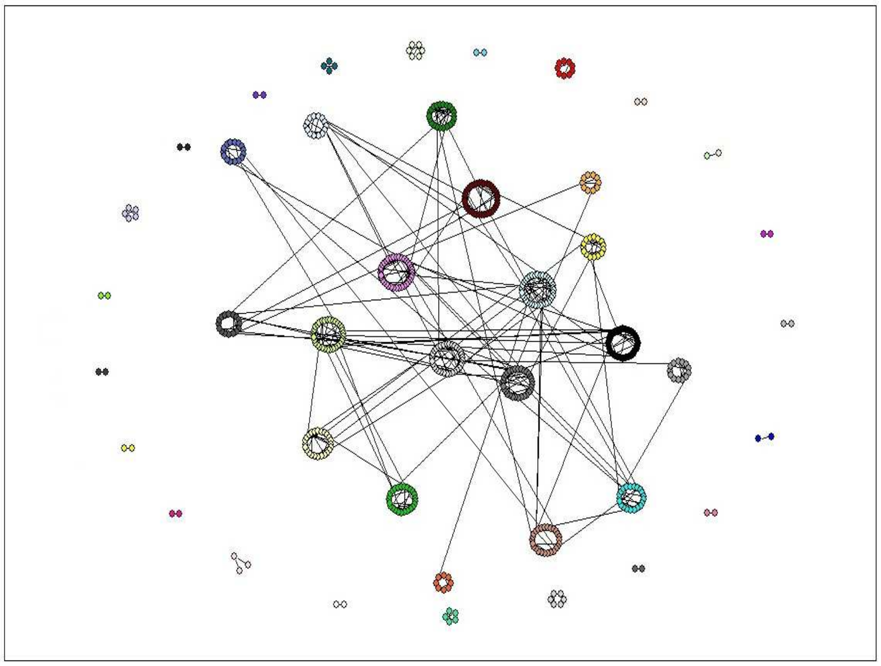


Figure C.6: *A modular model of a "sparse" random-network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into 41 modules and the modularity score for this network is M = 0.8216082. This network is not fully connected; it is comprised of a giant strong component (GSC), which is the most interconnected part (i.e., the core) of the network, and 22 smaller modules around the outside that are isolated from the GSC.*

C.2 *Listeria monocytogenes*

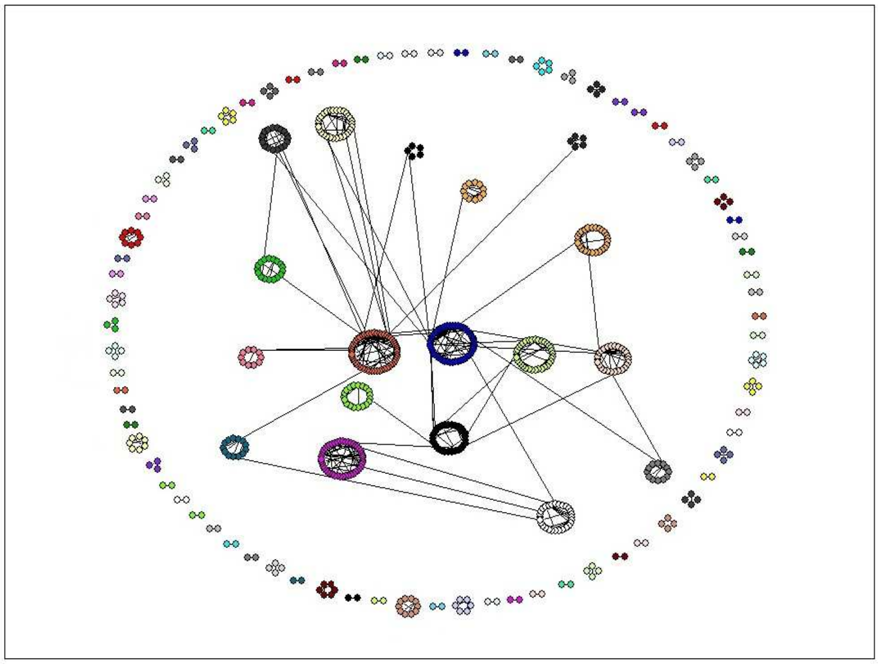


Figure C.7: *A modular model of the Listeria monocytogenes bacterial network. The modules, as shown in this drawing, were identified by the SS algorithm. The network is partitioned into 95 modules and the modularity score for this network is M = 0.8899461. This network is not fully connected; it is comprised of a giant strong component (GSC), which is the most interconnected part (i.e., the core) of the network, and 77 smaller modules around the outside that are isolated from the GSC.*

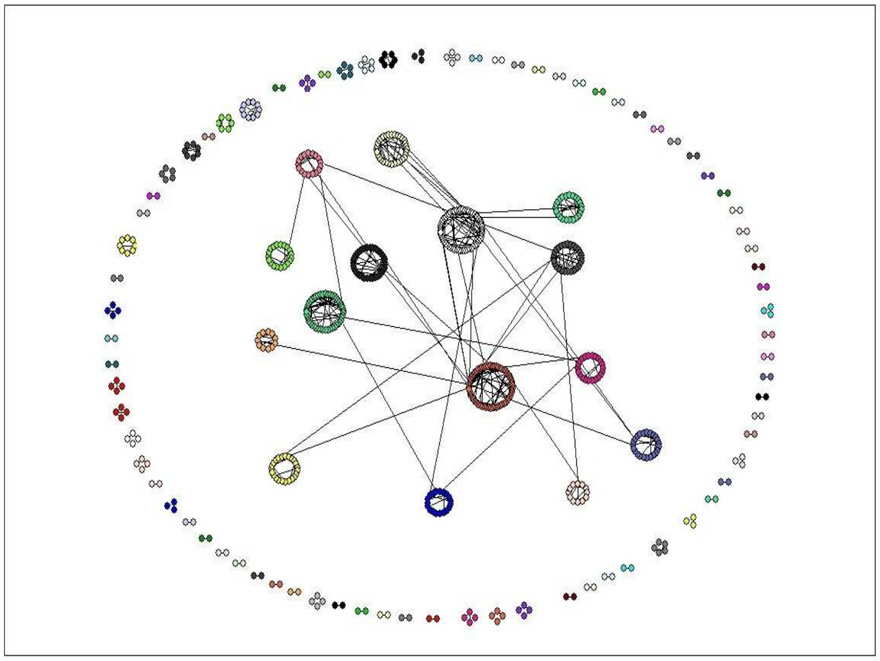


Figure C.8: *A modular model of the Listeria monocytogenes bacterial network. The modules, as shown in this drawing, were identified by the SA algorithm. The network is partitioned into 92 modules and the modularity score for this network is M = 0.890524. This network is not fully connected; it is comprised of a giant strong component (GSC), which is the most interconnected part (i.e., the core) of the network, and 77 smaller modules around the outside that are isolated from the GSC.*

Bibliography

[1] Agresti, A. (2002). Logit models for multinomial responses. In: *Categorical data analysis, 2nd Ed.* New Jersey: John Wiley & Sons, Inc. pp 267-288.

[2] Bielawski, J.P., Yang, Z. (2005). Maximum likelihood methods for detecting adaptive protein evolution. In Rasmus Nielson (Ed.), *Statistical methods in molecular evolution* (p 103). New York: Springer Science+Business Media, Inc.

[3] Brock, G., Goode, J. (Eds.). (2002). *'In silico' simulation of biological processes* (p 91). Chichester, West Sussex, UK: John Wiley & Sons Ltd.

[4] Chakrabarti, D., Modha, D.S., Papadimitriou, S., Faloutsos, C. (2004). Fully automatic cross-associations.

[5] Danon, L., Díaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008.

[6] Dunn, K.A., Bielawski, J.P., Ward, T.J., Urquhart, C., Gu, H. (2009). Reconciling ecological and genomic divergence among lineages of listeria under an "extended mosaic genome concept". *Moleculary Biology and Evolution*, 26(11):2605-2615.

[7] Guimerà, R., Amaral, L.A.N. (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028):895-900.

[8] Guimerà, R., Sales-Pardo, M., and Amaral, L.A.N. (2005). Cartography of complex networks: Modules and universal roles. *Journal of Statistical Mechanics (Online)*, 2005(P02001):P02001+.

[9] Guimerà, R., Sales-Pardo, M., and Amaral, L.A.N. (2004). Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101+.

[10] Guimerà, R., Sales-Pardo, M., and Amaral, L.A.N. (2007). Module identification in bipartite and directed networks. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, 76(3 Pt 2).

[11] Jing, Z., Hong, Y., Jianhua, L., Cao, Z.W., Yi-Xue, L. (2006). Complex networks theory for analyzing metabolic networks. *Chinese Science Bulletin*, 51(13):1529-1537.

[12] Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations Research*, 37(6):865-892.

[13] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671-680.

[14] Krebs, V.E. (2002). Mapping networks of terrorist cells. *Connections*, 24:43-52.

[15] Ma, H., Zeng, A-P. (2003). Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. *Bioinformatics*, 19(2):270-277.

[16] Morine, M.J. (2007). *Functional topology and evolution in prokaryotic metabolic networks* (Master's thesis). Dalhousie University, Halifax, NS.

[17] (2008). Metabolite. *The Columbia encyclopedia, 6th Ed.* Retrieved from http://www.encyclopedia.com/doc/1E1-metabolit.html

[18] Newman, M.E.J., Barabasi, A-L., Watts, D.J. (2006). *The structure and dynamics of complex networks.* New Jersey: Princeton University Press. pp 3-6.

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

[20] Newman, M.E.J., Girvan, M. (2004). Finding and evaluating community structure in networks.

[21] Nightingale, K.K., Windham, K., Wiedmann, M. (2005). Evolution and molecular phylogeny of listeria monocytogenes isolated from human and animal listeriosis cases and foods. *Journal of Bacteriology*, 187(16):5537-5551.

[22] Reichle, A., Hildebrandt, G.C. (2009). Principles of modular tumor therapy. *Cancer Microenvironment*, 2(Suppl 1):S227S237.

[23] Russell, S., Norvig, P. (1995). *Artificial intelligence: A modern approach* Prentice Hall, Inc.

[24] Smith, R.D. (2008). Average path length in complex networks: Patterns and predictions.

[25] Wang, X.F., Chen, G. (2003). Complex networks: Small-world, scale-free and beyond.

[26] Watts, D.J., Strogatz, S. (1998). Collective dynamics of 'small world' networks. *Nature*, 393:440442.

[27] Wiedmann, M. (2002). Molecular subtyping methods for listeria monocytogenes. *Journal of AOAC International*, 85(2):524-532.