5-2015

Bioinformatic Game Theory and Its Application to Cluster Multi-domain Proteins

Brittney Keel
University of Nebraska-Lincoln, s-bhinds1@math.unl.edu

BIOINFORMATIC GAME THEORY AND ITS APPLICATION TO CLUSTER MULTI-DOMAIN PROTEINS

by

Brittney Nicole Keel

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Mathematics

Under the Supervision of Professors Bo Deng and Etsuko Moriyama

Lincoln, Nebraska

May, 2015

BIOINFORMATIC GAME THEORY AND ITS APPLICATION TO CLUSTER MULTI-DOMAIN PROTEINS

Brittney Nicole Keel, Ph. D.

University of Nebraska, 2015

Advisers: Bo Deng and Etsuko Moriyama

The exact evolutionary history of any set of biological sequences is unknown, and all phylogenetic reconstructions are approximations. The problem becomes harder when one must consider a mix of vertical and lateral phylogenetic signals. In this dissertation we propose a game-theoretic approach to clustering biological sequences and analyzing their evolutionary histories. In this context we use the term evolution as a broad descriptor for the entire set of mechanisms driving the inherited characteristics of a population. The key assumption in our development is that evolution tries to accommodate the competing forces of selection, of which the conservation force seeks to pass on successful structures and functions from one generation to the next, while the diversity force seeks to maintain variations that provide sources of novel structures and functions. One branch of the mathematical theory of games is brought to bear. It translates this evolutionary game hypothesis into a mathematical model in two-player zero-sum games, with the zero-sum assumption conforming to one of the fundamental constraints in nature in mass and energy conservation. We demonstrate why and how a mechanistic and localized adaptation to seek out greater information for conservation and diversity may always lead to a global Nash equilibrium in phylogenetic similarity.

DEDICATION

I would like to dedicate this dissertation to my husband and best friend, Ryne Keel, and my parents, Bob and Barbie Courtright.

To Ryne: I have been so lucky to have you in my life. Your love and kindness blow me away every day. I give my deepest love and appreciation for the encouragement that you have given and the sacrifices you have made during my time in the graduate program. Thank you for always supporting me and for your company during the hours and hours of work.

To my parents: You instilled in me the values that shape my life today, and you are literally my biggest inspirations. Dad, thanks for never allowing me to say "I can't." You taught me that the sky is the limit and to never say never. Without this encouragement, I'd never be where I am today. Mom, thanks for always being there when I need to someone to talk to and for always being able to put things into perspective for me. Your support has stuck with me and helped me through this journey.

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my advisors, Dr. Bo Deng and Dr. Etsuko Moriyama, for their continuous guidance and support. I have really enjoyed the time we spent working together over the last four years. Thank you for all of the mentoring and helping me prepare for the next step in my career.

I would also like to thank my committee members Dr. Steve Dunbar, Dr. Stephen Hartke, and Dr. Khalid Sayood, and fellow members of my research group (past and present), Dr. Shunpu Zhang, Ling Zhang, Yixiang Zhang, Neethu Shah, and Ximeng Zheng for their support over the years. Special thanks to Dr. Sayood for inviting me to participate in his lab meetings. Thanks to the office staff in the UNL Mathematics Department, Liz Youroukos, Marilyn Johnson, Lori Mueller, and Tom Danaher for all of their support.

I would also like to say thank you to the Department of Mathematics at the University of Central Missouri. In particular, I would like to thank Dr. Rhonda McKee, Dr. Nick Baeth, and Dr. Lianwen Wang for their encouraging me to go to graduate school.

Lastly, I would like to thank my friends and fellow classmates in the UNL Department of Mathematics. All of you are so awesome and have had such an impact on me. If I took the time to individually mention everyone who deserves something to be said about them, this acknowledgment would be longer than my dissertation! So just simply thank you to you all for everything.

Contents

Chapter 1

Introduction

In the last decade, research in biology has been revolutionized by the onset of next-generation sequencing technologies, such as whole-genome DNA sequencing. The number of completely sequenced genomes is increasing exponentially. Sequencing the genome, however, is only the beginning of many more steps required to decipher the complete information contained in the genome. The two main steps in this process are: locating gene candidates on the genome and identifying the functions of their products (i.e. proteins). With the large amount of data being generated by sequencing technologies, it is necessary to examine the relationships and behavior of many functioning biological molecules simultaneously, rather than individually.

Phylogenetic methods are used to reconstruct the evolutionary history of amino acid and nucleotide sequences. Functionality of biological sequences in the context of phylogenetic trees is investigated by positioning the sequences among others whose functions may be known. The number and diversity of tools for phylogenetic analysis are continually increasing. Classic phylogenetic methods assume that evolution is a tree-like (bifurcating) branching process, where genetic information arises through the divergence and vertical transmission of existing genes, from parent to offspring

[39]. However, when there are reticulate evolutionary events, such as lateral gene transfer (LGT), domain shuffling, or hybridization of species, the evolutionary process is no longer tree-like. Therefore evolutionary histories are often more accurately represented by networks [30, 48, 56, 114].

1.1 Phylogenetic Networks

The development of analytical tools to generate network topologies that accurately describe evolutionary history is an open field of research. Early network construction methods often employed some appropriate notion of distance between taxa. Posada and Crandall [100] explain why networks are appropriate representations for several different types of reticulate evolution and describe and compare available methods and software for network estimation. One of the earliest methods for phylogenetic network construction was the statistical geometry method [34]. The authors in [79] use a least-squares fitting technique to infer a reticulated network. Other network construction methods can be found in [17, 48, 62, 73], each of which is useful for modeling a particular kind of data.

Differentiating between vertical and lateral phylogenetic signals is a challenging task in developing accurate models for reticulate evolution. In order to establish a definition for vertical versus lateral transfer it must be that some component of evolutionary signal recovered from a set of genes be awarded privileged status [56]. In the genomic context, vertical signals are assumed to reside within a core set of genes, shared between genomes. The best examples of such sets are the 16S and 18S ribosomal DNA sequences, often used to infer species phylogeny [10].

When conflicting phylogenetic signals are combined, relationships amongst taxa that appear to be vertical may in fact be lateral and vice versa, resulting in a set of

invalid evolutionary connections [56]. This phenomena is observed, for example, in the thermophilic bacterium *Aquifex aeolicus*, which has been described as an early branching bacterium with similarities to thermophiles [14] or a Proteobacterium with strong LGT connections to thermophiles [19].

A comprehensive map of genetic similarities which encapsulates the results of all phylogenetic signals is a desirable goal. Lima-Mendez et al. [76] developed a methodological framework for representing the relationships across a bacteriophage population as a weighted edge network graph, where the edges represent the phage-phage similarities in terms of their gene content. The genes within the phage were assigned to modules, groups of proteins that share a common function. The authors used graph theory techniques to cluster the phage in the network, and then analyzed the 'module profile' for each of the clusters in order to identify modules that were common to phage within the clusters.

Holloway and Beiko [56] introduced the framework for an evolutionary network known as an intergenomic affinity graph (IAG). An IAG is a directed, weighted edge graph, where each node represents an individual genome, and an edge between two nodes denotes the relative affinity of the genetic material in the source genome to the target genome. The assignment of edge weights in the IAG is based on the solutions to a set of linear programming (LP) problems. A noteworthy feature of the IAG is that the LP derivation of the edge weights does not force the relationship between two genomes to be symmetric.

Phylogenetic networks play an important role in describing and understanding the evolutionary history of biological sequences. They are beginning to replace trees as the basic paradigm for data interpretation in phylogenetics, as they are increasingly being recognized as providing a more complete picture of evolutionary events. There are many research efforts being dedicated to the development of computational

approaches for constructing these networks, and the techniques mentioned above describe only a subset of those currently available.

1.2 Proteins and Their Domains

Proteins are polymers of amino acids that perform a wide variety of functions in living organisms. A protein domain is a highly conserved part of a protein sequence, a structural unit, that can function and evolve almost independently of the rest of the protein. Proteins often include multiple domains. In fact in eukaryotes, 70% or more of proteins are multi-domain proteins [4]. Domain shuffling [45] or domain accretion [69] is an important mechanism in protein evolution underlying the evolution of complex functions and life forms. Multiple evolutionary events including domain duplication, domain loss, recombination, and sequence divergence generate complex proteins [43, 126]. Figure 1.1 is a simple example of evolution illustrating how several different multi-domain proteins can be evolved from one single-domain protein.

1.3 Clustering Protein Sequences

As the size of protein databases continues to grow, large scale sequence comparison is becoming an increasingly more common method for extracting biological information from the sequence data. Protein clustering algorithms are designed to take sets of proteins and assign them to protein families, based on some measure of similarity (or distance). One of the simplest and most common ways to identify sequence similarity for the purpose of clustering is to perform a pairwise-alignment-based sequence similarity search, such as BLAST [3] or FASTA [94]. More sensitive similarity searches can be done using profile hidden Markov models (pHMMs) [40, 61]. Results

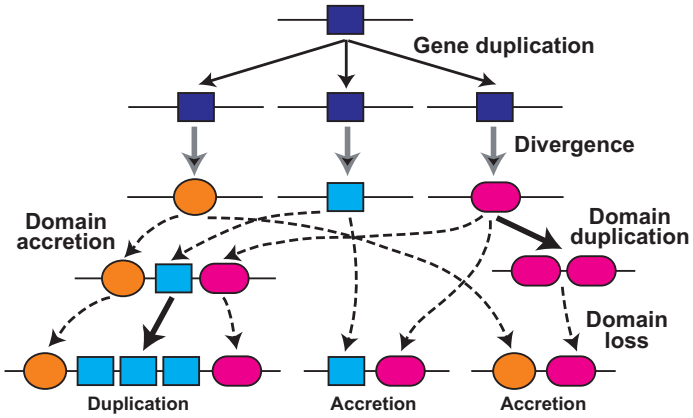


Figure 1.1: **Evolution of Multi-domain Proteins.** Several different multi-domain proteins can be evolved from one single-domain protein. Here the amino acid sequence of a protein is represented by a linear string of domains. Each colored shape (dark blue rectangle, light blue rectangle, orange circle, etc.) represents a distinct domain. Figure adapted from [28].

from these sequence comparison programs can be used to obtain a list of significantly similar sequence pairs, along with their associated similarity scores.

Clustering algorithms can then use this information to generate clusters for the data. For proteins that share a domain, phylogenetic analysis can be performed. In phylogenetic analysis a multiple alignment is generated using the domain shared across the sequences, and then the phylogeny is reconstructed. Pairwise-alignment based methods, including TRIBE-MCL, are used to cluster proteins on a larger scale, where more variation in domain composition exists in the proteins and there is no commonly shared domain across all of the proteins.

There are two fundamental issues that arise when trying to classify groups of divergent proteins, proteins with heterogeneous domain compositions, into protein families. The first is how to construct clusters given that the entirety of the sequence similarity information is not given in the search results. Clustering of sequences in

the context of phylogenetic trees gives an understanding of protein functions by positioning them among other proteins whose functions may be known. However, these methods require sequences to be aligned, limiting their application to proteins that share at least one alignable (shared) domain. If common domains are not found throughout the proteins, each subgroup of proteins needs to be independently analyzed based on different sets of domains. As mentioned before, another drawback to phylogenetic clustering is that these methods assume that evolution is a bifurcating process, where genetic information arises through the divergence and vertical transmission of existing genetic information, from parent to offspring. However, as seen in Fig. 1.1 reticulate evolutionary events, such as domain shuffling, lead to evolutionary histories that are more accurately represented by networks.

In order to get a more complete picture of the evolutionary process of multi-domain proteins the "domain architectures" of the proteins, i.e. the domain content, locations, order, etc., must be considered. Phylogenetic profile methods [12, 21, 57, 68] have tried to address this issue by constructing a phylogenetic tree that takes into consideration the entire domain content by viewing each protein sequence as a vector of domain scores and building the tree with the Euclidean distance for the vectors serving as the pairwise distance between the proteins. Just as in the case of classic phylogenetic methods, network relationships among the proteins cannot be detected using this approach.

Protein similarity networks have been introduced to address the multi-domain protein clustering problem [6, 98]. Many protein similarity networks are constructed using local sequence similarities such as BLAST E-values [3]. The Markov clustering (TRIBE-MCL) algorithm, a graph clustering algorithm that simulates random walks within a graph [124], has been used to cluster proteins in a similarity network into families [22, 35, 128]. Sequence similarity networks based only on local similarities,

such as TRIBE-MCL, in some sense incorporate domain conservation information. However, these methods use information from only one region of similarity between two proteins. More detailed domain architecture information (such as the entire domain content and domain order) needs to be utilized in order to get a clearer picture of the evolutionary histories.

Domain co-occurrence networks [127, 129] and related graph-theoretic approaches [71, 91, 102, 130, 132] incorporate domain composition (and sometimes order) information. However, these methods are employed to depict relationships among the domains, and the relationships between the proteins are usually not considered.

A handful of studies have investigated the application of bi-dimensional clustering (biclustering) for protein classification. Cohen-Gihon et al. [24] used a bipartite graph representation of proteins and domains to identify co-occurring domain sets. Later bipartite graphs were also applied to investigate the distribution of proteins and domains and various other network properties [86]. More recently, Shah et al. [111, 112] have developed a protein clustering method that uses biclustering to cluster protein sequences in terms of their domain composition information.

1.4 Objectives

The key objective of this dissertation is to develop a game-theoretic approach to cluster biological sequences and analyze their evolutionary history. In this context we use the term evolution as a broad descriptor for the entire set of mechanisms driving the inherited characteristics of a population. The key assumption in our development is that evolution (or some subset of the mechanisms therein) tries to accommodate the competing forces of selection, of which the conservation force (e.g. functional constraints) seeks to pass on successful structures and functions from one generation

to the next, while the diversity force seeks to maintain variations that provide sources of novel structures and functions. This hypothesis is naturally modeled through the use of game theory, a framework used to optimize competing goals in various applied fields.

The organization of the remainder of the dissertation is as follows: Chapter 2 describes the relevant background information in the theory of two-player, zero-sum games, and various methods and tools from the field of bioinformatics. Chapter 3 gives an overview of related work in clustering multi-domain proteins, including phylogenetic profile methods, Markov clustering, and protein-domain biclustering. Chapter 4 contains a discussion of definitions and notation for a game-theoretic similarity network, as well as the development of our game theory model and a description of how to use the results to construct the similarity network graph and component profiles. In Chapter 5 we describe an algorithm for simulating multi-domain proteins. This simulator is used to obtain benchmark multi-domain protein sets, whose true evolutionary history is known, in order to test the performance of our game-theoretic protein clusters against clusters obtained from various other methods. Chapter 6 explains the methodologies in each of the steps involved in developing a game-theoretic similarity network for a given set of proteins, the data sets used in this study, and the approach used to compare clusters generated by two different clustering methods. Results are presented in Chapter 7, and the dissertation concludes with an overall discussion of these results and future work in Chapter 8.

Chapter 2

Preliminaries

2.1 Theory of Two-Player, Zero-Sum Games

In this section we review and compile all pertinent results and proofs about two-player, zero-sum games. In particular, we establish definitions and notation that will be present throughout the remainder of this work. Although most of the proofs for the results seen here are readily available in the literature [70, 77, 87, 88, 89, 90], they are provided for completeness.

2.1.1 Existence of Nash Equilibrium for Non-Cooperative n-Player Games

Here we consider a non-cooperative game of n-players, where the term non-cooperative signifies the absence of coalitions amongst players. More specifically, we require that each participant acts independently, without collaboration or communication with the rest [88].

An n-player game consists of a set of n players, each with a finite set of pure

strategies. Let $S_i = \{s_{1i}, s_{2i}, \ldots, s_{n_i i}\}$ denote the set of pure strategies for player i, and $S = \prod_{i=1}^{n} S_i$ be the product set of all pure strategies. For a particular play, player i uses one of his strategies $s_{j_i i} \in S_i$, and we denote by $s = \{s_{j_1 1}, s_{j_2 2}, \ldots, s_{j_n n}\} \in S$ the n-tuple consisting of one such play by each player.

A mixed strategy for player i is a vector $x_i = [x_{1i}, x_{2i}, \ldots, x_{n_i i}]^T$ whose entries represent the relative frequency with which player i will play his pure strategies. That is, letting x_{ji} be the frequency that player i plays the strategy s_{ji}, we have that $x_{ji} \geq 0$ for all $1 \leq j \leq n_i$ and $\sum_{j=1}^{n_i} x_{ji} = 1$.

Let $X_i = \{x_i \in \mathbb{R}_+^{n_i} : \sum_j x_{ji} = 1\}$ be the probability simplex for player i and $X = \prod_{i=1}^{n} X_i$ the product simplex space for all players, where $x \in X$ means $x = \{x_1, x_2, \ldots, x_n\}$ with each $x_i \in X_i$. Each X_i and their product X are convex, compact, and finite dimensional.

Let $a_i(s)$ denote the payoff to player i for the pure strategy play $s = \{s_{j_1 1}, s_{j_2 2}, \ldots, s_{j_n n}\} \in S$. In addition for $x \in X$ let $x(s) = [x_{j_1 1}, x_{j_2 2}, \ldots, x_{j_n n}]^T \in [0,1]^n$ and $\prod x(s) = \prod_{k=1}^{n} x_{j_k k}$. We will use the dynamic notation $x = \{x_i, x_{-i}\}$ to separate player i's play frequency $x_i \in X_i$ from its opponents play frequencies $x_{-i} = \{x_k : k \neq i\} \in X_{-i} = \prod_{k \neq i} X_k$. Similarly, we will use $s = \{s_{j_i i}, s_{-i}\}$ to denote any strategy play s where player i uses strategy $s_{j_i i}$ and his opponents use strategy $s_{-i} = \{s_{j_1 1}, \ldots, s_{j_{i-1}(i-1)}, s_{j_{i+1}(i+1)}, \ldots, s_{j_n n}\} \in S_{-i} = \prod_{k \neq i} S_k$. Thus $x = \{x_1, x_{-1}\} = \{x_2, x_{-2}\} = \cdots = \{x_n, x_{-n}\}$ all denote the same play frequencies, and $x(s) = \{x_i(s_{j_i i}), x_{-i}(s_{-i})\}$. With this notation in place, the mixed strategy's expected payoff for player i is given by

$$p_i(x) = p_i(x_i, x_{-i}) = \sum_{s \in S} a_i(s) \prod x(s) = \sum_{j_i=1}^{n_i} \sum_{s_{-i} \in S_{-i}} a_i(s_{j_i i}, s_{-i}) \left[x_i(s_{j_i i}) \prod x_{-i}(s_{-i}) \right]$$

Let $e_{ji} \in X_i$ be player i's jth pure strategy play, i.e. e_{ji} has all zero frequencies

except for the jth strategy $s_{ji} = 1$. Then substituting e_{ji} for x_i in the formula above we have the expected payoff for player i when he switches to his jth pure strategy while the strategies of his opponents are held fixed:

$$p_i(e_{ji}, x_{-i}) \overset{x_i = e_{ji}}{=} \sum_{s_{-i} \in S_{-i}} a_i(s_{ji}, s_{-i}) \prod x_{-i}(s_{-i})$$

because $x_i(s_{j_i i}) = e_{ji}(s_{j_i i}) = 1$ if $j_i = j$ and 0 if $j_i \neq j$.

Definition 1. *A play frequency $\bar{x} \in X$ is a Nash equilibrium point if*

$$p_i(e_{ji}, \bar{x}_{-i}) \leq p_i(\bar{x}_i, \bar{x}_{-i}) \text{ for all } 1 \leq j \leq n_i \text{ and all } 1 \leq i \leq n.$$

Thus a Nash equilibrium point is an n-tuple such that each player will not improve his payoff by switching to any pure strategy from his mixed play frequency when the strategies of the other players are held fixed.

Theorem 1. *Every n-player game has a Nash equilibrium point (NE).*

Proof. Nash's proof from [88] is based on a map $T : X \to X$ with the property that T has a fixed point by Brouwer's Fixed Point Theorem [108] and the property that a point is a fixed point of T if and only if it is a Nash equilibrium point.

The definition of the map T follows. The *excess payoff* for player i changing to his jth pure strategy from a mixed strategy x is given by

$$\varphi_{ji}(x) = [p_i(e_{ji}, x_{-i}) - p_i(x_i, x_{-i})]_+$$

where $[t]_+ = \max\{0, t\}$, and the total excess payoff (from the mixed strategy x) for

player i is

$$\phi_i(x) = \sum_{j=1}^{n_i} \varphi_{ji}(x).$$

Notice that x is a NE if and only if $\varphi_{ji}(x) = 0$ for all $1 \leq j \leq n_i$ and all $1 \leq i \leq n$ which occurs if and only if $\phi_i(x) = 0$ for all $1 \leq i \leq n$. The Nash map $T : X \to X$ is defined component-wise for each player:

$$[T(x)]_i = \frac{x_i + \varphi_i(x)}{1 + \phi_i(x)},$$

where $\varphi_i = [\varphi_{1i}, \varphi_{2i}, \ldots, \varphi_{n_i i}]^T$. Clearly T is a continuous map which maps into X since $\sum_{j=1}^{n_i}[T(x)]_{ji} = (\sum_{j=1}^{n_i} x_{ji} + \sum_{j=1}^{n_i} \varphi_{ji}(x))/(1 + \phi_i(x)) = 1$.

Let x be any fixed point of T which is guaranteed to exist by Brouwer's Fixed Point Theorem. Note that player i's payoff $p_i(x_i, x_{-i})$ is linear in all x_{ji} and is a weighted probability average in $x_{1i}, x_{2i}, \ldots, x_{n_i i}$. In fact, we have explicitly

$$p_i(x_i, x_{-i}) = \sum_{j=1}^{n_i} \sum_{s_{-i} \in S_{-i}} a_i(s_{ji}, s_{-i}) \left[x_i(s_{ji}) \prod x_{-i}(s_{-i}) \right] = \sum_{j=1}^{n_i} x_{ji} p_i(e_{ji}, x_{-i}).$$

Thus, among all non-zero probability weights $x_{ji} > 0$ there must exist $j \in 1, 2, ..., n_i$ such that the pure s_{ji}-strategy payoff $p_i(e_{ji}, x_{-i})$ is no greater than the mixed strategy payoff $p_i(x_i, x_{-i})$, that is $p_i(e_{ji}, x_{-i}) \leq p_i(x_i, x_{-i})$. Otherwise we will have that $p_i(x_i, x_{-i}) > p_i(x_i, x_{-i})$ because $x_{ji} > 0$ and $\sum_j x_{ji} = 1$, a clear contradiction. As a result, the corresponding excess payoff for this j is zero, that is $\varphi_{ji}(x) = 0$.

Now since x is fixed by T, we have $x_{ji} = \frac{x_{ji} + \varphi_{ji}(x)}{1 + \phi_i(x)} = \frac{x_{ji}}{1 + \phi_i(x)}$, which implies $\phi_i(x) = 0$ since $x_{ji} > 0$. This holds for all i, demonstrating that x is a NE. The converse is straightforward since the excess payoff from every Nash equilibrium x is zero, $\phi_i(x) = 0$, which leads to $T(x) = x$. □

2.1.2 Linear Programming Method for Nash Equilibria of Two-Player, Zero-Sum Games

The theory of two-player, zero-sum games was first developed by von Neumann in 1928 [89, 90]. All results surveyed below are known [70, 77, 89, 90] but this exposition is meant to be more concise and succinct than the others. As a starting point, it is assumed that the reader has knowledge of the simplex method for linear programming, (see [8, 46]).

For the remainder of the section, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are column vectors and $A = A_{m\times n}$ is a matrix. For two vectors, a and b, $a \leq b$ means the inequality holds component-wise. In addition, $\mathbf{1} = [1, \ldots, 1]^T$ denotes the vector whose entries all equal 1, for an appropriate dimension. The linear programming (LP) aspect of two-player, zero-sum game theory is based on the following theorem which encapsulates the simplex method and can be found in most linear optimization textbooks, (see [8, 46]).

Theorem 2. *The primal LP problem*

$$\begin{aligned} &\max z = c^T y \\ &\textit{subject to} \qquad Ay \leq b \\ &\qquad\qquad\quad\; y \geq \boldsymbol{0} \end{aligned}$$

has a solution $y \in \mathbb{R}^n_+$ if and only if the dual LP problem

$$\begin{aligned} &\min z = b^T x \\ &\textit{subject to} \qquad A^T x \geq c \\ &\qquad\qquad\quad\; x \geq \boldsymbol{0} \end{aligned}$$

has a solution $x \in \mathbb{R}^m_+$. *Moreover, if the primal and dual LPs do have solutions,* y^* *and* x^* *respectively, the solutions must satisfy* $c^T y^* = b^T x^*$, *i.e. the optimal objective values for both LP problems must be the same.*

We note that the optimal solution x^* for the dual LP problem is referred to as the *shadow price* or the *Lagrange multipliers* of the primal LP problem and vice versa. Also, the simplex algorithm for the primal problem will simultaneously produce both the optimal solution y^* and its shadow price x^*. The same holds for the dual problem. For convenience, we also need the following result.

Lemma 1. *Let* S *be the simplex defined by* $w_i \geq 0$ *for all* $1 \leq i \leq k$ *and* $\sum_{i=1}^k w_i = 1$, *then* $\max_{w \in S} c^T w = \max_{1 \leq i \leq k}\{c_i\}$. *Similarly,* $\min_{w \in S} c^T w = \min_{1 \leq i \leq k}\{c_i\}$.

Proof. Here we consider the maximization case. Let $c_{i_0} = \max_{1 \leq i \leq k}\{c_i\}$. Then $c_i \leq c_{i_0}$ for all i and since $w_i \geq 0$ we have that $c^T w = c_1 w_1 + \cdots + c_n w_n \leq c_{i_0}(w_1 + \cdots + w_n) = c_{i_0}$. Hence $\max_{w \in S} c^T w \leq c_{i_0}$. Note that equality must hold since the value c_{i_0} is obtained from the function $z = c^T w$ with $w_{i_0} = 1, w_i = 0$ for $i \neq i_0$. A similar argument is used to show the minimization case. □

In the previous section existence of Nash equilibria was demonstrated for a general n-player game. We will now restrict our attention to the special case of a two-player game, i.e. a *matrix game*. For $i \in \{1, 2\}$ let $S_i = \{s_{1i}, s_{2i}, \ldots, s_{n_i i}\}$ denote the set of pure strategies for player i, $x = [x_1, x_2, \ldots, x_m]^T$ a mixed strategy vector for player 1 (player $\mathbf{x}$), and $y = [y_1, y_2, \ldots, y_n]^T$ a mixed strategy vector for player 2 (player $\mathbf{y}$). Then for each player, i, there exists a matrix $A_i \in \mathbb{R}^{m \times n}$ such that the entry $A_i(q, r) = a_i(s_{q1}, s_{r2})$, which denotes the payoff to player i when player $\mathbf{x}$ plays pure strategy q and player $\mathbf{y}$ plays pure strategy r.

A zero-sum game is a game in which a player's gain or loss is exactly balanced by the losses and gains of the other player(s). That is, in the case of a two-player,

		Player 2				
Player 1			y_1	y_2	$\cdots$	y_n
		Strategy	s_{12}	s_{22}	$\cdots$	s_{n2}
	x_1	s_{11}	$a_2(s_{11}, s_{12})$	$a_2(s_{11}, s_{22})$	$\cdots$	$a_2(s_{11}, s_{n2})$
	x_2	s_{21}	$a_2(s_{21}, s_{12})$	$a_2(s_{21}, s_{22})$	$\cdots$	$a_2(s_{21}, s_{n2})$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	x_m	s_{m1}	$a_2(s_{m1}, s_{12})$	$a_2(s_{m1}, s_{22})$	...	$a_2(s_{m1}, s_{n2})$

Table 2.1: **Two-Player, Zero-Sum Game in Matrix Form.** This figure shows the matrix form of a two-player, zero-sum game, where the matrix is assumed to be the payoff table for player **y** against player **x**, i.e. $\mathbf{A} = \mathbf{A}_2$.

zero-sum game we have that $A_1 = -A_2$. Since A_2 is easily deduced from A_1, we may focus our attention on only one of the matrices, denoted A. However, we must specify ahead of time to which player the matrix corresponds.

The matrix form of a two-player, zero-sum game is exhibited in Table 2.1. This matrix is assumed to be the payoff table for player **y** against player **x**, i.e. $A = A_2$. The pure strategies as well as the mixed strategy frequencies are listed along the rows and columns for easy reference. Since A is the payoff matrix for player **y**, the expected payoff for player **y** is $p_2(x, y) = x^T Ay$, and the expected payoff to player **x** is $p_1(y, x) = -x^T Ay = -p_2(x, y)$. The remainder of this section shows the Nash equilibria can be found by means of linear optimization, i.e. by the simplex method from linear programming.

Again for a two-player, zero-sum game, $x = [x_1, \ldots, x_m]^T$ and $y = [y_1, \ldots, y_n]^T$ are the mixed strategy probability vectors, and $A = A_{m\times n} = [a_{ij}]$ is the payoff matrix for player **y** against player **x**. Let A_j be the column vectors of the matrix A and $\tilde{A}_i$ be the row vectors of A, i.e. $A = [A_1, A_2, \ldots, A_n]$ and $A^T = [\tilde{A}_1^T, \ldots, \tilde{A}_m^T]$. Then, the expected payoff per play for player **y** is $p_2(x, y) = \sum_i \sum_j a_{ij} x_i y_j$. For ease of notation we let $E(x, y) = p_2(x, y) = \sum_i \sum_j a_{ij} x_i y_j$. Note that the bi-linear payoff function $E(x, y) = x^T Ay$ can be summed in two different ways, each as a

probabilistically weighted linear form: $E(x,y) = \sum_i x_i(\sum_j a_{ij}y_j) = x^T(Ay)$ and $E(x,y) = \sum_j(\sum_i a_{ij}x_i)y_j = (x^TA)y$.

Proposition 1. *$(\bar{x},\bar{y})$ is a NE if and only if*

$$\max_y E(\bar{x},y) = E(\bar{x},\bar{y}) = \min_x E(x,\bar{y})$$

or equivalently

$$E(\bar{x},y) \leq E(\bar{x},\bar{y}) \leq E(x,\bar{y})$$

for all $(x,y) \in S = \prod_{i=1}^2 S_i$.

Proof. By definition, $(\bar{x},\bar{y})$ is a NE if and only if $E(\bar{x},e_j) \leq E(\bar{x},\bar{y})$ for all pure strategy vectors e_j of player **y** and $-E(e_i,\bar{y}) \leq -E(\bar{x},\bar{y})$ for all pure strategy vectors e_i of player **x** since $-E(x,y)$ is the expected payoff for player **x**. That is, $E(\bar{x},e_j) \leq E(\bar{x},\bar{y}) \leq E(e_i,\bar{y})$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. Because E is a probabilistically weighted linear form in both x and y, we have by Lemma 1 that $\max_y E(\bar{x},y) = \max_j\{E(\bar{x},e_j)\} \leq E(\bar{x},\bar{y}) \leq \min_i\{E(e_i,\bar{y})\} = \min_x E(x,\bar{y})$. Since both extreme values are reached by a NE point, $(\bar{x},\bar{y})$, the equalities hold. The second equivalence is obvious from the first. □

A NE as a solution $(\bar{x},\bar{y})$ to the optimization problem described in the previous proposition is known as an *optimal game solution* or simply a game solution, and $E(\bar{x},\bar{y})$ is referred to as the *game value* for player **y**.

Proposition 2. *The game value for a two-player, zero-sum game is unique.*

Proof. Let $(\bar{x},\bar{y}),(x',y')$ be two optimal solutions with game values $u = E(\bar{x},\bar{y})$ and $v = E(x',y')$, respectively. Then by the result above, $u = E(\bar{x},\bar{y}) \leq E(x',\bar{y}) \leq E(x',y') = v$, where the first inequality holds since $(\bar{x},\bar{y})$ is a NE, and the second

inequality holds since (x', y') is a NE. Since u and v are two arbitrary Nash equilibria, we have by the same argument $v \leq u$, which gives $u = v$. □

Proposition 3. *For the primal LP problem*

$$\begin{aligned} &\max z = u \\ &\textit{subject to} \quad Ay \geq u\boldsymbol{1} \\ &\qquad\qquad y \geq 0, \sum_j y_j = 1 \end{aligned}$$

the dual LP problem is given by

$$\begin{aligned} &\min w = v \\ &\textit{subject to} \quad x^T A \leq v\boldsymbol{1}^T \\ &\qquad\qquad x \geq 0, \sum_i x_i = 1 \end{aligned}$$

Therefore, the optimal objective value is the same, and the solution of one problem is exactly the shadow price of the other.

Proof. By introducing $u_1 \geq 0, u_2 \geq 0$ for $u = u_1 - u_2$ and $\sum_j y_j \leq 1$ and $-\sum_j y_j \leq -1$ for $\sum_j y_j = 1$, we can recast the LP problem

$$\begin{aligned} &\max z = u \\ &\text{subject to} \quad Ay \geq u\mathbf{1} \\ &\qquad\qquad y \geq 0, \sum_j y_j = 1 \end{aligned}$$

as follows:

$$\begin{aligned} \max z &= C^T Y \\ \text{subject to} \quad & \mathcal{A}Y \leq B \\ & Y \geq 0, \end{aligned}$$

where $Y = [y_1, \ldots, y_n, u_1, u_2]^T$, $C = [0, \ldots, 0, 1, -1]^T$, $B = [0, \ldots, 0, 1, -1]^T$, and $\mathcal{A} = \begin{bmatrix} -A & \mathbf{1} & -\mathbf{1} \\ \mathbf{1}^T & 0 & 0 \\ -\mathbf{1}^T & 0 & 0 \end{bmatrix}$. By Theorem 2, the dual LP problem is given by

$$\begin{aligned} \min z &= B^T X \\ \text{subject to} \quad & \mathcal{A}^T X \geq C \\ & X \geq 0, \end{aligned}$$

where $X = [x_1, \ldots, x_m, v_1, v_2]^T$. Writing the latter in $\mathcal{A}$'s block component form, we obtain

$$\begin{aligned} \min z &= v_1 - v_2 \\ \text{subject to} \quad & -A^T x \geq -(v_1 - v_2)\mathbf{1} \\ & \mathbf{1}^T x \geq 1 \\ & -\mathbf{1}^T x \geq -1. \end{aligned}$$

Letting $v = v_1 - v_2$ we have the equivalent form of the dual LP problem

$$\begin{aligned} \min z &= v \\ \text{subject to} \quad & x^T A \leq v\mathbf{1}^T \\ & x \geq 0, \sum_i x_i = \mathbf{1}^T x = 1. \end{aligned}$$

□

The following two theorems complete this compilation of the basic theory for two-player, zero-sum games. The first is the LP algorithm for finding Nash equilibria, and the second is the minimax theorem.

Theorem 3. *$(\bar{x}, \bar{y})$ is an optimal game solution with game value $\bar{v} = E(\bar{x}, \bar{y})$ if and only if $(\bar{x}, \bar{v})$ is a solution to the LP problem*

$$\begin{aligned} &\min z = u \\ &\textit{subject to} \quad x^T A \leq u\mathbf{1}^T \\ &\qquad\qquad x \geq 0, \sum x_i = 1, \end{aligned} \tag{2.1}$$

and $(\bar{y}, \bar{v})$ is a solution to the dual LP problem

$$\begin{aligned} &\max z = u \\ &\textit{subject to} \quad Ay \geq u\mathbf{1} \\ &\qquad\qquad y \geq 0, \sum y_j = 1. \end{aligned} \tag{2.2}$$

Proof. For necessity: As an optimal game solution

$$\bar{v} = E(\bar{x}, \bar{y}) = \max_y E(\bar{x}, y) = \max_y (\bar{x}^T A)y = \max_j \{\bar{x}^T A_j\}$$

by Lemma 1, which implies $\bar{x}^T A_j \leq \bar{v}$ for all j and equivalently $\bar{x}^T A \leq \bar{v}\mathbf{1}^T$. That is, $(\bar{x}, \bar{v})$ is a basic feasible point for the LP problem (2.1).

We claim $(\bar{x}, \bar{v})$ must be an optimal solution to the LP problem. For if not, there exists (x', u) such that $(x')^T A \leq u\mathbf{1}^T$ with $u < \bar{v} = E(\bar{x}, \bar{y})$. That is, $\max_j \{(x')^T A_j\} \leq u < \bar{v}$ componentwise. By Lemma 1, we have

$$\max_y E(x', y) = \max_y ((x')^T A)y = \max_j \{(x')^T A_j\} \leq u < \bar{v} = E(\bar{x}, \bar{y}).$$

So $E(x', \bar{y}) \leq \max_y E(x', y) \leq u < \bar{v} = E(\bar{x}, \bar{y})$, which contradicts the property that $(\bar{x}, \bar{y})$ is a NE. Similar arguments apply to the primal LP problem, completing the proof of the necesary condition.

Conversely, because $\bar{x}, \bar{y}$ are the optimal solutions for the dual pair with the optimal value $\bar{v}$, from $\bar{x}^T A \leq \bar{v}\mathbf{1}^T$ we have $E(\bar{x}, \bar{y}) = (\bar{x}^T A)\bar{y} \leq (\bar{v}\mathbf{1}^T)\bar{y} = \bar{v}(\mathbf{1}^T\bar{y}) = \bar{v}$ and from $A\bar{y} \geq \bar{v}\mathbf{1}$ we have $E(\bar{x}, \bar{y}) = \bar{x}^T(A\bar{y}) \geq x^T(\bar{v}\mathbf{1}) = \bar{v}$ and hence $E(\bar{x}, \bar{y}) = \bar{v}$. Also, for any x, $E(x, \bar{y}) = x^T(A\bar{y}) \geq \bar{v} = E(\bar{x}, \bar{y})$ and for any y, $E(\bar{x}, y) = (\bar{x}^T A)y \leq \bar{v} = E(\bar{x}, \bar{y})$, showing $(\bar{x}, \bar{y})$ is an optimal game solution with game value $\bar{v}$. □

Theorem 4. *Let $(\bar{x}, \bar{y})$ be a NE, then $E(\bar{x}, \bar{y}) = \min_x[\max_y E(x, y)]$ over the mixed strategy probability vectors and symmetrically $E(\bar{x}, \bar{y}) = \max_y[\min_x E(x, y)]$.*

Proof. Note that the primal LP problem (2.1) can be equivalently written as

$$x^T A \leq u\mathbf{1}^T \Leftrightarrow \max_j x^T A_j \leq u \Leftrightarrow \max_y (x^T A)y \leq u \Leftrightarrow \max_y E(x, y) \leq u$$

with the smallest such u. This implies $\min_x(\max_y E(x, y)) \leq \min_x u = \bar{v} = E(\bar{x}, \bar{y})$.

We claim the equality $\min_x(\max_y E(x, y)) = \min_x u$ must hold. If not, let $w(x) = \max_y E(x, y)$ and x' have the property that $u' = w(x') = \min_x w(x)$ with $u' < \min_x u = \bar{v}$. Then $x^T A \leq w(x)\mathbf{1}^T$ for all x. In particular, $(x')^T A \leq w(x')\mathbf{1}^T = u'\mathbf{1}^T$, showing that (x', u') is a basic feasible point to the LP problem. Since $\bar{v}$ is the optimal value of the LP solution, it must be that $u' \geq \bar{v}$, contradicting the assumption $u' < \bar{v}$. The same argument applies to the dual LP problem. □

2.2 Basic Methods and Tools in Bioinformatics

2.2.1 Sequence Similarity and Function

Given that a only a small portion of known proteins have been experimentally characterized, protein annotation relies heavily upon the accurate exploitation of evolutionary relationships, as functional information is extrapolated following the identification of a sequence relative, on the basis that family members commonly exhibit some similarity in function [122]. Hence sequence similarity detection plays a key role in the functional annotation of proteins.

2.2.1.1 BLAST

The most well-known similarity detection tool is the Basic Local Alignment Search Tool (BLAST) [1]. In short, BLAST is an algorithm that finds short matches between two sequences of interest and builds local alignments based on these "hot spots" [84]. BLAST searches a protein or nucleotide database, specified by the user, and reports both the significant sequence hits and their local alignment to the query sequence (Fig. 2.1).

BLAST is a heuristic search algorithm that seeks to approximate the results of

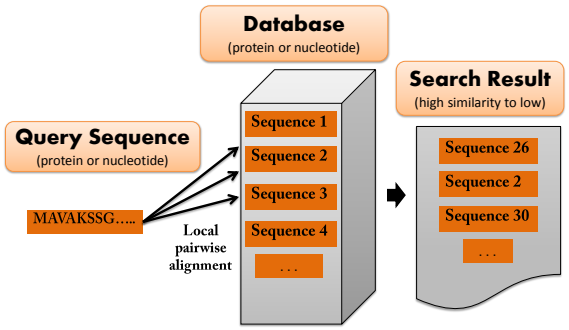


Figure 2.1: **Searching in BLAST.**

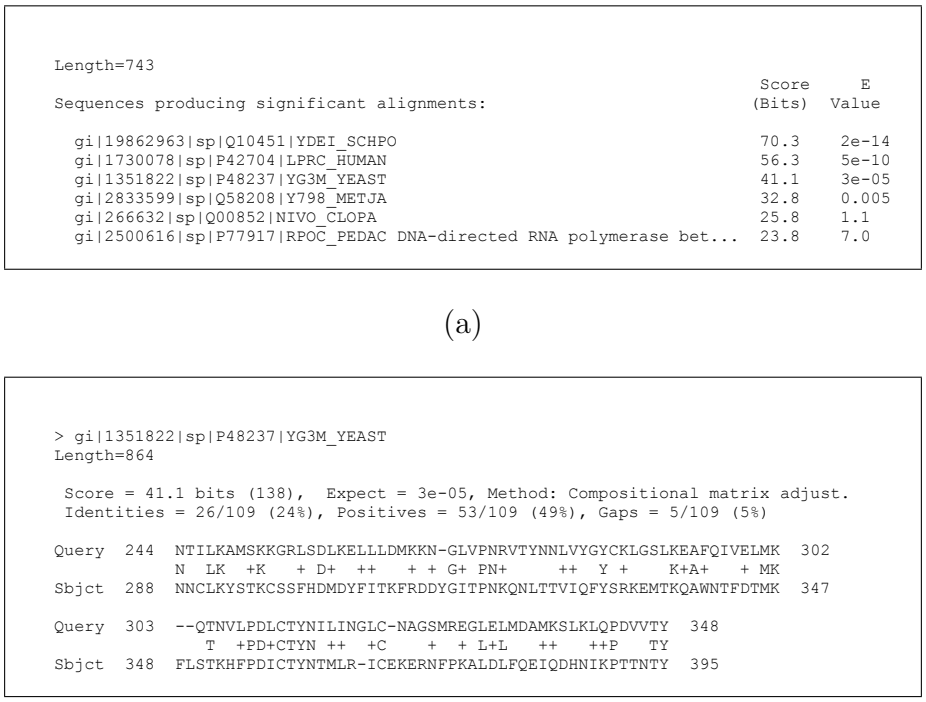
the Smith-Waterman algorithm [113]. The BLAST algorithm begins by dividing the query sequence into words of a specified length and scanning the database for words that the query sequence and sequences in the database have in common. A word of length k is simply defined to be a k-tuple in the query sequence. For example, if the query sequence is ABCDE and the prescribed word length is $k = 3$ the algorithm scans the database for the words ABC, BCD, and CDE. For each word in the query sequence, BLAST determines all possible words that could be aligned to it with a score greater than a threshold T, called the neighborhood threshold score. BLAST builds these words into a table and scans a database sequence for exact matches, each of which constitutes a hit.

For each hit, the alignment is then extended in both directions (gap-free) until the score falls below the best score seen so far. The alignment scores are based on empirically derived nucleotide or amino acid substitution matrices, such as BLOSUM and PAM [27, 50, 51, 63, 67]. These high-scoring, un-gapped alignments are called maximal segment pairs (MSPs) [1]. Once the un-gapped MSPs are identified, these alignments are extended to gapped alignments using a dynamic programming algorithm [31].

BLAST reports the list of sequences in the database that contained significant local alignments, as well as the actual alignments that were significant. Figure 2.2 exhibits the contents of a typical BLAST report.

2.2.1.2 BLAST E-value

The BLAST program employs a variety of refinements to adjust the raw similarity score of a pairwise alignment (i.e. the score based only on the substitution matrix and gap penalties). These refinements fine-tune the score to more accurately predict the evolutionary significance of the similarity between aligned sequences. In this work,



```
Length=743

                                                                      Score     E
Sequences producing significant alignments:                          (Bits)  Value

  gi|19862963|sp|Q10451|YDEI_SCHPO                                    70.3    2e-14
  gi|1730078|sp|P42704|LPRC_HUMAN                                     56.3    5e-10
  gi|1351822|sp|P48237|YG3M_YEAST                                     41.1    3e-05
  gi|2833599|sp|Q58208|Y798_METJA                                     32.8    0.005
  gi|266632|sp|Q00852|NIVO_CLOPA                                      25.8    1.1
  gi|2500616|sp|P77917|RPOC_PEDAC DNA-directed RNA polymerase bet...  23.8    7.0
```

(a)

```
> gi|1351822|sp|P48237|YG3M_YEAST
Length=864

 Score = 41.1 bits (138),  Expect = 3e-05, Method: Compositional matrix adjust.
 Identities = 26/109 (24%), Positives = 53/109 (49%), Gaps = 5/109 (5%)

Query  244  NTILKAMSKKGRLSDLKELLLDMKKN-GLVPNRVTYNNLVYGYCKLGSLKEAFQIVELMK  302
            N  LK  +K   + D+  ++   + + G+ PN+     ++  Y +    K+A+   + MK
Sbjct  288  NNCLKYSTKCSSFHDMDYFITKFRDDYGITPNKQNLTTVIQFYSRKEMTKQAWNTFDTMK  347

Query  303  --QTNVLPDLCTYNILINGLC-NAGSMREGLELMDAMKSLKLQPDVVTY  348
               T  +PD+CTYN ++  +C    +  + L+L   ++   ++P   TY
Sbjct  348  FLSTKHFPDICTYNTMLR-ICEKERNFPKALDLFQEIQDHNIKPTTNTY  395
```

(b)

Figure 2.2: **Components of A Typical BLAST Report.** (a) A sample listing of the database sequences with significant local alignments. (b) A sample significant local alignment to a database sequence.

we use the *expectation value* (E-value) to assess the statistical significance of a single pairwise alignment within a database search.

In general, the E-value is given by the equation

$$E = D * P(S \geq x),$$

where D denotes the number of sequences in the database and $P(S \geq x)$ denotes the probability of obtaining an alignment score, S, greater than some observed score x [1]. Thus the E-value ranges from 0 to D, and it gives the number of database sequences not related to the query that are expected to have alignment scores higher

than the observed score. Thus small E-values (i.e. values close to 0) indicate more significant alignments.

The BLAST E-value is calculated based on the stochastic model of Karlin and Altschul [2, 64]. In this model, the probability of getting an alignment score x or higher is given by

$$P(S \geq x) = 1 - \exp(-Kmne^{-\lambda x}), \tag{2.3}$$

where K and λ are constants calculated from scoring matrix and amino acid composition (empirically calculated), m is the length of the query sequence, and n the length of the database.

2.2.2 Protein Domain Identification

The BLAST algorithm is very effective in identifying sequences with high pairwise similarity. In order to identify more distantly related similar sequences, it is necessary to employ more sensitive search methods (sequence profile methods), such as PSI-BLAST (Position-Specific Iterative BLAST) [3] or profile hidden Markov models [33]. In this work, we use HMMER [33], a profile hidden Markov model (pHMM) search algorithm, for protein domain detection, which is described in the remainder of this section.

2.2.2.1 Profile Hidden Markov Models

A hidden Markov model (HMM) is a very general, widely used, probabilistic model for sequences of symbols. In an HMM, as suggested by the name, one of the important underlying features is the assumption that the system being modeled is a Markov process, a set of states and transition probabilities between those states, where the probability of a given state depends only on the previous state. Mathematically, a

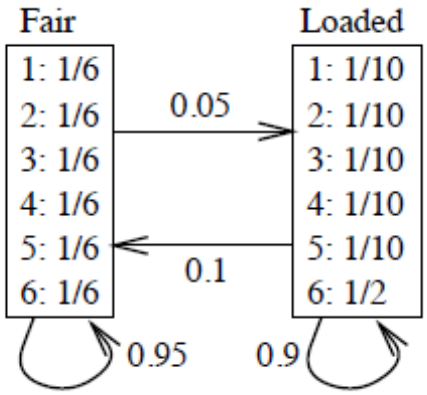


Figure 2.3: **The Occasionally Dishonest Casino Hidden Markov Model.** Each node represents a state in the model with the emission probabilities for each state given inside the node. The directed edges represent the transitions and their weights the transition probabilities. Figure taken from [31].

Markov process is one in which the following condition, known as the Markov property, holds:

$$P(X_k|X_{k-1}, X_{k-2}, ..., X_1) = P(X_k|X_{k-1}). \tag{2.4}$$

In a standard Markov process we are given a set of states and a set of transition probabilities between those states. There is a clear one-to-one correspondence between the states and the emitted characters in the sequence. However, in many situations there are likely unobserved (hidden) states that affect the output of the model. A classic textbook example of such a scenario is the "Occasionally Dishonest Casino" problem, where the casino uses a fair die most of the time but occasionally switches to a loaded die [31]. Here when we examine a sequence of rolls, say $X =< 1, 2, 4, 3, 1, 6, 5 >$, we observe only the outcome and cannot identify whether the fair or loaded die was used. In this case the hidden state is the type of die being used.

A hidden Markov model (HMM) incorporates a Markov process, but decouples the state sequence from the symbol sequence, as they are no longer in one-to-one correspondence. A single symbol could be emitted by many different states in a given HMM. For instance in the casino model described above, the symbol 4 could have been emitted in the fair die state or in the loaded die state. Thus in the case of an HMM we must differentiate between the state and emission probabilities. HMMs are often illustrated graphically, with the states representing nodes and the directed edges transitions. Figure 2.3 shows a graphical representation of the Occasionally Dishonest Casino problem.

A profile HMM (pHMM) is a particular type of hidden Markov model well-suited to model the contents of a multiple sequence alignment [31]. As mentioned above, in order to identify distantly related similar biological sequences one must employ sensitive search methods that incorporate information from multiple sequences rather than a single sequence. A "profile" is a representation of the composite information within a set of sequences. Profiles are constructed by creating a multiple sequence alignment (MSA) for the set of sequences. In an MSA, homologous residues within the set of sequences are aligned together in columns [31]. Figure 2.4 shows an example of an MSA for seven protein sequences.

```
VGA--HAGEY
V----NVDEV
VEA--DVAGH
VKG------D
VYS--TYETS
FNA--NIPKH
IAGADNGAGV
```

Figure 2.4: **Multiple Sequence Alignment.** A multiple sequence alignment for seven protein sequences. The letters each represent an amino acid, while "-" represents a gap in the alignment. Example taken from [31].

Profile HMMs consist of three states for each alignment position (i.e. each column

in the MSA) that model the three possible outcomes that can occur when aligning a residue of the query sequence with the MSA. The query residue may align (match) with the next residue of the MSA, it may correspond to an insertion relative to the MSA, or it may correspond to a deletion (a gap) relative to MSA.

The first step in building a pHMM, is to determine its length. The length of the pHMM is determined by the number of columns in the MSA that are assigned to match states. There are many heuristic rules for assigning the MSA columns as match states, one of which considers a column to be an match column if less than half of the characters are gap characters. Using this heuristic for the MSA shown in Fig. 2.4, columns 1-3 and 6-10 are match columns. Hence the length of the corresponding pHMM is 8.

The most basic state is the match state, which matches (i.e. aligns) residues at a specific position (column) in the MSA. Each match state in the pHMM has its own corresponding set of emission probabilities, generated from counting the frequencies of each amino acid in the corresponding column.

For insertions, i.e. portions of the query sequence that do not match anything in the multiple alignment, an insert state is added into the model. Just as in the case of the match states, each insert state has its own set of emission probabilities. The insert state emission probabilities are typically generated using the background amino acid distribution, that is the distribution of amino acids over the entire MSA.

Lastly, in the case of a deletion, the insertion of a gap in the alignment, the query sequence residue aligns to a residue later in the MSA. This could be viewed as a set of jump transitions between match states, but this would require a large number of transitions for long gaps [31]. Thus a delete state is introduced for each of the positions in the MSA. The delete state is an example of a silent state in the model, as it does not emit any residues.

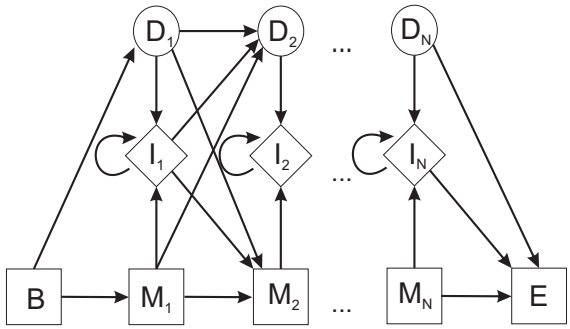


Figure 2.5: **Profile HMM Structure.** Each node represents a state in the model. The square nodes indicate the match states, the diamonds the insert states, and the circles the delete states. The directed edges represent the transitions.

In the general profile HMM structure, only a few transitions are acceptable at each state for the j^{th} alignment position. From the match state, M_j there is a transition to the next match state M_{j+1}, a transition to the insert state I_j, and a transition to the delete state D_j. From the insert state I_j there is a transition to the next match state M_{j+1} and the next delete state D_{j+1}. In addition for the insert state there is a self-loop, to handle the case when multiple residues are inserted. Lastly, from the delete state D_j there is a transition to the insert state I_j, a transition to the next delete state D_{j+1}, and a transition to the next match state M_{j+1}. All transitions have an associated probability, such that at each state the outgoing transition probabilities sum to 1. Adding in the silent begin and end states gives the general profile HMM structure shown in Fig. 2.5.

Once the general structure of the profile HMM has been established, the emission and transition probabilities must be determined. Having both the multiple sequence alignment and the HMM structure, the state sequence is known, and therefore the parameters can be found by simply counting the frequencies. That is, the transition

probability from state k to state l is given by

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}},$$

where A_{kl} denotes the frequency of transitioning from state k to state l, and the emission probability of symbol a at state k is

$$e_k(a) = \frac{E_k(a)}{\sum_{a'} E_k(a')},$$

where $E_k(a)$ is the frequency that symbol a was emitted in state k.

One issue that can arise is the event in which some transition or emission does not occur in the MSA. If this is the case, the probability of that event will be assigned a zero value, which means that it is not allowed to occur in the future. As a minimal approach to ensure non-zero prior probabilities, pseudocounts should be added to the initial frequencies. The most basic pseudocount is to simply add one to each of the base frequencies. Another commonly used pseudocount is the Dirichlet mixture prior, which is based on counting the observations of each amino acid at each column in the MSA (for details see [16]). In fact, Dirichlet priors are the default pseudocounts in the HMMER [33] pHMM software, which is used in this work.

Once the profile HMM has been generated, it is then most frequently used to determine significant matches in a query sequence to the profile and to use these matches to detect the potential membership of the query sequence to the pHMM. This is done by computing the log-odds ratio between the probability that the query sequence s came from the model M, $P(s|M)$, and the probability that the query was

generated at random via the model R, $P(s|R)$:

$$L = \log \frac{P(s|M)}{P(s|R)}.$$

A high log-odds score indicates that the query sequence is likely related to the sequences in the pHMM, while a low score (a score near 0) indicates an insignificant match.

2.2.2.2 Pfam and HMMER

In this work, we use profile hidden Markov models to detect protein domains along a given query protein. Pfam [103] is a large database of protein families, which includes both their annotations as well as their multiple sequence alignments. In our work, the profile HMMs from Pfam-A (a manually curated, high quality sector of Pfam) are used as the domain profiles for our pHMM domain searches. Domain sequences along a query protein were identified using HMMER [33], a set of pHMM search algorithms.

2.2.3 Phylogeny

Phylogenetic analysis seeks to understand the evolutionary history of organisms (or groups of organisms) by establishing a relationship based on shared characteristics. The framework for such analysis is fundamentally driven by the assumption that genetic information arises through the divergence and vertical transmission of existing genes, from parent to offspring. In other words, the assumption is that evolution is a tree-like, branching process. With this assumption in place, the reconstruction of the evolutionary history of a given set of taxa can be represented graphically by a stucture known as a phylogenetic tree.

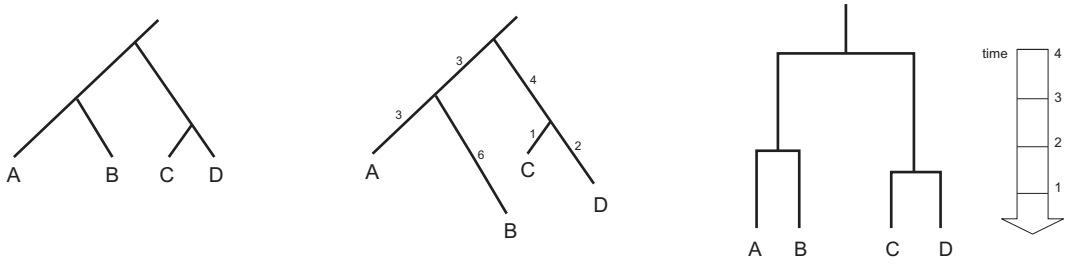
(a) Rooted Tree (b) Unrooted Tree

Figure 2.6: **Phylogenetic Tree Structure.**

A phylogenetic tree is a graphical representation of a set of inferred evolutionary relationships between a set of taxa. The basic features of such a graph are shown in Fig. 2.6. The taxa are connected by edges (branches) which indicate the presence of an evolutionary relationship. The leaves (external nodes) of the tree are the currently existing taxa that have yet to evolve or taxa whose lineage died without producing any descendants. The edges in the tree do not directly connect the leaves, but do so through internal nodes. These internal nodes represent hypothesized ancestral states that occurred throughout the evolutionary process.

Phylogenetic trees can be rooted or unrooted (Fig. 2.6). Rooted trees are used when the direction of evolution, i.e. the common ancestor, is known. Thus, rooted trees exhibit the divergence of a group of taxa from the last common ancestor. In contrast, unrooted trees demonstrate the evolutionary relationships between taxa without identifying their last common ancestor. Hence, in an unrooted tree the order of ancestral descendence is unclear once one gets to the internal nodes.

There are two other key components of phylogenetic tree - the tree topology and the length of the branches. The topology of a tree refers to the way that it branches. Even for a small number of taxa there are many possible tree topologies. For example, in [93] it is shown that the number of possible binary rooted trees



(a) Cladogram (b) Additive Tree (c) Ultrametric Tree

Figure 2.7: **Types of Phylogenetic Trees.**

that can be produced with N leaf nodes is $\frac{(2N-4)!}{2^{N-2}(N-2)!}$. Each distinct topology represents a possible evolutionary history, where the differences occur in event order and ancestry. The most important task in phylogenetic tree construction is to recover the tree that best describes the evolution given the data.

The branch lengths of a phylogenetic tree provide additional information about the evolution of the taxa. Branch lengths can be incorporated into both rooted and unrooted tree topologies. In a cladogram (Fig. 2.7 (a)) the branch lengths do not have a meaning, i.e. the genealogy of the taxa is exhibited but nothing is claimed about the timing or extent of their divergence. Additive trees (Fig. 2.7 (b)) use the branch lengths as a quantitative measure of evolution. Typically the branch lengths are proportional to the number of mutations that occurred through the evolutionary process. Moreover, as the name suggests, in such a tree the branch lengths are additive, i.e. the distance between any pair of nodes is the sum of the branch lengths connecting them [31]. Lastly, an ultrametric tree (Fig. 2.7 (c)) has, in addition to the properties of an additive tree, the same constant rate of evolution along each of its branches [31].

There are several different techniques used for constructing a phylogenetic tree. We can classify them as clustering approaches or search approaches, where the search

is for a tree that satisfies some optimality criterion. We can also classify them as distance-based approaches, where the sequence relationships are abstracted in terms of distances between them, or character-based approaches in which we deal with the actual sequences themselves.

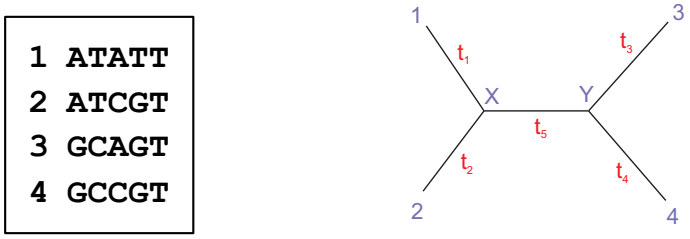
There are several well-established methods for inferring the phylogeny of a given set of taxa. Distance-based methods, such as UPGMA [115] and neighbor-joining [109], build the phylogentic tree based on the pairwise distance between sequences. Maximum parsimony [41] is a tree-building algorithm used to find the tree which explains the relationships amongst the sequences with a minimal number of substitutions. There are many other more complex approaches to phylogeny construction, including maximum likelihood methods [37] and methods that employ Bayesian inference [60, 106].

One commonly used search algorithm for reconstructing a phylogeny, the one used in this work, is the maximum likelihood method [37]. The maximum likelihood (ML) method chooses the tree that makes the observed data the most probable evolutionary outcome. The likelihood is given by

$$L(\tau, \theta) = P(\text{Data}|\tau, \theta),$$

where τ denotes a tree topology and θ denotes a set of parameters in a model of evolution. That is, the likelihood is the conditional probability of obtaining the observed sequences given a tree topology and an evolutionary model.

We demonstrate the maximum likelihood algorithm through the following example. We begin with a multiple alignment of four observed sequences (Fig. 2.8(a)). In Fig. 2.8(b), each of the tree's four leaves (nodes labeled 1 to 4) represents an observed sequence. The internal nodes, labeled X and Y, represent unknown ancestral



(a) Multiple Sequence Alignment for Observed Sequences

(b) Tree Topology

Figure 2.8: **Beginning a Maximum Likelihood Phylogeny.**

sequences. Each branch in the tree topology will be considered separately, and the probability of base i mutating to base j in time t is denoted $P(i|j,t)$. The calculation of these probabilities is determined by the evolutionary model that is used.

In the ML method we are considering each base in the sequence, which means that we should calculate the likelihood at each individual position in the alignment ignoring sites with gaps. For a given sequence position (i.e. column in the MSA), the likelihood of bases z_X and z_Y occurring at internal nodes X and Y with tree topology T and branch lengths t_i is given by:

$$L = P(z_1, z_2, z_3, z_4, z_X, z_Y|T, t_1, t_2, t_3, t_4, t_5) \tag{2.5}$$

$$= P(z_1|z_X, t_1)P(z_2|z_X, t_2)P(z_Y|z_X, t_5)P(z_3|z_Y, t_3)P(z_4|z_Y, t_4). \tag{2.6}$$

Each branch is considered to evolve independently of the others, so the probabilities are multiplied.

For example, consider the first column in the MSA from Fig. 2.8(a) and suppose we want to calculate the likelihood of seeing these sequences assuming $z_X = A$ and

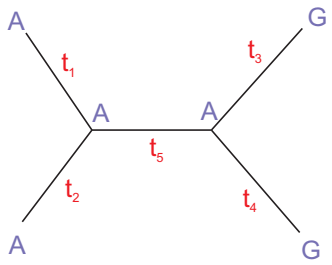


Figure 2.9: **Calculating the Likelihood of a Tree Topology.**

$z_Y = A$ as shown in Fig. 2.9. Then we have that the likelihood of this scenario is given by:

$$L = P(A|A, t_1)P(A|A, t_2)P(A|A, t_5)P(G|A, t_3)P(G|A, t_4).$$

Internal nodes X and Y could take on any possible base in the ith position, and this must be taken into account. Therefore the likelihood of observing column i is:

$$L_i = \sum_{z_X} P(z_1|z_X, t_1)P(z_2|z_X, t_2)(\sum_{z_Y} P(z_Y|z_X, t_5)P(z_3|z_Y, t_3)P(z_4|z_Y, t_4)).$$

Since we are assuming independence between columns in the MSA, the total likelihood, $\mathscr{L}$, is found by multiplying these probabilities together, $\mathscr{L} = \prod_i L_i$, or alternatively by taking logarithms $\ln \mathscr{L} = \sum_i \ln L_i$.

Now, for a given tree topology our goal is to find the maximum likelihood. This is done by optimizing the available parameters, including the branch lengths and possibly the parameters in the evolutionary model being used. The partial derivatives with respect to each of the parameters are derived and used in standard optimization routines to determine the parameter values that yield the maximum likelihood.

For all phylogenetic reconstructions, an estimate of the uncertainty of the result is often desirable. Bootstrap analysis is a simple and effective method that uses the original data set to generate estimates for the variability of the results. If we

knew the true distribution from which the data came, then we could calculate any relevant statistics. The key idea to bootstrap analysis is to assume that the data was sampled without bias from a real but unknown distribution. A new data set can be produced by sampling the original data set. If sampled correctly, this should also be an unbiased sample of the true distribution and can therefore be used to measure property variances. In bootstrap analysis, hundreds of new data sets, called bootstrap replicates, are generated in this fashion and their properties are analyzed to obtain estimates on the uncertainty of the results.

Bootstrapping phylogenetic trees was first introduced by Felsenstein [36]. Phylogenetic bootstrapping proceeds as follows. A bootstrap replicate is formed by randomly selecting columns from the original multiple sequence alignment (MSA), with replacement. For example, the first column of the bootstrap replicate might be the 17th column of the MSA, the second might be the 209th column, the third the 17th column, etc. Then the original tree-building algorithm is applied to the bootstrap data, giving a bootstrap tree. This whole process is independently repeated for each bootstrap replicate. Then, the consensus tree is constructed. That is, for each node, the proportion of bootstrap trees where this branch point occurred is calculated and indicated in the tree. "Agreeing" here refers to the topology of the tree and not to the length of its branches. Figure 2.10 shows a sample phylogenetic bootstrap calculation.

2.2.4 Gene Ontology (GO) Terms

As mentioned earlier, the key objective of this dissertation is to develop a game-theoretic approach to cluster multi-domain protein sequences. The aim of our method is to generate a network that groups proteins with similar or related functions. Analyzing the Gene Ontology (GO) annotation of proteins in the network is one way to

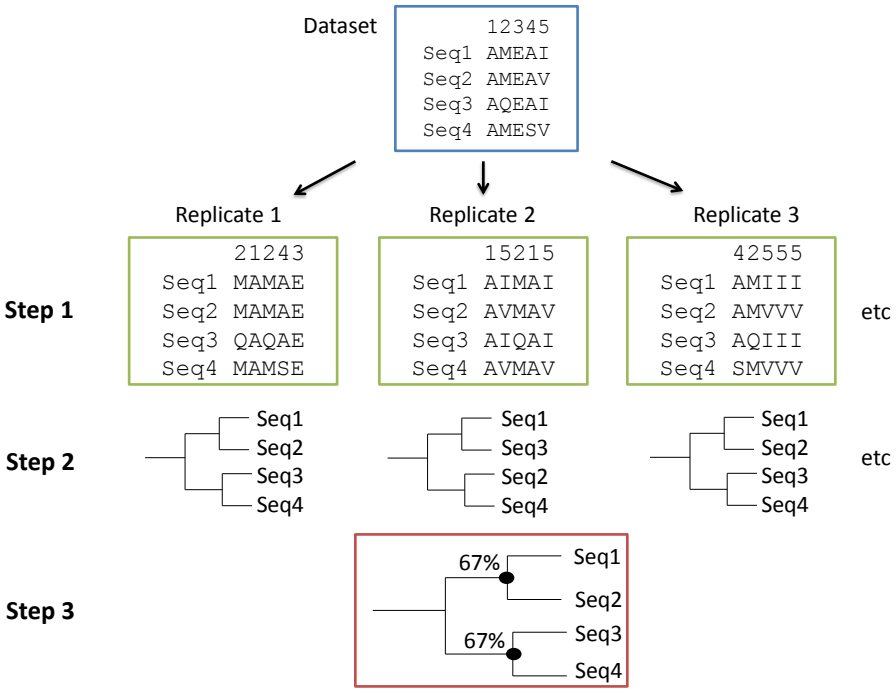


Figure 2.10: **Phylogenetic Bootstrap Analysis.** The original data (columns of the multiple sequence alignment) is sampled randomly, with replacement, to create multiple bootstrap replicates that have the same length as the original alignment (Step 1). The tree building algorithm is then used to construct a tree for each of the replicates Step 2). For each node in the original tree, the proportion of bootstrap trees whose clade at this node agree with the original tree are calculated (Step 3). This proportion is called the bootstrap value. Figure adapted from [7].

validate the functional relationships of the proteins.

The Gene Ontology (`www.geneontology.org`) [121] initiative was established in 1998 as a means of summarizing information about gene and gene product attributes consistently across different databases by using a common set of defined controlled vocabulary terms that are designed to be species neutral. The GO provides gene product properties, i.e. GO terms, in three different areas: cellular component (the parts of the cell or extracellular environment), biological process (operations related to the functioning of integrated living units), and molecular function (activities of the gene product at the molecular level). The GO also gives relationships between GO

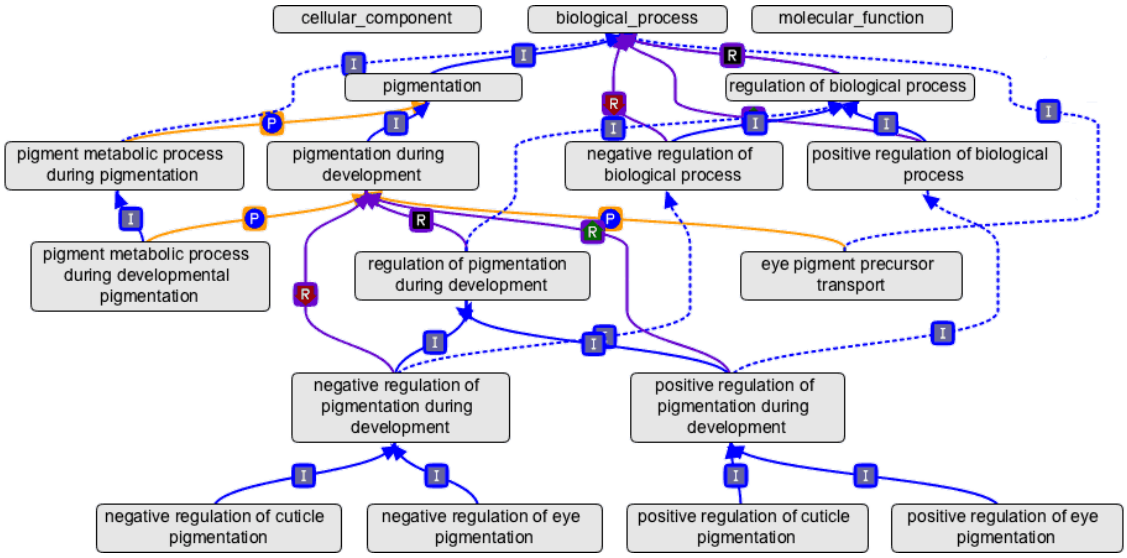


Figure 2.11: **Structure of the Gene Ontology.** This directed acyclic graph shows the relationships between a set of GO terms in the biological process component. Each node represents a GO term, and each edge gives the relationship between GO terms. The edge labels indicate the type of relationship represented by the edge, where R denotes a regulation relationship, I denotes an is_a relationship, and P denotes a part_of relationship. The GO terms become more specialized moving down the graph. Figure taken from `www.geneontology.org`.

terms.

The structure of the GO can be described as a directed acyclic graph, where the nodes represent individual GO terms, and the edges indicate relationships between the GO terms. The relationships between GO terms are loosely hierarchical, where a child term is more specialized than its parent term. For example, the GO term 'mitotic cell cycle' is a more specific type of cell cycle, and hence it is child of the GO term 'cell cycle.' However, the GO is not strictly hierarchical, as a child node may have more than one parent node.

Figure 2.11 shows the relationships between a set of GO terms in the biological

process component. Each node represents a GO term, and each edge gives the relationship between GO terms. Each edge in the graph represents a specific type of relationship. One type of relationship is the *regulates* relationship (denoted by R in Fig. 2.11), where one GO term is responsible for regulating another. As an example, in Fig. 2.11 we see an edge with a regulation relationship between the terms 'negative regulation of pigmentation during development' and 'pigmentation during development,' and it is quite clear from the names of the GO terms that one regulates the other. Other edge types include the *is_a* relationship (denoted I in Fig. 2.11), which means that one term is a subclass of the other, and *part_of* (denoted P in Fig. 2.11), which means that one term is necessarily part of the other, i.e. the presence of one term implies the presence of the other.

GO annotation is useful for both small-scale and large-scale analyses. It can help provide a first indication of the nature of a gene product. Many model organism databases and genome annotation groups use the GO and contribute their annotation sets to the GO resource. The GO database is continually evolving and being updated as new data becomes available.

Chapter 3

Related Work

This chapter describes a few currently used approaches to clustering multi-domain protein sequences: (a) phylogenetic profile methods (Section 3.1), (b) Markov clustering (Section 3.2), and (c) protein-domain biclustering methods (Section 3.3).

3.1 Phylogenetic Profile Methods

Originally, a phylogenetic profile was proposed as a vector whose entries quantify the existence of a protein in a genome. Phylogenetic profile methods [66, 95, 105] use these vectors to infer evolutionary relationships between genomes. These methods are independent of multiple sequence alignments, which are essential for classic phylogenetic inference. Inference of phylogenetic relationships using pairwise alignment is nearly impossible when the pairwise sequence identity falls below 25% identity, that is when sequences fall into the sequence identity "twilight zone" [29, 99, 125]. Phylogenetic profile methods [12, 21, 57, 58] have also been used to infer evolutionary relationships between highly divergent and/or rapidly evolving proteins.

The Gestalt Domain Detection Algorithm-Basic Local Alignment Tool (GDDA-

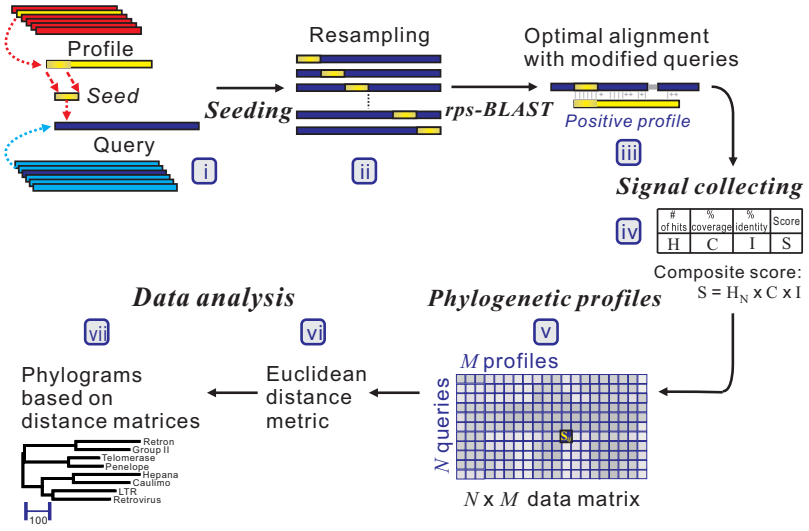


Figure 3.1: **Gestalt Domain Detection Algorithm-Basic Local Alignment Tool (GDDA-BLAST) Workflow.** Figure taken from [57].

BLAST) [21] uses a variation of a phylogenetic profile, where a protein is a vector whose entries quantify the existence of an alignment to a domain profile. Figure 3.1 shows the workflow for GDDA-BLAST. The first step is to compile the set of domain sequence profiles, i.e. position-specific scoring matrices (PSSMs) [3], to which the query sequence will be compared. PSSMs for domains can be obtained from any domain database, such as Pfam [103], SMART [74], and National Center for Biotechnology Information Conserved Domain Database (CDD) [80], or even generated locally using PSI-BLAST [3].

The GDDA-BLAST algorithm begins by modifying the query sequence to produce a set of seeded query sequences. Since BLAST algorithms are based on determining a hit and then extending the hit (see Section 2.3.1), a portion of the N-terminus (the start of the peptide sequence) or C-terminus (the end of the peptide sequence) region of each of the PSSM consensus sequences, called a "seed" region, is inserted

into the query sequence in order to establish a consistent initiation site. This step is necessary to allow the BLAST algorithm to extend alignments between highly divergent sequences. The seed sequence is inserted at each amino acid position in the query sequence to produce one seeded query. Thus for a data set consisting of m PSSMs, a query sequence composed of n amino acids will have $n \times m$ seeded query sequences.

Once the query sequence has been seeded, reverse specific position BLAST (rps-BLAST) [3] is used to align the each of the query sequences to the parent PSSM. Each optimal pairwise alignment is evaluated and recorded as a hit if it satisfied the given domain coverage and sequence identity thresholds.

Next, for a pair of the query and a profile a composite score is calculated by taking the product of the normalized hit number, mean percent identity, and mean percent coverage from the rps-BLAST results. A composite score of 0 indicates no significant hit, while a positive score measures the degree of successful matching between the query and the profile. The vectors of composite scores are used to generate an $N \times M$ data matrix, where N denotes the total number of queries (proteins) and M the total number of profiles. The data in this matrix is then used to compute the Euclidean distance between the vector representations of the proteins, and these distances are used to construct the phylogeny.

Although GDDA-BLAST has shown potential to reconstruct phylogenetic relationships for highly diverse proteins, its application is limited due to its high computational cost. To address this issue, the authors of GDDA-BLAST introduced Adaptive GDDA-BLAST (Ada-BLAST) [58], a sequence alignment algorithm similar to and as sensitive as GDDA-BLAST but much faster in computation. In fact, Ada-BLAST was shown to be 19 times faster than the original GDDA-BLAST. The key difference in these methods is that instead of inserting a seed sequence at each amino

acid position, as was done in GDDA-BLAST, Ada-BLAST inserts a seed sequence into the query sequence only in positions where the seed is likely to be extended to an alignment [58]. Although Ada-BLAST is much faster than its predecessor GDDA-BLAST, its scalability is still questionable, as it has only been tested on sequence sets of < 1000 proteins.

3.2 Markov Clustering

The Markov clustering (MCL) algorithm is an unsupervised graph clustering technique that is based on simulating stochastic flow in a graph [124]. TRIBE-MCL [35] is a direct application of the MCL algorithm used to cluster protein sequences into families. The TRIBE-MCL algorithm begins with an all-against-all BLASTp protein similarity search. The pairwise BLAST E-values are stored in a square matrix, which serves as an adjacency matrix for a weighted protein similarity network. An example of such a network is shown in Fig. 3.2(A).

A symmetric, weighted transition matrix is then generated using negative log-transforms of the BLAST E-values ($-\log(E)$), and the entries in the matrix are rescaled to produce a column stochastic transition matrix (Fig. 3.2(B)). A column stochastic matrix is a matrix whose entries are non-negative and whose columns sum to 1. The columns in the Markov matrix give the probabilities of transitioning from one protein to another within the network graph. This Markov matrix serves as the input for the MCL algorithm.

The MCL algorithm finds the cluster structure in a network using a bootstrapping process. Random walks within the graph are simulated using two operations, expansion and inflation. The expansion operator simply squares the Markov matrix using normal matrix multiplication. Inflation corresponds to taking the Hadamard power

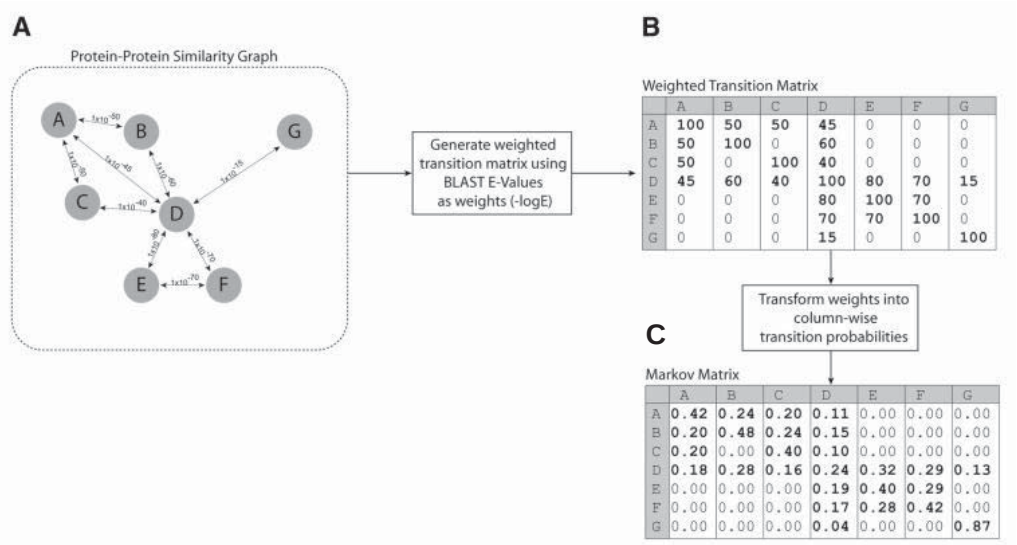


Figure 3.2: **TRIBE-MCL Data Processing.** This figure illustrates the initial processing of the data in the TRIBE-MCL algorithm. (A) An example of a protein-protein similarity network for seven proteins. Each node represents a protein, and a weighted edge between two nodes gives the BLAST E-value. (B) The weighted transition matrix and associated column stochastic matrix for the seven proteins in the network. The Markov matrix serves as the input for the MCL algorithm. Figure taken from [35].

of a matrix (i.e. taking powers entrywise) and then rescaling to ensure the resulting matrix is column stochastic.

The expansion operator coincides with computing "higher length" walks between node pairs in the graph. Intuitively, higher length walks between node pairs in the graph will be more likely to occur within dense clusters than across clusters, since there will be a larger number of ways of moving from one node to another within a cluster. Thus, using the language of stochastic flow, expansion scatters the flow within clusters, dissipating the flow between clusters. The inflation operator, on the other hand, changes the probabilities of single-step random walks departing from a

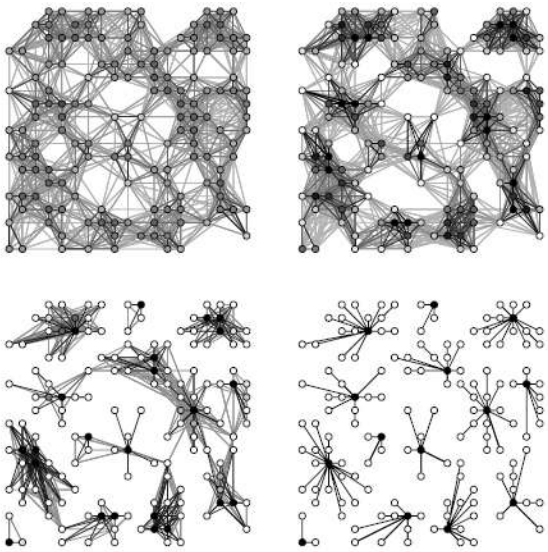


Figure 3.3: **Successive Stages of Flow Simulation by MCL.** Figure taken from [124].

particular node by favoring more probable walks over less probable walks. That is, inflation will boost the probability of intra-cluster walks. Successive stages of the MCL algorithm are exhibited in Fig. 3.3.

Iterating the expansion and inflation operators will eventually result in separating the graph into clusters. An equilibrium is reached when no change is observed in the matrix after applying inflation and expansion. With respect to convergence, it can be proven that the process simulated by the algorithm converges quadratically around its equilibrium states [35]. Global convergence has not been proven, but it is conjectured that the algorithm will always converge if the input matrix is symmetric [124].

TRIBE-MCL has been shown to detect protein families accurately for large-scale datasets [35]. However, as is the case in phylogenetic tree clustering, TRIBE-MCL does not incorporate information from all domains in a multi-domain protein. Phylogenetic clustering relies on aligning a region that is shared across all proteins in the data set. As explained above, the TRIBE-MCL method uses only the most significant

local region between two proteins to define the pairwise similarity. Therefore in these two approaches important information in the domain architecture is being excluded during the clustering process.

3.3 Protein-Domain Biclustering

Biclustering, also known as co-clustering, bidimensional clustering, and subspace clustering, refers to a distinct class of clustering algorithms that simultaneously clusters row and column data in a matrix [78]. One of the earliest applications of biclustering to biological data was the method of Cheng and Church [23], developed for gene expression analysis. As is the case with many other early biclustering algorithms, including Samba [120], the Order Preserving Submatrix Algorithm [11], and xMotif [85], the method of Cheng and Church is based on a greedy search algorithm. The shared feature of these approaches is that the data is clustered according to some real-valued attribute type that relates the two types of data in a bicluster. Binary Inclusion-Maximal Biclustering (Bimax) [101] is a simple divide-and-conquer based approach to biclustering that clusters a binary matrix into submatrices where the members of both the rows and columns are connected to each other. Speaking in terms of graph theory, the Bimax algorithm seeks to identify maximal bicliques in a bipartite graph.

Recently, the Bimax algorithm has been applied to cluster proteins in terms of their domain content and to construct a protein-domain bicluster network [111, 112]. The complete workflow for this method is shown in Fig. 3.4. First, the domain architecture for each protein is determined using HMMER profile hidden Markov model search. The results of the HMMER search are then filtered according to a specified E-value threshold. Then a binarized protein-domain matrix is constructed.

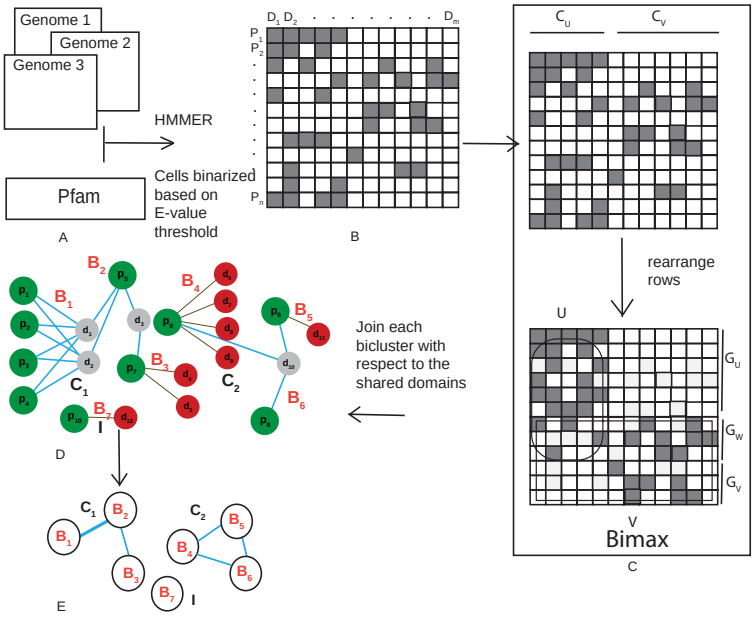


Figure 3.4: **Protein-Domain Biclustering Workflow.** Figure taken from [111].

In this matrix, an entry of 1 indicates the presence of the domain on the protein, and an entry of 0 indicates the domain's absence on the protein. This binary matrix then serves as the input to the Bimax algorithm (for details of the algorithm see [101, 111]).

Once the protein-domain inclusion maximal bicliques have been determined, these biclusters are converted to a network (Steps D and E in Fig. 3.4). First the biclusters generated from Bimax are connected based on their shared domains (Fig. 3.4(D)). These biclusters contain redundant proteins, i.e. proteins may be present in more than one bicluster. The biclusters are then refined with respect to their unique domain compositions by removing overlapping proteins from the bicluster that has the smallest domain set. This produces a set of biclusters, each containing a unique set of proteins. Finally the set of biclusters is converted to a network, where a node represents a bicluster and a weighted edge between two nodes indicates the number of domains shared between the biclusters (Fig. 3.4(E)).

The protein-domain biclustering approach can be easily applied to large-scale protein-domain analyses [112]. Unlike other clustering techniques, such as TRIBE-MCL, the biclustering method uses domain composition information from each protein. However, one drawback to this method is that the domain composition information is binarized prior to clustering. Thus the method is unable to make use of quantitative similarity information among domains on the proteins.

Chapter 4

Bioinformatic Game Theory

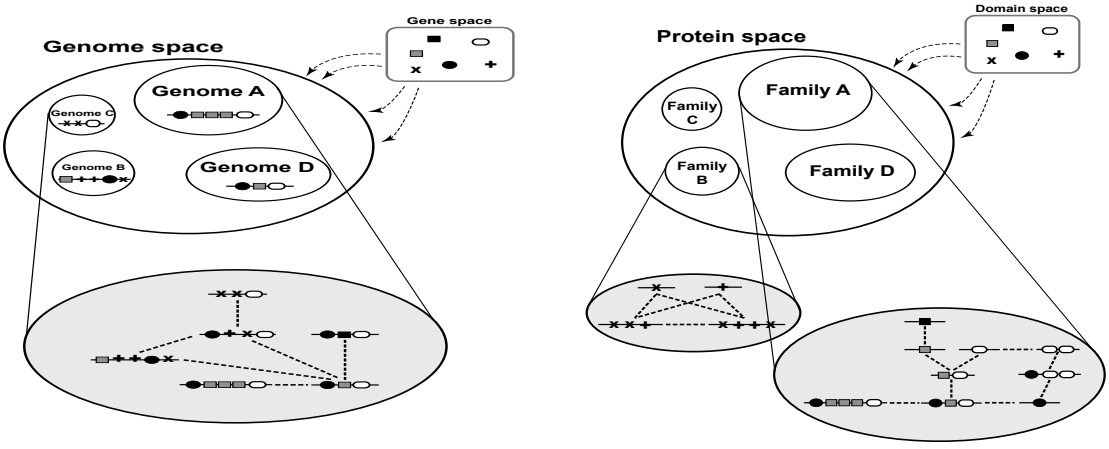
In this chapter we propose a game-theoretic approach to constructing biological networks. The key hypothesis is that evolution is driven by distinct mechanisms that seek to maximize two competing objectives, conservation to maintain diversity and diversity for adaptation. One branch of the mathematical theory of games is brought to bear. It translates this evolutionary game hypothesis into a mathematical model in two-player, zero-sum games, with the zero-sum assumption conforming to one of the fundamental constraints in nature in mass and energy conservation.

We begin by defining a game-theoretic similarity graph for a given set of entities. Next, we establish the game-theoretic approach to evolution and demonstrate how this can be used to formulate a linear programming problem for a given reference entity in the set, whose solution yields the set of evolutionary neighbors for the reference entity with respect to all other entities in the network. In addition we demonstrate why and how a mechanistic and localized adaptation to seek out greater information for conservation and diversity may always lead to a global Nash equilibrium in phylogenetic similarity.

4.1 Game-Theoretic Similarity Graphs

A entity space is defined by a set of biological entities, each of which is in turn defined by a set of components. For example, if the entities in the space are defined to be genomes the components will be genes (Fig. 4.1(a)) , while if proteins are the entities the components will be functional or structural domains (Fig. 4.1(b)). Given the entity space and the corresponding component space we will construct a similarity network.

We use a network definition similar to that proposed by Holloway and Beiko [56]. A *game-theoretic similarity graph*, $G = (V, E)$, for a given set of entities is a directed



(a) (b)

Figure 4.1: (a) **Genome/Gene Space and Genome Networks.** Each genome is composed of a set of genes (shown in the gene space). For a given set of genomes there exists a network graph. In this diagram the individual genomes are represented by a linear string of genes and their network edges are exhibited with dashed lines. (b) **Protein/Domain Space and Protein Networks.** Each protein is composed of a set of domains (shown in the domain space), and groups of proteins from the protein space form protein families (i.e. protein clusters). Within each protein family there exists a network graph. In this diagram the individual proteins are represented by a linear string of domains and their network edges are exhibited with dashed lines. Figure from [28].

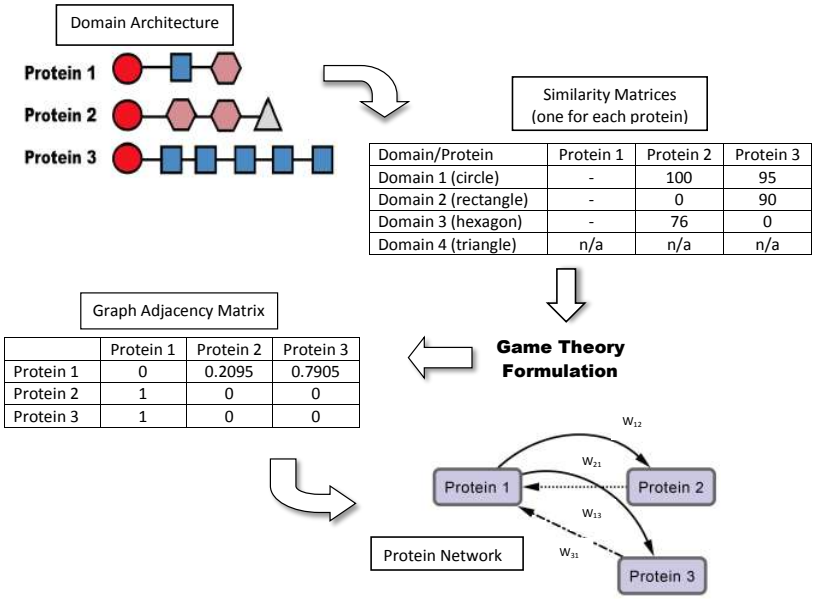


Figure 4.2: **Method Overview.** This diagram shows the game theory pipeline for constructing a protein similarity network. We begin with a set of domain architectures for the proteins in the protein space. Using these architectures, a similarity matrix is constructed for each of the proteins. The similarity matrix shown is the similarity matrix for Protein 1, which is indicated by the entries '-' for the domains that exist in Protein 1. The entries 'n/a' indicate that Domain 4 is not present in the reference protein, Protein 1. The similarity matrices serve as the payoff tables in the game theory optimization problems and lead to a graph adjacency matrix, which is used to construct the protein network. Figure from [28].

graph such that each vertex in the set $V = \{P_1, P_2, ..., P_n\}$ uniquely corresponds to one entity, and all edges have nonzero weights with the incoming edges to any given vertex summing to 1. (The distinction between whether the edges are incoming or outgoing is determined by the construction of the similarity matrix, to be discussed in more detail later.) An example of such a network can be seen in Fig. 4.2. An edge from vertex P_j to P_i is present if and only if the edge weight w_{ij} is strictly positive. The edge weight w_{ij} is a measure for the similarity of entity P_j to P_i relative to all other entities in V. No edge is drawn from any node to itself (i.e., $w_{ii} = 0$) by convention.

The network graph is constructed in an entity-by-entity approach. For each member of the entity space we construct a similarity matrix. In these matrices, we compare the amino acid (or nucleotide) sequences of each component as a first order approximation. Suppose there are a total of m components, $d_1, d_2, ..., d_m$, found in the n members of V. Then, as exhibited in Fig. 4.3, the similarity matrix, $A_i = a_i(s, j)$, of a given reference member P_i is an $m \times n$ matrix, where $a_i(s, j)$ is the similarity score of component s in member P_i to member P_j. This entry may be considered as a proxy for the mutual information between P_i and P_j with respect to component s in that the higher the value the more similar the pair are in component s. The values in the reference column (the i^{th} column) will not be used in calculation and are therefore marked as '-'.

As mentioned above, the edge directionality in the network graph depends on the construction of the similarity matrix. If the scores are established using the reference entity, P_i, as the intended parent sequence, the edges with nonzero weights will be the outgoing edges of the corresponding node, representing a likely ancestor-descendent directionality. Similarly, if the scores are constructed to permit the inference that the reference entity is a descendent sequence, the edges with nonzero weights will be the incoming edges to the reference node. If there is no obvious parent-offspring directionality, as is the case in the similarity matrix construction for our analysis, either convention may be used but only to keep track of the model solutions.

Once the edge weights are found for all P_i, they in turn define the network matrix $\mathbf{W} = [w_{ij}]$ as shown in Fig. 4.3. The directed network graph is then constructed according to the weight matrix $\mathbf{W}$. Furthermore if the matrix is block diagonalizable as explained in Fig. 4.3, then each (irreducible) block defines a distinct subnetwork graph, a cluster. Therefore, the construction of any directed similarity network graph is reduced to finding the weight vector $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{in})$ for node i in a node-

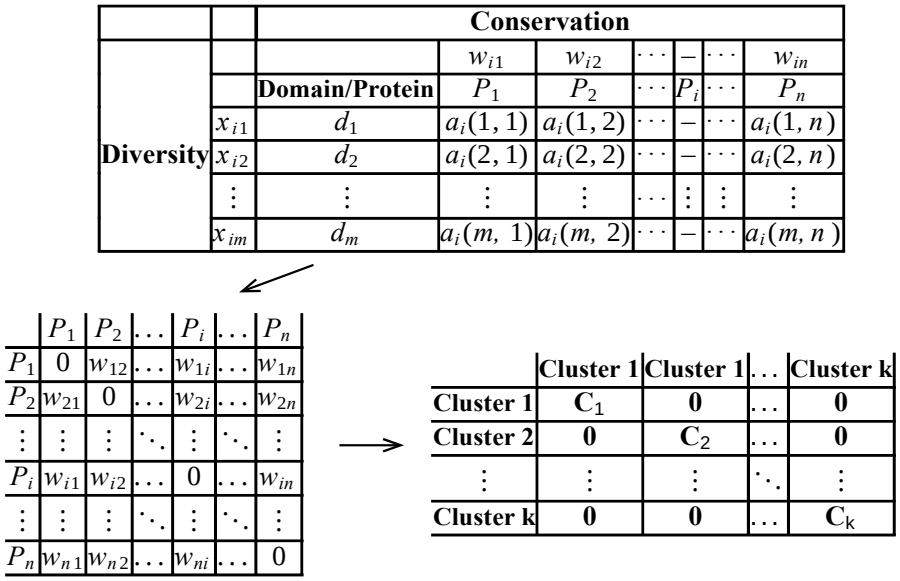


Figure 4.3: **Similarity Matrix.** The similarity matrix $A_i = [a_i(s,j)]$ (top) for node P_i that generates P_i's incoming edge weights w_{ij} which in turn generates the $n \times n$ network matrix $\mathbf{W} = [w_{ij}]$ (lower left) in the original ordering of the nodes P_i. (The weight row vector $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, 0, \ldots, w_{in})$ is the $\mathbf{y}$ probability solution vector from the game theory minimax problem (4.1) for the node i.) The matrix on the right is the same network matrix $\mathbf{W}$ except for the procedure that reorders the nodes $P_1, P_2, \ldots, P_n$ together with their rows and columns so that the resulting network matrix is a block diagonal in $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k$, with the other blocks being zero matrices $\mathbf{0}$. Figure from [28].

by-node basis for each $i = 1, 2, ..., n$.

4.2 Evolution As A Game

The construction of the edge weights w_{ij} is based on the assumption that during evolution, sequence conservation and component architecture diversity are both maximized. We will view the interwoven relationships of entities as the result of evolutionary processes such as mutations, insertions/deletions, and domain transfers, all taking place amongst thousands of individual organisms contemporaneously in space and repeated for thousands of generations, and all driven by some particular selective

forces. We will also view the net effect of the processes as a two-player game in which one player, or one force, is to maximize the genetic (sequence) conservation so that deleterious changes are eliminated and successful (or non-deleterious) structures and functions are passed on from one generation to the next; and the other player, or the other force, is to maintain the genetic diversity and to maximize evolutionary resources where novel structures (e.g., domain architectures) and functions can be tried out. The outcome is a set of entity families that tends to increase organism-level fitness.

We will assume, for the purpose of being a primary approximation, that the two goals are polar opposites because conservation as characterized by structural and functional similarity is negatively correlated with diversity which is characterized by the reverse. This first order approximation can also be justified by the principle of mass and energy conservation in nature. That is, genetic materials and natural resources that are devoted for conservation will not be available for divergence (or in population genetics, extinction vs. fixation). In short we will view each process event as one game play with the aim to maximize both conservation and divergence simultaneously. The net effect of this repeating game of evolution is the set of clusters of entities with similar component architectures in the entity space, and this effect is to be captured by the frequencies with which various strategies in the evolutionary game are played.

The goal of our game-theoretic model for evolution is, for each entity in the entity space, to find a Nash equilibrium [77, 87, 88] of the expected payoff for similarity, $E_i = \sum_{s \in S_i} \sum_j a_i(s,j) x_s y_j$, by determining the similarity probability vector $\mathbf{y} = (y_1, y_2, ..., y_n)$ to maximize E_i while simultaneously minimizing it (i.e., maximizing the diversity) by finding the novelty probability vector $\mathbf{x} = (x_1, x_2, ..., x_m)$. Here S_i denotes the subset of the m components that are present in the reference entity P_i.

The conservation-diversification dichotomy interpretation for the $\mathbf{y}$ and $\mathbf{x}$ probability vectors can be explained as follows. In the case of a pure 'diversity' or 'component' strategy being played, say $x_s = 1$, $x_t = 0$ for $s \neq t$ and $s \in S_i$, that is, when component s of the reference member P_i is used to measure divergence it is the entity (or entities) P_j having the largest similarity score $a_i(s, j)$ that should be picked as the countering 'conservation' or 'entity' strategy to maximize the similarity score E_i. Namely, for a constituent component whichever entity is closest to the reference entity is picked to conserve. Thus $y_j > 0$ for these j, and the y_j sum to 1. This gives the conservation interpretation of the $\mathbf{y}$ solutions.

On the other hand, in the case of a pure 'conservation' or 'entity' strategy being played, say $y_j = 1$, $y_k = 0$ for $k \neq i, j$ and $j \neq i$, that is, when P_j is used to measure similarity to the reference P_i it is the component(s) having the smallest similarity score $a_i(s, j)$ that stands out and should be picked as the countering 'diversification', or 'component' strategy to minimize the similarity score E_i. Namely, for a conserved entity whichever component is the least similar to the reference entity is picked to distinguish between the two entities. That is, these x_s are positive and sum to 1, and permit the interpretation for divergence.

As mentioned before, since all evolutionary processes – all kinds of genetic transfers or otherwise – take place amongst all organisms all the time, the evolutionary state we observe today would be the result of the frequencies with which all pure conservation strategies and all pure diversity strategies are played one event a time, and by our proposed game-theoretical model these frequencies are approximated by the solution $\mathbf{y}$ and $\mathbf{x}$ to the following minimax problem:

$$\begin{aligned} \min_{\mathbf{x}}\max_{\mathbf{y}} \quad & E_i = \sum_{s\in S_i}\sum_{j} a_i(s,j)x_s y_j \\ \text{subject to} \quad & \sum_j y_j = 1,\ y_j \geq 0,\ j = 1,2,...,n,\ y_i = 0 \\ & \sum_{s\in S_i} x_s = 1,\ x_s \geq 0,\ s \in S_i. \end{aligned} \tag{4.1}$$

The solution to this problem exists and is exactly a Nash equilibrium point (see Section 2.1). The optimal expected similarity score, E_i, is the so-called *game value*.

There are two different ways to find a Nash equilibrium (NE) for a two-player, zero-sum game. One is through a dynamical play of the game to find a NE asymptotically which is modeled by the Brown-von-Neumann-Nash (BNN) system of differential equations [15, 28, 54]. The other is by the simplex method in linear programming (see Section 2.1).

Here we present a mechanistic derivation of Nash's map. Nash used this map to prove the existence of NE for all non-cooperative games (Section 2.1). This derivation is extremely relevant to our game theory formulation for bioinformatics. It gives a plausible answer to the question how a NE is realized by nature. It shows that evolution or individual organisms need only be driven by their immediate, short term gain in game play payoff to reach a globally attractive Nash equilibrium. Here is an outline of the scenario, which works for any n-player game.

In the case that a game is played by large populations of all types of players repeatedly for a long period of time, the time between consecutive plays can be blurred to view the game as played continuously, and the play strategy frequency for player type-i, $x_i(t)$, changes continuously, where $x_i = (x_{1i}, x_{2i}, \ldots, x_{n_i i})^T$ is the mixed strategy probability or frequency vector, $\sum_j x_{ji} = 1, x_{ji} \geq 0$. Each $1 \leq j \leq n_i$ corresponds to the jth strategy of the type-i players, and x_{ji} can be interpreted to

be the fraction of the type-i player population that uses its pure strategy j. Consider it at time t and a Δt time later, $x_i(t), x_i(t+\Delta t)$. We would like to understand how $x_i(t+\Delta t)$ changes from $x_i(t)$. We will do so probabilistically.

Let α_i be the scalar *inertia* probability by which an individual of the type-i population plays the same strategy with probability $x_i(t)$ at time $t+\Delta t$ as at time t. Then, $1-\alpha_i$ is the non-inertia or *kinetic* probability with which an individual of the type-i population chooses or adapts to a particular strategy, including the choice of playing the same strategy at time $t+\Delta t$ as at time t because it is advantageous to do so, or because the individual organism is driven to do so due to selection. Let $\beta_i = (\beta_{1i}, \ldots, \beta_{n_i i})^T$ be the conditional play probability vector given that the play is kinetic, $\beta_{ji} \geq 0, \sum_j \beta_{ji} = 1$, then by elementary probability rules

$$x_i(t+\Delta t) = \alpha_i x_i(t) + (1-\alpha_i)\beta_i. \tag{4.2}$$

The scalar marginal probability α_i and the conditional probability vector β_i are derived as follows. First we assume the advantage for type-i player's kinetic strategy switch or adaptation depends on its total (scalar) excess payoff $\phi_i(x(t))$ from the current play frequency (Section 2.1, also [87, 88]), where $x = \{x_1, \ldots, x_n\}$ denotes the current play frequency for all player types with probability vector x_k for player type-k. That is, if $\phi_i(x(t)) = 0$, then all plays are of the inertia kind, $\alpha_i = 1$, and $x_i(t+\Delta t) = x_i(t)$ without adaptation. Next, for the $\phi_i(x(t)) > 0$ case, we assume the jth kinetic frequency is $\beta_{ji} = \varphi_{ji}(x(t))/\phi_i(x(t)) \geq 0$, where $\varphi_{ji}(x(t))$ (Section 2.1) is the jth strategy's excess payoff from the current play for the player type-i, $\varphi_i = (\varphi_{1i}, \ldots, \varphi_{n_i i})^T$. That is, the strategy switch to strategy j for the type-i players is strictly proportional to its excess payoff against the total $\phi_i(x(t)) = \sum_j \varphi_{ji}(x(t))$.

As for the scalar marginal inertial probability α_i, we assume it is a function of

the total excess payoff as well as the time increment Δt. Specifically, consider the probability equivalently in its reciprocal form $1/\alpha_i$, which represents all fractional possible choices for each inertia choice. The fractional possible choices automatically include the inertia choice itself so that $1/\alpha_i \geq 1$ always holds. Then at $\Delta t = 0$, we must have this trivial boundary condition $1/\alpha_i(x(t), 0) = 1$, the default inertia choice only for lack of time to adapt.

Assume the fractional possible choices increase linearly for a small time increment Δt, we have $1/\alpha_i = 1+r\Delta t$ where 1 represents the inertia choice itself and r represents the rate of increase in the kinetic choices, which may include the choice of maintaining the same strategy play, because of its excess payoff is positive, and all other kinetic strategy adaptations. We assume the rate of the kinetic choice change is proportional to the total excess payoff, $r = h\phi_i(x(t))$, i.e., the greater the excess payoff the more play switches in the population for a greater payoff gain. As a result, $1/\alpha_i = 1 + h\phi_i(x(t))\Delta t$ and equivalently, $\alpha_i = \frac{1}{1+h\phi_i(x(t))\Delta t}$.

With the functional forms for α_i and $\beta_i = \varphi_i/\phi_i$ above, we have

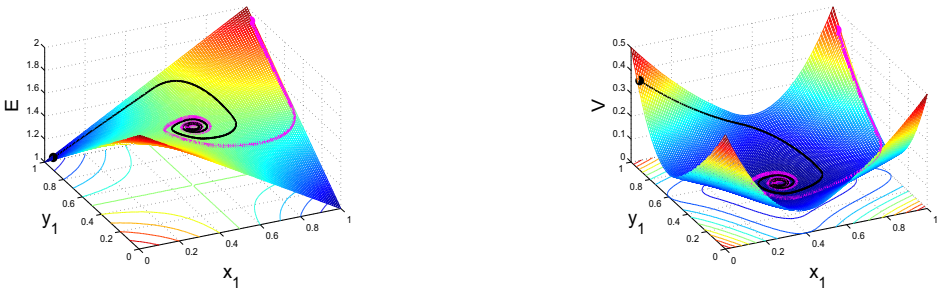
$$\begin{aligned} x_i(t+\Delta t) &= \alpha_i x_i(t) + (1-\alpha_i)\beta_i \\ &= \frac{x_i(t)}{1+h\phi_i(x(t))\Delta t} + \left(1 - \frac{1}{1+h\phi_i(x(t))\Delta t}\right)\beta_i \\ &= \frac{x_i(t) + h\varphi_i(x(t))\Delta t}{1+h\phi_i(x(t))\Delta t} \end{aligned} \tag{4.3}$$

which is exactly the Nash map [88] if $h = 1$ and $\Delta t = 1$. From (4.3) we also have

$$\lim_{\Delta t\to 0} \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} = \lim_{\Delta t\to 0} \frac{h(\varphi_i(x(t)) - \phi_i(x(t))x_i(t))}{1+h\phi_i(x(t))\Delta t} = h(\varphi_i(x(t)) - \phi_i(x(t))x_i(t))$$

and hence the following equivalent system of differential equations

$$\dot{x}_i(t) = \varphi_i(x(t)) - \phi_i(x(t))x_i(t) \qquad \text{for all} \qquad 1 \leq i \leq n \tag{4.4}$$



(a) (b)

Figure 4.4: **Two-Player Zero-Sum Game Dynamics.** An example of a prototypical two-player zero-sum game with the payoff matrix $A = [a_{ij}], a_{11} = a_{22} = 2, a_{12} = a_{21} = 1$, for player $\mathbf{y}$, for which $(x_1, 1 - x_1)$ is the mixed strategy for player $\mathbf{x}$ and $(y_1, 1 - y_1)$ is the mixed strategy for player $\mathbf{y}$. The trajectory starting near the point $(x_1, y_1) = (0, 1)$ is for the Nash map and that starting near $(1, 1)$ is for the BNN. Both are plotted on the expected payoff function E for player $\mathbf{y}$, (a), and on the total excess payoff potential function V, (b). Both converge to a NE which is the saddle point of the expected payoff function and the global minimum point of the excess potential function, respectively. Figure from [28].

after a time scaling by h. This type of equation was first introduced in [15] by Brown and von Neumann to compute a NE for symmetrical zero-sum games, and the derivation of (4.4) from the Nash map $x_i(t+\Delta t) = \frac{x_i(t)+\varphi_i(x(t))}{1+\phi_i(x(t))}$ was first noted in [54].

The derivation immediately suggests an evolutionary mechanism as to how a Nash equilibrium point *may* be realized or reached because the process or the game play is driven by the excess payoff at every step of the way, which can be interpreted as a mechanism for adaptation and a force of selection.

In fact, let

$$V(x(t)) = \frac{1}{2}\sum_{i,j}[\varphi_{ji}(x(t))]^2$$

define the total excess payoff potential. Then for any two-player zero-sum game,

it has been shown that $V(x(t))$ always decreases $\dot{V}(x(t)) < 0$ if $V(x(t)) > 0$, and $\lim_{t\to\infty} V(x(t)) \searrow 0$ for every solution $x(t)$ of the BNN equation (4.4) [28, 54]. A NE is reached when there is no excess payoff left, i.e. $V = 0$. It shows that as a global dissipative system, any mixed play frequency trajectory will *always* find a Nash equilibrium by following the down gradient of the excess potential function V. That is, in their search of greater excess payoffs, the total excess payoff for the players can never increase along any time evolution of their game plays.

A computational implication of this theorem is that a Nash equilibrium of any two-player, zero-sum game can always be found by iterating the Nash map or the solution to the BNN equations for any initial strategy frequency. This result solves the important problem as to how dynamical plays of a zero-sum game driven by individual players seeking out only localized advantages can eventually and collectively find a globally stable Nash equilibrium. Fig. 4.4 shows, for a prototypical two-player zero-sum game, the trajectories of the Nash map for a small time increment Δt and the BNN equation converge to a NE which is a saddle point on the payoff surface of one player and a global minimum on the excess potential surface, which can be viewed as a energy function for the dynamics of the BNN equation.

With the existence problem and the search problem for a NE solved, we can employ alternative and practical methods to find them. One standard procedure to solve the minimax problem (4.1) is to solve the following linear programming problem as reviewed in Section 2.1:

$$\begin{aligned} \max_{(\mathbf{y},v)} \quad & E_i = v \\ \text{subject to} \quad & \sum_j a_i(s,j) y_j \geq v, \text{for } s \in S_i \\ & \sum_j y_j = 1,\ y_j \geq 0,\ j = 1, 2, ..., n,\ y_i = 0. \end{aligned} \tag{4.5}$$

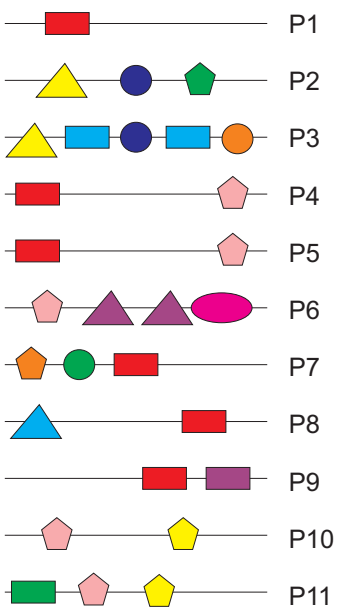
It is well-known that the $\mathbf{y}$ solution and the optimal value of the objective function $E_i = v$ for the LP problem (4.5) are exactly the $\mathbf{y}$ solution and the game value to the minimax problem (4.1), respectively, and the shadow price or the set of Lagrange multipliers for the LP problem is exactly the $\mathbf{x}$ solution to the minimax problem.

4.3 Similarity Network and Component Profile Construction

In the final step of our method we use the solution of (4.1) for each of the entities in the entity space to construct both the game-theoretic similarity network and the component profile. We demonstrate these constructions with an example of a protein space consisting of 11 proteins. Figure 4.5 shows the domain architecture for each of the proteins, which was found using HMMER [33]. There were a total of 15 domains identified on the proteins.

In the construction of the similarity network (Fig. 4.6(a)), the edge weight w_{ij} from P_j to P_i is assigned to be y_j from the Nash equilibrium of the minimax problem for node P_i. That is, the $\mathbf{y}$ solution, which obviously depends on i but the dependence is suppressed for simplicity, for each node P_i gives the ith row of the network matrix $\mathbf{W}$ of Fig. 4.3. The $\mathbf{x}$ solution vector for each i, is used to define the *domain profile* for the node P_i. Both the protein similarity network and the domain profile for the example above are shown in Fig. 4.6.

Thus, by our game-theoretic approach the edge weights of the network and the component profiles are the result of both conservation and diversity being maximized. More specifically, a high edge weight in the network indicates a strong similarity between entity pairs relative to the others, and a high row weight, x_s of node i,



(a)

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
P1	0	0	0	1.0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0.7761	0	0
P3	0	0.4224	0.2239	0	0	0.5776	0	0	0	0	0
P4	0	0	0	0	0.4520	0.5480	0	0	0	0	0
P5	0	0	0	0.5178	0	0.4822	0	0	0	0	0
P6	0	0.8953	0	0	0	0	0	0	0	0	0.1047
P7	0	0	0	0	0	1.0	0	0	0	0	0
P8	0	0	0	0	0	0.7076	0.2924	0	0	0	0
P9	0.2481	0	0	0.2481	0.2481	0	0	0.2558	0	0	0
P10	0	0	0	0	0	0	0	0	0	0	1.0
P11	0	0	0	0	0	0.8709	0	0	0	0.1291	0

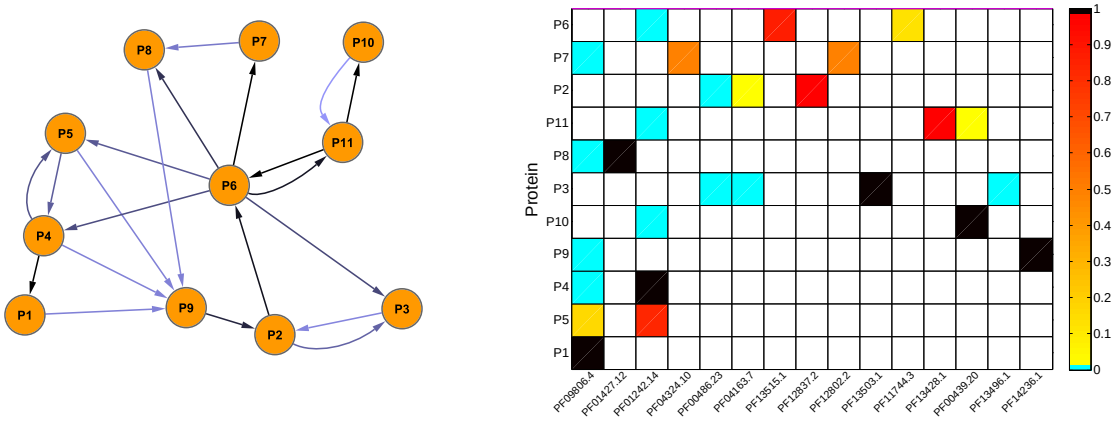
(b)

Figure 4.5: (a) **Domain Architectures.** This diagram shows the domain architectures for 11 proteins, labeled P1 to P11. The architectures were identified using HMMER profile hidden Markov search against the Pfam database. Each unique color/shape combination corresponds to one of the 15 domains that were identified in the search. (b) **Graph Adjacency Matrix.** The adjacency matrix for the game-theoretic protein similarity network generated using the approach described above. Each row gives the $\mathbf{y}$ probability solution vector from the game theory minimax problem (4.1) for the protein listed in the row's first entry.

indicates that the reference individual, P_i, is somewhat unique or dissimilar with respect to the component s compared to the other members of the entity space.

The game values also yield important information about the similarity network. For example, for two topologically identical clusters, it is their average game values that set them apart, which in this sense the cluster with the higher average game value is a 'tighter' or a more similar subnetwork than the latter.

The importance of a Nash equilibrium lies in the property that if we change the similarity frequency vector $\mathbf{y}$ from its optimal, then we may find a different diversity



(a) (b)

Figure 4.6: (a) **Protein Similarity Network.** In this graph, the proteins (nodes) are labeled from P1 to P11. The edges are directed so that the incoming edge weights of each node sum to 1. These edge weights are taken from the adjacency matrix shown in Fig. 4.5(b). The edge color indicates the edge weight, with darker edges indicating high weights and lighter edges indicating low weights. (b) **Domain Profile.** In this diagram, each row represents one of the proteins in the protein space, while each column represent a domain. The color bar on the right shows the color scheme for the diversity weights $\mathbf{x}$. A black shaded entry denotes a weight of 1 (or nearly 1), while a cyan shaded entry indicates that the domain was present in the protein but was assigned a weight of 0 for the optimal solution. A blank entry indicates the absence of a corresponding domain. The proteins are arranged according to their game value, with the largest appearing as the lowest row.

frequency vector $\mathbf{x}$ such that the corresponding expected similarity score is lower than the game value. Similarly, if we change the diversity frequency vector $\mathbf{x}$ from its optimal, we may then find a different similarity vector $\mathbf{y}$ so that the corresponding expected similarity score is larger. That is, deviating from the conservation optimal distribution may give rise to a greater diversification, and deviating from the diversity optimal distribution may give rise to a greater conservation. The dynamical state of the evolution, according to our model, is literally sitting at a saddle point of the expected similarity function; and the game value is a balanced tradeoff between reproducibility and diversity, a minimally guaranteed similarity.

Chapter 5

Protein Simulation

The true phylogenetic relationships between multi-domain proteins cannot be known with certainty. Computational methods for reconstructing these complex evolutionary histories use current data to infer past events. For this reason, testing and validation of these computational techniques is notoriously difficult.

There are two main categories for biological sequence simulation techniques. The first, which is used in population genetics, involves simulations that account for changes within and across entire populations which arise from models of sex, gene conversion, and recombination. Algorithms in this category have been developed to perform both forward [20, 53, 55, 96, 97] and backward [59, 110, 116] in time.

In phylogenetics and evolutionary biology, simulation involves single representatives of a species being linked by a tree and evolved forward in time. Various algorithms of this type exist, each differentiated by the model of evolution used to generate the simulated sequences. Early sequence simulation methods dealt only with substitution processes, supporting models of nucleotide and amino acid substitution as well as site-specific hetereogeneity [44, 104, 123, 131]. These programs, however, did not include indels, that is processes of insertion and deletion.

ROSE [117] was one of the earliest simulation algorithms to introduce indel events by extending the Gamma distribution model for site specific mutations to include indel constraints. Recently other simulation techniques have begun to include indel events, as well as other sequence simulation features such as sequence motifs and mutation rates that vary over time: EvolveAGene3 [49], DAWG [18], indel-Seq-Gen [118, 119], INDELible [42], MySSP [107], and SIMPROT [92].

Simulation programs with the goal of simulating complex genome evolution, such as gene duplication and lateral gene transfer exist [9, 26], but these simulators work at the genome level rather than at the protein level. The majority of simulation methods require the user to specify a guide tree for sequence evolution. We have developed a simulator that generates a protein family through probabilistic determination of evolutionary events. Moreover, in this simulation technique the evolution of each domain sequence is not constrained under a single guide tree, and insertion/deletion, duplication of domains, and domain swapping events can be freely incorporated.

5.1 Overview of the Simulation Process

A protein family is generated from a single root sequence and a set of domains as shown in Fig. 5.1. The ancestral sequence may be supplied by the user or generated randomly according to user specifications. The simulation starts by reading (or generating) the root protein, the domain locations on the root protein, and the given set of domain sequences. For the first generation, a randomly chosen number (a number between from 0 to K, a user-specified parameter) of child sequences are generated. Domain diversification, insertion, deletion, and duplication alter the content of each protein. Each child protein sequence incorporates an evolutionary event governed by a user-specified probability distribution, event location, and the sequence to be

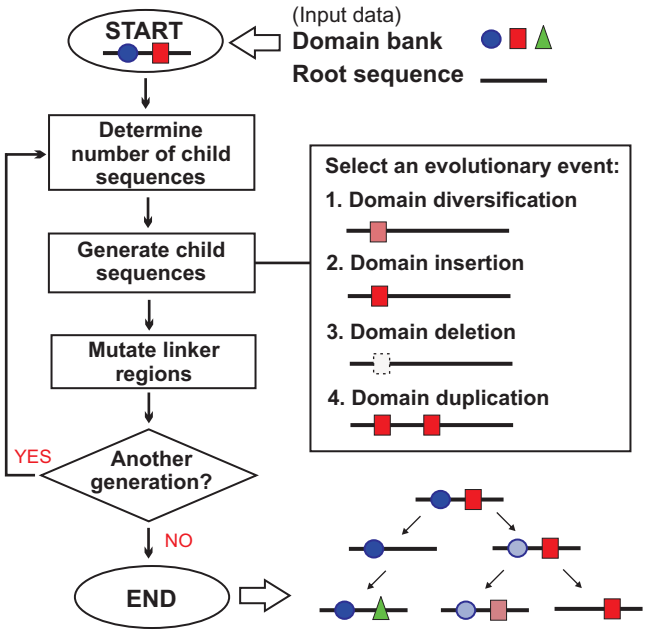


Figure 5.1: **Multi-domain Protein Simulation Algorithm Workflow.**

inserted (for a domain insertion event). After the chosen evolutionary event is incorporated, the domain linker regions (i.e. the regions between domains on the protein) are mutated using the JTT amino acid substitution model [63] and a mutation rate supplied by the user.

Once the first generation is completed, for subsequent generations the process described above is repeated, using each child sequence from the previous generation as a parent sequence (see Fig. 5.1). A generation is complete when this process has been carried out for each of the child sequences in the previous generation.

5.2 Evolutionary Events

Domain Diversification

In a domain diversification event the domain remains at the same location on the

sequence, and its amino acid sequence is mutated according to the JTT amino acid substitution model [63]. Once a domain has been mutated the resulting domain is then added to the existing domain bank, i.e. the set of domains available to be incorporated into insertion events.

Domain Insertion

For a domain insertion, the domain to be inserted is chosen uniformly at random from the domain bank, which includes the domain sequences initially supplied by the user as well as the mutated versions of these domains that were created during previous diversification events. Once a domain is chosen, the location of the insertion along the sequence is chosen uniformly at random. If a domain is to be inserted in a position that will cause an overlap with existing domains on the protein sequence, the overlapping domain(s) are removed before the new domain is inserted.

Domain Deletion

In the case of a domain deletion, the entire domain sequence, chosen uniformly at random, is removed from the protein.

Domain Duplication

For a domain duplication, the domain sequence, chosen uniformly at random, is duplicated and inserted along the sequence, resulting in two identical copies of the domain on the protein. The location for the insertion is chosen uniformly at random. If the duplicated domain is to be inserted in a position that will cause an overlap with existing domains on the protein sequence, the overlapping domain(s) are removed before the new domain is inserted.

5.3 Defining Protein Families

As mentioned above, in order to test the accuracy of our protein clustering algorithm, we need to have the exact evolutionary history of the proteins, and we need to define each unique protein family existing in the protein space. We define a protein family (or subfamily) to be a group of proteins that share unique functions. Since different domain architectures are likely to be associated to different functions, when a protein acquires sufficiently different domains compared to the ancestral form, it becomes a member of a new family.

During the simulation process, the entire history, including both sequences and evolutionary events, is saved. Therefore we can track the family-membership of each protein using domain content. In this study we define the occurrence of a new protein subfamily to be when the domain content of a protein differs by greater (strictly greater) than 50% from the root of the protein family to which it currently belongs. For example, in Fig. 5.2 proteins P4 and P9 define new subfamilies, as they each share less than 50% of their domains with protein P1. Note that 50% difference in domain content is an arbitrary choice for defining a new subfamily, and its validity needs to be evaluated in the future.

In order to cluster proteins and identify family structures from the simulated proteins, following the simulated phylogeny, starting from the root protein, we assign a length for each branch based on domain events that happened in the child protein. The three domain events, insertion, deletion, and duplication, alter the domain content of the protein and contribute more to the emergence of a new protein family. If a new subfamily is to be defined, we set the branch length to be 1.0, and if not the branch length is assigned to be 0.000001. Note that these numerical values are used only to obtain the cluster structure in the phylogeny.

In the case of a domain diversification event, the observed per-site mutation rate for the domain is used as the branch length between the ancestor and child sequences in the phylogenetic tree. The observed per-site mutation rate, μ, is calculated as follows:

$$\mu = \frac{\text{number of observed mutations}}{\text{length of domain sequence}} \tag{5.1}$$

From our simulated proteins, families can be identified using an agglomerative hierarchical method. We used the `cluster` function in MATLAB's Bioinformatics Toolbox [81] with the maximum within-cluster pairwise distance, $W_{\max}$, where the cluster splitting stops when $W_{\max}$ is drops below the given threshold. For example, if we cluster the leaf proteins (P5,P7,P8,P10,P11,P12) using the threshold $W_{\max} = 1$ we get the following four protein clusters: {P7,P8}, {P5}, {P11,P12}, {P10}. If instead we choose $W_{\max} = 2$ we obtain two clusters: {P7,P8,P5}, {P11,P12,P10}.

5.4 Program Output

Due to the probabilistic nature of this simulation technique, simulations using the same model parameters will yield different simulated sequence sets. Therefore it is important to track all information during the simulation process. The program outputs the entire simulation history, i.e. a complete tracking of the evolutionary events that occurred in the simulation at each step as well as the sequences of all proteins that arise. The program also produces a FASTA file containing the leaf sequences in the tree and a phylogenetic tree for the sequences in Newick-format (where the branch lengths are assigned by the process described above).

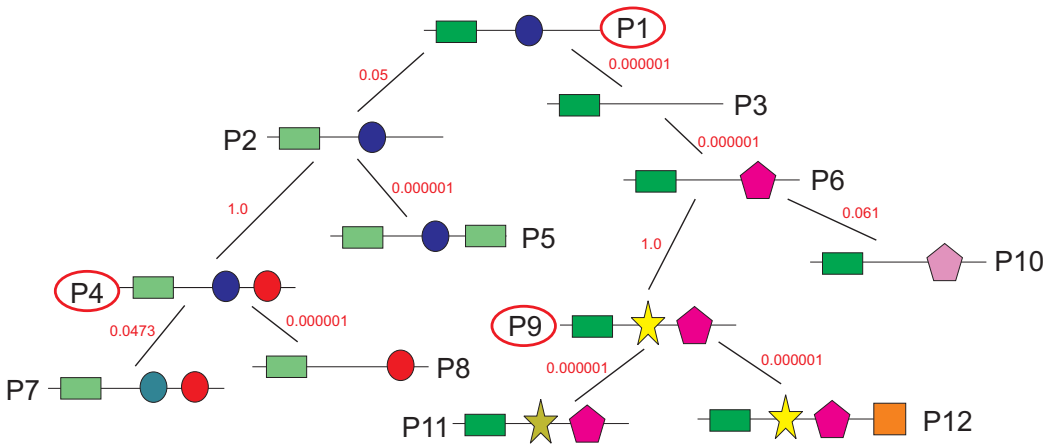


Figure 5.2: **Simulated Protein Families.** A set of six proteins (P5,P7,P8,P10,P11,P12) was generated using our simulation technique. This diagram shows the entire evolutionary history. The branch lengths are assigned as follows: in the case of a domain diversification (indicated by changing the shade of the domain's color) the branch length is the observed per-site mutation rate (Eq. 5.1), for domain insertion/deletion/duplication a branch length of 1.0 is assigned if a new protein subfamily occurred and 0.000001 otherwise. We define the occurrence of a new protein subfamily to be when the domain content of a protein differs by strictly greater than 50% from the root of the protein family to which it currently belongs. The circled nodes indicate the places where a new protein subfamily was defined during the simulation.

Chapter 6

Methods

In this chapter we describe the methodologies used in this dissertation for clustering a given set of multi-domain proteins. It describes: (a) an overview of the game theory method that produces the network clusters (Section 6.1), (b) domain identification (Section 6.2), (c) the structure of the protein-domain similarity matrix (Section 6.3), (d) the construction of the game-theoretic similarity network, (e) the data sets used in this study (Section 6.5), and (f) the evaluation of protein clusters (Section 6.6).

6.1 Workflow for Game-Theoretic Protein Clustering

A protein space is defined to be a set of proteins, each of which is in turn defined by a set of domains. Given the protein space and the corresponding domain space we can construct a similarity network that gives us a clustering of the proteins. As mentioned before, a *game-theoretic similarity graph*, $G = (V, E)$, for a given set of proteins, is a directed graph such that each vertex in the set $V = \{P_1, P_2, ..., P_n\}$ uniquely corresponds to one protein and all edges have nonzero weights with the

incoming edges to any given vertex summing to 1 (see Fig. 6.1).

The complete workflow for the game-theoretic clustering of proteins is shown in Fig. 6.1. The first step is to identify the domain architecture for each of the proteins. This can be done using any domain sequence search algorithm, for example profile hidden Markov model search. Next, for each of the proteins we construct a similarity matrix, which serves as the payoff table in a game theory minimax optimization problem. We solve each of these optimization problems and use the solutions to construct a graph adjacency matrix for the protein network. Each step in this process is described in greater detail in the sections that follow.

6.2 Domain Architecture Identification

6.2.1 Domain Identification Using HMMER

As mentioned before, the network graph is constructed in a protein-by-protein approach. We begin by first determining the domain architecture for each protein. This is done by using the `hmmscan` function of HMMER3 (version v3.1) [33] to search against the Pfam protein families database (release 27.0) [103]. The HMMER search was done using the following options:

```
hmmscan -o hmmer.out --acc --domtblout hmmer.tab Pfam-A.hmm RGS.fa
```

where "RGS.fa" is the file containing the proteins. The option "-o hmmer.out" specifies the name of the standard HMMER output file, "–acc" specifies the use of domain accessions rather than names in the output files, "–domtblout hmmer.out" specifies the output of a tabular file where each line summarizes the output of one domain hit on a query sequence, and "Pfam-A.hmm" is the Pfam database to be searched. The HMMER E-value [32] is used to measure the statistical significance of the align-

ments found in the pHMM search. The E-value is defined as the expected number of sequences in the database to have a score at least as high as a particular alignment score. In our analysis the results of the HMMER search are filtered to include only those domains whose E-value lies below the threshold 1.0.

6.2.2 Overlapping and Non-Overlapping Domain Predictions

It is possible to have domain regions identified by HMMER to be overlapped within a protein sequence. To examine how allowing overlapping domains in a protein affects the outcome of protein clustering, we examined three different overlap scenarios:

1. inclusion of all domains within the given E-value threshold, regardless of overlap;
2. inclusion of only those domains within the E-value threshold whose overlap is at most 5% of the protein length (for example if the protein has length 100 amino acids, then only those domains with overlap 5 amino acids or less are included. Note: if the overlap of two domains is greater than 5%, the domain with the lower E-value is kept); and
3. include only domains within the E-value threshold that are strictly non-overlapping (if two domains are overlapped, the domain with the lower E-value is kept).

Figure 6.2 shows the construction of each of these predictions for a given protein sequence.

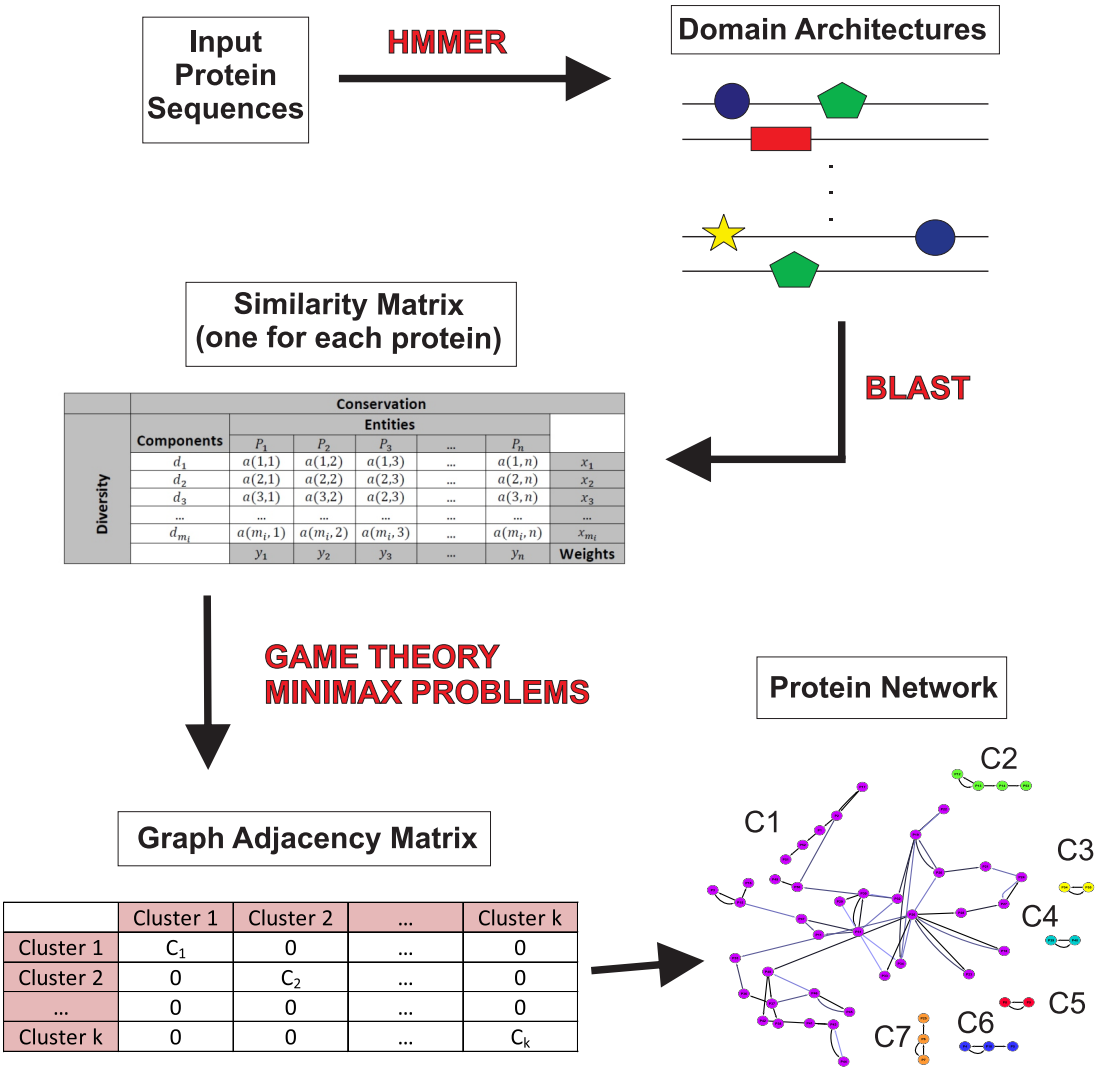


Figure 6.1: **Workflow For Constructing A Protein Similarity Network From A Set Of Protein Sequences.** First the domain architecture of each protein in the set is identified by a profile hidden Markov model search. Then for each of the proteins a similarity matrix is constructed, using a log-transformed BLAST E-value as the similarity score. This matrix serves as the payoff table in a game theory minimax optimization problem. Each of these optimization problems is solved, and the solutions are used to construct a graph adjacency matrix for the protein network. Block-diagonalizing the adjacency matrix gives the clusters in the network (each irreducible block defines a cluster).

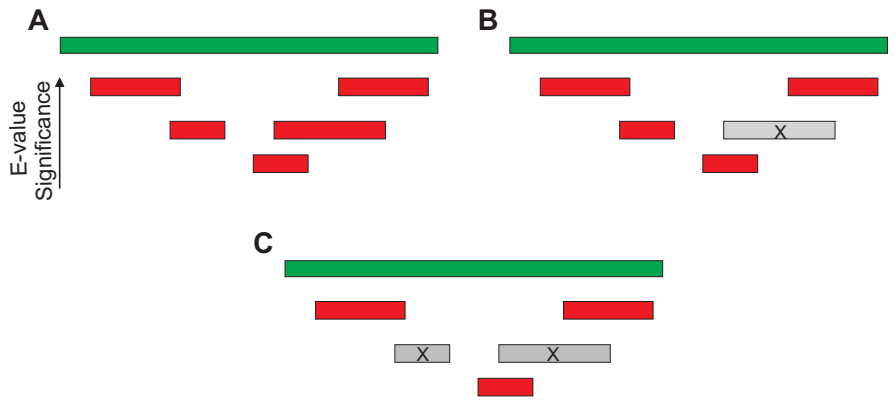


Figure 6.2: **Overlapping Domain Predictions. (A)** For a protein (green bar) several domains (red bars), with varying E-values (below some specified threshold) were predicted by HMMER. Some regions on the protein are predicted to contain more than one domain. The domains that share a region along the protein are called overlapping domains. This figure shows the case when all domains regardless of overlap are included in the prediction (included domains shown in red). **(B)** This figure shows the prediction for the case when only those domains whose overlap is at most 5% of the protein length are included (if two domains are overlapped by more than 5% the domain with the lower E-value is included). We see that in this case one of the domains (shown in gray) must be removed from the set of domains predicted by HMMER. **(C)** This figure shows the prediction for the case when only completely non-overlapping domains are to be included (if two domains overlapped the domain with the lower E-value is included). We see that in this case two of the domains (shown in gray) must be removed from the set of domains predicted by HMMER.

6.3 Similarity Matrix Construction

Once the domains have been identified, we begin the construction of the similarity matrices. In these matrices, we compare the amino acid sequences of the domains to the protein sequences using BLASTp. For a given reference protein, P_i, we extract the exact amino acid sequence for the top hit of each domain on the protein (the same domain could exist in multiple locations on the protein, but we use only the copy of the domain with the smallest E-value, i.e. the top hit). Suppose there are a total of m domains, $d_1, d_2, ..., d_m$, found on the proteins in the protein space, V. Then, as

exhibited in Fig. 4.3, the similarity matrix, $A_i = a_i(s, j)$, for P_i is an $m \times n$ matrix, where $a_i(s, j)$ is the similarity score of domain s from protein P_i to protein P_j, and n is the number of proteins in the protein space.

For each entry in the similarity matrix we used a log-transformed score of the BLAST E-value [3], $a_i(s, j) = -\log(\alpha_{sj})$, where α_{sj} is the E-value obtained for the domain query d_s against the subject protein P_j. The BLAST search was done using the following options:

```
blastp -query domains.fa -db RGS.fa -outfmt 6 -searchsp 53000 -out
blastreport.tab
```

where "domains.fa" is the query file containing all of the domain sequences, "RGS.fa" is the database file containing the protein sequences, and "blastreport.tab" is the name of the output file. The option "-outfmt 6" specifies that the output file is in tabular format and "-searchsp 53000" sets the size of the search space to be 53000 amino acids. Since the BLAST E-value is dependent on the size of the database and query sequence, setting a constant search space allows the BLAST E-value to be compared for data sets of different sizes.

The score used in our similarity matrix could be considered a proxy for the mutual information between proteins P_i and P_j with respect to domain s in that the higher the value the more similar the pair are in domain s. In fact, this log-transformed score has the dimension of information. If the base 2 is used, the score's dimension is exactly bit. The choice of base is not important, and here we will use the bit unit for the scores. The values in the reference column (the i^{th} column) are not used in calculation and are therefore marked as '-' as shown in Fig. 4.3.

6.4 Game-Theoretic Protein Similarity Network Construction

We construct the protein similarity network and domain profiles for the proteins using the game-theoretic algorithm described in Chapter 4. For each protein, P_i, in the protein space, its similarity matrix serves as the input (payoff table), $A_i = [a_i(s,j)]$, to the game theory optimization problem:

$$\begin{aligned} \min_{\mathbf{x}}\max_{\mathbf{y}} \quad & E_i = \sum_{s\in S_i}\sum_{j} a_i(s,j)x_s y_j \\ \text{subject to} \quad & \sum_j y_j = 1,\ y_j \geq 0,\ j = 1,2,...,n,\ y_i = 0 \\ & \sum_s x_s = 1,\ x_s \geq 0,\ s \in S_i. \end{aligned} \tag{6.1}$$

Here S_i denotes the set of domains present in the reference protein, P_i. As mentioned earlier, the solution to this problem, the Nash equilibrium, always exists (see Section 2.1.1).

Once the game theory optimization problem (6.1) has been solved for each of the proteins in the protein space, we use the solutions to construct the network graph. The edge weight w_{ij} from node P_j to P_i is assigned to be y_j from the solution to minimax problem for node P_i. That is, the $\mathbf{y}$ solution, which obviously depends on i but the dependence is suppressed for simplicity, for node P_i gives the i^{th} row of the network matrix $\mathbf{W}$ in Fig. 4.3. The $\mathbf{x}$ solution vector for each i is used to define the domain profile for the protein P_i. These constructions are explained in detail in Section 4.3.

Distributions	# Generations	Domain M.R. (%)	Linker M.R. (%)
Uniform	10	5,10,25,40	5,10,25,40
Uniform	15	5,10,25,40	5,10,25,40

Table 6.1: **Simulation Parameters.** Protein families were simulated using the varying combinations of model parameters, including the number of generations, domain mutation rate (Domain M.R.), and linker mutation rate (Linker M.R.). For each parameter combination 25 data sets were generated. All distributions in the simulation were taken to be uniform, and all simulations were generated from the same root protein sequence and the same initial bank of thirty Pfam domain sequences.

6.5 Data Sets Used in the Study

Both simulated and real protein data sets were used in this study. Simulated protein families were generated using the simulation algorithm described in Chapter 5. Simulations were performed using varying model parameters, and 25 simulated sets of each type were generated. All of the simulated sets were generated from the same root protein sequence and the same initial bank of thirty Pfam domain sequences (Table A.1). The root sequence was produced as follows. First, a random amino acid sequence of length 300 was generated and scanned using the sequence search tool on the Pfam website (`http://pfam.xfam.org/`) to ensure that no Pfam domains can be identified on the sequence. Next, three domain sequences from the domain bank, chosen randomly, were inserted into the sequence at locations that were chosen randomly such that the domains did not overlap. This resulted in the root protein sequence of length 649 containing three domains. Table 6.1 shows the parameter combinations used to generate the simulated data sets.

In addition to the simulated protein sets, two types of real protein data sets were used in this study. The Regulator of G-Protein Signaling (RGS) data set was comprised of 55 proteins from the mouse (*Mus musculus*) genome. RGS sequences were found by performing a profile hidden Markov model search in HMMER [33] using the Pfam [103] domains PF00615 (RGS) and PF09128 (RGS-like) as query sequences

and with an E-value threshold of 1.0. This RGS family sequence set was subsequently used to HMMER search against the Pfam database to find other domains present in the sequences. This step identifies the other domains that coexist with the RGS and RGS-like domains on the proteins.

Twenty six non-overlapping Pfam domains (including RGS and RGS-like) were identified on the proteins. The non-overlapping domain architecture predictions for each of the proteins is shown in Table A.10. Domain architecture predictions were also generated allowing only 5% overlap (architecture predictions shown in Table A.11) and unrestricted overlap (architecture prediction shown in Table A.12). When restricting the allowed overlap to 5%, 29 Pfam domains were identified, while 41 domains were identified when no restriction was placed on the overlap percentage. (See Section 6.2.2 for a description of the various overlap detection strategies).

The second real data set, a larger-scale set, consisted of 12,222 proteins from the mouse (*Mus musculus*) genome downloaded from the Swiss-Prot section of the UniProt Knowledgebase (UniProtKB) (`www.uniprot.org`). UniProtKB is a curated database, a central access point for integrated protein information with cross-references to multiple sources. The manually-curated Swiss-Prot sector of the database provides records for non-redundant, manually annotated protein sequences. That is, it contains only those sequences with information extracted from literature and curator-evaluated computational analysis. The mouse proteome (proteome ID UP000000589) was downloaded from the Swiss-Prot database (release 2014_08). This proteome consisted of 16,677 proteins. Our mouse family data set consists of the 12,222 of these proteins that had UniProt protein family annotation supplied. UniProt makes use of Interpro [5] to assign protein family membership. Interpro uses predictive signatures from various family and domain databases to assign proteins to families. For proteins that do not have any of these predictive signatures, UniProt employs sequence sim-

ilarity searches along with scientific literature to assign proteins to families. There are 10,914 protein families represented in UniProtKB.

6.6 Evaluation of Protein Clustering

6.6.1 Phylogenetic Clustering

As mentioned earlier, the reconstruction of a phylogeny for multi-domain proteins requires at least one domain sequence to be shared across all of the proteins. Therefore phylogenetic clustering could only be applied to the RGS data set in this study. A multiple alignment of the 55 RGS sequences was done using MAFFT (version v7.182 [65]). The MAFFT alignment was constructed using the L-INS-i algorithm with the default parameters:

```
mafft -linsi rgs_domains.fa > rgs_msa
```

where "rgs_domains.fa" is the FASTA file containing only the RGS domain for each of the proteins and "rgs_msa" is the output multiple sequence alignment.

The maximum likelihood phylogeny was reconstructed using PHYML (version v3.1 [47]) using the following options:

```
phyml -i rgs_msa -d aa -m LG -a e -b 1000
```

where "rgs_msa" is the file containing the multiple sequence alignment of RGS proteins. The option "-m LG" uses the LG amino acid substitution model [72], "-a e" specifies the gamma distribution shape parameter with the maximum likelihood estimate, and "-b 1000" specifies the bootstrap analysis with 1000 pseudoreplicates. Bootstrap values of 70% were used to define the clusters of RGS sequences. That is,

groups of sequences that have a bootstrap support of 70% or higher form a cluster, while the rest form single protein clusters.

6.6.2 Markov Clustering

The Markov Clustering (MCL) algorithm was used to generate protein clusters to be compared with the game-theoretic protein clusters. The MCL package (version v12-068) was downloaded from `www.micans.org/mcl`. The details of the TRIBE-MCL method were described in Section 3.2. The TRIBE-MCL process uses the following steps: (a) for a given set of proteins, an all-vs.-all BLAST hit table is generated using the `blastp` program, (b) the `mcxdeblast` application is used to parse the BLAST table and generate an all-vs.-all similarity matrix using an E-value threshold of 1.0, (c) the `mcxassemble` program creates a probability matrix from the similarity matrix, and (d) the probability matrix is used as the input to the `mcl` program, which generates the protein clusters. Clusters were obtained using varied values of the inflation parameter, including the default $I = 2.0$ as well as $I = 1.2, 3.0, 5.0$).

6.6.3 Protein-Domain Biclustering

The protein-domain biclustering network approach was used to construct clusters of proteins to be compared to the clusters arising from the application of our game theory method. A detailed description of the biclustering method is provided in Section 3.3. Using the principle of biclustering (co-clustering), proteins and domains are simultaneously clustered, giving rise to biclusters which each contain a subset of proteins and domains that form a complete bipartite graph. These protein-domain biclusters (with redundant proteins removed) served as the clusters to which we compared the results of our method. The biclustering pipeline from [111, 112] was used

to generate the biclusters for each of the data sets.

6.7 Cluster Comparison Metric

The average maximum Jaccard index [101] was used to compare the sets of clusters generated by two different methods. Given two sets of protein clusters, A and B, the average maximum Jaccard index for A against the alternative set of clusters B is given by:

$$J(A,B) = \frac{1}{|A|}\sum_{A_1 \in A} \max_{B_1 \in B} \frac{|A_1 \bigcap B_1|}{|A_1 \bigcup B_1|}$$

Chapter 7

Results

This chapter describes: (a) the application of the game theory network approach to simulated protein families and comparison of the resulting network clusters with the clusters obtained by other methods including TRIBE-MCL and protein-domain biclustering (Section 7.1), (b) the application of the game theory network approach to RGS family proteins and comparison of the RGS network clusters with the clusters obtained by other methods including TRIBE-MCL, protein-domain biclustering, and phylogenetic clustering (Section 7.2), (c) application of the game theory network approach to the Swiss-Prot mouse proteome and its evaluation (Section 7.3), and (d) analysis of the effect of overlapping and non-overlapping domain predictions on the game-theoretic clusters (Section 7.4).

7.1 Game-Theoretic Clustering of Simulated Proteins

The game theory approach was used to cluster several different sets of simulated proteins. As mentioned earlier, the simulated protein families were generated using

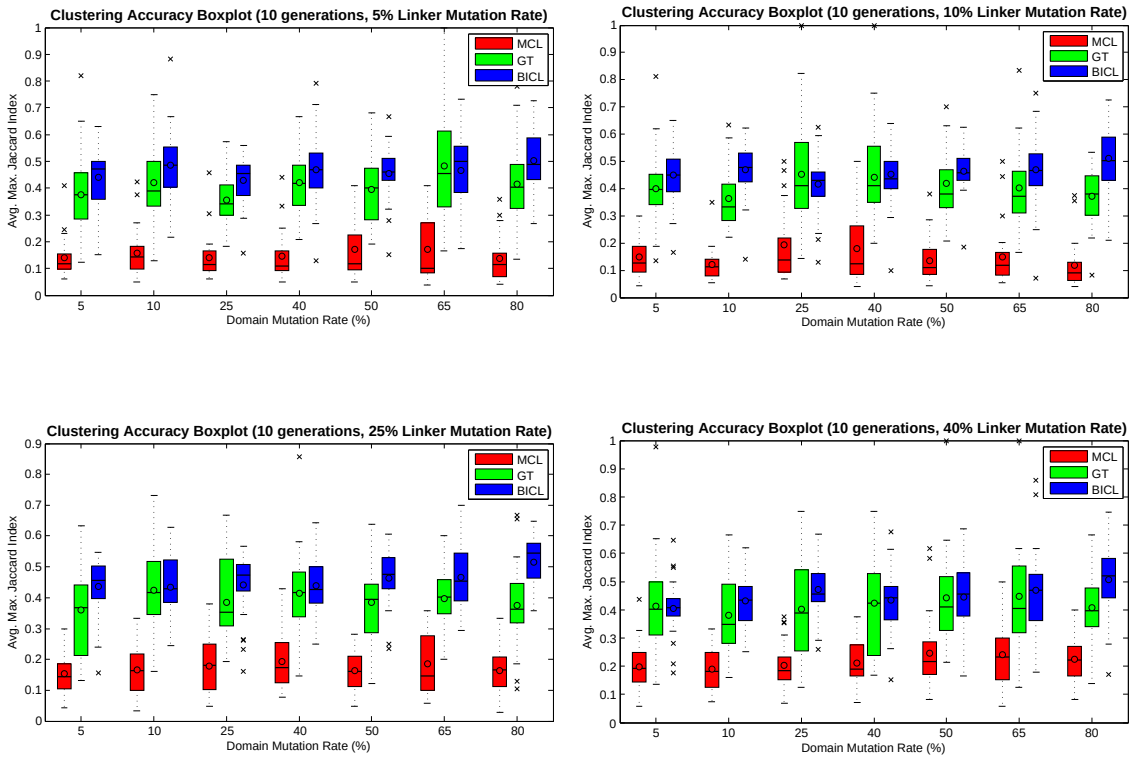


Figure 7.1: **Clustering Performance Among Game Theory, Biclustering, and TRIBE-MCL methods (10 Generations of Simulated Proteins).** For each linker mutation rate (% shown in the title of the figure) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). Clustering accuracy was assessed using the average maximum Jaccard index. Note that as described in Chapter 5, the domain mutation rate is the mutation rate used in each individual domain mutation event. Tables A.2-A.9 show information on the total domain sequence divergence for the simulated proteins.

the simulation algorithm described in Chapter 5. Simulations were performed using varying model parameters, and 25 simulated sets of each type were generated. All of the simulated sets were generated from the same root protein sequence and the same initial bank of thirty domain sequences. The various parameter combinations used to generate these data sets are shown in Table 6.1.

Using these data sets, clusters were obtained by the game theory (GT), biclus-

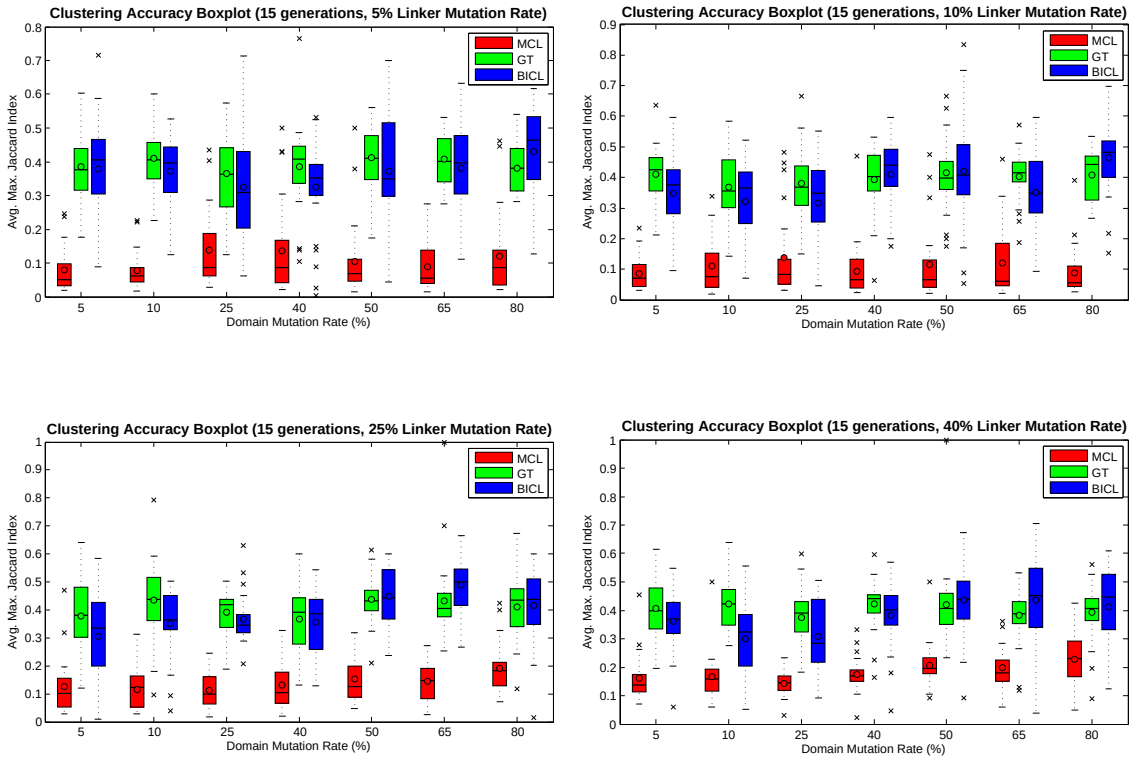


Figure 7.2: **Clustering Performance Among Game Theory, Biclustering, and TRIBE-MCL methods (15 Generations of Simulated Proteins).** For each linker mutation rate (% shown in the title of the figure) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). Clustering accuracy was assessed using the average maximum Jaccard index. Note that as described in Chapter 5, the domain mutation rate is the mutation rate used in each individual domain mutation event. Tables A.2-A.9 show information on the total domain sequence divergence for the simulated proteins.

tering (BICL), and TRIBE-MCL (MCL) methods (see Section 6.6) and compared to the true clusters of protein families, obtained by the process described in Section 5.3. Figures A.1-A.4 show examples of the game theory and biclustering networks for the simulated data sets. Clustering accuracy was assessed using the average maximum Jaccard index (see Section 6.7), a comparison metric whose values run from 0 to 1, with 0 indicating no proteins were clustered correctly and 1 indicating all

proteins were clustered correctly. Figures 7.1 and 7.2 show boxplots of the clustering accuracy results when 10 and 15 generations are simulated, respectively, using the various combinations of linker and domain mutation rates, while Figures 7.3 and 7.4 summarize these results using cluster variance heat maps. These four plots show the results when the default TRIBE-MCL inflation parameter, $I = 2.0$, is used. Figures A.5-A.10 show the results for other values of the inflation parameter.

In all cases TRIBE-MCL is outperformed by both the game theory and protein-domain biclustering methods. The TRIBE-MCL algorithm considers only the most significant local region between two proteins when defining the pairwise similarity. That is, the clustering of the proteins is based on only a limited amount of the similarity information that is available. On the other hand, the game theory and protein-domain biclustering approaches incorporate information from all domains on all of the proteins in the data set and therefore tend to provide a clearer distinction between protein families.

Figures 7.1-7.4 demonstrate that the results of the game theory method were more favorable when simulations were done with a larger number of generations (10 generations vs. 15 generations). That is, when the data sets become more complex (increased divergence in domain sequences, increased numbers of insertion/deletion events, etc.) the game theory approach tends to outperform both TRIBE-MCL and protein-domain biclustering with a lower variance. Figure 7.5 shows the complexity of the data sets in terms of the average number of predicted non-overlapping domains (predicted using HMMER). This supports the claim that the game theory clustering algorithm is a useful tool for detecting protein families even when the sequences in the data set contain a large number of domains and domain architectures.

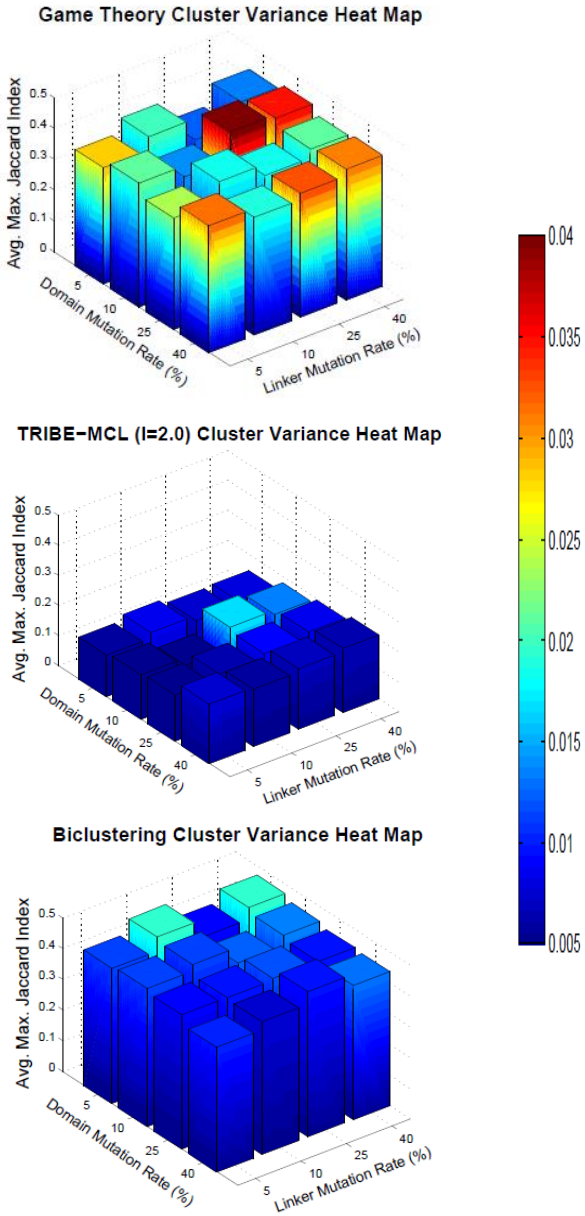


Figure 7.3: **Cluster Variance Heat Maps (10 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The default TRIBE-MCL inflation parameter, $I = 2.0$, was used. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

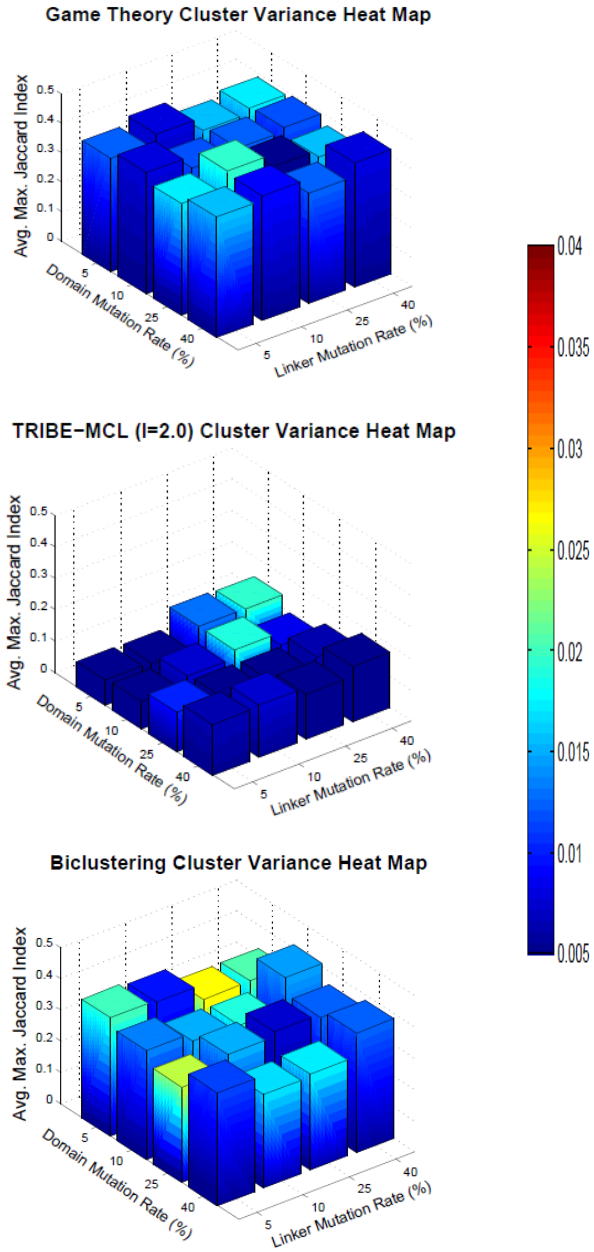


Figure 7.4: **Cluster Variance Heat Maps (15 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The default TRIBE-MCL inflation parameter, $I = 2.0$, was used. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

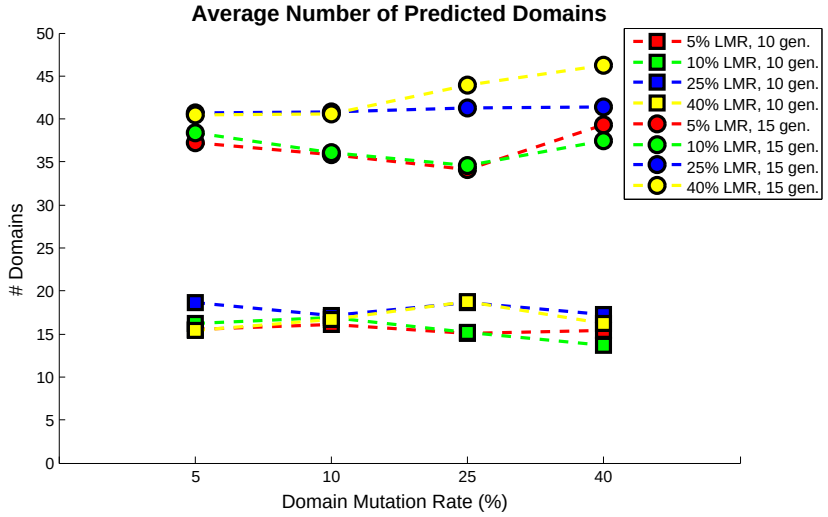


Figure 7.5: **Average Total Number of Domains Predicted on Simulated Protein Sets Using HMMER.** For each linker mutation rate (%, denoted LMR, shown in legend), domain mutation rate (% shown along the x-axis), and number of generations (shown in legend), 25 sets of proteins were simulated. This figure shows the average number of non-overlapping domains predicted by HMMER for the 25 data sets with the given mutation parameter values.

7.2 Game-Theoretic Clustering of RGS Family Proteins

The set of 55 Regulator of G-Protein Signaling (RGS) proteins from the mouse genome was clustered using the game-theoretic method. Twenty six Pfam domains were identified by HMMER using the non-overlap domain detection strategy. These domain composition predictions are shown in Table A.10. Nine clusters, shown in Fig. 7.6, were found using the game theory method. The clusters are labeled according to their average game values, in descending order. As mentioned before, the game values yield important information about the inter-cluster similarities in network. For example,

Cluster 2 and Cluster 5 each include three proteins (nodes) and are topologically identical. However, their average game values are 146.7074 and 71.9517, respectively, indicating that Cluster 2 is a tighter or more similar subnetwork than Cluster 5.

Even though all 55 proteins in this data set belong to the RGS family (they all contain either the RGS or RGS-like domain), the proteins in each of the network clusters differ from those in the other clusters with respect to their domain composition and varying levels of similarity between their domain sequences. The domain profile across the proteins in the RGS game-theoretic similarity network is exhibited in Fig. 7.7. The proteins are grouped according to the clusters in the network graph.

There are some clear profile pattern differences that exist between the clusters. For many of the clusters, specifically Clusters 1, 2, 7, 8, and 9, the proteins all contain the same domains, and the weights placed on these domains are the same. The profile also highlights domains that are unique to specific clusters. For example, Pfam domain PF00018.23 is hallmark to Cluster 8 because it is present in all members of Cluster 8 and not elsewhere.

As an initial validation of the network clusters, we provide the information for the proteins within each cluster in Table A.13. Figure A.11 clearly shows that different domain architectures are represented in different clusters. Sequence divergence within the same domain type (e.g., RGS domain for RGS 17 vs. RGS 19/20 proteins) is also recognized in separating Clusters 1 and 2. Furthermore we note that isoforms (proteins coded in alternatively spliced transcripts derived from the same gene) of the same gene fall into the same cluster even if some domains are missing in different isoforms as shown in the beta-adrenergic receptor kinase 2 isoforms 1 and 2 in Cluster 4. Therefore, using our game-theoretic framework, we incorporated both sequence diversity and domain information and produced a valid RGS protein network.

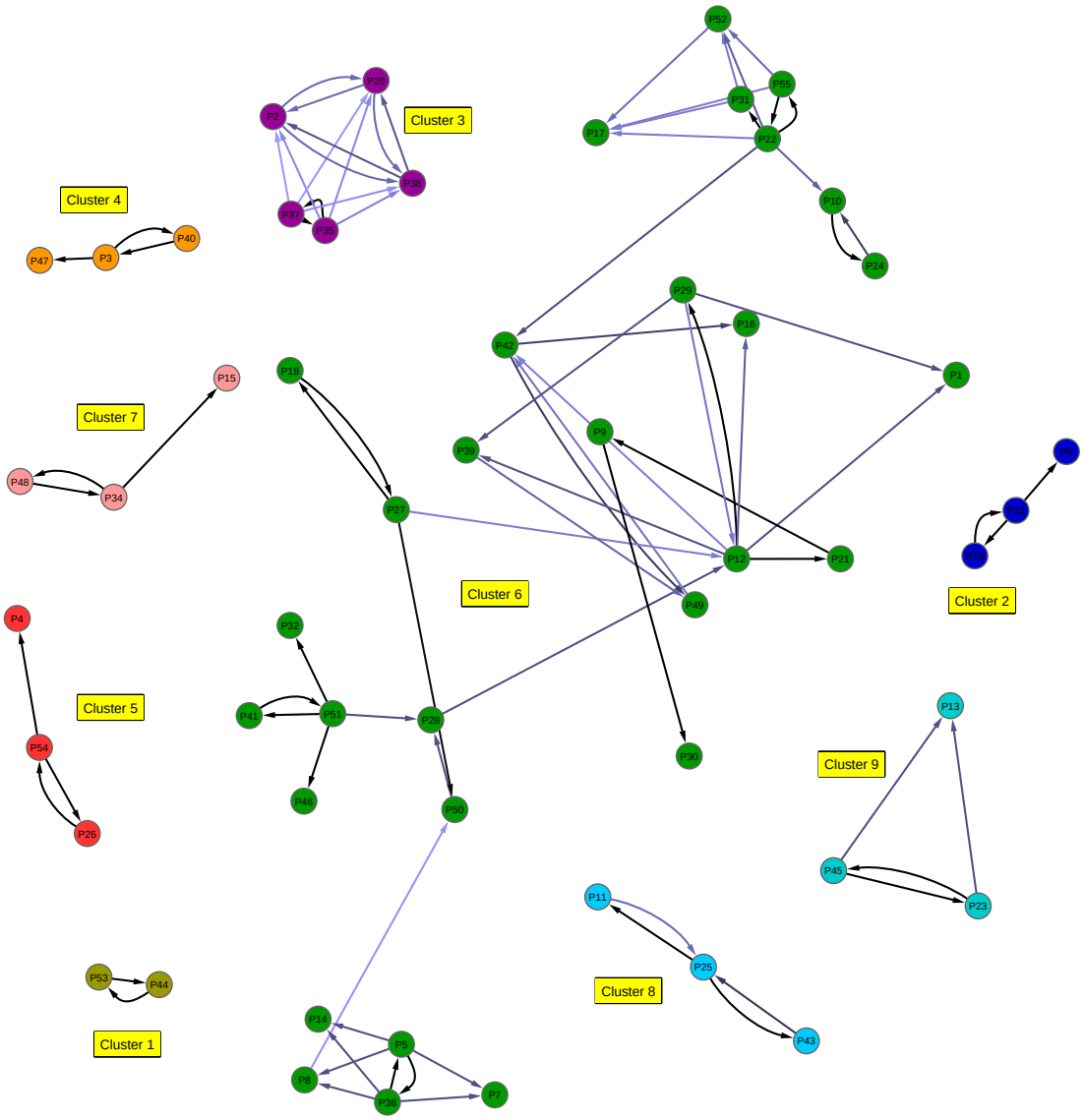


Figure 7.6: **RGS Family Game-Theoretic Protein Similarity Network (Non-Overlapping Domains).** From 55 mouse RGS family proteins, using the non-overlapping domain detection strategy, 9 clusters were identified using the game theory method. The nodes represent distinct proteins and the edges are directed so that the incoming edge weights of each node sum to 1. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. The game value for each protein is represented in the network by the length of its incoming edges, with longer edges corresponding to small game values and longer edges indicating high game values. Clusters in the network, represented by different node colors, are labeled in descending order according to their average game value, i.e. Cluster 1 has the largest average game value. See Fig. 7.7 for the domain profiles for the proteins.

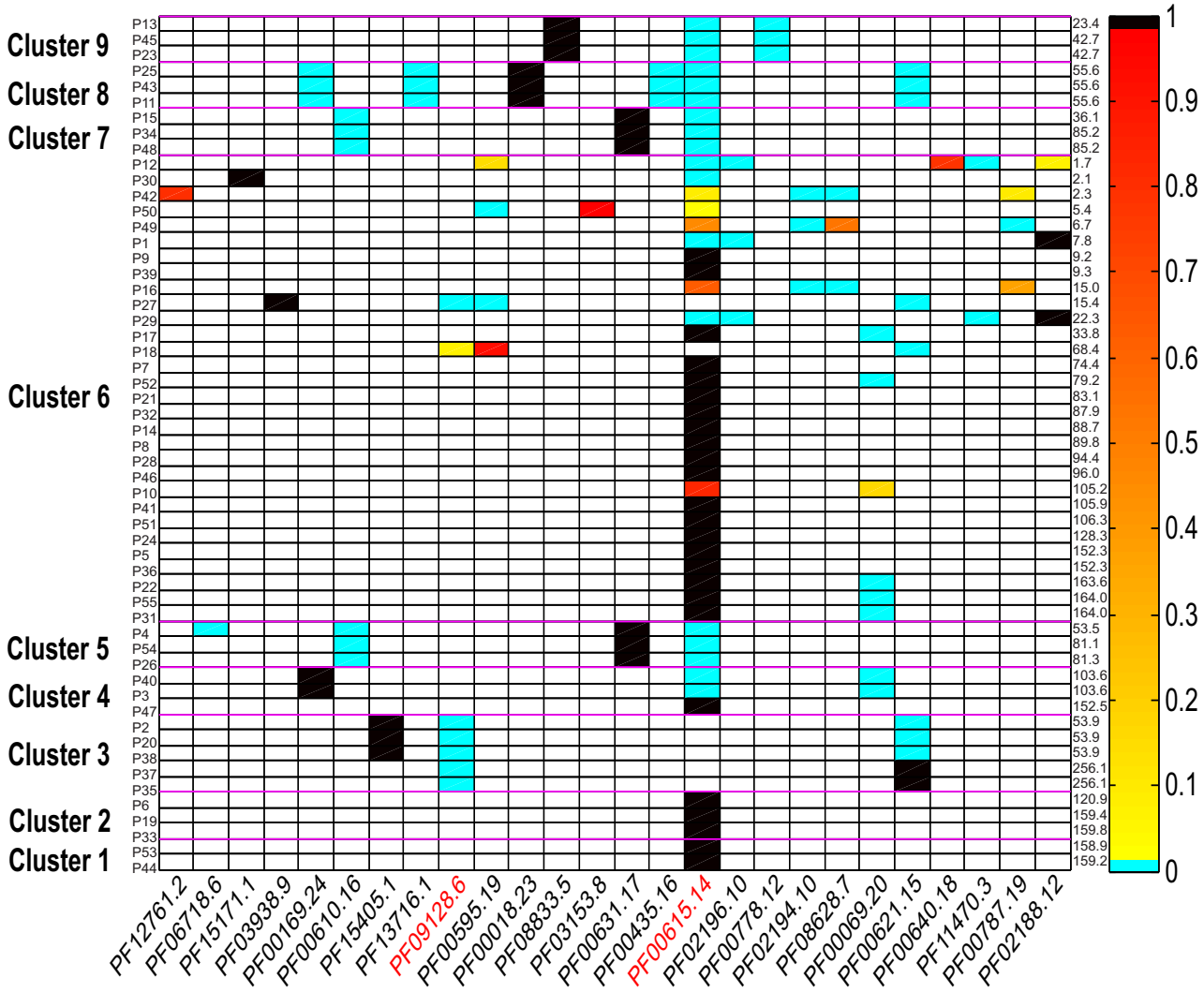


Figure 7.7: **RGS Family Domain Profile (Non-Overlapping Domains).** In this diagram, each row represents one of the 55 RGS proteins, while each column represents one of 26 Pfam domains (RGS and RGS-like domains are shown in red). Each cell is color-coded based on the diversity weights x: black for $x_i = 1$, cyan for $x_i = 0$ (but domain exists), and from dark to light orange for $1 > x_i > 0$. Blank cells indicate domain absence. The clusters are separated by horizontal magenta lines. Proteins in each cluster (identifers shown on the right) are arranged according to their game value, with the largest appearing as the lowest row in the cluster. The game values for each protein are shown on the left.

7.2.1 Comparison of Game Theory Clustering To TRIBE-MCL, Protein-Domain Biclustering, and Phylogenetic Clustering

As mentioned before, for a regular phylogenetic analysis of multi-domain proteins, usually only sequence information of domains shared across all member proteins can be used. The maximum likelihood method (explained in Section 2.2.3) was used to construct phylogenetic clusters for the RGS proteins. A multiple sequence alignment was constructed for the commonly shared domain across all proteins (i.e. RGS or RGS-like domain). Bootstrap values of 70% were used to define the clusters of RGS sequences (Fig. 7.8). As shown in Table 7.1, the game theory clusters were largely consistent with the phylogenetic clusters, with $J(GT, PHY) = 0.7120$. One thing to note is that the average maximum Jaccard index of the game theory clusters against the phylogenetic clusters is larger ($J(GT, PHY) = 0.7120$) than that of the phylogenetic clusters against the game theory clusters ($J(PHY, GT) = 0.4198$). We conjecture that the differences in the cluster sizes contribute to this difference in average maximum Jaccard indices. Although the game theory clusters were quite similar to the phylogenetic clusters, the phylogenetic analysis cannot represent information from domains that are not shared across all proteins, and therefore misses some relationships that our network approach reveals.

There are seven proteins in the protein space that contain the RGS-like domain (PF09128.6). Proteins P2, P20, P35, P37, and P38 are grouped into the same cluster by both the game theory and phylogenetic methods. For the other two proteins containing the RGS-like domain, proteins P18 and P27, there is a discrepancy between the two approaches. The phylogenetic method clusters P18 with the five proteins containing RGS-like domain mentioned above (Cluster 22 in Fig. 7.8), while P27

	GT (9)	BICL (17)	MCL (10)	PHY (24)
GT	1.0000	0.5245	0.4335	0.7120
BICL	0.3479	1.0000	0.3952	0.6133
MCL	0.4521	0.5662	1.0000	0.8342
PHY	0.4198	0.5346	0.5000	1.0000

Table 7.1: **Comparison of RGS Game Theory (GT), Bicluster (BICL), TRIBE-MCL (MCL) and Phylogeny (PHY) Clusters Based On The Average Maximum Jaccard Index.** The average maximum Jaccard indices shown in each column are computed using the method in the column header as the reference method. For example, the entries in the first column, from top to bottom, are $J(GT, GT)$, $J(BICL, GT)$, $J(MCL, GT)$, and $J(PHY, GT)$. The number of clusters produced by each method is shown in parenthesis.

exists as a single protein cluster (Cluster 15 in Fig. 7.8) located in a completely different region of the tree than the other six proteins.

The game-theoretic approach places P18 and P27 in the same cluster (Cluster 6 in Fig. 7.6). Moreover, the edge weights indicate that relative to the other proteins in the RGS data set, P18 is P27's closest neighbor and vice versa. This relationship, missed by the phylogenetic method, is a direct result of the game theory method's use of information from all domains. P18 contains three domains, PF09128.6, PF00595.19, and PF00621.15, which comprise three of P27's four domains. In addition P27 has relationships with both P50 and P12, as a result their sharing Pfam domain PF00595.19. Moreover, Fig. 7.7 shows that PF00595.19 only exists in these four proteins (P12, P18, P27, and P50). Thus, the game theory clusters for the RGS data set shed light on evolutionary relationships that would otherwise be overlooked in a classic phylogenetic analysis.

Clusters for the RGS sequences with non-overlapping domain predictions were also constructed using TRIBE-MCL and protein-domain biclustering. The biclustering approach identified 17 clusters (see Fig. 7.9 and Table A.14), while the TRIBE-MCL method identified 10 clusters (see Table A.15). There are some noteworthy differences

between the clusters that arise from protein-domain biclustering and TRIBE-MCL when compared to the game theory clusters. TRIBE-MCL clusters all seven of the proteins that contain the RGS-like domain (P2, P18, P20, P27, P35, P37, and P38) into a single cluster (Cluster 3 in Table A.15). As mentioned before, the MCL method clusters the sequences based on the single most significant region between them. The proteins mentioned here were clustered together by MCL based on their RGS-like domain, which is unique to these proteins relative to the others in the protein space.

Our game theory approach clusters these proteins into two clusters. The first, Cluster 3 in Fig. 7.6, contains proteins P2, P20, P35, P37, and P38. As mentioned before, the other proteins containing the RGS-like domain, P18 and P27, each contain the Pfam domain PF00595.19, which leads to them belonging to Cluster 6. Hence by incorporating information from all of the domain regions on the proteins, the game theory method is able to cluster proteins on a finer scale, in terms of domain content, than the TRIBE-MCL method.

There were also key differences between the game theory clusters and the protein-domain biclusters. For example, in the bicluster network, all of the proteins that contain only the RGS domain (PF00615.14) appear in a single cluster (B1 in Fig. 7.9). The protein-domain biclustering approach binarizes the domain composition information prior to clustering, while the game theory method makes use of quantitative similarity information among domains on the proteins. So in the game-theoretic network we see that the proteins containing only the RGS domain are further separated due to the varying levels of similarities in their RGS domains. For instance, Cluster 1 in Fig. 7.6 consists of P44 (RGS 17 isoform 1) and P53 (RGS 17 isoform 2). These two proteins are isoforms and have only the RGS domain. Since these two sequences are almost identical, relative to the other proteins in the protein space, that they appear in their own distinct cluster.

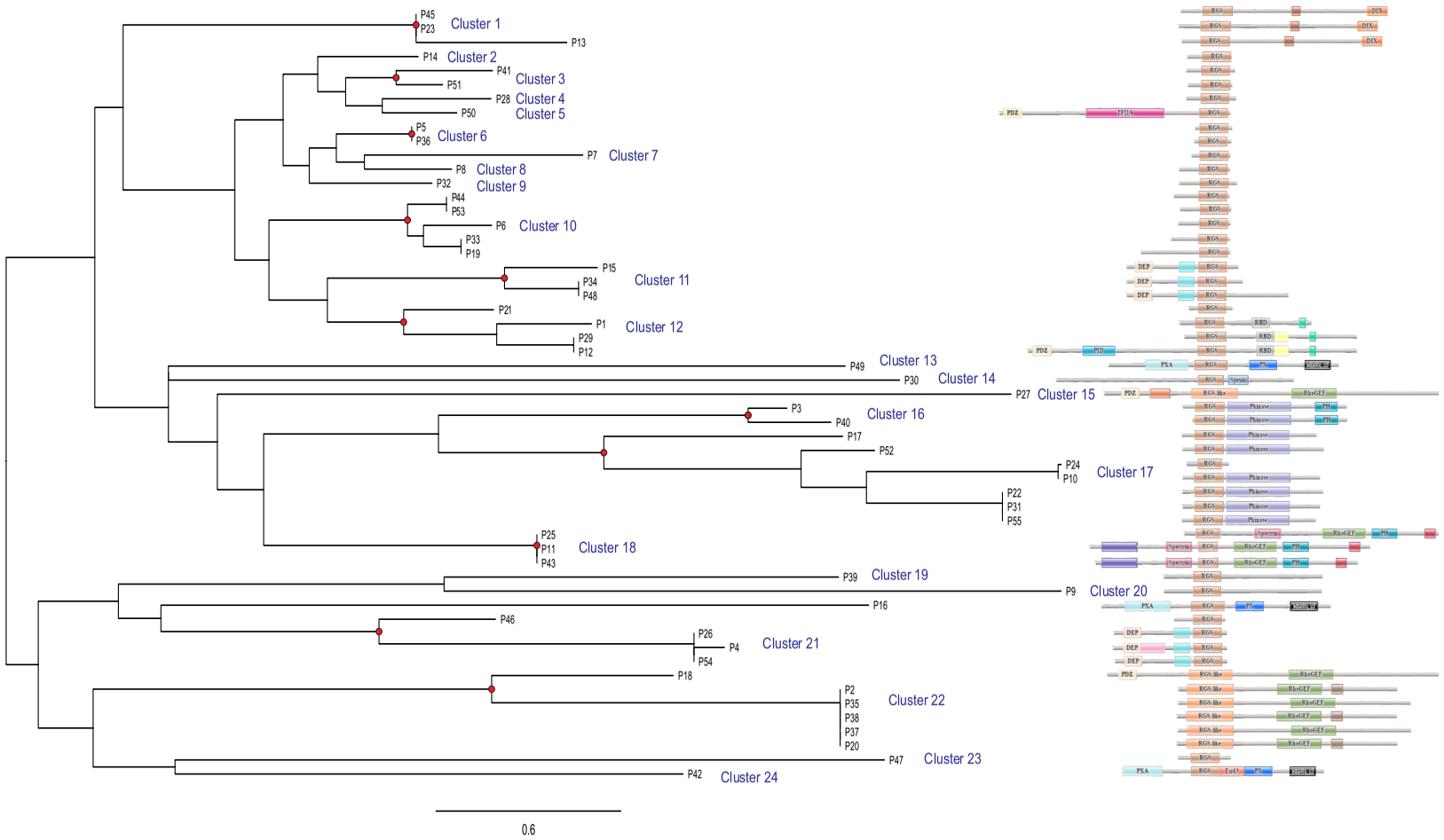


Figure 7.8: **RGS Family Maximum Likelihood Phylogeny.** The maximum likelihood method was used to construct a phylogeny of the 55 RGS proteins, based on their multiple sequence alignment. Nodes with at least 70% bootstrap support, shown by a red dot, define phylogenetic clusters. There are 24 clusters in the tree (labeled Cluster 1-24).

Another interesting difference between the biclustering and game theory methods is the placement of P47 in the network. During the HMMER domain search, only the RGS domain is identified on this protein. So in the biclustering network, P47 is located in Bicluster 1, with the other proteins that have only the RGS domain. However, in the game theory network P47 is located in Cluster 3 with proteins P3 and P40. Table A.14 shows that P3, P40, and P47 are all Beta-Adrenergic Receptor Kinase proteins. In fact, in this data set these are the only Beta-Adrenergic Receptor Kinase proteins. The game theory method was able to pick out the relationship between these proteins because they were highly similar in their shared RGS domain sequence, and their "version" of the RGS domain was unique, relative to the other proteins. This example highlights that the game theory method's use of quantitative similarity information can lead to a better clustering of protein families.

7.2.2 GO Analysis of RGS Game Theory Clusters

As shown in Fig. 7.7, clusters in the RGS game-theoretic protein similarity network share relatively large number of domains. We suspect that the proteins in these clusters may also perform similar or related functions and hence belong to similar protein subfamilies. Gene Ontology (GO) [121] is widely accepted as the standard vocabulary for describing the biological process, molecular function, and cellular component of genes (see Section 2.2.4). GO annotation of these proteins was generated using BLAST2GO [25], and the molecular function annotations were analyzed to support our hypothesis.

Table 7.2 shows the molecular function GO terms associated to proteins in each of the game-theoretic clusters. Following each of the GO terms is a list of proteins in the cluster that were annotated with the GO term. GO terms that were assigned to

every protein in the cluster are shown in red. Figure 7.10 shows a subgraph of the Gene Ontology containing the GO terms found in Table 7.2. This graph, generated using QuickGO [13], shows the hierarchical relationships between the GO terms.

Most of the GO terms are shared between clusters due to the fact that all of the proteins in the data set belong to the Regulator of G-Protein Signaling family, i.e. these proteins should all have somewhat similar functions. With the exception of Cluster 6 each cluster in the game theory network has a GO term shared by all proteins in the cluster. The combination of the annotation information in Table A.13 and the shared GO terms in each cluster suggests that in addition to generating clusters that are consistent with other clustering algorithms, the game-theoretic approach is able to assign proteins to functionally meaningful clusters and effectively identify multi-domain protein families in a given data set.

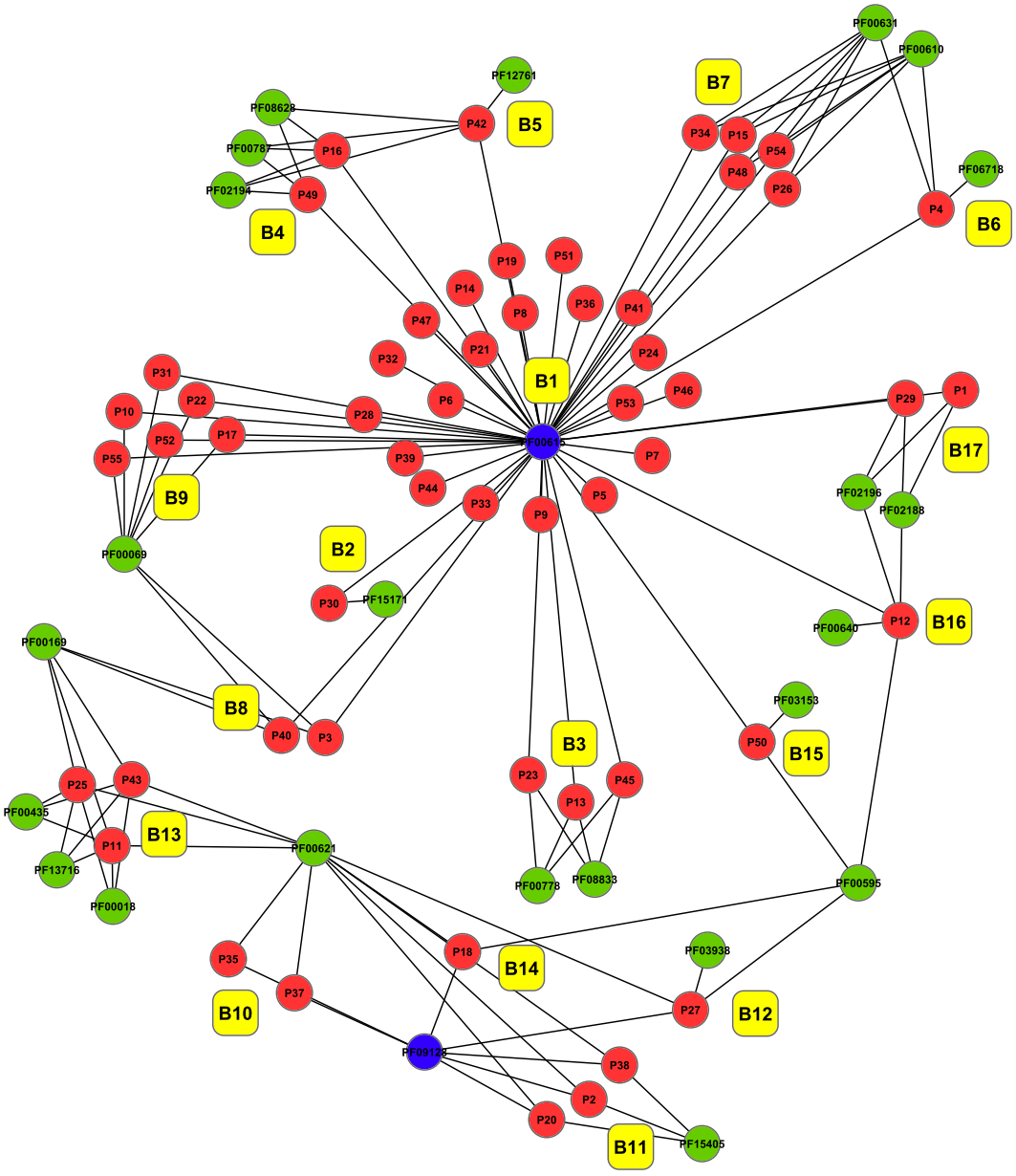


Figure 7.9: **RGS Family Bicluster Network.** The protein-domain biclustering method was used to generate a network of biclusters. The 55 RGS proteins (red nodes) were biclustered with their respective domains (green nodes), resulting in 17 biclusters (labeled B1-B17). Either the RGS (PF00615) or RGS-like (PF09128) domain exists in every protein, and the nodes corresponding to these domains are shown in blue.

Cluster	GO Term (Protein)
1	GO:0030234 (P44, P53)
2	GO:0030234 (P33,P19,P6), GO:003674 (P33,P19,P6)
3	GO:0030234 (P2,P20,P35,P37,P38), GO:0003674 (P2,P20,P35,P37,P38), GO:0003723 (P2,P20,P35,P37,P38)
4	GO:0043167 (P3,P40,P47), GO:003674 (P3,P40,P47), GO:0016301 (P3,P40,P47)
5	GO:0030234 (P4,P26,P54), GO:0003674 (P4,P26,P54), GO:0004871 (P4,P26,P54)
6	GO:0043167 (P10,P16,P17,P22,P24,P31,P42,P49,P52,P55), GO:0016301 (P10,P17,P22,P24,P31,P52,P55), GO:0019899 (P1,P52), GO:0003674 (P1,P7,P8,P12,P18,P21,P22,P27,P28,P29,P31,P39,P42, P46, P55), GO:0030234 (P1,P5,P7,P8,P9,P12,P14,P21,P18,P27,P28,P29,P32,P36, P41,P46,P50,P51), GO:0008289 (P16,P42,P49,P52), GO:0005198 (P1), GO:0008092 (P1,P46), GO:0004871 (P1,P12,P29)
7	GO:0004871 (P15,P34,P48), GO:0003674 (P15,P34,P48), GO:0030234 (P15,P34,P48)
8	GO:0043167 (P11,P25,P43), GO:0008289 (P11,P25,P43), GO:0003674 (P11,P25,P43)
9	GO:0005198 (P23,P45),GO:0030234 (P13,P23,P45), GO:0003674 (P13,P23,P45), GO:0004871 (P13,P23,P45), GO:0019899 (P13,P23,P45)

Table 7.2: **GO Terms for RGS Game-Theoretic Clusters.** Gene Ontology (GO) molecular function annotations for the 55 RGS proteins were generated using BLAST2GO. GO terms associated to proteins in each of the game theory clusters are shown, followed by the list of proteins in the cluster annotated by the particular GO term. The GO terms shown in red were associated to every protein in the cluster.

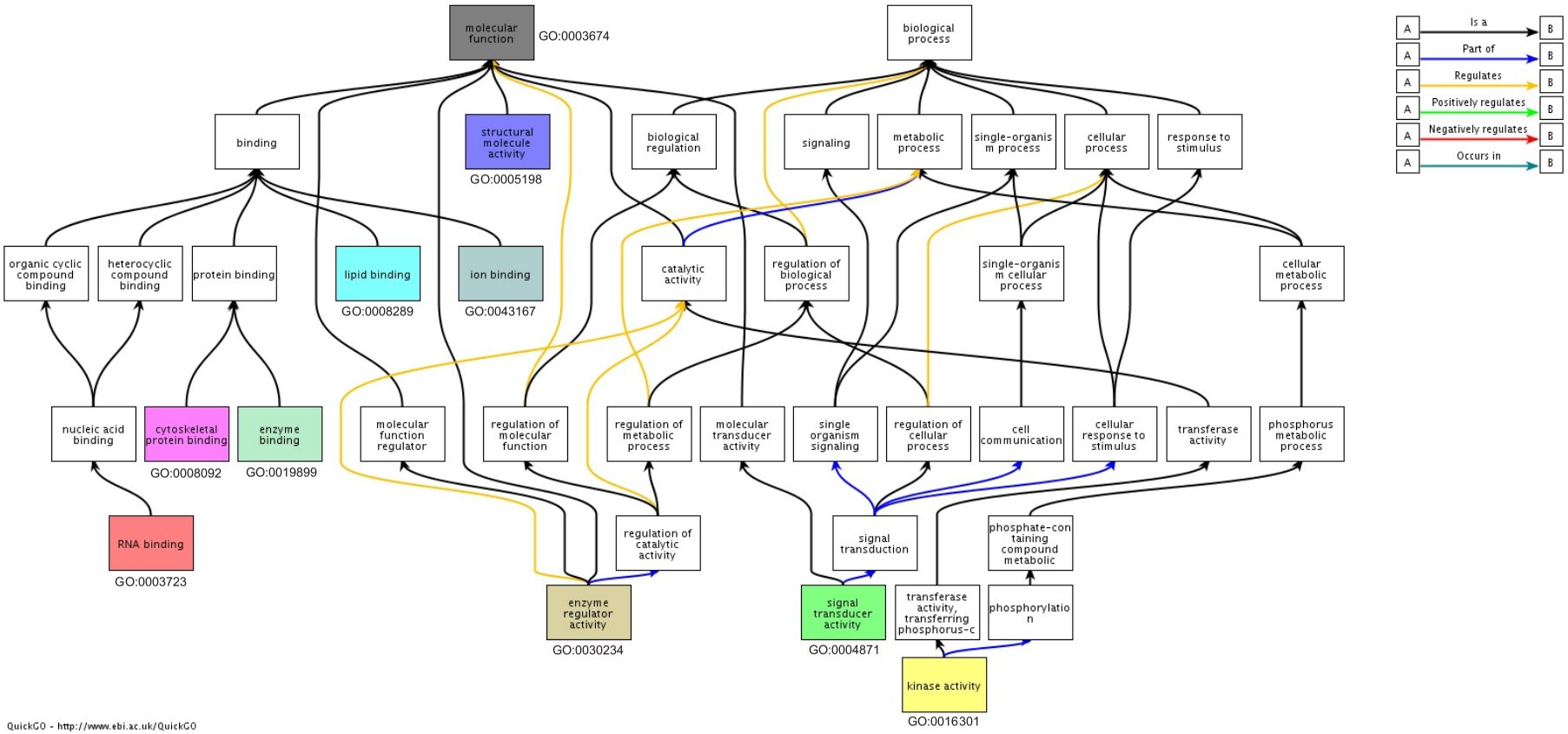


Figure 7.10: **Gene Ontology Subgraph Containing GO Terms Associated to RGS Proteins.** This directed acyclic graph, generated using QuickGO [13], shows the hierarchical relationships in the Gene Ontology between the terms associated to the 55 RGS proteins (see Table 7.2). The nodes corresponding to these GO terms are shown as colored nodes with the GO identifier listed next to the node. The edges in the GO graph are colored according to the relationship they represent (given in the legend).

7.3 Game-Theoretic Clustering for Swiss-Prot Mouse Proteome

For a large-scale real data, we used the set of all proteins in the Swiss-Prot mouse genome that had UniProt protein family annotations, which included 12,222 proteins. The game theory and biclustering methods are unable to be used for proteins that do not contain a Pfam domain. Therefore 302 proteins had to be removed from the analysis. For this data set, which consisted of 11,920 proteins, UniProt assigned 3,703 protein families. We used this UniProt family assignment as the reference clusters and tested the performance of our game-theoretic method compared to protein-domain biclustering and TRIBE-MCL. Note that phylogenetic clustering is not possible for this data set since no domain is shared amongst all of the proteins.

7.3.1 Comparison of Game Theory Clusters To TRIBE-MCL and Protein-Domain Biclustering

The HMMER search identified 5,381 non-overlapping Pfam domains from these proteins. By applying the game theory clustering approach 1,207 clusters were identified as shown in Fig. 7.11. In comparison, biclustering and TRIBE-MCL identified 3,033 and 1,217 clusters, respectively. We compared the results of these clustering techniques to the UniProt reference protein families. Table 7.3 shows the clustering accuracy for each of the methods, which was assessed using the average maximum Jaccard index.

In the case of the non-overlapping domain detection strategy (as well as the other strategies), the number of clusters identified from biclustering was exceedingly higher than that of game theory and TRIBE-MCL. This comes from the fact that the bi-

Method	Domain Type	# Clusters	Avg. Max. Jaccard Index
Game Theory	Non-Overlap	1,207	0.5110
	5% Overlap	1,201	0.5108
	Unrestricted Overlap	909	0.5098
Biclustering	Non-Overlap	3,033	0.5112
	5% Overlap	3,596	0.4574
	Unrestricted Overlap	5,692	0.3872
TRIBE-MCL	NA	1,217	0.1975

Table 7.3: **Comparison of Clusters For Swiss-Prot Mouse Proteome.** The results of the game theory, biclustering, and TRIBE-MCL clustering were compared to the 3,703 UniProt reference protein families, i.e. the clusters of proteins that have the same UniProt family annotation.

clustering approach is based on binarized inclusion of domains, which means that the number of clusters is directly proportional to the number of distinct domain compositions present in the data [111]. So unlike biclustering, which cannot separate proteins that share domains even if some domains are shared with very weak similarity, game theory and TRIBE-MCL are able to cluster proteins based on highly conserved domains since these methods use quantitative similarity information.

The TRIBE-MCL method had the lowest clustering performance of the three methods. As mentioned before, this is because TRIBE-MCL uses information from only one region of local similarity, i.e. it does not use information from all conserved domains on the proteins. Both game theory and biclustering use information from all domains, and their clusters are more consistent with the UniProt family assignments. Interestingly, the performance of the game theory method, assessed by the average maximum Jaccard index, was roughly the same as that of biclustering, even though the biclustering method identified many more clusters.

Twenty one of the 66 proteins in the RGS data set, discussed in Section 7.2, were present in our Swiss-Prot mouse data set (see Table A.16). The remainder of the RGS family proteins were not part of the Swiss-Prot database, as they had either not

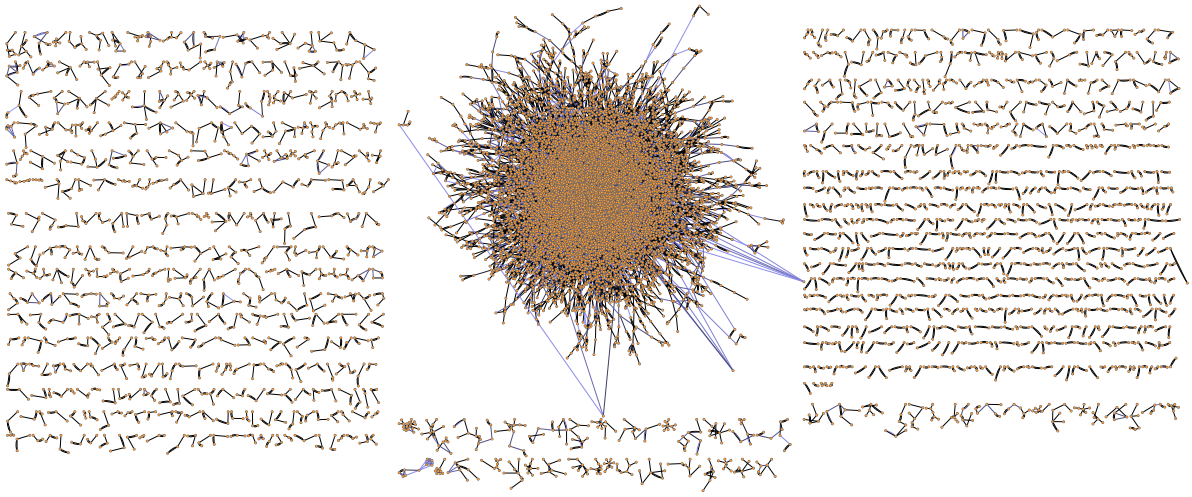


Figure 7.11: **Game-Theoretic Protein Similarity Network (Non-Overlapping Domains) For the Swiss-Prot Mouse Proteome.** A total of 1,207 clusters were identified for the 11,920 proteins (orange nodes). The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. The game value for each protein is represented in the network by the length of its incoming edges, with longer edges corresponding to small game values and longer edges indicating high game values.

been manually annotated and reviewed or were an isoform of another RGS protein family (Swiss-Prot merges these isoforms into a single entry). Three of the 21 RGS proteins, P8453, P8454, and P8456 (in the RGS network P4, P26, P48), were clustered along with one other protein in a small cluster. This cluster is shown in Fig. A.16. In the RGS network (Fig. 7.6) these proteins existed in Cluster 5 and Cluster 9. The other protein that was clustered with these three RGS proteins in Fig. 7.11 is P3752, Guanidinoacetate N-methyltransferase protein. This protein contained only one domain, PF00278.17, and it showed highest similarity to P8454. However, the similarity was quite weak, as the game value for P3752 was small, only 11.5.

The other RGS proteins in Table A.16 belonged to the largest network cluster, which consisted of 8,290 proteins. Within the large cluster, 10 of the RGS proteins

exist in one small neighborhood, while 3 others exist is another neighborhood. These neighborhood (subgraphs of the large cluster) are shown in Fig. A.17 and A.18, respectively.

7.4 Effect of Overlapping and Non-overlapping Domain Identification

To analyze the impact of overlapping and non-overlapping domains on the network clusters, clustering was done using the non-overlap, 5% overlap, and unrestricted overlap domain identification strategies (described in Section 6.2.2). Figures 7.12(A) and 7.12(B) show the number of proteins that have a given number of domains using each of the domain identification strategies for the RGS and Swiss-Prot mouse data sets, respectively. The number of proteins identified to contain a single domain is much larger when using the non-overlap strategy. For example, as shown in Fig. 7.12(B) the number of single domain proteins in the Swiss-Prot mouse proteome is 6,484 when using the non-overlap strategy and 2,969 when using the unrestricted overlap strategy. On the other hand, as the percentage of allowed overlap increases there is an increase in the number of domains identified in a single protein.

7.4.1 Effect of Domain Identification Strategy For RGS Family

As mentioned before, for the RGS family protein set there was a total of 26 non-overlapping domains (see Table A.10), 29 domains when 5% overlap was allowed (see Table A.11), and 41 domains when no restriction was placed on the overlap percentage (see Table A.12). For the non-overlapping case (described in detail in Section 7.2)

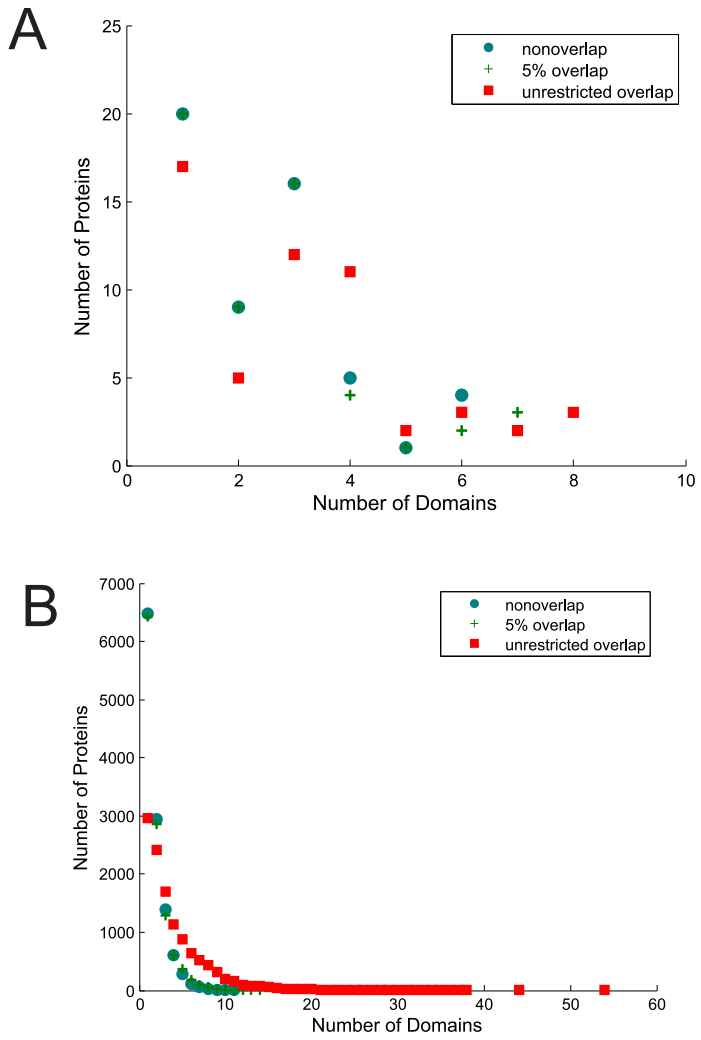


Figure 7.12: **Domain Identification From the RGS Family (A) and Swiss-Prot Mouse Proteome (B).** The number of identified domains vs. the number of proteins containing the given number of domains.

9 clusters were identified by the game theory method (Fig. 7.6). Figures A.12 and A.13 show the game theoretic similarity networks for the 5% overlap and unrestricted domain identification, respectively.

Although allowing a 5% overlap instead increases the total number of identified domains from 26 to 29, the network structure is identical to that of the non-overlapped domain case. The three additional domains, PF13180.1, PF14604.1, and PF13476.1,

	Non-Overlap	5% Overlap	Unrestricted Overlap
Cluster			
1	159.0567 (2)	159.0567 (2)	146.7074 (3)
2	146.7074 (3)	146.7074 (3)	134.7646 (5)
3	134.7646 (5)	134.7646 (5)	109.9268 (3)
4	119.9047 (3)	119.9047 (3)	106.6325 (2)
5	71.9517 (3)	71.9517 (3)	71.9517 (3)
6	71.1609 (30)	71.1609 (30)	68.8465 (3)
7	68.8465 (3)	68.8465 (3)	68.1064 (30)
8	55.5742 (3)	55.5742 (3)	55.5742 (3)
9	36.2278 (3)	36.2278 (3)	36.2278 (3)

Table 7.4: **Average Game Values for Game Theory Clusters.** The game theory method was used to build networks for the 55 RGS proteins using the non-overlapping (Fig. 7.6), 5% overlap (Fig. A.12), and unrestricted (Fig. A.13) domain detection strategies. This figure shows the average game value and the number of proteins (the number in parentheses) for each of the clusters in each of the three networks.

are each identified on only one of the 55 RGS proteins. Domain PF13180.1 is identified on P27 in Cluster 6. However, the domain profile for this protein (Fig. A.13) shows that even though PF13180.1 is hallmark to this protein, the domain receiving the highest weight (in fact, a weight of 1) is PF03938.9, another domain which exists only in P27. The same phenomena is seen for domains PF14604.1 and PF13476.1. So since our game theory method consists of finding minimally shared regions of maximal similarity, we can conclude that these domain sequences were not unique enough (with respect to the other proteins in the data set) to change the network edge structure.

Removing all restrictions on domain overlap gives rise to a different network structure (Fig. A.12). The proteins in each cluster remain the same, but there are differences in edge weights and game values. Table 7.4 shows the average game values and number of proteins in each cluster for all three networks. The average game values for many of the clusters decrease when the unrestricted domain detection strategy is used, and the ordering of the clusters changes (e.g. Cluster 2 in Fig. 7.6 becomes

Cluster 1 in Fig. A.12). Hence the domain identifications have a direct effect on the 'tightness' of the clusters.

In addition to the changes in the average game values for the clusters, there is a difference in the edge structure in the network. In particular, the incoming edges for proteins P1, P10, P11, P25, and P43 have different edge weights. This is because the additional domains identified on the proteins affect the overall similarity of the protein to each of its neighbors in the network. For example, in the non-overlap network P1 has incoming edges from P12 (edge weight 0.5027) and P29 (edge weight 0.4973). In the unrestricted network, these edge weights change to 0.0493 and 0.9506, respectively. Domain PF07631.6 is added to the domain content of P1 in the unrestricted network, and hence it is the similarities of P12 and P29 to P1 with respect to this domain that govern the change in edge weight.

7.4.2 Effect of Domain Identification Strategy For Swiss-Prot Mouse Proteome

As described earlier, for the 11,920 proteins in the Swiss-Prot mouse data set clusters were generated using the game theory, biclustering, and TRIBE-MCL methods and compared to the UniProt reference protein families For this data set, Pfam domains were identified using the non-overlap (5,381 Pfam domains identified), 5% overlap (5,444 Pfam domains identified), and unrestricted overlap (7,512 Pfam domains identified) detection strategies. The results for the various domain detection strategies are summarized in Table 7.3.

As mentioned earlier, increasing the allowed percentage of overlap in domain identification results in larger domain sets. In the Swiss-Prot mouse data set, the increase in domain overlap percentage led to a decrease in the number of clusters in the game-

theoretic network. In contrast, the number of biclusters increased as the allowed overlap increased. Although the number of biclusters was much larger than the number of game theory clusters in all cases, the game theory method outperformed biclustering in both the 5% and unrestricted overlap scenarios and was quite comparable to biclustering in the non-overlap case. In addition, for the game theory method the average maximum Jaccard index remained nearly constant as the domain overlap percentage increased, while for biclustering a decrease in Jaccard index was observed as the overlap percentage increased. This suggests that the game theory clustering is more robust with respect to domain identification strategies.

Chapter 8

Discussion

Using game theory to study biological problems was first introduced by John Maynard Smith [82, 83]. The formulation of evolution as a game derived in this dissertation is *much* different from Smith's evolutionarily stable strategy (ESS) theory for animal behavior and conflict. In ESS, there are literal players, i.e. individual animals, as well as literal strategies that the players use in competition for reproduction and ecological resources.

In contrast, in our formulation evolution as a process is modeled as a game in which the players are the selective forces which operate everywhere, all of the time, in every biological process. Whenever genetic variation exists, evolution operates, in a simplistic model, in two modes: conservation and diversification. While conservation is to decrease the deleterious effect of mutational changes, diversification is to increase functional innovation, e.g. by acquiring advantageous mutations, new genes, or new domains. At the genome level, any enhancement of similarity between two genomes is the play of conservation whereas any widening of difference in a gene or gene composition between the genomes is the play of diversification. At the protein level, the sequence similarity in domains between two proteins is a play of conserva-

tion, while any diverging difference in domains, sequence or composition, is a play of diversity. In each level, the payoff of the game or process is not literal. It is their evolutionary similarity or dissimilarity broadly construed, which can be measured in terms of various informational distances.

Our bioinformatic game theory gives a plausible mechanistic explanation as to why and how evolution should sit at an informational Nash equilibrium. In fact, our derivation of the Nash map gives the same explanation to all games, including the games of ESS theory. The evolutionary selection force is local in time, space, and genetic sequences — meaning organisms or biological processes only need to seek out excess similarity for conservation and excess dissimilarity for diversity one step and one nucleotide at a time before eventually an informational Nash equilibrium is reached for the competing objectives. This evolutionary scenario is based on the dissipative dynamics of our localized Nash map or the Brown-von-Neumann-Nash equations. In searching for greater information for both conservation and diversity, the total excess information potential cannot increase but eventually converge to a state from which any deviation will not enhance the information for one of the two purposes. Since Nash equilibria are usually saddle points of the expected payoff functions, in this sense we can say that evolution should sit at a saddle point forged by the opposite pulls of the conservation and diversity strategies that evolution plays.

The problem of classifying biological sequences into similar families has been studied intensively in bioinformatics research. In this work we addressed the multi-domain protein clustering problem. Identifying or clustering large scale protein sequences into similar families based on their domain composition is a challenging problem. It is known that proteins sharing a single domain do not always perform the same or even related functions [52]. Traditional approaches to the protein clustering problem, including TRIBE-MCL and phylogenetic clustering, use information from only one

local region of similarity between proteins, i.e. information from only one shared domain. To get an accurate classification of protein families one needs to incorporate information from the entire domain composition. The protein-domain biclustering method takes this information into account, but in a limited binarized form.

In this work, the protein classification problem was addressed using our game-theoretic approach to building biological similarity networks. This method is able to utilize quantitative sequence similarity information from all domains on the proteins, to build a network which house clusters of similar protein sequences. The game theory method is completely scalable to the size of the data set. It has been used to classify proteins within a single protein family (the RGS protein data set), as well as proteins at the complete proteome level (the Swiss-Prot mouse data set). Comparison and evaluation of the game theory protein clusters with clusters from the TRIBE-MCL and biclustering methods showed higher average maximum Jaccard index scores both at a single protein family and a complete proteome level (using both real and simulated protein data).

Our novel clustering method hinges on the well-known theory of two-player zero-sum games. The algorithm allows efficient and rapid clustering of protein sequences, given a set of similarity matrices built from the results of another method, such as BLAST. The actual implementation of our method uses linear programming, a widely available, computational tool. The game-theoretic similarity network is built on a protein-by-protein basis, where each game theory minimax problem is solved independently of the others. This allows us to further speed up the algorithm by solving the game theory minimax problems in parallel.

One caveat of our game theory approach is that the network structures and the domain profiles depend critically on the similarity matrices that serve as the input to the game theory model. Therefore in future work we must examine different

scoring schemes for their robustness and sensitivity. Some possible scoring schemes include: the mutual information of pairwise sequences, the informational distance between sequences [75], and E-values based on profile HMMs instead of BLASTP. We could also look at composition-based similarity scores (e.g., frequencies of amino acid, k-mers, or reduced alphabets based on amino acid properties). It would be most interesting to introduce context to our similarity scores, including but not limited to context in terms of the combination, the size, and the order of domains.

The game-theoretic approach gives rise to clustered networks on minimally shared regions of maximal similarity. We can extend the method to obtain secondary clusters from within a primary network cluster. This is done by removing the domain having the largest domain weight for each reference protein and then finding the solution to the new corresponding linear programming problem. This secondary clustering structure will capture the next minimally shared region of maximal similarity for the proteins inside the primary cluster. This zoom-in procedure can continue to reveal the tertiary, the quaternary similarity relationship of the proteins, and so on, down to the 'root' at which the similarity is the maximum (having the maximal network game value). Unlike other methods, there are no arbitrary thresholds used to define clustering structure in the initial game theory network. However, since the solution vectors in the game theory minimax problem are real numbers rather than binary integers, we could judiciously set thresholds to prune the networks for finer clustering as an alternative to the zoom-in procedure described here.

Appendix A

Supplementary Materials

A.1 Figures

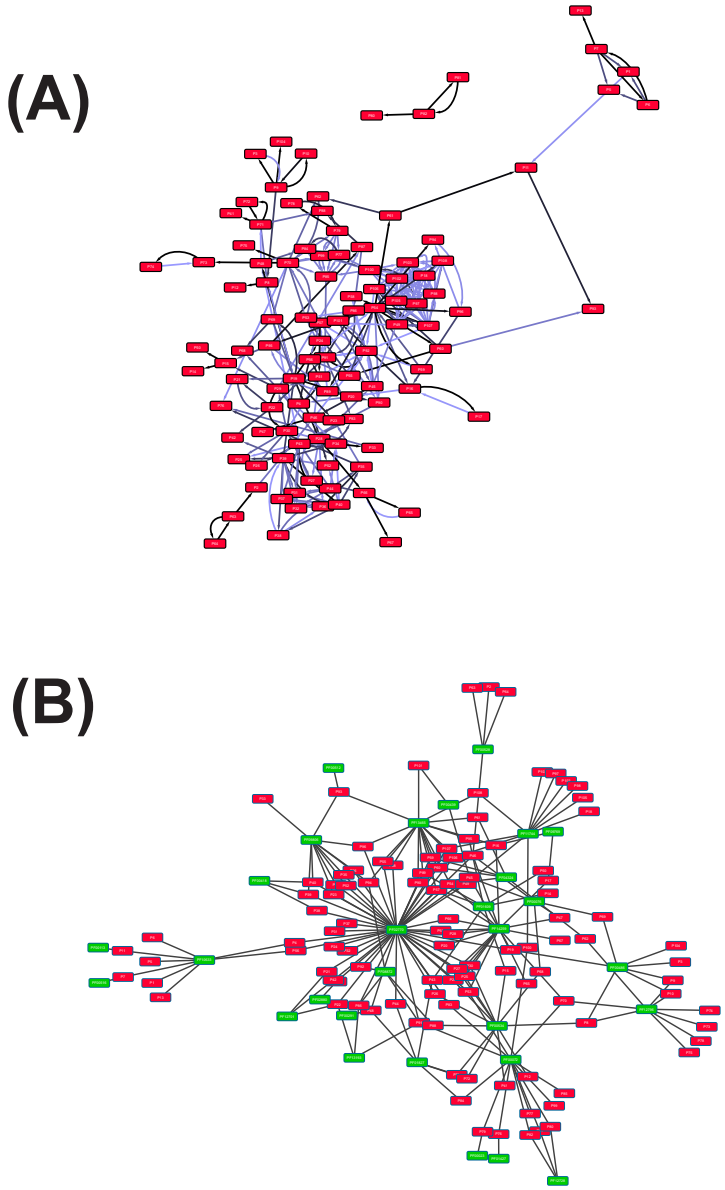


Figure A.1: **Sample Simulation Game Theory and Biclustering Network.** (A) The game theory network for a 10 generation simulated protein set consisting of 108 proteins. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. (B) The biclustering network for the same simulated set. The red nodes are proteins, while the green nodes are domains. In this simulation the linker mutation rate was 5% while the domain mutation rate was 10%.

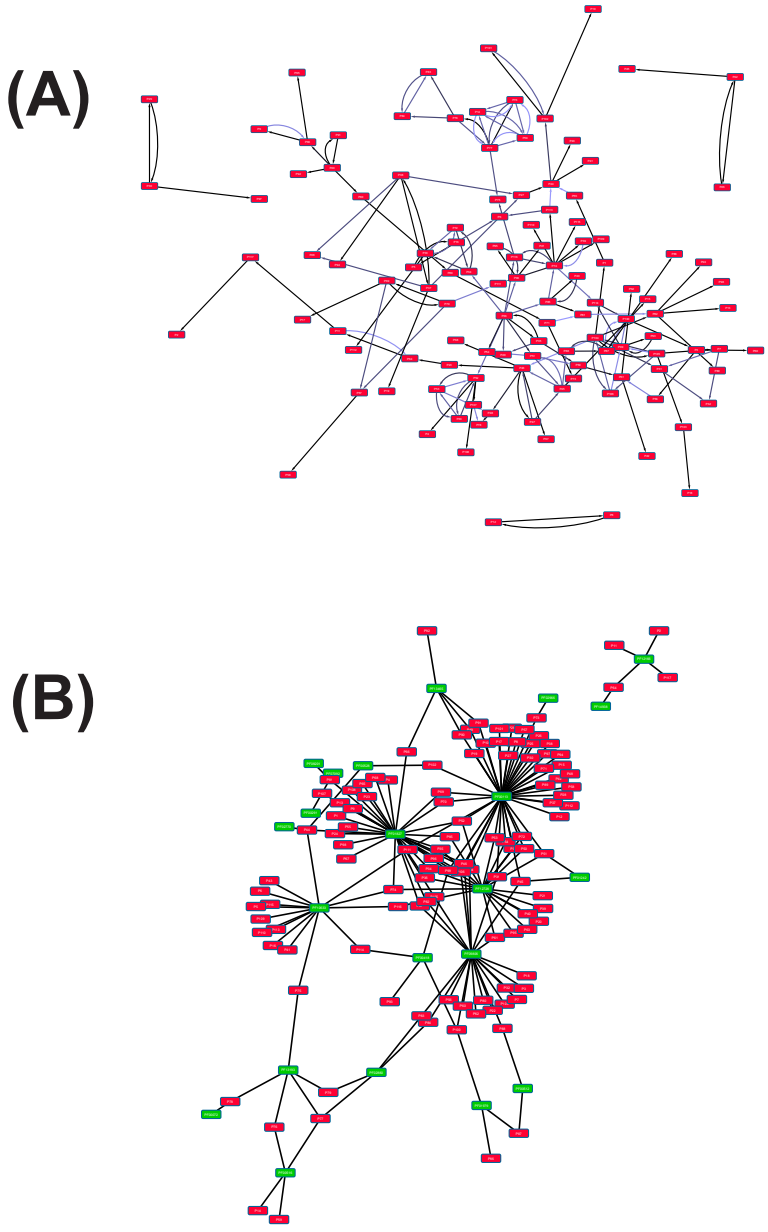


Figure A.2: **Sample Simulation Game Theory and Biclustering Network.** (A) The game theory network for a 15 generation simulated protein set consisting of 117 proteins. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. (B) The biclustering network for the same simulated set. The red nodes are proteins, while the green nodes are domains. In this simulation the linker mutation rate was 5% while the domain mutation rate was 10%.

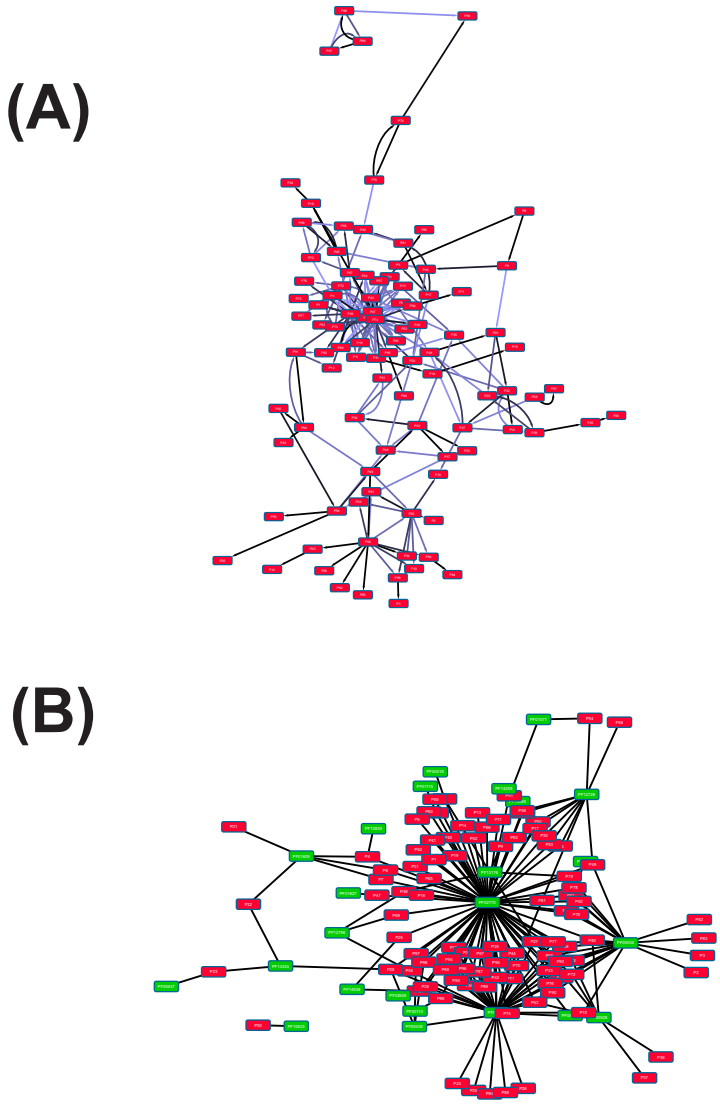


Figure A.3: **Sample Simulation Game Theory and Biclustering Network.** (A) The game theory network for a 10 generation simulated protein set consisting of 98 proteins. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. (B) The biclustering network for the same simulated set. The red nodes are proteins, while the green nodes are domains. In this simulation the linker mutation rate was 25% while the domain mutation rate was 5%.

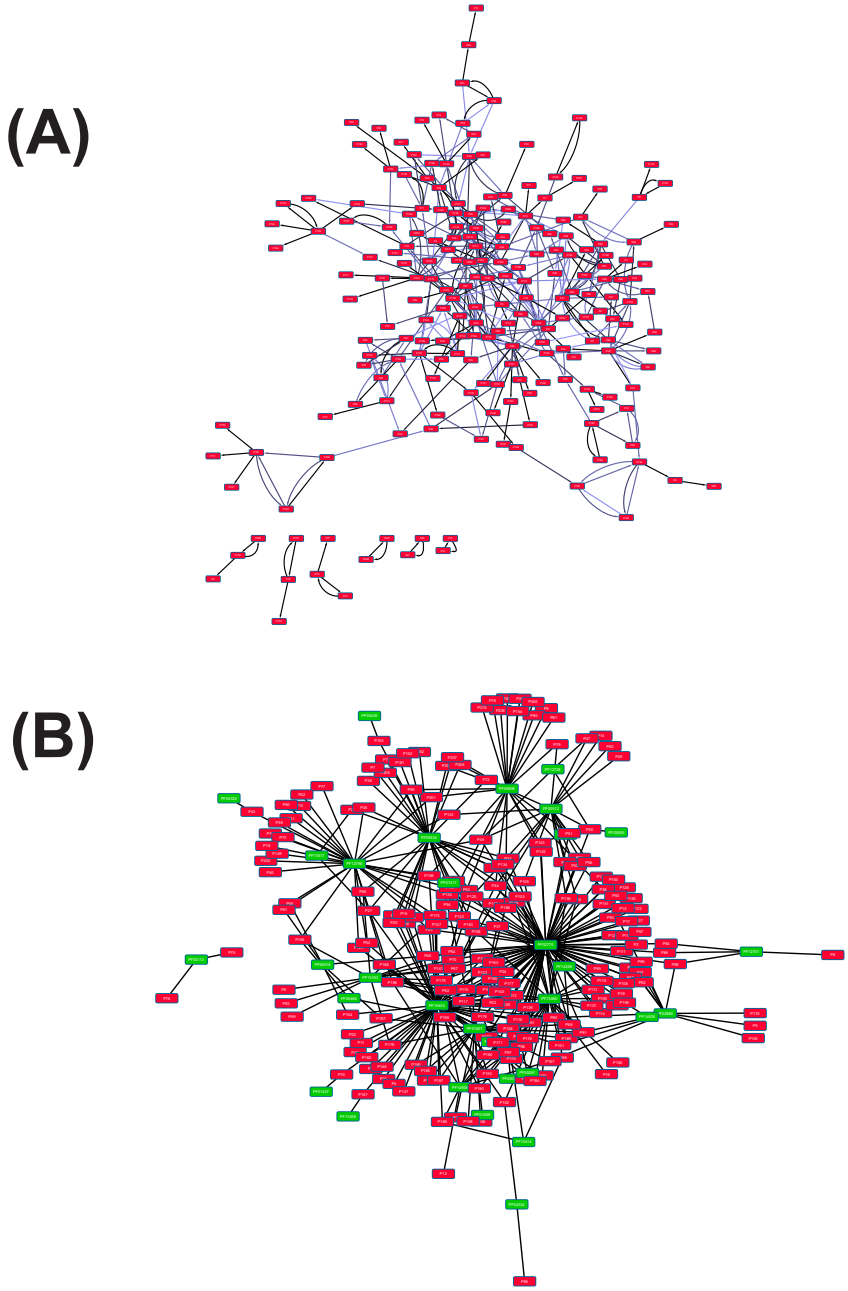


Figure A.4: **Sample Simulation Game Theory and Biclustering Network.** (A) The game theory network for a 15 generation simulated protein set consisting of 210 proteins. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. (B) The biclustering network for the same simulated set. The red nodes are proteins, while the green nodes are domains. In this simulation the linker mutation rate was 25% while the domain mutation rate was 5%.

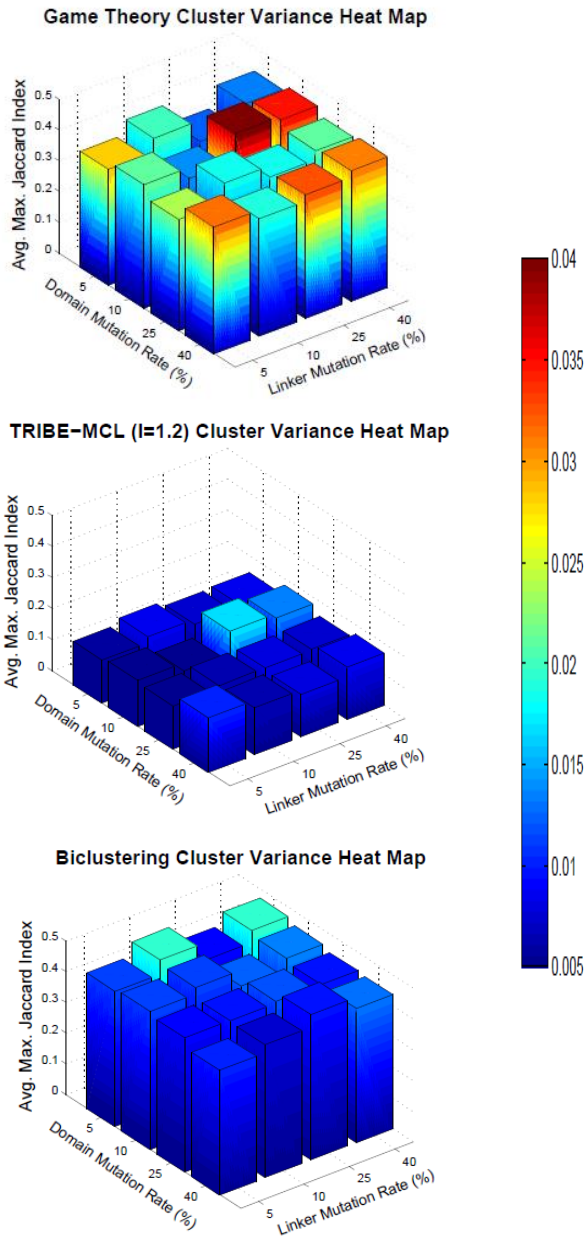


Figure A.5: **Cluster Variance Heat Maps (10 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The inflation parameter used in TRIBE-MCL was $I = 1.2$. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

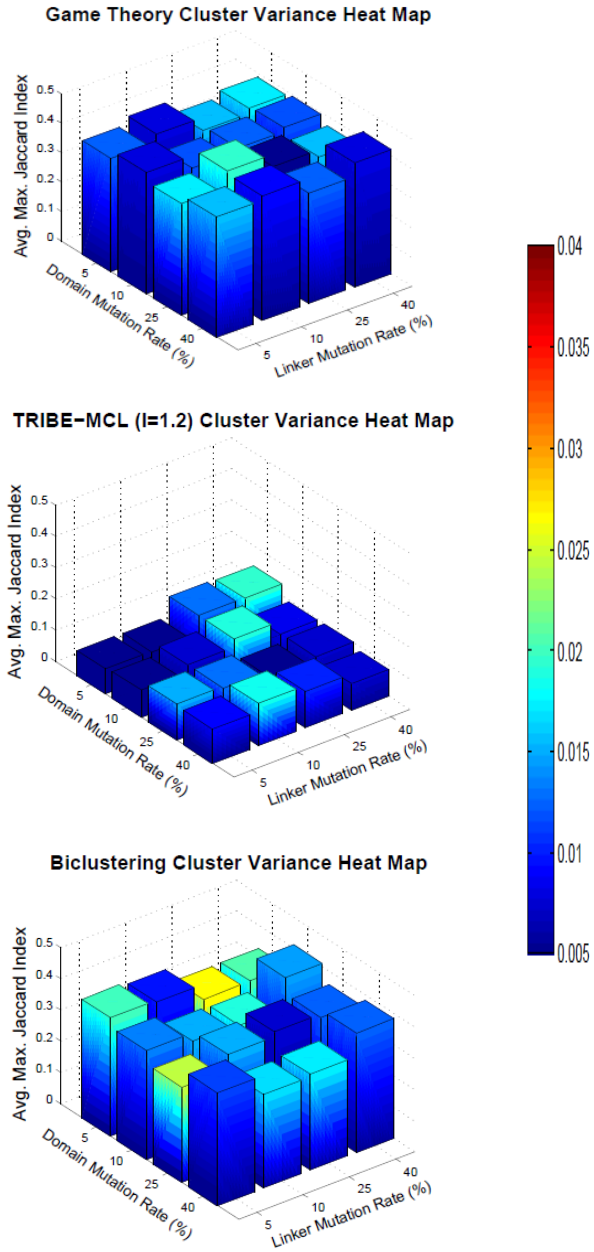


Figure A.6: **Cluster Variance Heat Maps (15 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The inflation parameter used in TRIBE-MCL was $I = 1.2$. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

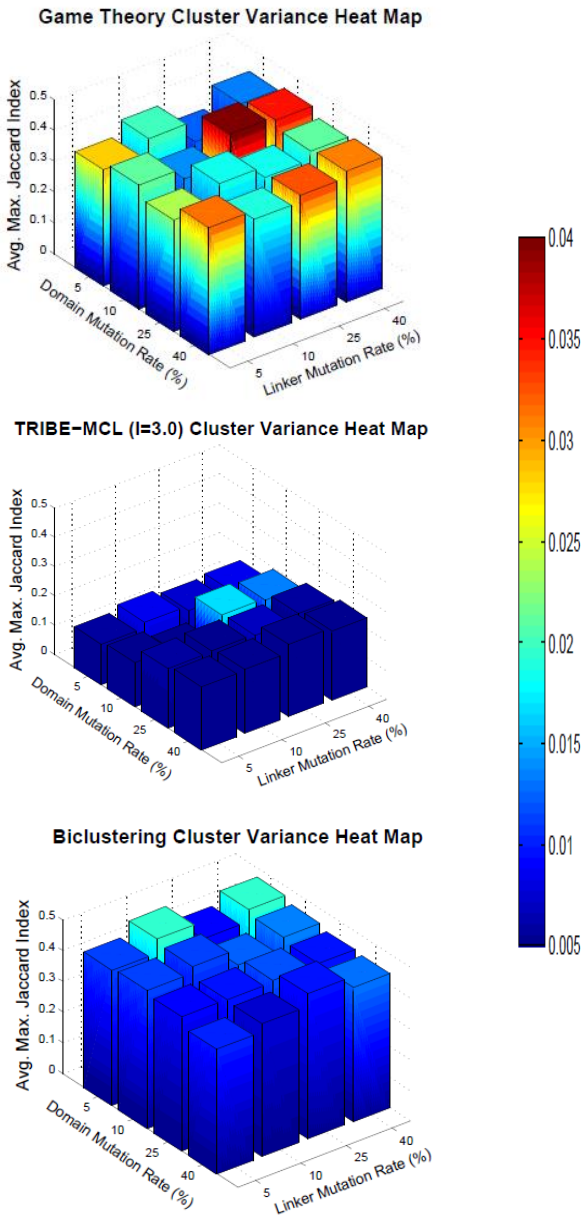


Figure A.7: **Cluster Variance Heat Maps (10 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The inflation parameter used in TRIBE-MCL was $I = 3.0$. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

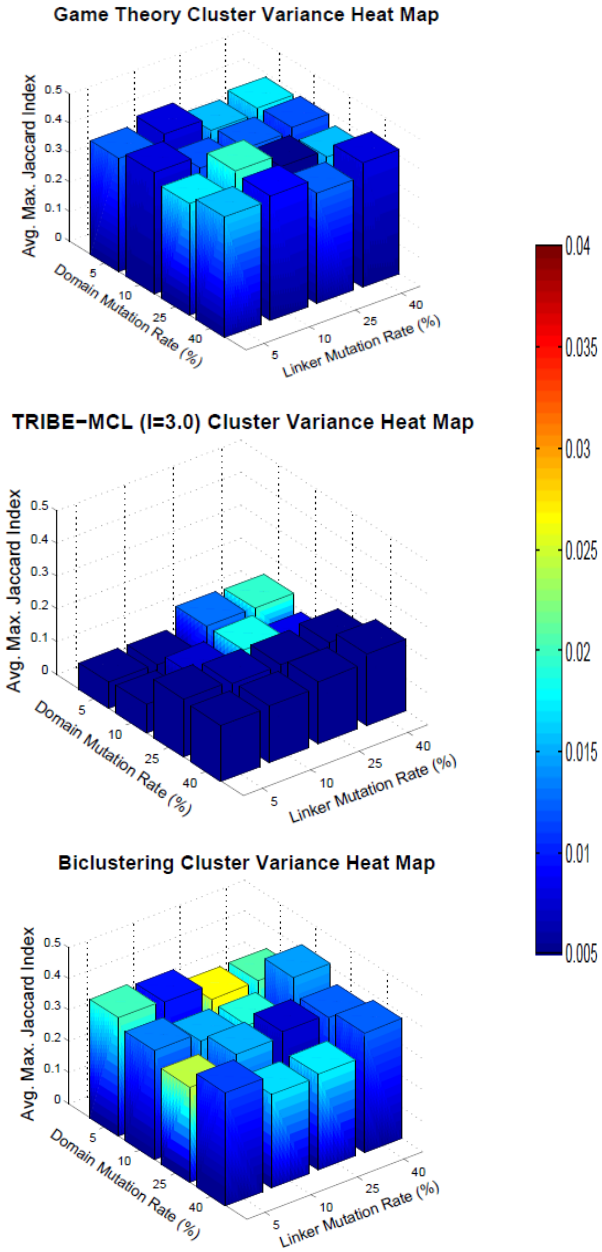


Figure A.8: **Cluster Variance Heat Maps (15 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The inflation parameter used in TRIBE-MCL was $I = 3.0$. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

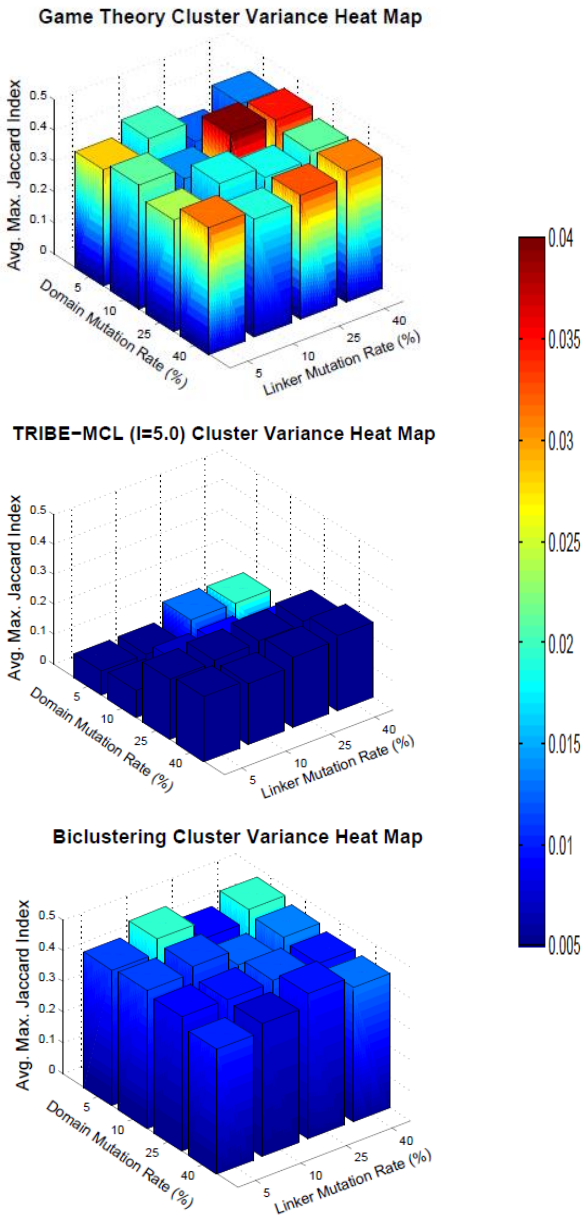


Figure A.9: **Cluster Variance Heat Maps (10 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The inflation parameter used in TRIBE-MCL was $I = 5.0$. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

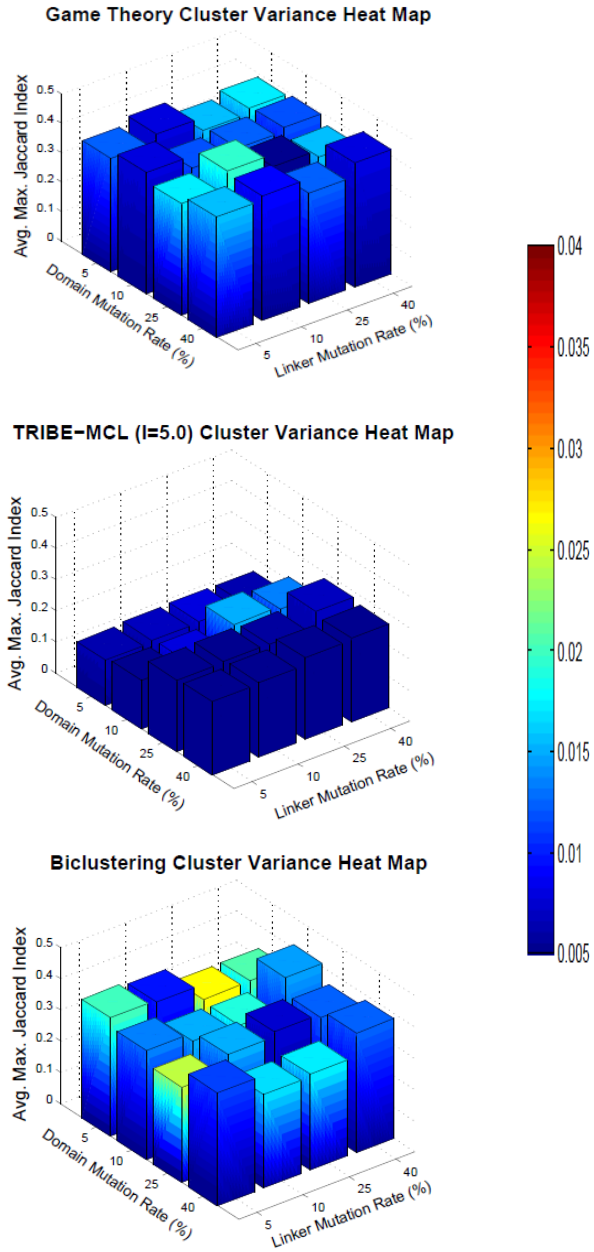


Figure A.10: **Cluster Variance Heat Maps (15 Generations of Simulated Proteins).** For each linker mutation rate (% shown along the y-axis) and domain mutation rate (% shown along the x-axis), 25 sets of proteins were simulated. Using these data sets, clusters were obtained by game theory (GT), biclustering (BICL), and TRIBE-MCL (MCL). The inflation parameter used in TRIBE-MCL was $I = 5.0$. Clustering accuracy was assessed using the average maximum Jaccard index. The height of each bar gives the mean of the average maximum Jaccard indices for the 25 simulations performed using the corresponding domain and linker mutation rates. The face color of each bar shows the variance in average maximum Jaccard index for the 25 data sets.

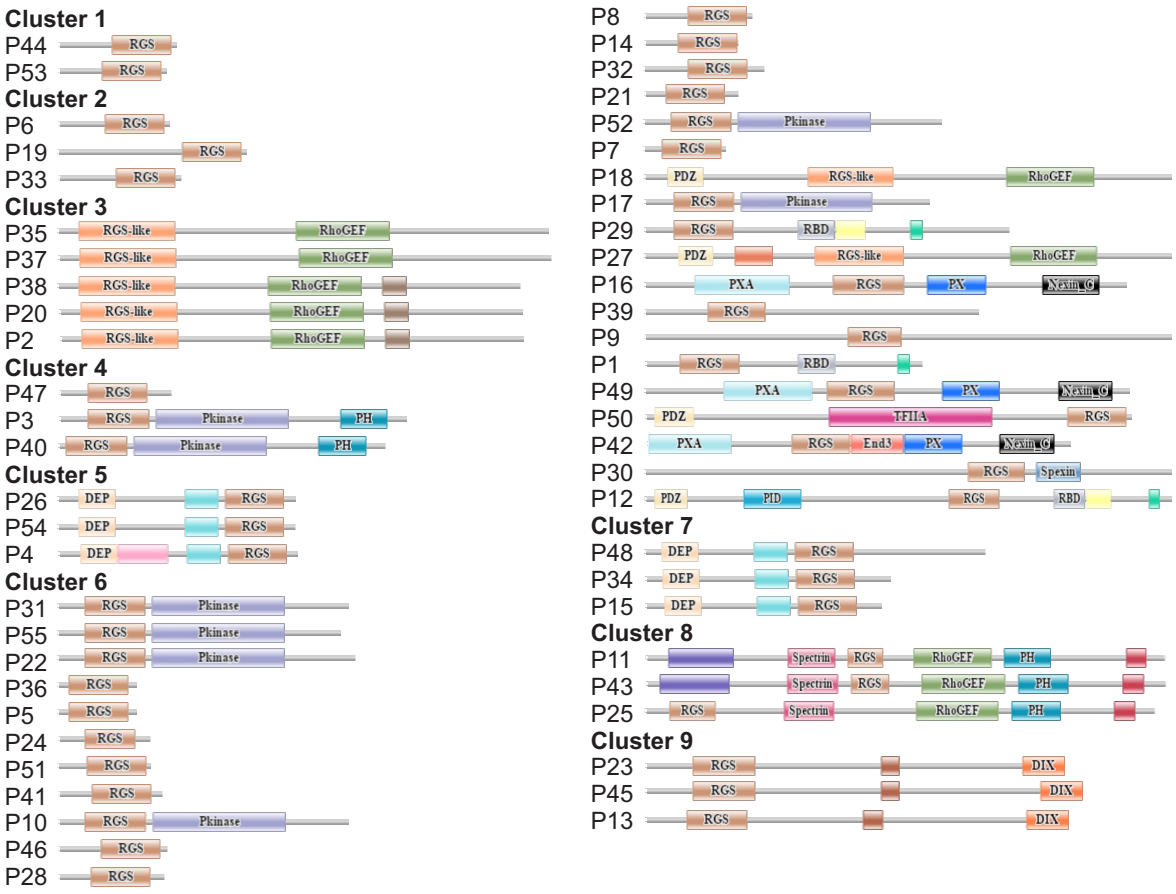


Figure A.11: **RGS Family Domain Architectures (Non-Overlapping Domains).** The non-overlapping domain architectures for the RGS family proteins identified by HMMER. The image for each protein was generated using the domain graphic generator on the Pfam website (`http://pfam.xfam.org/generate_graphic`).

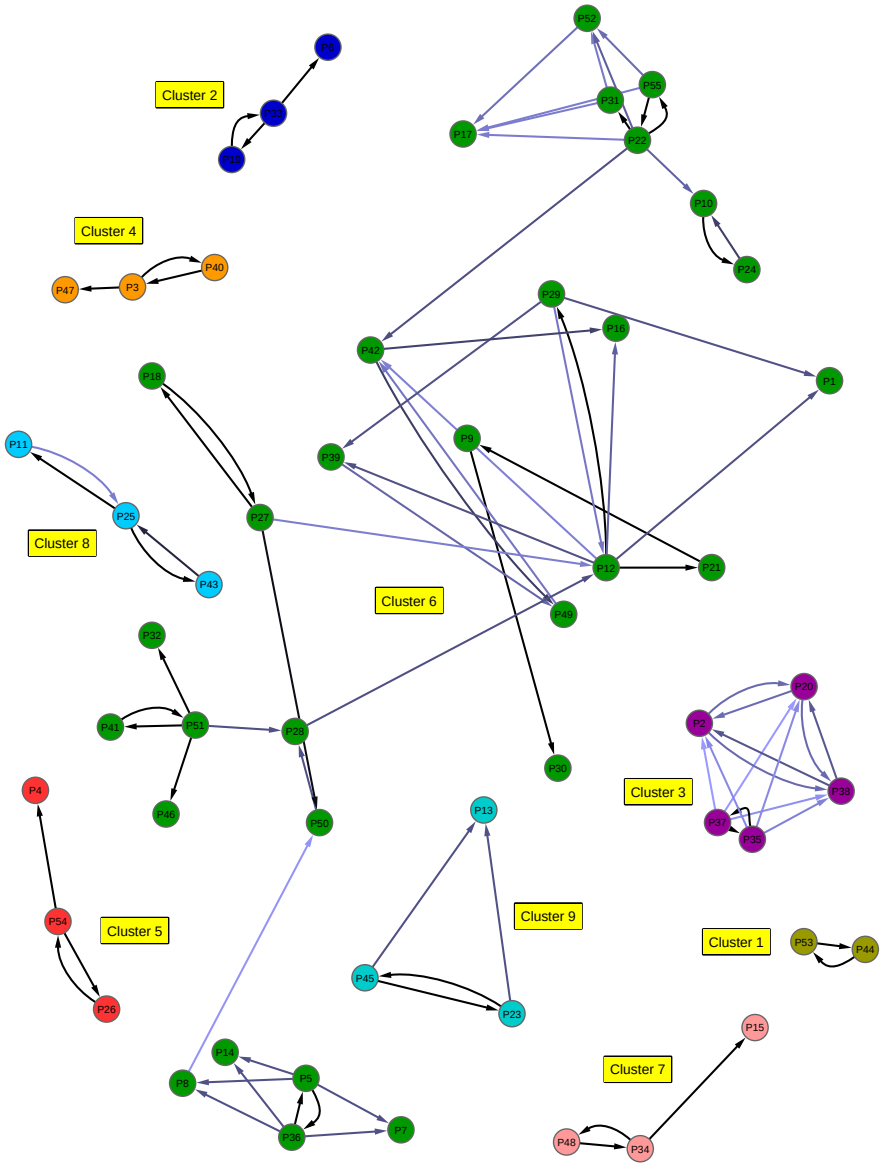


Figure A.12: **RGS Family Game-Theoretic Protein Similarity Network (5% Overlap Domains).** From 55 mouse RGS family proteins, using the 5% overlap domain detection strategy, 9 clusters were identified using the game theory method. The nodes represent distinct proteins and the edges are directed so that the incoming edge weights of each node sum to 1. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. The game value for each protein is represented in the network by the length of its incoming edges, with longer edges corresponding to small game values and longer edges indicating high game values. Clusters in the network, represented by different node colors, are labeled in descending order according to their average game value, i.e. Cluster 1 has the largest average game value. See Fig. A.14 for the domain profiles for the proteins.

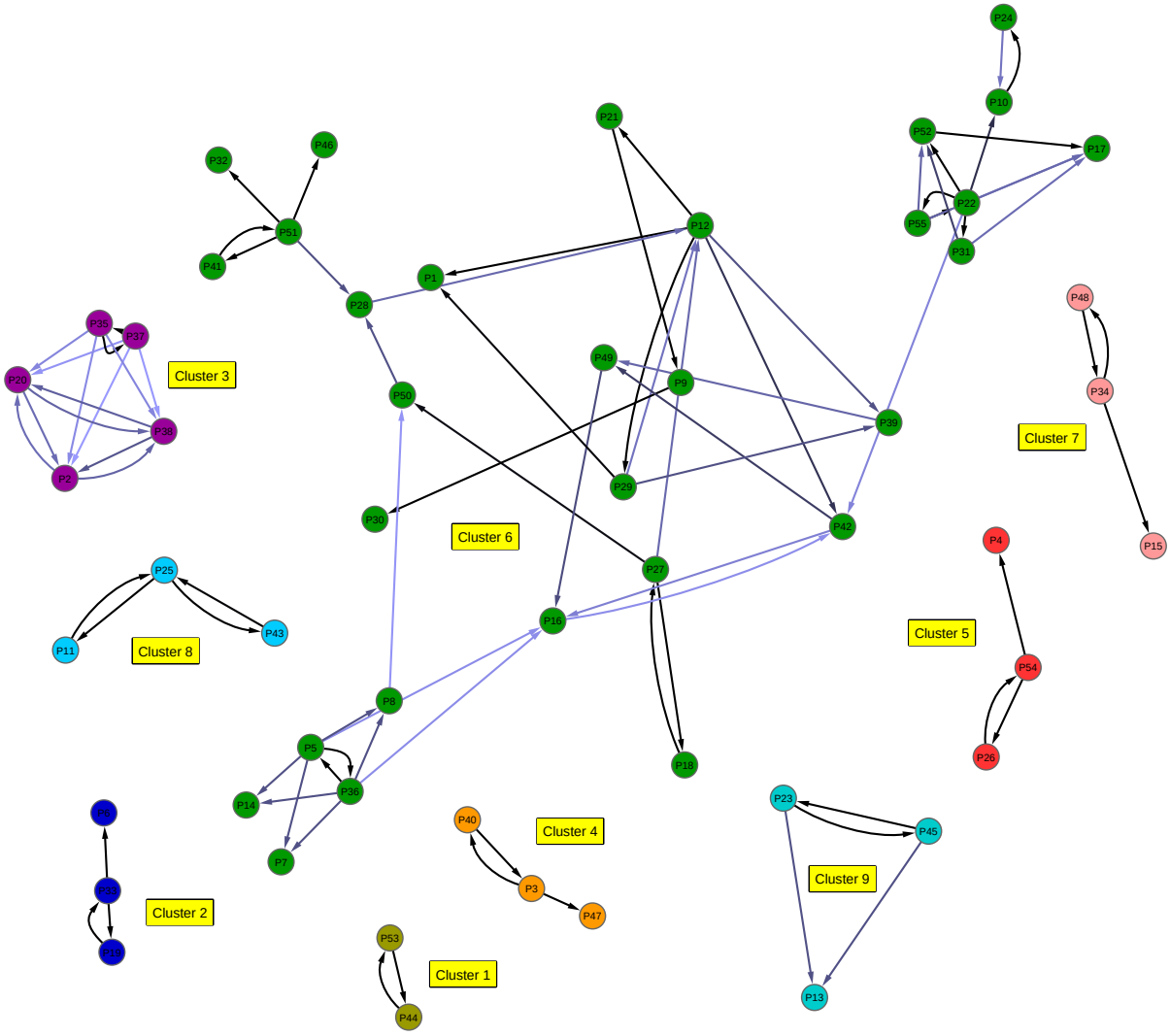


Figure A.13: **RGS Family Game-Theoretic Protein Similarity Network (Unrestricted Overlap Domains).** From 55 mouse RGS family proteins, using the unrestricted overlap domain detection strategy, 9 clusters were identified using the game theory method. The nodes represent distinct proteins and the edges are directed so that the incoming edge weights of each node sum to 1. The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. The game value for each protein is represented in the network by the length of its incoming edges, with longer edges corresponding to small game values and longer edges indicating high game values. Clusters in the network, represented by different node colors, are labeled in descending order according to their average game value, i.e. Cluster 1 has the largest average game value. See Fig. A.15 for the domain profiles for the proteins.

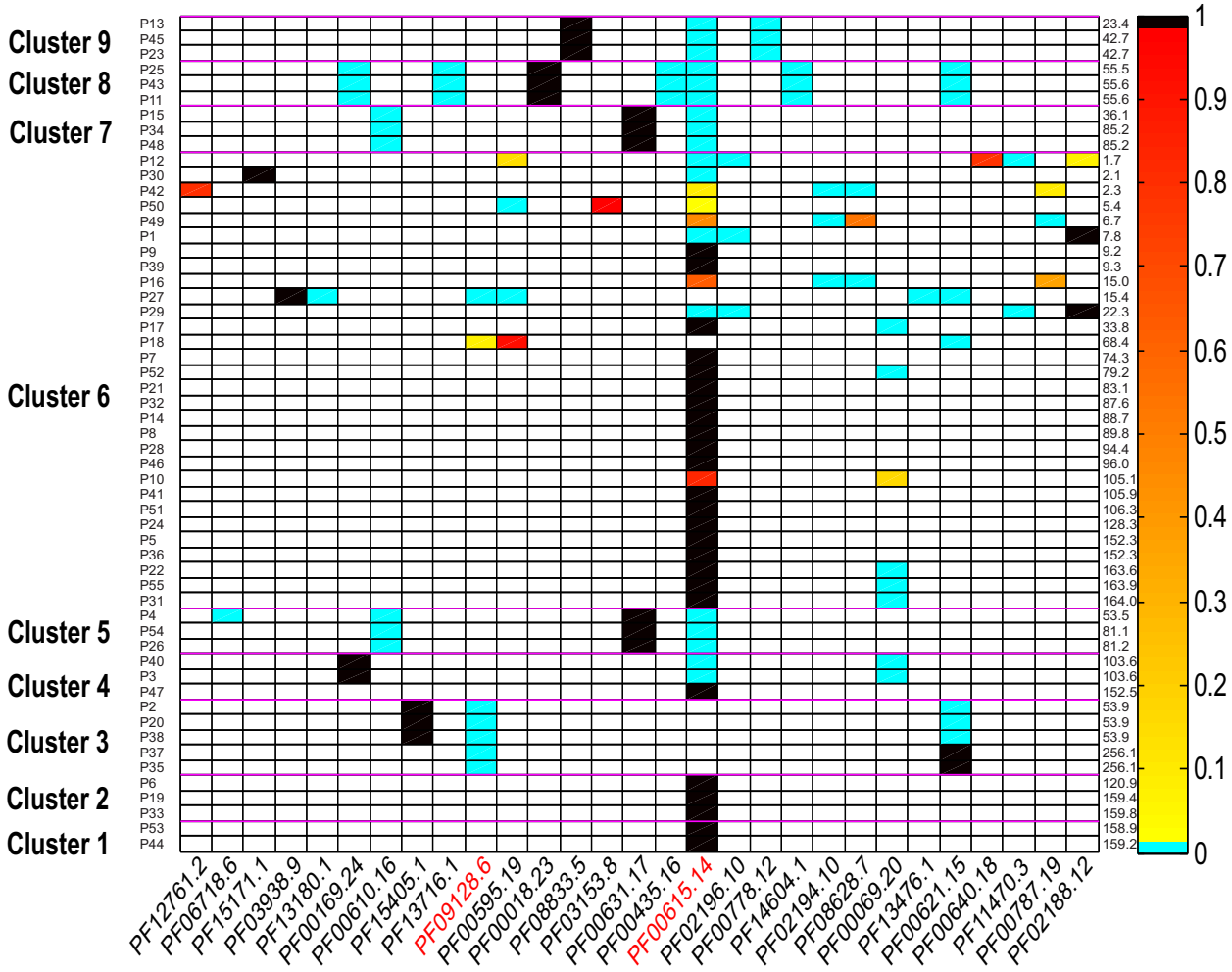


Figure A.14: **RGS Family Domain Profile (5% Overlap Domains).** In this diagram, each row represents one of the 55 RGS proteins, while each column represents one of 29 Pfam domains (RGS and RGS-like domains are shown in red). Each cell is color-coded based on the diversity weights x: black for $x_i = 1$, cyan for $x_i = 0$ (but domain exists), and from dark to light orange for $1 > x_i > 0$. Blank cells indicate domain absence. The clusters are separated by horizontal magenta lines. Proteins in each cluster (identifers shown on the right) are arranged according to their game value, with the largest appearing as the lowest row in the cluster. The game values for each protein are shown on the left.

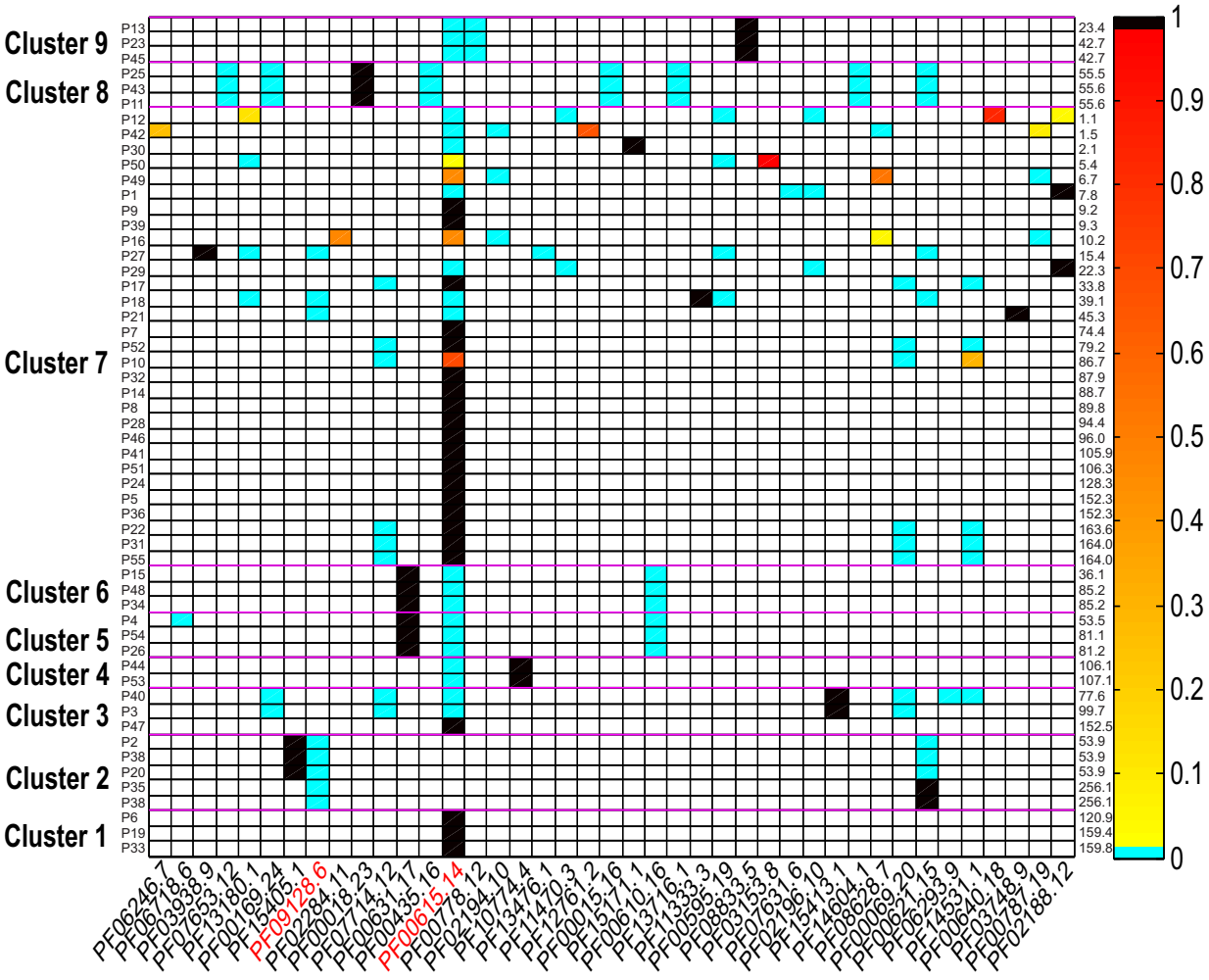


Figure A.15: **RGS Family Domain Profile (Unrestricted Overlap Domains).** In this diagram, each row represents one of the 55 RGS proteins, while each column represents one of 41 Pfam domains. Each cell is color-coded based on the diversity weights x: black for $x_i = 1$, cyan for $x_i = 0$ (but domain exists), and from dark to light orange for $1 > x_i > 0$. Blank cells indicate domain absence. The clusters are separated by horizontal magenta lines. Proteins in each cluster (identifers shown on the right) are arranged according to their game value, with the largest appearing as the lowest row in the cluster. The game values for each protein are shown on the left.

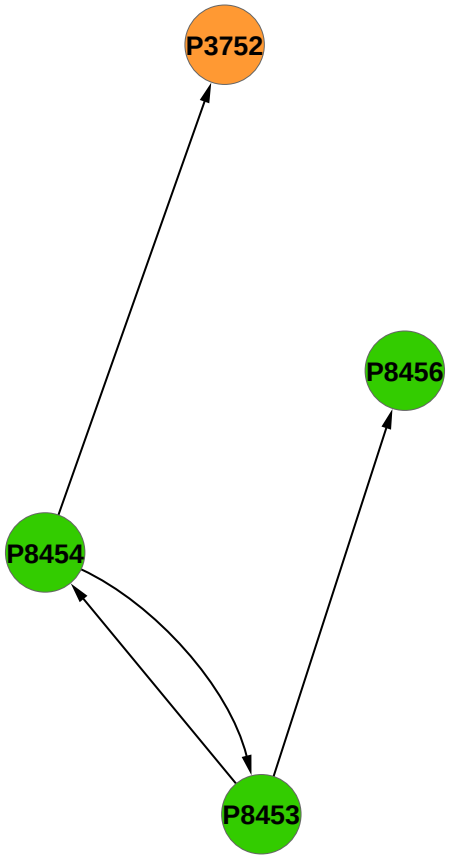


Figure A.16: **RGS Proteins In Swiss-Prot Mouse Proteome.** Twenty one of the 66 RGS proteins discussed in Section 7.2 were also found in the Swiss-Prot mouse data set (see Table A.16). This figure shows one cluster in the game theory similarity network for the Swiss-Prot mouse proteome. Three of the four proteins (shown in green) belong to the RGS data set.

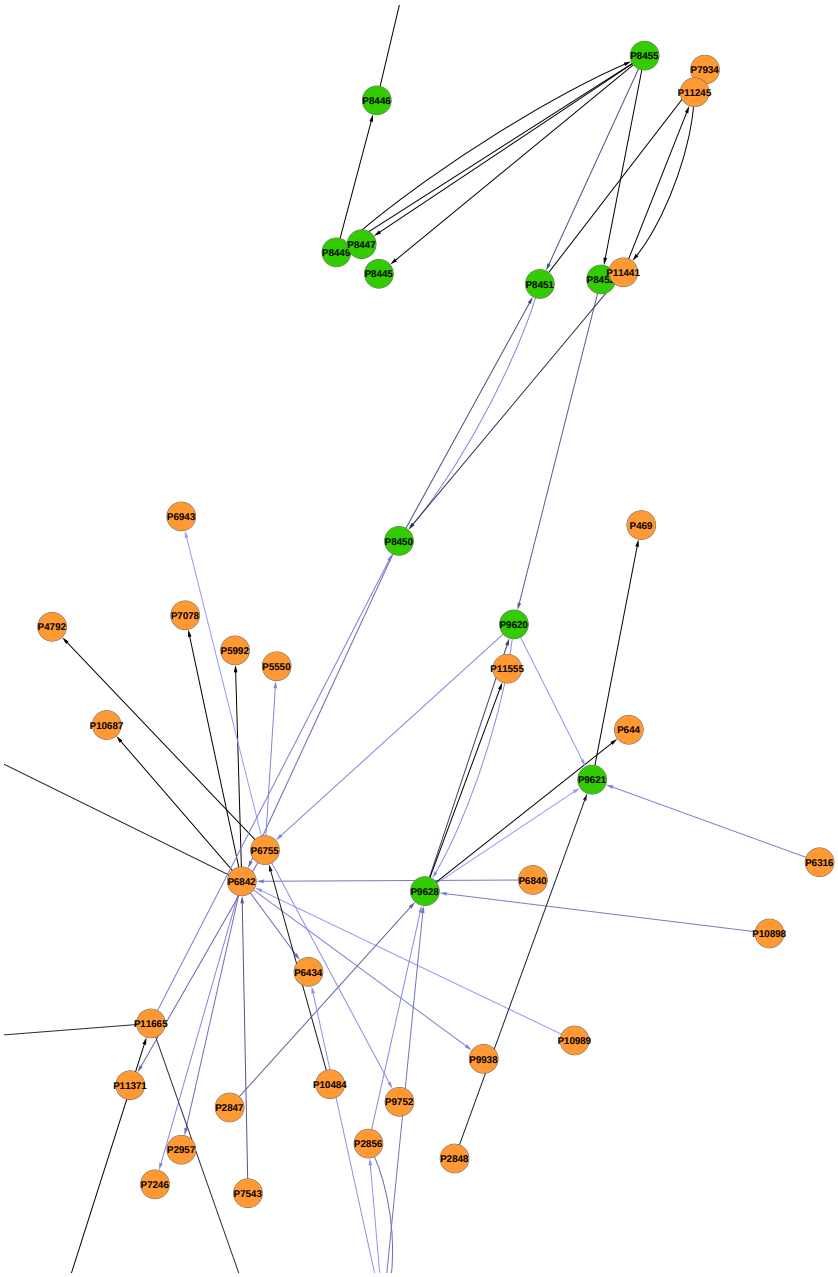


Figure A.17: **RGS Proteins In Swiss-Prot Mouse Proteome.** Twenty one of the 66 RGS proteins discussed in Section 7.2 were also found in the Swiss-Prot mouse data set (see Table A.16). This figure shows a small neighborhood in the largest cluster from the game theory similarity network for the Swiss-Prot mouse proteome. Eleven of the proteins (shown in green) belong to the RGS data set.

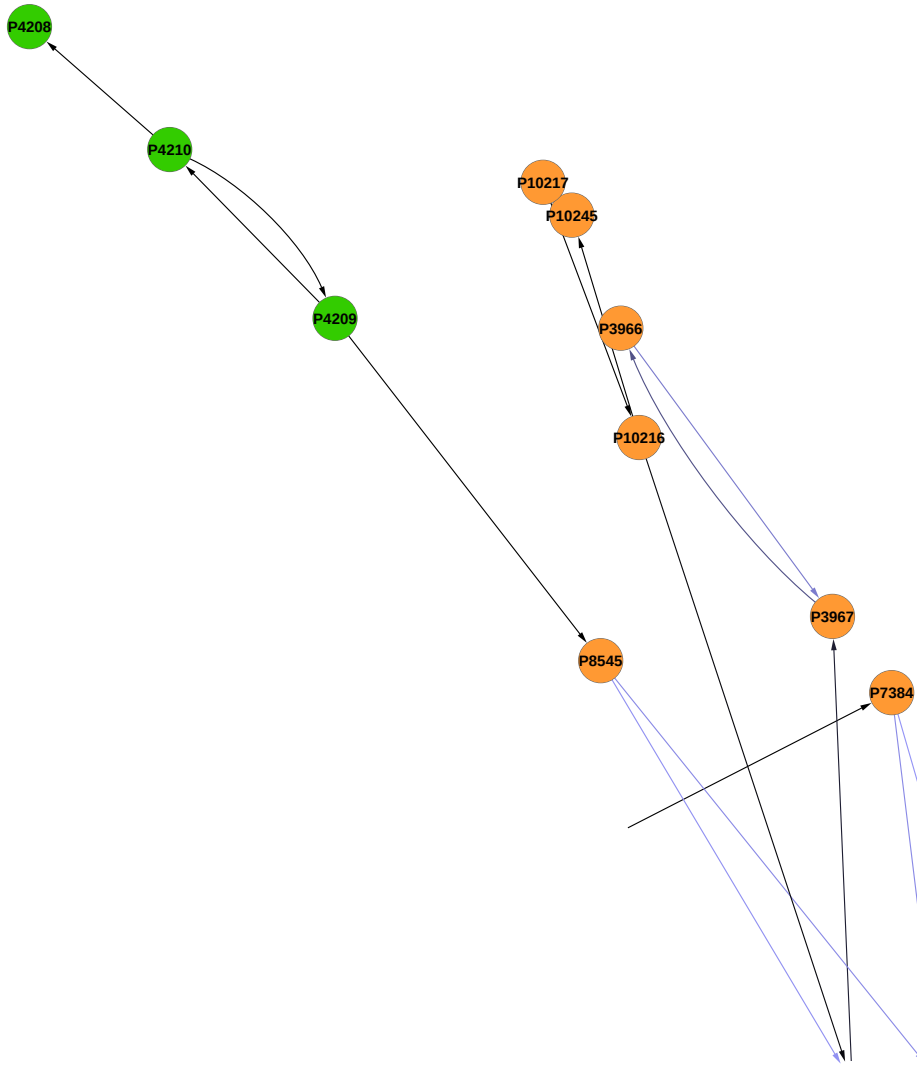


Figure A.18: **RGS Proteins In Swiss-Prot Mouse Proteome.** Twenty one of the 66 RGS proteins discussed in Section 7.2 were also found in the Swiss-Prot mouse data set (see Table A.16). This figure shows a small neighborhood in the largest cluster from the game theory similarity network for the Swiss-Prot mouse proteome. Three of the proteins (shown in green) belong to the RGS data set.

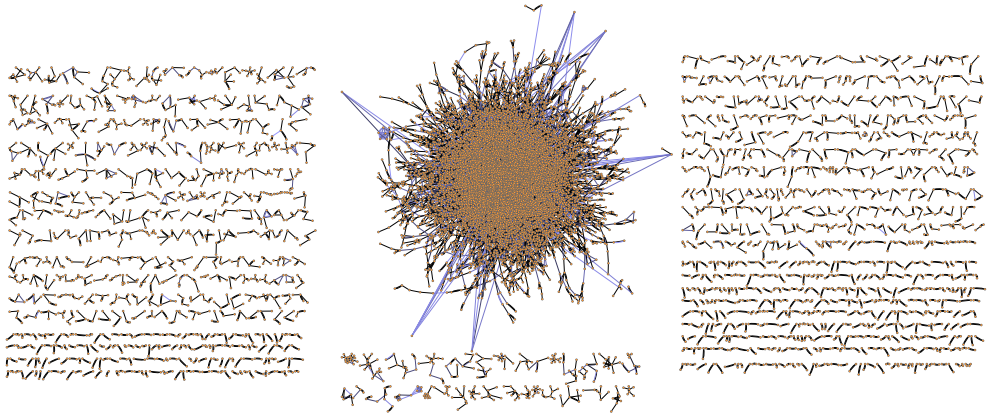


Figure A.19: **Game-Theoretic Protein Similarity Network (5% Overlap Domains) For the Swiss-Prot Mouse Proteome.** A total of 1,201 clusters were identified for the 11,920 proteins (orange nodes). The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. The game value for each protein is represented in the network by the length of its incoming edges, with longer edges corresponding to small game values and longer edges indicating high game values.

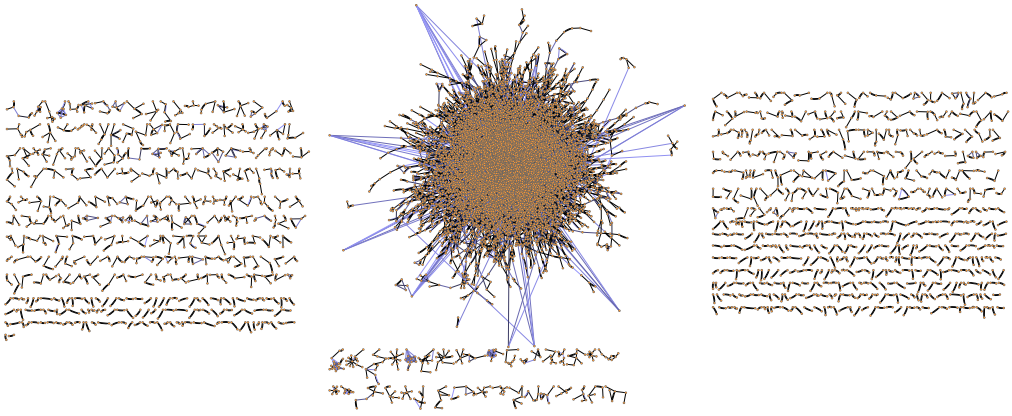


Figure A.20: **Game-Theoretic Protein Similarity Network (Unrestricted Overlap Domains) For the Swiss-Prot Mouse Proteome.** A total of 909 clusters were identified for the 11,920 proteins (orange nodes). The edge color (in varying shades of blue) indicates the edge weight, with darker edges (black) indicating high weights and lighter edges indicating low weights. The game value for each protein is represented in the network by the length of its incoming edges, with longer edges corresponding to small game values and longer edges indicating high game values.

A.2 Tables

Pfam Accession	Domain Length		Pfam Accession	Domain Length
PF02770	59		PF00516	417
PF11744	457		PF00512	69
PF01979	323		PF12728	51
PF13193	76		PF12701	97
PF12796	71		PF10633	75
PF00528	190		PF00291	343
PF00439	80		PF01427	49
PF09806	119		PF02880	111
PF01609	185		PF01242	117
PF12166	305		PF00072	112
PF00113	290		PF14259	60
PF04324	56		PF13181	29
PF01827	149		PF00486	77
PF01571	206		PF00418	30
PF00534	171		PF13465	26

Table A.1: **Pfam Domain Bank Used For Simulation Study.**

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	216	.00001	.1721	.0345
PF00113	68	.00001	.0948	.0237
PF00291	105	.00001	.3117	.0446
PF00418	359	.00001	.3521	.0597
PF00439	169	.00001	.2043	.0357
PF00486	139	.00001	.1828	.0348
PF00512	186	.00001	.1741	.0222
PF00516	89	.00001	.1492	.0161
PF00528	123	.00001	.2020	.0323
PF00534	929	.00001	.3547	.0770
PF01242	171	.00001	.1893	.0322
PF01427	131	.00001	.2437	.0267
PF01571	136	.00001	.2901	.0575
PF01609	223	.00001	.1832	.0405
PF01827	125	.00001	.1577	.0352
PF01979	114	.00001	.1421	.0308
PF02770	3198	.00001	.3846	.0718
PF02880	104	.00001	.1252	.0221
PF04324	178	.00001	.2440	.0343
PF09806	1099	.00001	.3598	.0750
PF10633	257	.00001	.2283	.0532
PF11744	80	.00001	.2228	.0442
PF12166	91	.00001	.1576	.0317
PF12701	109	.00001	.2256	.0555
PF12728	184	.00001	.2394	.0318
PF12796	125	.00001	.1637	.0177
PF13181	247	.00001	.3222	.0436
PF13193	159	.00001	.1770	.0342
PF13465	183	.00001	.3434	.0581
PF14259	132	.00001	.1797	.0328

Table A.2: **Domain Sequence Divergence For Simulation Study (10 generations, 5% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	292	.00001	.5927	.0995
PF00113	110	.00001	.3084	.0461
PF00291	85	.00001	.3014	.0592
PF00418	271	.00001	.6572	.0814
PF00439	167	.00001	.5908	.0805
PF00486	98	.00001	.5071	.0486
PF00512	273	.00001	.4335	.0610
PF00516	84	.00001	.3921	.0713
PF00528	114	.00001	.2203	.0552
PF00534	638	.00001	.6484	.1547
PF01242	134	.00001	.3176	.0470
PF01427	202	.00001	.4150	.0635
PF01571	125	.00001	.3723	.0631
PF01609	95	.00001	.2409	.0375
PF01827	155	.00001	.3346	.0820
PF01979	103	.00001	.4162	.0757
PF02770	2867	.00001	.7103	.1467
PF02880	147	.00001	.4139	.0534
PF04324	193	.00001	.4868	.0621
PF09806	1405	.00001	.6239	.1298
PF10633	152	.00001	.3059	.0509
PF11744	128	.00001	.4149	.1166
PF12166	97	.00001	.3710	.0795
PF12701	123	.00001	.3735	.0584
PF12728	245	.00001	.3134	.0438
PF12796	148	.00001	.4204	.0406
PF13181	242	.00001	.3779	.0599
PF13193	141	.00001	.3785	.0523
PF13465	250	.00001	.5726	.0906
PF14259	162	.00001	.3288	.0447

Table A.3: **Domain Sequence Divergence For Simulation Study (10 generations, 10% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	188	.00001	.8741	.2121
PF00113	105	.00001	1.1403	.1741
PF00291	105	.00001	.8921	.2367
PF00418	285	.00001	1.2044	.1114
PF00439	163	.00001	.7981	.1821
PF00486	206	.00001	.7507	.1817
PF00512	176	.00001	.7429	.1567
PF00516	92	.00001	1.0311	.1805
PF00528	106	.00001	1.1956	.2086
PF00534	820	.00001	1.6910	.3053
PF01242	118	.00001	.7781	.1303
PF01427	165	.00001	.7812	.1053
PF01571	99	.00001	.8865	.1947
PF01609	157	.00001	.7415	.1407
PF01827	136	.00001	1.0805	.2198
PF01979	105	.00001	.5625	.1228
PF02770	2795	.00001	1.4419	.2849
PF02880	187	.00001	.5464	.1254
PF04324	216	.00001	1.0056	.1551
PF09806	1285	.00001	1.2243	.2809
PF10633	114	.00001	.5334	.1374
PF11744	97	.00001	.8168	.2159
PF12166	129	.00001	1.0245	.2394
PF12701	107	.00001	.9542	.1290
PF12728	274	.00001	1.0581	.2192
PF12796	180	.00001	.9818	.1451
PF13181	216	.00001	1.4731	.1804
PF13193	188	.00001	1.1947	.2359
PF13465	293	.00001	1.6849	.1576
PF14259	182	.00001	1.2385	.1529

Table A.4: **Domain Sequence Divergence For Simulation Study (10 generations, 25% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	166	.00001	1.9835	.3101
PF00113	105	.00001	1.1790	.1987
PF00291	104	.00001	1.2815	.2610
PF00418	131	.00001	1.4427	.3166
PF00439	182	.00001	2.2229	.3105
PF00486	160	.00001	1.4399	.2371
PF00512	283	.00001	1.7876	.3182
PF00516	85	.00001	1.6101	.3261
PF00528	111	.00001	1.7047	.3072
PF00534	851	.00001	2.7046	.5324
PF01242	95	.00001	.9996	.2301
PF01427	181	.00001	1.9270	.3338
PF01571	114	.00001	1.1002	.2341
PF01609	104	.00001	1.8162	.2556
PF01827	181	.00001	1.2179	.2714
PF01979	108	.00001	1.5831	.3055
PF02770	2688	.00001	3.5951	.5377
PF02880	171	.00001	1.0860	.2087
PF04324	163	.00001	1.3672	.1267
PF09806	823	.00001	2.1084	.5241
PF10633	199	.00001	1.7192	.2101
PF11744	78	.00001	.8478	.2293
PF12166	106	.00001	1.3340	.2317
PF12701	184	.00001	2.6217	.2613
PF12728	146	.00001	.8739	.1883
PF12796	210	.00001	2.0612	.2849
PF13181	213	.00001	1.8760	.1982
PF13193	169	.00001	1.2438	.1900
PF13465	206	.00001	.5726	.0899
PF14259	167	.00001	.9294	.1265

Table A.5: **Domain Sequence Divergence For Simulation Study (10 generations, 40% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	1092	.00001	.4058	.0913
PF00113	412	.00001	.2286	.0673
PF00291	496	.00001	.3104	.0716
PF00418	1688	.00001	.5500	.0896
PF00439	750	.00001	.3198	.0544
PF00486	1452	.00001	.3956	.0771
PF00512	947	.00001	.3960	.0531
PF00516	388	.00001	.3648	.0784
PF00528	542	.00001	.3043	.0738
PF00534	3228	.00001	.4953	.1217
PF01242	1530	.00001	.3253	.0735
PF01427	1445	.00001	.3266	.0707
PF01571	874	.00001	.2853	.0780
PF01609	586	.00001	.2941	.0680
PF01827	788	.00001	.3182	.0639
PF01979	499	.00001	.3333	.0800
PF02770	16204	.00001	.5819	.1120
PF02880	844	.00001	.3136	.0648
PF04324	1878	.00001	.4656	.0873
PF09806	5979	.00001	.4810	.1106
PF10633	1327	.00001	.3967	.0703
PF11744	377	.00001	.3406	.0741
PF12166	600	.00001	.3151	.0739
PF12701	1039	.00001	.2775	.0769
PF12728	1710	.00001	.3527	.0697
PF12796	969	.00001	.4648	.0773
PF13181	2342	.00001	.4912	.0836
PF13193	844	.00001	.2843	.0632
PF13465	2239	.00001	.4881	.0851
PF14259	1603	.00001	.4215	.0728

Table A.6: **Domain Sequence Divergence For Simulation Study (15 generations, 5% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	733	.00001	.5408	.1155
PF00113	415	.00001	.5492	.1337
PF00291	472	.00001	.6747	.1505
PF00418	1836	.00001	1.2456	.1318
PF00439	1466	.00001	.6510	.1438
PF00486	1039	.00001	.6896	.1528
PF00512	1134	.00001	.6907	.1097
PF00516	565	.00001	.5747	.1637
PF00528	547	.00001	.5109	.1234
PF00534	3311	.00001	.8435	.2120
PF01242	932	.00001	.5227	.1227
PF01427	1544	.00001	.7476	.1243
PF01571	474	.00001	.5644	.1333
PF01609	650	.00001	.7300	.1463
PF01827	717	.00001	.5591	.1280
PF01979	430	.00001	.8273	.1534
PF02770	14918	.00001	1.0764	.1995
PF02880	1245	.00001	.6382	.1589
PF04324	1388	.00001	.6237	.1258
PF09806	5866	.00001	1.0638	.2062
PF10633	1079	.00001	.7911	.1435
PF11744	392	.00001	.5696	.1459
PF12166	479	.00001	.6433	.1781
PF12701	1041	.00001	.7591	.1238
PF12728	1395	.00001	.6268	.1422
PF12796	1067	.00001	.5692	.1162
PF13181	1819	.00001	.7246	.1442
PF13193	1349	.00001	.6813	.1427
PF13465	1541	.00001	.8756	.1431
PF14259	1679	.00001	.6406	.1598

Table A.7: **Domain Sequence Divergence For Simulation Study (15 generations, 10% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	1053	.00001	1.6188	.3630
PF00113	493	.00001	1.6496	.3873
PF00291	483	.00001	1.6658	.3562
PF00418	1381	.00001	2.4142	.3745
PF00439	841	.00001	1.2248	.2819
PF00486	890	.00001	1.2955	.2688
PF00512	1202	.00001	1.4160	.3046
PF00516	287	.00001	1.2925	.3083
PF00528	626	.00001	1.3498	.3254
PF00534	4092	.00001	2.2503	.5562
PF01242	1192	.00001	2.0210	.4148
PF01427	1384	.00001	1.5072	.2451
PF01571	607	.00001	1.4329	.3520
PF01609	724	.00001	1.4936	.3427
PF01827	821	.00001	1.5797	.3294
PF01979	447	.00001	1.4111	.3823
PF02770	16764	.00001	2.7732	.4721
PF02880	1081	.00001	1.6471	.3253
PF04324	1228	.00001	1.3597	.2306
PF09806	6157	.00001	2.2985	.5130
PF10633	1401	.00001	1.5854	.3665
PF11744	320	.00001	1.1888	.2637
PF12166	656	.00001	1.5326	.3740
PF12701	578	.00001	1.3839	.2246
PF12728	1200	.00001	2.1197	.2792
PF12796	1527	.00001	1.8043	.3056
PF13181	1433	.00001	1.5268	.2883
PF13193	1083	.00001	1.2918	.2923
PF13465	1262	.00001	1.9528	.3686
PF14259	978	.00001	1.4898	.2565

Table A.8: **Domain Sequence Divergence For Simulation Study (15 generations, 25% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Pfam Accession	# Observations	Min. Distance	Max. Distance	Avg. Distance
PF02770	794	.00001	2.8696	.3979
PF00113	494	.00001	2.1698	.5421
PF00291	587	.00001	2.6268	.6126
PF00418	1425	.00001	2.7114	.4092
PF00439	1506	.00001	2.8624	.5506
PF00486	915	.00001	2.4956	.4396
PF00512	1049	.00001	2.0375	.3760
PF00516	386	.00001	2.7561	.6586
PF00528	575	.00001	2.2565	.5428
PF00534	4558	.00001	3.2343	.8539
PF01242	1403	.00001	2.8605	.5896
PF01427	1366	.00001	2.3335	.5269
PF01571	624	.00001	4.6530	.7246
PF01609	700	.00001	2.1328	.5438
PF01827	725	.00001	2.0549	.5422
PF01979	402	.00001	2.3132	.4405
PF02770	18786	.00001	5.6419	.7800
PF02880	963	.00001	2.6428	.4993
PF04324	989	.00001	2.2610	.4199
PF09806	7709	.00001	4.9030	.8293
PF10633	1323	.00001	2.3718	.5131
PF11744	389	.00001	2.6458	.5843
PF12166	494	.00001	2.1157	.6153
PF12701	1270	.00001	2.6350	.5074
PF12728	2207	.00001	2.5400	.5242
PF12796	1520	.00001	2.6481	.5056
PF13181	2086	.00001	3.0850	.4338
PF13193	1112	.00001	2.1758	.5644
PF13465	1060	.00001	2.2210	.4891
PF14259	998	.00001	2.4839	.4272

Table A.9: **Domain Sequence Divergence For Simulation Study (15 generations, 40% Domain Mutation Rate).** The domain sequence divergence was calculated for each of the domains present in the simulated proteins by aligning the domain to the original domain sequence (from the domain bank at the start of the simulation) and using the `protdist` function in the Phylip [38] package to compute the distance using the JTT amino acid substitution model. The number of observations of each domain is shown, along with the minimum, maximum, and average distance from the root domain sequence, computed by Phylip.

Protein Identifier	**Pfam Domains**
P1	PF00615.14, PF02196.10, PF02188.12
P2, P20, P38	PF15405.1, PF09128.6, PF00621.15
P3, P40	PF00169.24, PF00615.14, PF00069.20
P4	PF06718.6, PF00610.16, PF00631.17, PF00615.14
P5, P6, P7, P8, P9, P14, P19, P21, P24 P28, P32, P33, P36, P39, P41, P44, P46, P47 P51, P53	PF00615.14
P10, P17, P22, P31, P52, P55	PF00615.14, PF00069.20
P11, P25, P43	PF00169.24, PF13716.1, PF00018.23 PF00435.16, PF00615.14, PF00621.15
P12	PF00595.19, PF00615.14, PF02196.10 PF00640.18, PF02188.12, PF11470.3
P13, P23, P45	PF08833.5, PF00615.14, PF00778.12
P15, P26, P34, P48, P54	PF00610.16, PF00631.17, PF00615.14
P16	PF00615.14, PF02194.10, PF08628.7, PF00787.19
P18	PF09128.6, PF00595.19, PF00621.15
P27	PF03938.9, PF09128.6, PF00595.19, PF00621.15
P29	PF00615.14, PF02196.10, PF02188.12, PF11470.3
P30	PF15171.1, PF00615.14
P35, P37	PF09128.6, PF00621.15
P42	PF12761.2, PF00615.14, PF02194.10 PF08628.7, PF00787.19
P49	PF00615.14, PF02194.10, PF08628.7, PF00787.19
P50	PF00595.19, PF03153.8, PF00615.14

Table A.10: **Non-Overlapping HMMER Domain Architectures for RGS Protein Family.**

Protein Identifier	**Pfam Domains**
P1	PF00615.14, PF02196.10, PF02188.12
P2, P20, P38	PF15405.1, PF09128.6, PF00621.15
P3, P40	PF00169.24, PF00615.14, PF00069.20
P4	PF06718.6, PF00610.16, PF00631.17 PF00615.14
P5, P6, P7, P8, P9, P14, P19, P21, P24 P28, P32, P33, P36, P39, P41, P44, P46, P47 P51, P53	PF00615.14
P10, P17, P22, P31, P52, P55	PF00615.14, PF00069.20
P11, P25, P43	PF00169.24, PF13716.1, PF00018.23, PF00435.16 PF00615.14, PF14604.1, PF00621.15
P12	PF00595.19, PF00615.14, PF02196.10 PF00640.18, PF11470.3, PF02188.12
P13, P23, P45	PF08833.5, PF00615.14, PF00778.12
P15, P26, P34, P48, P54	PF00610.16, PF00631.17, PF00615.14
P16, P49	PF00615.14, PF02194.10, PF08628.7, PF00787.19
P18	PF09128.6, PF00595.19, PF00621.15
P27	PF03938.9, PF13180.1, PF09128.6 PF00595.19, PF13476.1, PF00621.15
P29	PF00615.14, PF02196.10, PF11470.3, PF02188.12
P30	PF15171.1, PF00615.14
P35, P37	PF09128.6, PF00621.15
P42	PF12761.2, PF00615.14, PF02194.10 PF08628.7, PF00787.19
P50	PF00595.19, PF03153.8, PF00615.14

Table A.11: **Restricted Overlap (5%) HMMER Domain Architectures for RGS Protein Family.**

Protein Identifier	**Pfam Domains**
P1	PF00615.14, PF07631.6, PF02196.10, PF02188.12
P2, P20, P38	PF15405.1, PF09128.6, PF00621.15
P3	PF00169.24, PF00615.14, PF00069.20 PF07714.12, PF15413.1
P4	PF06718.6, PF00631.17, PF00615.14, PF00610.16
P5, P6, P7, P8, P9, P14, P19, P24, P28 P32, P33, P36, P39, P41, P46, P47, P51	PF00615.14
P10, P17, P22, P31, P52, P55	PF00615.14, PF00069.20, PF07714.12, PF14531.1
P11, P25, P43	PF00169.24, PF00018.23, PF00435.16, PF00015.16 PF13716.1, PF14604.1, PF00621.15, PF00615.14
P12	PF00615.14, PF11470.3, PF00595.19, PF02196.10 PF00640.18, PF02188.12, PF13180.1
P13, P23, P45	PF00615.14, PF00778.12, PF08833.5
P15, P26, P34, P48, P54	PF00631.17, PF00615.14, PF00610.16
P16	PF00615.14, PF02194.10, PF08628.7 PF00787.19, PF02284.11
P18	PF09128.6, PF00595.19, PF11333.3 PF00621.15, PF13180.1, PF00615.14
P21	PF09128.6, PF00615.14, PF03748.9
P27	PF03938.9, PF13180.1, PF09128.6 PF13476.1, PF00595.19, PF00621.15
P29	PF00615.14, PF11470.3, PF02196.10, PF02188.12
P30	PF00615.14, PF15171.1
P35, P37	PF09128.6, PF00621.15
P40	PF00169.24, PF00615.14, PF00069.20, PF07714.12 PF15413.1, PF06293.9, PF14531.1
P42	PF06246.7, PF00615.14, PF02194.10 PF12761.2, PF08628.7, PF00787.19
P44, P53	PF00615.14, PF10774.4
P49	PF00615.14, PF02194.10, PF08628.7, PF00787.19
P50	PF00615.14, PF00595.19, PF03153.8, PF13180.1

Table A.12: **Unrestricted Overlap HMMER Domain Architectures for RGS Protein Family.**

Cluster (AGV)	Node	NCBI Accession	Protein Type
1 (159.0567)	P44	NP_001155294.1	RGS 17 isoform 1
	P53	NP_064342.1	RGS 17 isoform 2
2 (146.7074)	P33	NP_067349.2	RGS 20 isoform 2
	P19	NP_001171266.1	RGS 20 isoform 1
	P6	NP_080722.1	RGS 19
3 (134.7646)	P38	NP_032514.1	Rho guanine nuc. ex. factor 1 isoform d
	P37	NP_001123622.1	Rho guanine nuc. ex. factor 1 isoform a
	P35	NP_001123623.1	Rho guanine nuc. ex. factor 1 isoform b
	P20	NP_001123625.1	Rho guanine nuc. ex. factor 1 isoform c
	P2	NP_001123624.1	Rho guanine nuc. ex. factor 1 isoform c
4 (119.9047)	P3	NP_796052.2	Beta-adrenergic receptor kinase 2 isoform 1
	P40	NP_570933.1	Beta-adrenergic receptor kinase 1
	P47	NP_001030608.1	Beta-adrenergic receptor kinase 2 isoform 2
5 (71.9517)	P26	NP_001185932.1	RGS 7 isoform 2
	P54	NP_036010.2	RGS 7 isoform 1
	P4	NP_056627.1	RGS 6
6 (71.1609)	P31	NP_001106182.1	GPCR kinase 6 isoform c
	P55	NP_036068.2	GPCR kinase 6 isoform b
	P22	NP_001033107.1	GPCR kinase 6 isoform a
	P36	XP_894544.3	PREDICTED: RGS 21
	P5	XP_921002.3	PREDICTED: RGS 21
	P24	NP_001074212.1	GPCR kinase 4 isoform 2
	P51	NP_080656.2	RGS 8
	P41	NP_035397.2	RGS 16
	P10	NP_062370.2	GPCR kinase 4 isoform 1
	P46	NP_033087.2	RGS 2
	P28	NP_033088.2	RGS 4
	P8	NP_056626.2	RGS 1
	P14	NP_033089.2	RGS 5
	P32	NP_075019.1	RGS 18
	P21	NP_080694.1	RGS 10
	P52	NP_061357.3	GPCR kinase 5
	P7	NP_694811.1	RGS 13
	P18	NP_001003912.1	Rho guanine nuc. ex. factor 11
	P17	NP_036011.3	Rhodopsin kinase precursor
	P16	NP_001014973.2	Sorting nexin-13
	P27	NP_081420.2	Rho guanine nuc. ex. factor 12
	P39	NP_064305.2	A-kinase anchor protein 10, mit. precursor
	P9	NP_001182677.1	RGS 22
	P1	NP_058038.2	RGS 14
	P49	NP_766514.2	Sorting nexin-14
	P50	NP_599018.3	RGS 3 isoform 2
	P29	NP_001156984.1	RGS 12 isoform B
	P42	NP_997096.2	Sorting nexin-25
	P30	NP_001230152.1	RGS protein-like
	P12	NP_775578.2	RGS 12 isoform A
7 (68.8465)	P48	NP_035398.2	RGS 9 isoform 1
	P34	NP_001159406.1	RGS 9 isoform 2
	P15	NP_001074538.1	RGS 11
9 (55.5742)	P11	NP_835177.2	Guanine nuc. ex. factor DBS isoform 1
	P43	NP_001152958.1	Guanine nuc. ex. factor DBS isoform 2
	P25	NP_001152957.1	Guanine nuc. ex. factor DBS isoform 3
9 (36.2278)	P23	NP_001153070.1	Axin-1 isoform 1
	P45	NP_033863.2	Axin-1 isoform 2
	P13	NP_056547.3	Axin-2

Table A.13: **Game Theory Clustering For RGS Proteins (Non-Overlapping Domains).** Proteins are listed in descending order according to their game value with the average game value for the cluster shown next to cluster number.

Cluster	Node	NCBI Accession	Protein Type
1	P28	NP_033088.2	RGS 4
	P41	NP_035397.2	RGS 16
	P47	NP_001030608.1	Beta-adrenergic receptor kinase 2 isoform 2
	P32	NP_075019.1	RGS 18
	P33	NP_067349.2	RGS 20 isoform 2
	P9	NP_001182677.1	RGS 22
	P39	NP_064305.2	A-kinase anchor protein 10, mit. precursor
	P51	NP_080656.2	RGS 8
	P44	NP_001155294.1	RGS 17 isoform 1
	P8	NP_056626.2	RGS 1
	P21	NP_080694.1	RGS 10
	P36	XP_894544.3	PREDICTED: RGS 21
	P24	NP_001074212.1	GPCR kinase 4 isoform 2
	P46	NP_033087.2	RGS 2
	P14	NP_033089.2	RGS 5
	P53	NP_064342.1	RGS 17 isoform 2
	P6	NP_080722.1	RGS 19
	P5	XP_921002.3	PREDICTED: RGS 21
	P7	NP_694811.1	RGS 13
	P19	NP_001171266.1	RGS 20 isoform 1
2	P30	NP_001230152.1	RGS protein-like
3	P13	NP_056547.3	Axin-2
	P45	NP_033863.2	Axin-1 isoform 2
	P23	NP_001153070.1	Axin-1 isoform 1
4	P16	NP_001014973.2	Sorting nexin-13
	P49	NP_766514.2	Sorting nexin-14
5	P42	NP_997096.2	Sorting nexin-25
6	P4	NP_056627.1	RGS 6
7	P26	NP_001185932.1	RGS 7 isoform 2
	P54	NP_036010.2	RGS 7 isoform 1
	P15	NP_001074538.1	RGS 11
	P34	NP_001159406.1	RGS 9 isoform 2
	P48	NP_035398.2	RGS 9 isoform 1
8	P3	NP_796052.2	Beta-adrenergic receptor kinase 2 isoform 1
	P40	NP_570933.1	Beta-adrenergic receptor kinase 1
9	P55	NP_036068.2	GPCR kinase 6 isoform b
	P52	NP_061357.3	GPCR kinase 5
	P31	NP_001106182.1	GPCR kinase 6 isoform c
	P10	NP_062370.2	GPCR kinase 4 isoform 1
	P17	NP_036011.3	Rhodopsin kinase precursor
	P22	NP_001033107.1	GPCR kinase 6 isoform a
10	P35	NP_001123623.1	Rho guanine nuc. ex. factor 1 isoform b
	P37	NP_001123622.1	Rho guanine nuc. ex. factor 1 isoform a
11	P2	NP_001123624.1	Rho guanine nuc. ex. factor 1 isoform c
	P38	NP_032514.1	Rho guanine nuc. ex. factor 1 isoform d
	P20	NP_001123625.1	Rho guanine nuc. ex. factor 1 isoform c
12	P27	NP_081420.2	Rho guanine nuc. ex. factor 12
13	P43	NP_001152958.1	Guanine nuc. ex. factor DBS isoform 2
	P11	NP_835177.2	Guanine nuc. ex. factor DBS isoform 1
	P25	NP_001152957.1	Guanine nuc. ex. factor DBS isoform 3
14	P18	NP_001003912.1	Rho guanine nuc. ex. factor 11
15	P50	NP_599018.3	RGS 3 isoform 2
16	P12	NP_775578.2	RGS 12 isoform A
17	P29	NP_001156984.1	RGS 12 isoform B
	P1	NP_058038.2	RGS 14

Table A.14: **Biclusters For RGS Proteins (Non-Overlapping Domains).**

Cluster	Node	NCBI Accession	Protein Type
1	P50	NP_599018.3	RGS 3 isoform 2
	P29	NP_001156984.1	RGS 12 isoform B
	P12	NP_775578.2	RGS 12 isoform A
	P1	NP_058038.2	RGS 14
	P21	NP_080694.1	RGS 10
	P28	NP_033088.2	RGS 4
	P14	NP_033089.2	RGS 5
	P41	NP_035397.2	RGS 16
	P33	NP_067349.2	RGS 20 isoform 2
	P51	NP_080656.2	RGS 8
	P48	NP_035398.2	RGS 9 isoform 1
	P32	NP_075019.1	RGS 18
	P8	NP_056626.2	RGS 1
	P34	NP_001159406.1	RGS 9 isoform 2
	P19	NP_001171266.1	RGS 20 isoform 1
	P15	NP_001074538.1	RGS 11
	P6	NP_080722.1	RGS 19
	P36	XP_894544.3	PREDICTED: RGS 21
	P5	XP_921002.3	PREDICTED: RGS 21
	P53	NP_064342.1	RGS 17 isoform 2
	P4	NP_056627.1	RGS 6
	P44	NP_001155294.1	RGS 17 isoform 1
	P46	NP_033087.2	RGS 2
	P54	NP_036010.2	RGS 7 isoform 1
	P26	NP_001185932.1	RGS 7 isoform 2
	P7	NP_694811.1	RGS 13
2	P22	NP_001033107.1	GPCR kinase 6 isoform a
	P31	NP_001106182.1	GPCR kinase 6 isoform c
	P55	NP_036068.2	GPCR kinase 6 isoform b
	P40	NP_570933.1	Beta-adrenergic receptor kinase 1
	P10	NP_062370.2	GPCR kinase 4 isoform 1
	P3	NP_796052.2	Beta-adrenergic receptor kinase 2 isoform 1
	P47	NP_001030608.1	Beta-adrenergic receptor kinase 2 isoform 2
	P52	NP_061357.3	GPCR kinase 5
	P17	NP_036011.3	Rhodopsin kinase precursor
	P24	NP_001074212.1	GPCR kinase 4 isoform 2
3	P18	NP_001003912.1	Rho guanine nuc. ex. factor 11
	P2	NP_001123624.1	Rho guanine nuc. ex. factor 1 isoform c
	P20	NP_001123625.1	Rho guanine nuc. ex. factor 1 isoform c
	P38	NP_032514.1	Rho guanine nuc. ex. factor 1 isoform d
	P35	NP_001123623.1	Rho guanine nuc. ex. factor 1 isoform b
	P37	NP_001123622.1	Rho guanine nuc. ex. factor 1 isoform a
	P27	NP_081420.2	Rho guanine nuc. ex. factor 12
4	P43	NP_001152958.1	Guanine nuc. ex. factor DBS isoform 2
	P11	NP_835177.2	Guanine nuc. ex. factor DBS isoform 1
	P25	NP_001152957.1	Guanine nuc. ex. factor DBS isoform 3
5	P45	NP_033863.2	Axin-1 isoform 2
	P23	NP_001153070.1	Axin-1 isoform 1
	P13	NP_056547.3	Axin-2
6	P16	NP_001014973.2	Sorting nexin-13
	P42	NP_997096.2	Sorting nexin-25
7	P9	NP_001182677.1	RGS 22
8	P39	NP_064305.2	A-kinase anchor protein 10, mit. precursor
9	P30	NP_001230152.1	RGS protein-like
10	P49	NP_766514.2	Sorting nexin-14

Table A.15: **MCL Clusters For RGS Proteins (Non-Overlapping Domains).**

Node	NCBI Accession	UniProt ID	Protein Type	UniProt Family
P651	NP_796052.2	Q3UYH7	Beta-adrenergic receptor kinase 2	Protein kinase superfamily, AGC Ser/Thr protein kinase family, GPRK subfamily
P883	NP_035398.2	O54828	Axin-1	n/a
P884	NP_056547.3	O08849	Axin-2	n/a
P4208	NP_001074212.1	O70291	GPCR kinase 4	Protein kinase superfamily, AGC Ser/Thr protein kinase family, GPRK subfamily
P4209	NP_061357.3	Q8VEB1	GPCR kinase 5	Protein kinase superfamily, AGC Ser/Thr protein kinase family, GPRK subfamily
P4210	NP_001033107.1	O70293	GPCR kinase 6	Protein kinase superfamily, AGC Ser/Thr protein kinase family, GPRK subfamily
P8445	NP_694811.1	Q8K443	RGS 13	n/a
P8446	NP_075019.1	Q99PG4	RGS 18	n/a
P8447	NP_080722.1	Q9CX84	RGS 19	n/a
P8448	NP_001182677.1	G3UYX5	RGS 22	n/a
P8449	NP_766514.2	O08849	RGS 2	n/a
P8450	NP_599018.3	Q9DC04	RGS 3	n/a
P8451	NP_033088.2	O08899	RGS 4	n/a
P8452	NP_033089.2	O08850	RGS 5	n/a
P8453	NP_056627.1	Q9Z2H2	RGS 6	n/a
P8454	NP_036010.2	O54829	RGS 7	n/a
P8455	NP_080656.2	Q8BXT1	RGS 8	n/a
P8456	NP_035398.2	O54828	RGS 9	n/a
P9620	NP_001014973.2	Q6PHS6	Sorting nexin-13	Sorting nexin family
P9621	NP_766514.2	Q8BHY8	Sorting nexin-14	Sorting nexin family
P9628	NP_997096.2	Q3ZT31	Sorting nexin-25	Sorting nexin family

Table A.16: **RGS Proteins In Swiss-Prot Mouse Proteome.**

Bibliography

[1] S.F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman, "Basic Local Alignment Search Tool," Journal of Molecular Biology, Vol. 215, No. 3, pp. 403-410, 1990.

[2] S.F. Altschul and W. Gish, "Local Alignment Statistics," Methods in Enzymology, Vol. 266, pp. 460-480, 1996.

[3] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller and D.J. Lipman, "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs," Nucleic Acids Research, Vol. 25, No. 17, pp. 3389-3402, 1997.

[4] G. Apic, J.G. Gordana and S.A. Teichmann, "Domain Combinations in Archaeal, Eubacterial and Eukaryotic Proteomes," Journal of Molecular Biology, Vol. 310, No. 2, pp. 311-325, 2001.

[5] R. Apweiler, T.K. Attwood, A. Bairoch, A. Bateman, E. Birney, M. Biswas, P. Bucher et al. "The InterPro Database, An Integrated Documentation Resource For Protein Families, Domains, and Functional Sites," Nucleic Acids Research, Vol. 29, No. 1, pp. 37-40, 2001.

[6] H.J. Atkinson, J.H. Morris, T.E. Ferrin and P.C. Babbitt, "Using Sequence Similarity Networks For Visualization of Relationships Across Diverse Protein Superfamilies," PLoS ONE, Vol. 4, e4345, 2009.

[7] S.L. Baldauf, "Phylogeny For The Faint of Heart: A Tutorial," TRENDS in Genetics, Vol. 19, No. 6, pp. 345-351, 2003.

[8] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, "Linear Programming and Network Flows," John Wiley & Sons, 2011.

[9] R.G. Beiko and R.L. Charlebois, "A Simulation Test Bed For Hypotheses of Genome Evolution," Bioinformatics, Vol. 23, No. 7, pp. 825-831, 2007.

[10] R. Beiko, W. Doolittle and R. Charlebois, "The Impact of Reticulate Evolution on Genome Phylogeny," Systematic Biology, Vol. 57, No. 6, pp. 844-856, 2001.

[11] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering Local Structure In Gene Expression Data: The Order-Preserving Sub-Matrix Problem," In Proceedings of the 6th Annual Conference on Computational Biology, pp. 49-57, 2002.

[12] G. Bhardwaj, K.D. Ko, Y. Hong, Z. Zhang, N.L Ho, et al., "PHYRN: A Robust Method For Phylogenetic Analysis of Highly Divergent Sequences," PLoS ONE, Vol. 7, e34261, 2012.

[13] D. Binns, E. Dimmer, R. Huntley, D. Barrell, C. O'Donovan, and R. Apweiler, "QuickGO: A Web-based Tool For Gene Ontology Searching," Bioinformatics, Vol. 25, No. 22, pp. 3045-3046, 2009.

[14] B. Boussau, L. Guéuen and M. Gouy, "Accounting for Horizontal Gene Transfers Explains Conflicting Hypotheses Regarding the Position of Aquificales in the Phylogeny of Bacteria," BMC Evolutionary Biology, Vol. 8, No. 1, pp. 278, 2008.

[15] G.W. Brown and J. von Neumann, "Solutions of Games by Differential Equations," In: H.W. Kuhn and A.W. Tucker, Ed., Contributions to the Theory of Games I, In: Ann. of Math. Stud., Vol. 24; Princeton University Press, Princeton, pp. 73-79, 1950.

[16] M. Brown, R. Hughey, A. Krogh, I.S. Mian, K. Sjölander and D. Haussler, "Using Dirichlet Mixture Priors To Derive Hidden Markov Models For Protein Families," In Ismb, Vol. 1, pp. 47-55, 1993.

[17] D. Bryant and V. Moulton, "Neighbor-net: An Agglomerative Method For the Construction of Phylogenetic Networks," Molecular Biology and Evolution, Vol. 21, No. 2, pp. 255-265, 2004.

[18] R.A. Cartwright, "DNA Assembly With Gaps (Dawg): Simulating Sequence Evolution," Bioinformatics, Vol. 21, pp. iii31-iii38, 2005.

[19] T. Cavalier-Smith, "Rooting the Tree of Life by Transition Analyses," Biology Direct, Vol. 1, pp. 19, 2006.

[20] M. Chadeau-Hyam, C.J. Hoggart, P.F. O'Reilly, J.C. Whittaker, M. De Iorio and D.J. Balding, "Fregene: Simulation of Realistic Sequence-Level Data in Populations and Ascertained Samples," BMC Bioinformatics, Vol. 9, pp. 364, 2008.

[21] G.S. Chang, Y. Hong, K.D. Ko, G. Bhardwaj, E.C. Holmes, et al., "Phylogenetic Profiles Reveal Evolutionary Relationships Within the 'Twilight Zone' of Sequence Similarity," Proceedings of The National Academy of Sciences, USA,, Vol. 105, pp. 13474-13479, 2008.

[22] C.X. Chan, M. Mahbob and M.A. Ragan, "Clustering Evolving Proteins Into Homologous Families," BMC Bioinformatics, Vol. 14, pp. 120, 2013.

[23] Y. Cheng and G.M. Church, "Biclustering of Expression Data," Proceedings of the International Conference on Intelligent Systems for Molecular Biology, Vol. 8, pp. 93-103, 2000.

[24] I. Cohen-Gihon, R. Nussinov and R. Sharan, "Comprehensive Analysis of Co-occurring Domain Sets In Yeast Proteins," BMC Genomics, Vol. 8, pp. 161, 2007.

[25] A. Conesa, S. Götz, J.M. García-Gómez, J. Terol, M. Talón and M. Robles, "Blast2GO: A Universal Tool For Annotation, Visualization, and Analysis in Functional Genomics Research," Bioinformatics, Vol. 21, No. 18, pp. 3674-3676, 2005.

[26] D.A. Dalquen, M. Anisimova, G. Gonnet and C. Dessimoz, "ALF - A Simulation Framework For Genome Evolution," Molecular Biology and Evolution, Vol. 29, No. 4, pp. 1115-1123, 2012.

[27] M. Dayhoff and R. Schwartz, "A Model of Evolutionary Change in Proteins," In: Atlas of Protein Sequence and Structure, 1978.

[28] B. Deng, B. Hinds, E.N. Moriyama and X. Zheng, "Bioinformatic Game Theory and Its Application to Biological Affinity Networks," Applied Mathematics, Vol. 4, pp. 92-108, 2013.

[29] R.F. Doolittle, "Of URFs and ORFs: A Primer On How To Analyze Derived Amino Acid Sequences," University Science Books, 1986.

[30] W. Doolittle, "Phylogenetic Classification and the Universal Tree," Science, Vol. 284, No. 5423, pp. 2124-2129, 1999.

[31] R. Durbin, S. Eddy, A. Krogh and G. Mitchison, "Biological Sequence Analysis," Cambridge University Press, 1998.

[32] S.R. Eddy, "A Probabilistic Model of Local Sequence Alignment That Simplifies Statistical Significance Estimation," PLoS Compututational Biology, Vol. 4, No. 5, e1000069, 2008.

[33] S.R. Eddy, "Accelerated Profle HMM Searches," PLoS Computational Biology, Vol. 7, No. 10, 2011.

[34] M. Eigen, R. Winkler-Oswatitsch and A. Dress, "Statistical Geometry in sequence Space: A Method of Quantitative Comparative Sequence Analysis," Proceedings of the National Academy of Sciences, USA, Vol. 85, No. 16, pp. 5913-5917, 1988.

[35] A.J. Enright, S. Van Dongen and C.A. Ouzounis "An Efficient Algorithm For Large-Scale Detection of Protein Families," Nucleic Acids Research, Vol. 30, pp. 1575-1584, 2002.

[36] J. Felsenstein, "Confidence Limits On Phylogenies: An Approach Using The Bootstrap," Evolution, Vol. 39, pp. 783-791, 1985.

[37] J. Felsenstein, "Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach," Journal of Molecular Evolution, Vol. 17, No. 6, pp. 368-376, 1981.

[38] J. Felsenstein, "PHYLIP - Phylogeny Inference Package (Version 3.2)," Cladistics Vol. 5, pp. 164-166, 1989.

[39] J. Felsenstein, "Inferring Phylogenies," Sinauer Associates, Inc., Sunderland, MA, 2004.

[40] R.D. Finn, A. Bateman, J. Clements, P. Coggill, R.Y. Eberhardt, et al., "Pfam: The Protein Families Database," Nucleic Acids Research, Vol. 42, pp. D222-230, 2014.

[41] W. Fitch, "Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology," Systematic Biology, Vol. 20, No. 4, pp. 406-416, 1971.

[42] W. Fletcher and Z. Yang, "INDELible: A Flexible Simulator of Biological Sequence Evolution," Molecular Biology and Evolution, Vol. 26, No. 8, pp. 1879-1888, 2009.

[43] K. Forslund and E. L. Sonnhammer, "Evolution of Protein Domain Architectures," Methods in Molecular Biology, Vol. 856, pp. 187-216, 2012.

[44] N.C. Grassly, J. Adachi, and A. Rambaut, "PSeq-Gen: An Application for the Monte Carlo Simulation of Protein Sequence Evolution Along Phylogenetic Trees," Computer Applications in the Biosciences, Vol. 13, pp. 559-560, 1997.

[45] D. Graur and W.H. Li, "Fundamentals of Molecular Evolution," 2nd Ed., Sinauer Associates, Inc., Sunderland, MA, 2000.

[46] I. Griva, S. Nash and A. Sofer, "Linear and Nonlinear Optimization," Society for Industrial Mathematics, 2009.

[47] S. Guindon, J.F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk and O. Gascuel, "New Algorithms and Methods To Estimate Maximum-likelihood Phylogenies: Assessing the Performance of phyml 3.0," Systems Biology, Vol. 59, No. 3, pp. 307-321, 2010.

[48] S. Halary, J. W. Leigh, B. Cheaib, P. Lopez and E. Bapteste, "Network Analyses Structure Genetic Diversity in Independent Genetic Worlds," Proceedings of The National Academy of Sciences, USA, Vol. 107, No. 1, pp. 127-132, 2010.

[49] B.G. Hall, "Simulating DNA Coding Sequence Evolution With EvolveAGene 3," Molecular Biology and Evolution, Vol. 25, pp. 688-695, 2008.

[50] M. Hasegawa, H. Kishino and T. Yano, "Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA," Journal of Molecular Evolution, Vol. 22, No. 2, pp. 160174, 1985.

[51] S. Henikoff and J. Henikoff, "Amino Acid Substitution Matrices from Protein Blocks," Proceedings of The National Academy of Sciences, USA, Vol. 89, pp. 10915-10919, 1992.

[52] S. Henikoff, E.A. Greene, S. Pietrokovski, P. Bork, T. K. Attwood and L. Hood, "Gene Families: The Taxonomy of Protein Paralogs and Chimeras," Science, Vol. 278, No. 5338, pp. 609-614, 1997.

[53] R.D. Hernandez, "A Flexible Forward Simulator For Populations Subject to Selection and Demography," Bioinformatics, Vol. 24, pp. 2786-2787, 2008.

[54] J. Hofbauer, "Deterministic Evolutionary Game Dynamics," Proceedings of Symposia in Applied Mathematics, Vol. 69, pp. 61-79, 2011.

[55] C.J. Hoggart, M. Chadeau-Hyam, T.G. Clark, R. Lampariello, J.C. Whittaker, M. De lorio and D.J. Balding, "Sequence-level Population Simulations Over Large Genomic Regions," Genetics, Vol. 177, pp. 1725-1731, 2007.

[56] C. Holloway and R. Beiko, "Assembling Networks of Microbial Genomes Using Linear Programming," BMC Evolutionary Biology, Vol. 10, pp. 360, 2010.

[57] Y. Hong, D. Chalkia, K.D. Ko, G. Bhardwaj, G.S. Chang, et al., "Phylogenetic Profiles Reveal Structural and Functional Determinants of Lipid-binding," Journal of Proteomics & Bioinformatics, Vol. 2, pp. 139-149, 2009.

[58] Y. Hong, J. Kang, D. Lee and D.B. van Rossum, "Adaptive GDDA-BLAST: Fast and Efficient Algorithm For Protein Sequence Embedding," PLoS ONE, Vol. 5, No. 10, e13596, 2010.

[59] R.R. Hudson, "Generating Samples Under a Wright-Fisher Neutral Model of Genetic Variation," Bioinformatics, Vol. 18, pp. 337338, 2002.

[60] J. Huelsenbeck and F. Ronquist, "MRBAYES: Bayesian Inference of Phylogenetic Trees," Bioinformatics, Vol. 17, No. 8, pp. 754-755, 2001.

[61] S. Hunter, P. Jones, A. Mitchell, R. Apweiler, T.K. Attwood, et al., "InterPro in 2011: New Developments In the Family and Domain Prediction Database," Nucleic Acids Research, Vol. 40, pp. D306- D312, 2012.

[62] D.H. Huson and C. Scornavacca, "A Survey of Combinatorial Methods for Phylogenetic Networks," Genome Biology and Evolution, Vol. 3, pp. 23-35, 2011.

[63] D.T. Jones, W.R. Taylor and J.M. Thornton "The Rapid Generation of Mutation Data Matrices from Protein Sequences," CABIOS, Vol. 8, pp. 275-282, 1992.

[64] S. Karlin and S.F. Altschul, "Methods for Assessing the Statistical Significance of Molecular Sequence Features by Using General Scoring Schemes," Proceedings of The National Academy of Sciences, USA, Vol. 87, pp. 2264-2268, 1990.

[65] K. Katoh and D.M. Standley, "Mafft Multiple Sequence Alignment Software Version 7: Improvements In Performance and Usability," Molecular Biology and Evolution, Vol. 30, No. 4, pp. 772-780, 2013.

[66] Y. Kim and S. Subramaniam, "Locally Defined Protein Phylogenetic Profiles Reveal Previously Missed Protein Interactions and Functional Relationships," Proteins, Vol. 62, pp. 1115- 1124, 2006.

[67] M. Kimura, "A Simple Method for Estimating Evolutionary Rates of Base Substitutions Through Comparative Studies of Nucleotide Sequences," Journal of Molecular Evolution, Vol. 16, No. 2, pp. 111-120, 1980.

[68] K.D. Ko, G. Bhardwaj, Y. Hong, G.S. Chang, K. Kiselyov, et al., "Phylogenetic Profiles Reveal Structural/Functional Determinants of TRPC3 Signal-Sensing Antennae," Communcative & Integrative Biology, Vol. 2, pp. 133-137, 2009.

[69] E.V. Koonin, L. Aravind and A.S. Kondrashov, "The Impact of Comparative Genomics on Our Understanding of Evolution," Cell, Vol. 101, No. 6, pp. 573-576, 2000.

[70] H.W. Kuhn and A.W. Tucker, Ed., "Contributions to the Theory of Games," Ann. of Math. Stud., Vol. 24, Princeton University Press, Princeton, 1950.

[71] S.K. Kummerfeld and S.A. Teichmann, "Protein Domain Organisation: Adding Order," BMC Bioinformatics, Vol. 10, pp. 39, 2009.

[72] S. Le and O. Gascuel, "An Improved General Amino-acid Replacement Matrix," Molecular Biology and Evolution, Vol. 25, No. 7, pp. 1307-1320, 2008.

[73] P. Legendre and V. Makarenkov, "Reconstruction of Biogeographic and Evolutionary Networks Using Reticulograms," Systematic Biology, Vol. 51, No. 2, pp. 199-216, 2002.

[74] I. Letunic, R.R. Copley, S. Schmidt, F.D. Ciccarelli, T. Doerks, J. Schultz, C.P. Ponting and P. Bork, "SMART 4.0: Towards Genomic Data Integration," Nucleic Acids Research, Vol. 32, No. Suppl 1, pp. D142-D144, 2004.

[75] M. Li, J.H. Badger, X. Chen, S. Kwong, P. Kearney and H. Zhang, "An Information-Based Sequence Distance and Its Application To Whole Mitochondrial Genome Phylogeny," Bioinformatics, Vol. 17, No. 2, pp. 149-154, 2001.

[76] G. Lima-Mendez, J. Van Helden, A. Toussaint and R. Leplae, "Reticulate Representation of Evolutionary and Functional Relationships between Phage Genomes," Molecular Biology and Evolution, Vol. 25, No. 4, pp. 762-777, 2008.

[77] R. Luce and H. Raiffa, "Games and Decisions: Introduction and Critical Survey," Dover Press, 1989.

[78] S.C. Madeira and A.L. Oliveira, "Biclustering Algorithms For Biological Data Analysis: A Survey," IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 1, No. 1, pp. 24-45, 2004.

[79] V. Makarenkov and P. Legendre, "From a Phylogenetic Tree to a Reticulated Network," Journal of Computational Biology, Vol. 11, No. 1, pp. 195-212, 2004.

[80] A. Marchler-Bauer, S. Lu, J.B. Anderson, F. Chitsaz, M.K. Derbyshire, C. DeWeese-Scott, J.H. Fong et al., "CDD: A Conserved Domain Database For The Functional Annotation of Proteins," Nucleic Acids Research, Vol. 39, No. Suppl 1, pp. D225-D229, 2011.

[81] MATLAB and Statistics Toolbox Release 2013b, The MathWorks, Inc., Natick, Massachusetts, United States.

[82] J. Maynard Smith, "The Theory of Games and the Evolution of Animal Conflicts," Journal of Theoretical Biology, Vol. 47, No. 1, pp. 209-221, 1974.

[83] J. Maynard Smith, "Evolution and the Theory of Games," Cambridge University Press, 1982.

[84] S. McGinnis and T. Madden, "BLAST: At the Core of a Powerful and Diverse Set of Sequence Analysis Tools," Nucleic Acids Research, Vol. 32, Suppl 2, pp. W20-W25, 2004.

[85] K.M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs From Gene Expression Data," Pacific Symposium on Biocomputing, No. 8, pp. 77-88, 2003.

[86] J.C. Nacher, T. Ochiai, M. Hayashida and T. Akutsu, "A Bipartite Graph Based Model of Protein Domain Networks," In: Complex Sciences, Vol. 4 (J. Zhou, ed.). pp. 525-535 Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2009.

[87] J. Nash, "Equilibrium Points in N-person Games," Proceedings of the National Academy of Sciences, Vol. 36, No. 1, pp.48-49, 1950.

[88] J. Nash, "Non-cooperative Games," The Annals of Mathematics, Vol. 54, No. 2, pp. 286-295, 1951.

[89] J. von Neumann, "Zur Theorie der Gesellschaftsspiele," Mathematische Annalen, Vol. 100, No. 1, 1928, pp. 295-320, English translation: "On the Theory of Games of Strategy," In: A. W. Tucker and R.D. Luce, Ed., Contributions to the Theory of Games, Vol 24, Princeton University Press, 1959.

[90] J. von Neumann and O. Morgenstern, "Theory of Games and Economic Behavior," Princeton University Press, 1944.

[91] A. Ochoa, M. Llinas and M. Singh, "Using Context To Improve Protein Domain Identification," BMC Bioinformatics, Vol. 12, pp. 90, 2011.

[92] A. Pang, A.D. Smith, P.A.S. Nuin and E.R.M. Tillier, "SIMPROT: Using An Empirically Determined Indel Distribution In Simulations of Protein Evolution," BMC Bioinformatics, Vol. 6, pp. 236, 2006.

[93] L. Parida, "Pattern Discovery in Bioinformatics: Theory & Algorithms.," Chapman and Hall/CRC, 2007.

[94] W.R. Pearson and D.J. Lipman, "Improved Tools for Biological Sequence Comparison," Proceedings of The National Academy of Sciences, USA, Vol. 85, pp. 2444-2448, 1988.

[95] M. Pellegrini, E. Marcotte, M. Thompson, D. Eisenberg and T. Yeates, "Assigning Protein Functions By Comparative Genome Analysis: Protein Phylogenetic Profiles," Proceedings of The National Academy of Sciences, USA, Vol. 96, pp. 4285-4288, 1999.

[96] B. Peng and M. Kimmel, "simuPop: A Forward-time Population Genetics Simulation Environment," Bioinformatics, Vol. 21, pp. 3686-3687, 2005.

[97] B. Peng and X. Liu, "Simulating Sequences of the Human Genome With Rare Variants,", Human Heredity, Vol. 70, pp. 287-291, 2010.

[98] P. Pipenbacher, A. Schliep, S. Schneckener, A. Schonhuth, D. Schomburg, et al., "ProClust: Improved Clustering of Protein Sequences With An Extended Graph-based Approach," Bioinformatics, Vol. 18, Suppl 2, pp. S182-S191, 2002.

[99] C.P. Ponting and R.R. Russell, "The Natural History of Protein Domains," Annual Review of Biophysics and Biomoleular Structure, Vol. 31, pp. 45-71, 2002.

[100] D. Posada and K. Crandall, "Intraspecific Gene Genealogies: Trees Grafting Into Networks," Trends in Ecology and Evolution, Vol. 16, No. 1, pp. 37-45, 2001.

[101] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, et al., "A Systematic Comparison and Evaluation of Biclustering Methods For Gene Expression Data," Bioinformatics, Vol. 22, pp. 1122-1129, 2006.

[102] T. Przytycka, G. Davis, N. Song and D. Durand, "Graph Theoretical Insights Into Evolution of Multidomain Proteins," Journal of Computational Biology, Vol. 13, pp. 351-363, 2006.

[103] M. Punta, P.C. Coggill, R.Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E.L. Sonnhammer, S.R. Eddy, A. Bateman, and R.D. Finn, "The Pfam Protein Families Database," Nucleic Acids Research, Vol. 40, Database Issue, pp. D290-D301, 2012.

[104] A. Rambaut and N.C. Grassly, "Seq-Gen: An Application for the Monte Carlo Simulation of DNA Sequence Evolution Along Phylogenetic Trees," Bioinformatics, Vol. 13, pp. 235-238, 1997.

[105] J.A. Ranea, C. Yeats, A. Grant and C.A. Orengo, "Predicting Protein Function With Hierarchical Phylogenetic Profiles: The Gene3D Phylo-Tuner Method Applied To Eukaryotic Genomes," PLoS Computational Biolology, Vol. 3, e237, 2007.

[106] B. Rannalaa and Z. Yang, "Probability Distribution of Molecular Evolutionary Trees: A New Method of Phylogenetic Inference," Journal of Molecular Evolution, Vol. 43, No. 3, pp. 304-311, 1996.

[107] M.S. Rosenberg, "MySSP: Non-stationary Evolutionary Sequence Simulation, Including Indels," Evolutionary Bioinformatics Online, Vol. 1, pp. 81-83, 2005.

[108] W. Rudin, "Principles of Mathematical Analysis," Vol. 3, New York: McGraw-Hill, 1976.

[109] N. Saitou and M. Nei. "The Neighbor-joining Method: A New Method For Reconstructing Phylogenetic Trees," Molecular Biology and Evolution, Vol. 4, No. 4, pp. 406-425, 1987.

[110] S.F. Schaffner, C. Foo, S. Gabriel, D. Reich, M.J. Daly and D. Altschuler, "Calibrating a Coalescent Simulation of Human Genome Sequence Variation," Genome Research, Vol. 15, pp. 1576-1583, 2005.

[111] N. Shah, "Clustering and Classification of Multi-domain Proteins," Master's thesis. The United States: University of Nebraska-Lincoln, 2013.

[112] N. Shah, S.D. Scott and E.N. Moriyama, "Domain-Content Based Clustering of Multi-domain Proteins," 2014, In Review.

[113] T.F. Smith and Michael S. Waterman, "Identification of Common Molecular Subsequences," Journal of Molecular Biology, Vol. 147, No. 1, pp. 195-197, 1981.

[114] P.H.A. Sneath, "Reticulate Evolution in Bacteria and Other Organisms: How Can We Study It?" Journal of Classification, Vol. 17, pp. 159-163, 2000.

[115] R.R. Sokol and C.D. Michener, "Statistical Methods for Evaluating Systematic Relationships," University of Kansas Scientific Bulletin, Vol. 38, pp. 1409-1411, 1958.

[116] C.C.A. Spencer and G. Coop, "SelSim: A Program to Simulate Population Genetic Data With Natural Selection and Recombination," Bioinformatics, Vol. 20, pp. 3673-3675, 2004.

[117] J. Stoye, D. Evers and F. Meyer, "ROSE: Generating Sequence Families," Bioinformatics, Vol. 14, pp. 157-163, 1998.

[118] C.L. Strope, S.D. Scott and E.N. Moriyama, "indel-Seq-Gen: A New Protein Family Simulator Incorporating Domains, Motifs, and Indels," Molecular Biology and Evolution, Vol. 24, pp. 640-649, 2007.

[119] C.L. Strope, K. Abel, S.D. Scott and E.N. Moriyama, "Biological Sequence Simulation for Testing Complex Evolutionary Hypotheses: indel-Seq-Gen Version 2.0," Molecular Biology and Evolution, Vol. 26, No. 11, pp. 2581-2593, 2009.

[120] A. Tanay, R. Sharan and R. Shamir, "Discovering Statistically Significant Biclusters in Gene Expression Data," Bioinformatics, Vol. 18, Suppl. 1, pp. S136-S144, 2002.

[121] The Gene Ontology Consortium, "Creating the Gene Ontology Resource: Design and Implementation," Genome Research, Vol. 11, pp. 1425-1433, 2001.

[122] A.E. Todd, C.A. Orengo and J.M. Thornton, "Evolution of Function In Protein Superfamilies, From A Structural Perspective," Journal of Molecular Biology, Vol. 307, No. 4, pp. 1113-1143, 2001.

[123] P. Tufféry, "CS-PSeq-Gen: Simulating the Evolution of Protein Sequences Under Constraints," Bioinformatics, Vol. 18, pp. 1015-1016, 2002.

[124] S. Van Dongen, "Graph Clustering By Flow Simulation," PhD thesis. The Netherlands: University of Utrecht, 2000.

[125] C. Vogel, M. Bashton, N.D. Kerrison, C. Chothia and S.A. Teichmann, "Structure, Function, and Evolution of Multidomain Proteins," Current Opinions in Structural Biology, Vol. 14, pp. 208-216, 2004.

[126] C. Vogel, S.A. Teichmann and J. Pereira-Leal, "The Relationship between Domain Duplication and Recombination," Journal of Molecular Biology, Vol. 346, No. 1, pp. 355-365, 2005.

[127] Z. Wang, X.C. Zhang, M.H. Le, D. Xu, G. Stacey, et al., "A Protein Domain Co-occurrence Network Approach For Predicting Protein Function and Inferring Species Phylogeny," PLoS ONE, Vol. 6, pp. e17906, 2011.

[128] S. Wong and M.A. Ragan, "MACHOS: Markov Clusters of Homologous Subsequences," Bioinformatics, Vol. 24, pp. i77-i85, 2008.

[129] S. Wuchty and E. Almaas, "Evolutionary Cores of Domain Co-occurrence Networks," BMC Evolutionary Biology, Vol. 5, pp. 24, 2005.

[130] X. Xie, J. Jin and Y. Mao, "Evolutionary Versatility of Eukaryotic Protein Domains Revealed By Their Bigram Networks," BMC Evoluntionary Biology, Vol. 11, pp. 242, 2011.

[131] Z. Yang, "Estimating the Pattern of Nucleotide Substitution," Journal of Molecular Evolution, Vol. 39, pp. 105-111, 1994.

[132] Y. Ye and A. Godzik, "Comparative Analysis of Protein Domain Organization," Genome Research, Vol. 14, pp. 343-353, 2004.