Motif discovery in sequential data

by

Kyle L. Jensen

Submitted to the Department of Chemical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2006

Author . . . . . .
Department of Chemical Engineering
April 27, 2006

Certified by . . . . . .
Gregory N. Stephanopoulos
Bayer Professor of Chemical Engineering
Thesis Supervisor

Accepted by . . . . . .
William M. Deen
Chairman, Department Committee on Graduate Students

Motif discovery in sequential data

by

Kyle L. Jensen

Submitted to the Department of Chemical Engineering
on April 27, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this thesis, I discuss the application and development of methods for the automated discovery of motifs in sequential data. These data include DNA sequences, protein sequences, and real–valued sequential data such as protein structures and timeseries of arbitrary dimension. As more genomes are sequenced and annotated, the need for automated, computational methods for analyzing biological data is increasing rapidly. In broad terms, the goal of this thesis is to treat sequential data sets as unknown languages and to develop tools for interpreting an understanding these languages.

The first chapter of this thesis is an introduction to the fundamentals of motif discovery, which establishes a common mode of thought and vocabulary for the subsequent chapters. One of the central themes of this work is the use of grammatical models, which are more commonly associated with the field of computational linguistics. In the second chapter, I use grammatical models to design novel antimicrobial peptides (AmPs). AmPs are small proteins used by the innate immune system to combat bacterial infection in multicellular eukaryotes. There is mounting evidence that these peptides are less susceptible to bacterial resistance than traditional antibiotics and may form the basis for a novel class of therapeutics. In this thesis, I described the rational design of novel AmPs that show limited homology to naturally–occurring proteins but have strong bacteriostatic activity against several species of bacteria, including *Staphylococcus aureus* and *Bacillus anthracis*. These peptides were designed using a linguistic model of natural AmPs by treating the amino acid sequences of natural AmPs as a formal language and building a set of regular grammars to describe this language. This set of grammars was used to create novel, unnatural AmP sequences that conform to the formal syntax of natural antimicrobial peptides but populate a previously unexplored region of protein sequence space.

The third chapter describes a novel, GEneric MOtif DIscovery Algorithm (Gemoda) for sequential data. Gemoda can be applied to any dataset with a sequential character, including both categorical and real–valued data. As I show, Gemoda deterministically discovers motifs that are maximal in composition and length. As well, the algorithm allows any choice of similarity metric for finding motifs. These motifs are representation–agnostic: they can be represented using regular expressions, position weight matrices, or any other model for sequential data. I demonstrate a number of applications of the algorithm, including the discovery of motifs in amino acids and DNA sequences, and the discovery of conserved protein sub–structures.

The final chapter is devoted to a series of smaller projects, employing tools and methods indirectly related to motif discovery in sequential data. I describe the construction of a software tool, Biogrep that is designed to match large pattern sets against large biosequence databases

in a *parallel* fashion. This makes biogrep well–suited to annotating sets of sequences using biologically significant patterns. In addition, I show that the BLOSUM series of amino acid substitution matrices, which are commonly used in motif discovery and sequence alignment problems, have changed drastically over time. The fidelity of amino acid sequence alignment and motif discovery tools depends strongly on the target frequencies implied by these underlying matrices. Thus, these results suggest that further optimization of these matrices is possible.

The final chapter also contains two projects wherein I apply statistical motif discovery tools instead of grammatical tools. In the first of these two, I develop three different physiochemical representations for a set of roughly 700 HIV–I protease substrates and use these representations for sequence classification and annotation. In the second of these two projects, I develop a simple statistical method for parsing out the phenotypic contribution of a single mutation from libraries of functional diversity that contain a multitude of mutations and varied phenotypes. I show that this new method successfully elucidates the effects of single nucleotide polymorphisms on the strength of a promoter placed upstream of a reporter gene.

The central theme, present throughout this work, is the development and application of novel approaches to finding motifs in sequential data. The work on the design of AmPs is very applied and relies heavily on existing literature. In contrast, the work on Gemoda is the greatest contribution of this thesis and contains many new ideas.

Thesis Supervisor: Gregory N. Stephanopoulos
Title: Bayer Professor of Chemical Engineering

Acknowledgments

I am indebted to many people who both directly and indirectly contributed to this thesis. First, I would like to thank those collaborators who directly contributed. Most of all, I'm grateful for the help and friendship of Mark Styczynski. Mark was my collaborator on all matters computational for the past four years — his influence is evident throughout this document. I'm also grateful for my collaboration with Christopher Loose, who performed many of the experiments on antimicrobial peptides described in Chapter 2. Finally, I would like to thank Isidore Rigoutsos, who straddled the line between collaborator and advisor. Isidore taught me an attention to detail and a penchant for the UNIX command line and vi editor.

Most importantly, I am indebted to Greg Stephanopoulos, my advisor, whose guidance and support was unwavering these past six years. Greg is the perpetual optimist — always positive in the face of my many failures along the way. He also gave me the freedom to pursue projects of my own choosing, which contributed greatly to my academic independence, if not the selection of wise projects.

I am much obliged to my thesis committee members: my advisor Greg, Isidore, Bill Green, and Bob Berwick. My committee was always flexible in scheduling and judicious in their application of both carrots and sticks.

There are numerous people who contributed indirectly to this thesis. First among these is my intelligent, lovely, and vivacious wife Kathryn Miller–Jensen. Next, my parents Carl and Julie, my sister Heather, and my in-laws, Ron, Joyce, Suzanne, Jeff, and Mindi. Finally, there are innumerable friends who contributed and to whom I am greatly appreciative including Michael Raab, Joel Moxley, Bill Schmitt, Vipin Guptda, and Jatin Misra.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to motif discovery

1.1 Introduction

The field of biology is changing rapidly from a qualitative discipline to one rooted in quantitation. These changes are driven by advances in microfabrication and microelectronics that continue to yield ever more creative ways to probe cellular function. These advances, in turn, are producing a deluge of data, opening up new ways to think about and analyze life, and attracting engineers and scientists from other disciplines into biology.

Nowhere is this sea change of quantitation more pronounced than in DNA sequencing [227]. As shown in Figure 1-1 on the following page, improvements in DNA sequencing drove down the cost of sequencing many orders of magnitude over the past 30 years, making sequencing commonplace. This rate of sequencing produces a data storage nightmare — each dry gram of DNA can store approximately a zettabyte of information, or a million million gigabytes [205]. Consider the growth of the Genbank DNA database [33], as shown in Figure 1-1 on the next page. Genbank receives over 1000 new submissions of DNA sequences from scientists every day and has doubled in size every 18 months since 1982.

This torrent of data is not restricted to DNA sequencing. Technological advances in recent years produced myriad tools for quantifying biology including DNA microarrays, reverse transcriptase polymerase chain reaction (RT–PCR), flow cytometry, chromatin–immunoprecipitation (chip–chip), yeast two–hybrid assays, fluorescence and confocal microscopy, generalized robotic screening methods, and countless others. These tools enable the continual coining of new "omes" such as the transcriptome, proteome, and interactome — high–throughput counterparts to traditional areas of study in biology. (See Figure 1-4 on page 19 and Table 1.1 on page 20.) These new fields do not exist in isolation, but instead contribute to an ever–increasing network of information. Consider the Genbank annotation of the human insulin gene as shown an Figure 1-3 on page 18. The annotation includes detailed information about insulin culled from the scientific literature by human experts including not just the sequence of the gene, but also its post–translational modifications, cellular localization, interactions with other genes, role in energy metabolism and diabetes, and numerous links to external databases with yet more information.

The Genbank annotation of insulin is the rule rather than the exception. It is a small piece of the growing wealth of data describing life processes from the molecular level all the way to the organism and ecological levels. However, inasmuch as these data hold promise, they also

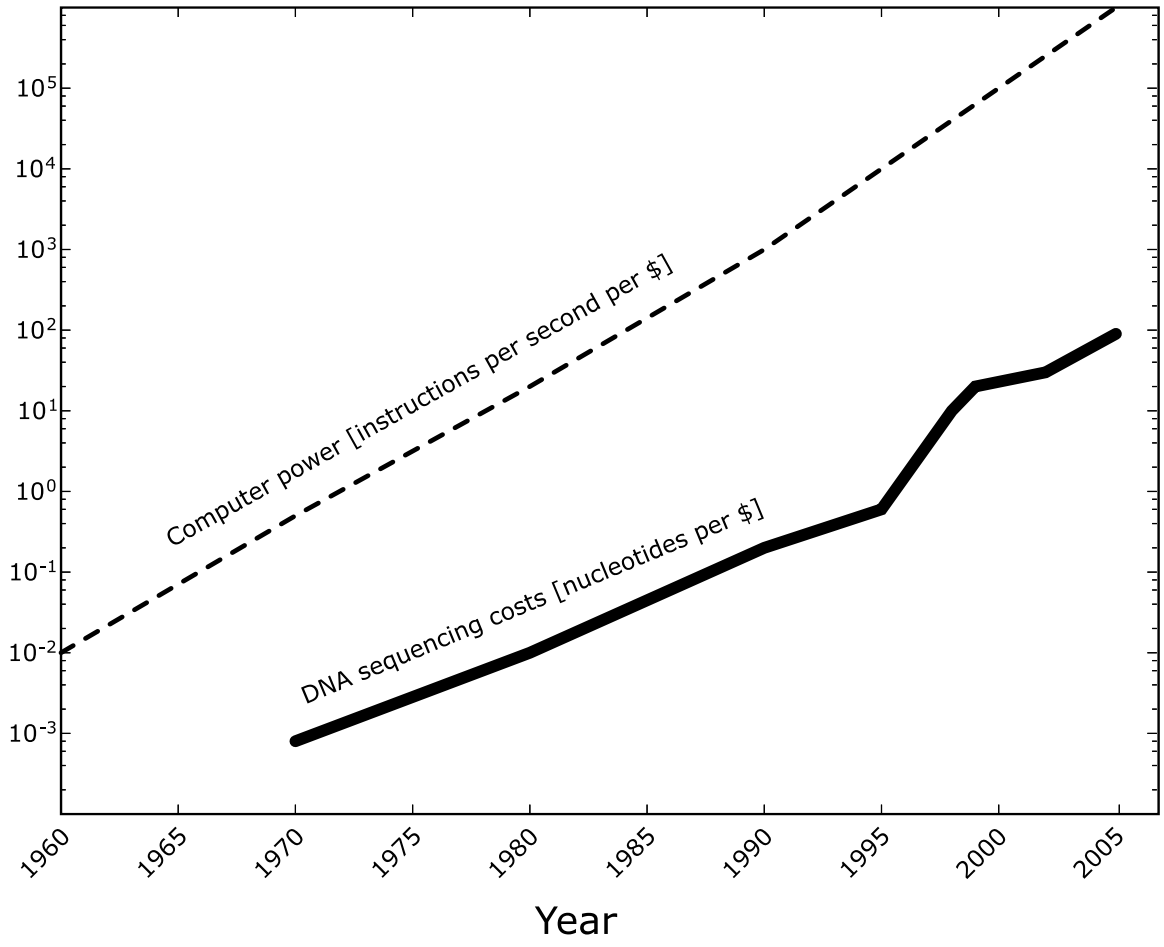


Figure 1-1: Exponential growth of sequencing throughput [227]. The figure tracks the number of nucleotides that can be sequenced for a dollar over time juxtaposed with the advancement of computing power. The hypothesis referred to as "Moore's Law" — that computational power doubles every 18 months — appears somewhat applicable to DNA sequencing. Engineering advancements in the basic electrophoretic method of DNA sequencing, the so–called "Sanger sequencing," over the past 30 years decreased the cost of sequencing many–fold. During the 15 year Human Genome Project, widespread investment into innovation and automation drove down the cost tenfold, greatly accelerating the completion of the project — 85% of the genome was sequenced in the project's last two years [61].

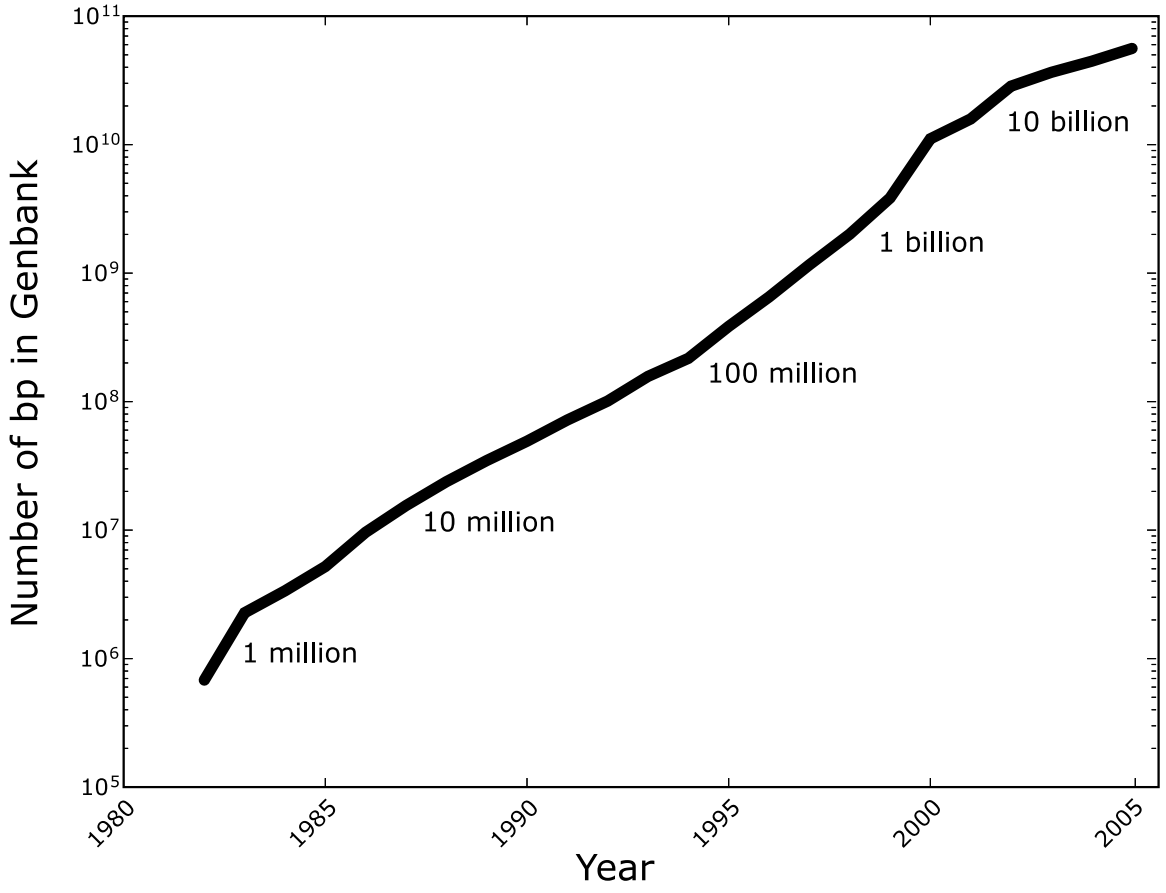


Figure 1-2: Exponential growth of Genbank [33]. Genbank is a comprehensive database of DNA sequences from over 200,000 organisms that were made public by direct submission from researchers. The database is maintained by the U.S. National Center for Biotechnology Information, and is updated daily. The sequences in Genbank include data from genome sequencing projects, expressed sequence tags, and international sequence data from the European Molecular Biology Laboratory and the DNA Databank of Japan. Genbank receives over 1000 submissions per day and has doubled roughly every 18 months since being started. It is the largest database of its kind and is the basis for many "information–added" databases and resources. See also 1-1 on the preceding page.

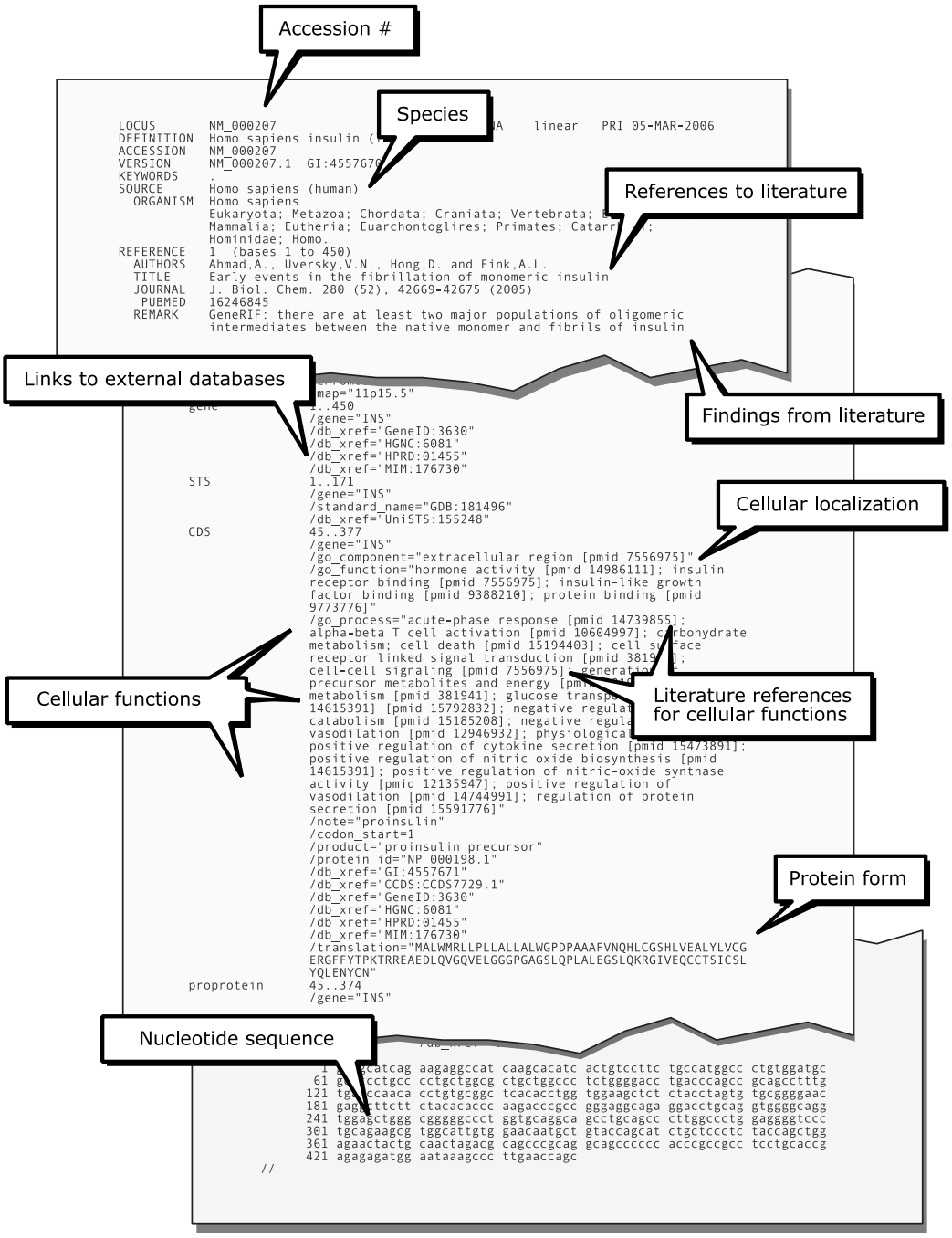


Figure 1-3: A sample Genbank record showing the litany of features annotating the human gene for insulin. As the figure suggests, this "extra" information typically dwarfs the gene sequence in size. The annotations are highly cross–referenced, linking genes to cellular functions, behaviors, localizations and to other databases with yet more information. Viewed *en masse*, a complex network of knowledge emerges linking primary sequences into the understanding of higher–order systems at the cellular, tissue, an organism levels.

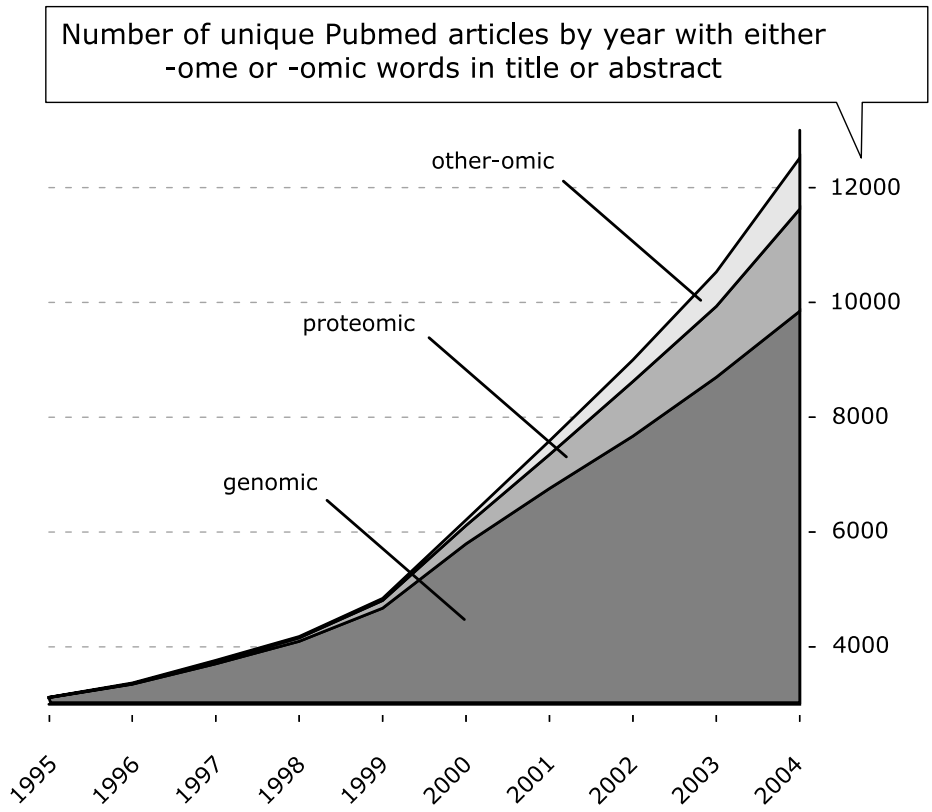


Figure 1-4: Usage of "ome" and "omic" words over time divided into three catagories: genome and genomic, proteome and proteomic, and other ome and omic. The latter category comprises those words shown in Table 1.1 on the following page.

Table 1.1: "Omic" fields other than genomic and proteomic. In many cases, fields have been reborn as "omic" fields due to a changing focus towards higher throughput empirical methods. This list is indicative of the shift in biology towards quantitation and the resulting need for new, automated modes of analysis. The rise of these words over time in the scientific vernacular is shown in Figure 1-4 on the preceding page. This list was compiled by searching over 5 million articles in the life sciences literature using the NCBI eutils API [76]. See also references in the bibliography [93, 244].

Anatomics	Biomics	Chromosomics	Cytomics
Enviromics	Epigenomics	Fluxomics	Glycomics
Glycoproteomics	Immunogenomics	Immunomics	Immunoproteomics
Integromics	Interactomics	Ionomics	Lipidomics
Metabolomics	Metabonomics	Metagenomics	Metallomics
Metalloproteomics	Methylomics	Mitogenomics	Neuromics
Neuropeptidomics	Oncogenomics	Peptidomics	Phenomics
Phospho-proteomics	Phosphoproteomics	Physiomics	Physionomics
Post–genomics	Postgenomics	Pre–genomics	Rnomics
Secretomics	Subproteomics	Surfaceomics	Syndromics
Transcriptomics			

present challenges: if our ability to collect data has increased exponentially, has our ability to interpret and derive meaningful conclusions from these data kept track? Will these diverse data allow us to "connect the dots," to reveal rich systems–level information? Or, instead, are they subject to the law of diminishing returns? The prevailing punditry lends credence to the former: witness the rise of systems biology [126, 127, 216].

But the problem remains: how can the potential of these volumous and diverse data sets be realized? The sheer amount of data points to the need for automated, computational methods for analysis. This need is driving the co–opting of computer science as a core discipline of biology. That is, computational methods are becoming increasingly necessary to analyze the vast data sets biologists have accrued, and consequently, computer science and mathematics are becoming as fundamental to biology as chemistry and physics. The particular sub–discipline of biology devoted to these computational methods is typically referred to as either bioinformatics or computational biology. Together, these fields comprise many research topics devoted to both data analysis and modeling of biological systems.

It is likely that the need for automated, computational analysis will only increase in the future. There are 334 completely sequenced genomes and over 700 genomes in various stages of completion in the genome database at the U.S. National Center for Biotechnology Information (NCBI) [267]. But, of the finished genomes, only two are mammalian: *Homo sapiens* [145, 256] and *Mus musculus* [265], human and mouse. This suggests that, rather than being in the "post–genomic" age, we have just begun the genomic age. Upcoming years will see the completion of many genomes with relevance to human health and disease — such as the rat, rabbit, and chimpanzee genomes — and to industry such as the cow, corn, and potato

genomes. Furthermore, these are just the sequencing data. Analogous progress in fields such as metabolomics and proteomics is likely to leave biologists awash in data, further exacerbating the need for automated methods of interpreting and modeling these data.

The motivation for automated data analysis techniques that I have painted in this introduction is quite broad, but what remains of this thesis is not. The focus of the remainder is automated, computational methods for discovering motifs in sequential data such as DNA and protein sequences. For example, imagine a scenario in which we would like to correlate one of the annotations of insulin shown in Figure 1-3 to a particular part of the DNA that encodes insulin. This is a very common problem in bioinformatics. In essence, this is like trying to learn a language we don't speak by reading many books. As suggested by Figure 1.2 on the next page, this is a nontrivial problem.

Motif discovery in sequential data is only one tiny sliver of the vast spectrum of topics spanned by bioinformatics and computational biology. However, many of the principles and tools I describe have broader implications for learning and automated discovery methods in biology. Chapter 1 of this thesis is devoted to familiarizing the reader with basic concepts of motif discovery, with a particular focus on linguistic methods of modeling sequences. In Chapter 2, these concepts are applied to the annotation and design of novel antibiotics, called antimicrobial peptides. The principal contribution of this thesis is the third chapter, in which I develop a framework and software tool for motif discovery that can be generalized to diverse types of data and is superior to existing tools in many ways. Finally, the last chapter comprises a series of vignettes that take a more broad approach to motif discovery and explore a number of issues adjoining the central theme of the first three chapters.

1.2 Fundamental tenets of motif discovery

I will begin with a series of elementary, almost philosophical examples that serve to illustrate some of the fundamental tenets of motif discovery. These examples may seem pedantic; however, the tenets they illustrate will be recurring themes throughout this thesis. Further, the following sections will serve to establish a common vocabulary and mode of thought that will enable the development of more complex ideas in later chapters.

In the most basic sense, the task of motif discovery is to find a repeated feature in a data set. Take, for example, the objects below.

■▲□■ (1.1)

What features are repeated at least twice in the objects? One answer is that the square shape appears three times. Yet another, is that three of the objects are darkened. And finally, a more sophisticated answer is that all of the objects are regular polyhedrons. Which of these is correct? All three are. The first two answers are intuitively obvious, whereas the final answer is rooted in a knowledge of geometry. The degree to which one answer is "more correct" than the others depends on what kinds of features we are interested in *a priori*. This is the first and most important tenet of motif discovery.

Consider a second question: is the dark square more similar to the dark triangle or to the

Table 1.2: Genes sequences from *Arabodopsis thaliana*, a popular plant model organism. These very small genes are only a few hundred bases long, whereas a typical gene can be many kilo–bases in length. Given these sequences, how can we find all the small repeated motifs such as CCACGCGTCCGAAAA? At a glance, the task seems difficult. Digging further it seems insurmountable — the sequences below are just a small snippet of Genbank . To print just the nucleotide sequences in Genbank using this font would require 30 million pages: a stack of paper 3 km high, or roughly the distance from MIT to Harvard.

>gi||8571926 Arabidopsis thaliana lipid transfer protein
CCACGCGTCCGAAAAAAAAAACAGAAAGTAACATGAGATCTCTCTTATTAGCCGTGTGCCTGGTTCTTGC
TTTACACTGCGGTGAAGCAGCCGTGTCTTGCAACACGGTGATTGCGGATCTTTACCCTTGCTTATCCTAC
GTGACTCAGGGCGGACCGGTCCCAACCCTCTGCTGCAACGGTCTCACAACACTCAAGAGTCAGGCTCAAA
CTTCTGTGGACCGTCAGGGGGTCTGTCGTTGCATCAAATCTGCTATTGGAGGACTCACTCTCTCTCCTAG
AACCATCCAAAATGCTTTGGAATTGCCTTCTAAATGTGGTGTCGATCTCCCTTACAAGTTCAGCCCTTCC
ACTGACTGCGACAGTATCCAGTGAGACAAGCAGAAAATCTTAAAGGAAGCTACTACAAGAACTATAATAA
CCTAATAATTAATAAATGAGGGCATTGGTTTGCTAGTTGCTAATTGATCAGTGATGTATTGTCATTTTGA
ATGTTCTAATATCAGCAGGCACTTATCTCTGAAAAAAAAAAAAAAAA

>gi||8571922 Arabidopsis thaliana lipid transfer protein
CCACGCGTCCGAAAACACAAGCGTAGAAAACAAAACTCAACTAATTGTGTTATCACCCAAAAGAGAAGAG
CAAACACAATGGCTTTCGCTTTGAGGTTCTTCACATGCTTTGTTTTGACAGTGTTCATCGTTGCATCAGT
GGATGCAGCAATAACATGTGGCACAGTGGCAAGTAGCTTGAGTCCATGTCTAGGCTACCTATCGAAGGGT
GGGGTGGTGCCACCTCCGTGCTGTGCAGGAGTCAAAAAGTTGAACGGTATGGCTCAAACCACACCCGACC
GCCAACAAGCATGCAGATGCTTACAGTCCGCTGCAAAAGGGGTTAATCCAAGTCTAGCCTCTGGCCTTCC
TGGAAAGTGCGGTGTTAGCATCCCCTATCCCATCTCCACGAGCACCAACTGCGCCACCATCAAGTGAAGT
GGGGAATAACGACATCATTTGCCTGAAGAGTATGGTTTCGTATACGTAAAATAAGACGGCTATCTAAGCT
GATATTTACCTTGTCTTTGTTTGTCTTGATGGCTTTGTAATCTTTTGCTTTGTTATGTTGTATACTTGTG
TCTTAACATGTTTAAGATATGATAATATATAGTATCGGTACCTTATTAAAAAAAAAAAAAAA

>gi||8571922 Arabidopsis thaliana lipid transfer protein
CCACGCGTCCGAAAACACAAGCGTAGAAAACAAAACTCAACTAATTGTGTTATCACCCAAAAGAGAAGAG
CAAACACAATGGCTTTCGCTTTGAGGTTCTTCACATGCTTTGTTTTGACAGTGTTCATCGTTGCATCAGT
GGATGCAGCAATAACATGTGGCACAGTGGCAAGTAGCTTGAGTCCATGTCTAGGCTACCTATCGAAGGGT
GGGGTGGTGCCACCTCCGTGCTGTGCAGGAGTCAAAAAGTTGAACGGTATGGCTCAAACCACACCCGACC
GCCAACAAGCATGCAGATGCTTACAGTCCGCTGCAAAAGGGGTTAATCCAAGTCTAGCCTCTGGCCTTCC
TGGAAAGTGCGGTGTTAGCATCCCCTATCCCATCTCCACGAGCACCAACTGCGCCACCATCAAGTGAAGT
GGGGAATAACGACATCATTTGCCTGAAGAGTATGGTTTCGTATACGTAAAATAAGACGGCTATCTAAGCT
GATATTTACCTTGTCTTTGTTTGTCTTGATGGCTTTGTAATCTTTTGCTTTGTTATGTTGTATACTTGTG
TCTTAACATGTTTAAGATATGATAATATATAGTATCGGTACCTTATTAAAAAAAAAAAAAAA

hollow square?

$$\blacksquare \sim \blacktriangle \sim \square \tag{1.2}$$

Again, there is no correct answer. The response obviously depends on our relative preference for color versus shape. This variety of question is even more difficult when we seek to quantify the degree of similarity or difference — the distance — between two objects. Is a human more similar to an alligator, or to an elephant? Based on body temperature, the human is more similar to the elephant; however, based on weight, the opposite true. This is the second tenet of motif discovery: the measurement of "distance" between objects is inherently relative, or dependent on predefined metrics.

Now consider a more complicated example shown schematically in Figure 1-5 on the following page. The figure shows an X–Y scatter plot with 12 data points. How many groups of points, or clusters, are in this figure? As before, there are many correct answers: three groups of four; one group of four and one group of eight; or one single group of twelve. The answer depends on our preconceived notion of how small the distance between objects must be in order for them to be grouped together. Or, equivalently, how similar objects must be to be considered the same. This is our third and final tenet of motif discovery.

The three tenets we have just developed can be rephrased as follows. The answer to any motif discovery problem will always depend on what kinds of motifs we are looking for and the search for these motifs will always depend on a predefined metric of similarity and method for grouping together similar objects.

1.3 Motif discovery in sequential data

In this section, I define "sequential data" and introduce some basic concepts of motif discovery in sequential data. I will build upon tenets developed in the previous section and develop more complex ideas in motif discovery that are specific to sequential data.

In the most general sense, sequential data are any data in which there is a natural ordering, such that rearranging the data would result in lost information. In later chapters, we will deal with the more general case of sequential data that encompasses series of multidimensional real–valued data; however, for the time being, we can consider sequential data to be a series of alphanumerical characters such as the four sequences shown below.

```
djndfduckdeicfjnfmABRAHAMLINCOLN
            idkeioddsnABRAHAMLINCOLNaknkwbad
            ioxcABRAHAMLINCOLNabkjwlkdaxlakj
                        xkasnkjlfABRAHAMLINCOLNkdsjkjsdl
```
(1.3)

We can refer to these sequences more formally by calling the set of sequences S, where S is defined such that $S = \{s_0, s_1, s_2, \ldots, s_n\}$, and where sequence s_i has length W_i. So, in the above example, $n = 3$ and $s_0 =$ `djndfduckdeicfjnfmABRAHAMLINCOLN`. Furthermore, let the the j^{th} member of the i^{th} sequence be denoted by $s_{i,j}$. So, $s_{0,0} = d$, $s_{0,1} = j$, etc. Each $s_{i,j}$ is a

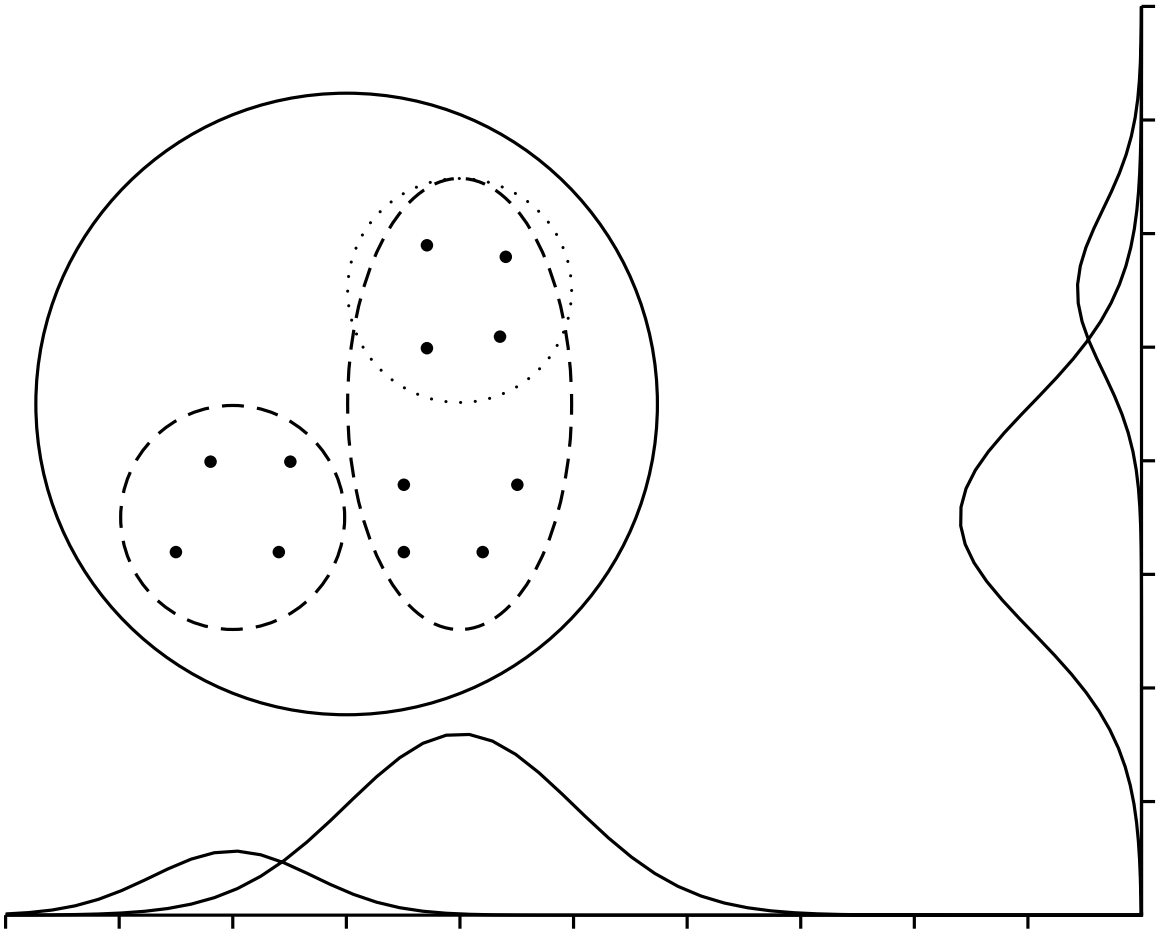


Figure 1-5: Clustering to find patterns in generic data. Given this data set, we would cluster the data together to find patterns (indicated here by the ellipses). But, there are many different patterns we could find. Which ones are correct?

primitive, or atomic unit, for the data that is being analyzed. For now, we will say that the primitives are alphanumerical characters selected from some alphabet Σ. In the example above, this alphabet is the set of 52 lowercase and uppercase characters from the Roman alphabet. However, this alphabet can be defined in many different ways depending on the context of our motif discovery problem. For example, for a DNA sequence the alphabet would comprise the characters {A, T, G, C}, representing the four bases found in DNA: adenosine, thymine, guanine, and cytosine (see Figure A-2 on page 175 in the Appendix. Or, for protein sequences, the alphabet would be the set of characters representing the one letter abbreviations for the 20 naturally occurring amino acids {A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V}, which are defined in Figure A-1 on page 174 in the Appendix.

In the set of sequences above, the motif "ABRAHAMLINCOLN" is inherently obvious in the otherwise random series of characters that make up each of the individual sequences. A similar motif is obvious even when the sequences are "mutated" as shown below.

$$\begin{aligned} s_0 &= \texttt{djndfduckdeicfjnfmABELINCOLN} \\ s_1 &= \texttt{idkeioddsnABRAHAMLINCOLNaknkwbad} \\ s_2 &= \texttt{ioxcALINCOLNabkjwlkdaxlakj} \\ s_3 &= \texttt{xkasnkjlfABRAHAMLINCOLNkdsjkjsdl} \end{aligned} \tag{1.4}$$

In two of these sequences, Abraham is abbreviated but is still recognizable. But now, it is not appropriate to call the motif simply "ABRAHAMLINCOLN." At this point, it is important to develop a more rigorous definition of "motif" that will allow us to describe all the possible permutations shown in the above mutated sequences. A motif is, henceforth, a mathematical model used to describe a set of locations in a set of sequences. These locations are referred to as "motif instances." In the example above, the motif instances are the substrings: "ABELINCOLN," "ABRAHAMLINCOLN," "ALINCOLN," and "ABRAHAMLINCOLN" in sequences s_0, s_1, s_2, and s_3, respectively.

Any mathematical model used to describe these instances should say that each instance begins with the letter A, optionally followed by either BE or BRAHAM, and necessarily followed by LINCOLN. That is, a model that describes the ordered arrangement of objects, in this case characters. Such models are commonplace in the field of linguistics and are called grammars.

1.4 Grammatical models of sequences

Introduction

The theory underlying grammatical models of sequences can be traced back to Noam Chomsky's early work on syntax theory [53–55]. Chomsky's work is the basis for much of formal language theory and computational linguistics. However, in general, most research in these areas has used grammars for pattern *recognition* vice pattern *discovery*, and focuses on machine learning techniques for computer understanding of natural languages. Only recently have these pattern recognition techniques been applied to problems of interest to biologists [222–224].

As I will show in the following sections, most motif discovery algorithms in bioinformatics

use motif models that can be reduced, in general, to a grammatical model. However, for the most part, such reductions are rare in the bioinformatics literature. This is because motif discovery and bioinformatics evolved independently from formal language theory. In a sense, the two fields are distant homologs of each other, both making use of models that can be reduced to grammars.

A grammar is a mathematical construct that describes the ordered arrangement of objects, usually words or characters, in a sequence [3]. More rigorously, a grammar is a 4–tuple $G = (N, \Sigma, P, S)$ wherein

1. N is a finite set of non-terminal symbols, also called variables or syntactic categories;
2. Σ is a finite set of terminal symbols, disjoint from N;
3. P is a finite subset of

$$(N \cup \Sigma) * N(N \cup \Sigma) * \times (N \cup \Sigma)*, \tag{1.5}$$

   each element in P is called a "production" and is written in the form $\alpha \rightarrow \beta$[1]; and,
4. S is a special symbol in N that is the start symbol.

To illustrate how a grammar can be used to model sequences, consider the following simple grammar:

$$G = (\{\alpha, S\}, \{\texttt{0}, \texttt{1}\}, P, S), \tag{1.6}$$

where the set of productions, P, is given by

$$P = \begin{cases} S \rightarrow \texttt{0}\alpha\texttt{1} \\ \texttt{0}\alpha \rightarrow \texttt{00}\alpha\texttt{1} \\ \alpha \rightarrow e. \end{cases} \tag{1.7}$$

Each line in the set of productions is essentially a replacement rule. For example, the operation $S \rightarrow \texttt{0}\alpha\texttt{1}$ should be read as "replace the character S with the sequence of characters $\texttt{0}\alpha\texttt{1}$." Then, the subsequent line should be read as "replace the two character sequence $\texttt{0}\alpha$ with the sequence $\texttt{00}\alpha\texttt{1}$." Finally, the last production should be read as "replace the character α with the character e, which is the termination character." These productions follow a few conventions that are used throughout this manuscript. First, as defined on page 26, S is a special non–terminal symbol that is always used as the starting symbol. Second, in order to distinguish terminal symbols from non–terminal symbols, the former will always be displayed in a fixed width font. Third, the character e is used always to represent the termination of a sequence and is referred to as the termination character.

[1] The star symbol, $*$, is called the Kleene star and is interpreted as "zero or more" of the expression it follows. For example, `ZA*` would be interpreted as a `Z` followed by zero or more `A` characters. The Kleene star and other similar operators will be discussed later in the section on regular grammars and regular expressions 1.4 on page 33.

Now consider how to construct a sequence using the set of productions in the grammar shown in Equations 1.6 and 1.7. Starting with the special non–terminal character S, the first production produces a three letter sequence.

$$S \Rightarrow \texttt{0}\alpha\texttt{1}$$

Using the second production, produces a four character sequence.

$$S \Rightarrow \texttt{0}\alpha\texttt{1} \Rightarrow \texttt{00}\alpha\texttt{1}$$

Finally, using the third production terminates the sequence.

$$S \Rightarrow \texttt{0}\alpha\texttt{1} \Rightarrow \texttt{00}\alpha\texttt{1} \Rightarrow \texttt{001} \tag{1.8}$$

The sequences produced by following the production rules of the grammar are called derivations of a grammar, or equivalently the sentences or sentential forms of the grammar. The collection of sentential forms of a grammar are collectively called the language generated by G, or L(G).

Notably, the final sequence shown in Equation 1.8 is not the only sequence that can be derived from the grammar G. Because the symbol α appears in two different productions, either production can be used and neither production is preferred *a priori*. For example the following sequence is an equally valid derivation of the same grammar.

$$S \Rightarrow \texttt{0}\alpha\texttt{1} \Rightarrow \texttt{00}\alpha\texttt{1} \Rightarrow \texttt{000}\alpha\texttt{1} \Rightarrow \texttt{0000}\alpha\texttt{1} \Rightarrow \texttt{00000}\alpha\texttt{1} \Rightarrow \texttt{000001} \tag{1.9}$$

As the derivation above suggests, any sequence that

- begins with a `0`,
- that is followed by zero or more `0`s, and
- is terminated by a single `1`

could be a derivation of the grammar shown in Equation 1.6 on the facing page. Collectively, these derivations form the language of the grammar.

Now I will return to the Abraham Lincoln example shown in Equation 1.4 on page 25. Recall that the motif instances are the substrings "ABELINCOLN," "ABRAHAMLINCOLN," "ALINCOLN," and "ABRAHAMLINCOLN" in sequences s_0, s_1, s_2, and s_3, respectively. A grammar that describes these motif instances is

$$G = (\{\alpha, S\}, \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{E}, \mathsf{H}, \mathsf{I}, \mathsf{L}, \mathsf{M}, \mathsf{N}, \mathsf{R}\}, P, S), \tag{1.10}$$

where P is given by the set of productions

$$P = \begin{cases} S \to A\alpha \\ \alpha \to \beta \\ \alpha \to BE\beta \\ \alpha \to BRAHAM\beta \\ \beta \to LINCOLN\gamma \\ \gamma \to e. \end{cases} \tag{1.11}$$

Again, in a manner similar to Equation 1.7 on page 26, the grammar shown in Equation 1.11 has a non–terminal character, α, in multiple productions. In such cases, the production can usually be abbreviated using the "|" character, which is to be read as "or." For example, the productions in Equation 1.11 can be written equivalently as

$$P = \begin{cases} S \to A\alpha \\ \alpha \to \beta \mid BE\beta \mid BRAHAM\beta \\ \beta \to LINCOLN\gamma \\ \gamma \to e. \end{cases} \tag{1.12}$$

Notice that the grammar shown in Equation 1.11 describes exactly all four of the motif instances in the sequences shown in Equation 1.4 on page 25. The three possible derivations of the grammar are shown below.

$$\begin{aligned} S \Rightarrow A\alpha \Rightarrow A\beta \Rightarrow ALINCOLN\gamma \Rightarrow ALINCOLN \\ S \Rightarrow A\alpha \Rightarrow ABE\beta \Rightarrow ABELINCOLN\gamma \Rightarrow ABELINCOLN \\ S \Rightarrow A\alpha \Rightarrow ABRAHAM\beta \Rightarrow ABRAHAMLINCOLN\gamma \Rightarrow ABRAHAMLINCOLN \end{aligned} \tag{1.13}$$

This is an ideal case. In general, in constructing any model describing any motif instances, we would like to use a grammar that is sensitive for the instances — i.e. all the instances are derivations of G — and specific for the instances — i.e. the language $L(G)$ includes few derivations that are not motif instances.

Now consider a more complicated case in which a grammar is used to model DNA sequences that are likely to assume a hairpin structure, such as those shown in Figure 1-6 on the next page. Hairpins in DNA and RNA sequences play an important role in the regulation of many processes, including transcription and translation. A hairpin is essentially a structure that bends back upon itself and is held together by Watson–Crick pairing. The paired bases in the hairpin structure are referred to as the "stem" and the unpaired, bulging bases are referred to as the "loop" (see Figure 1-6 on the facing page).

In order to form a stem–loop, or hairpin structure, the two sequences in the stem must be reverse complements of each other. This type of relationship is captured in the following

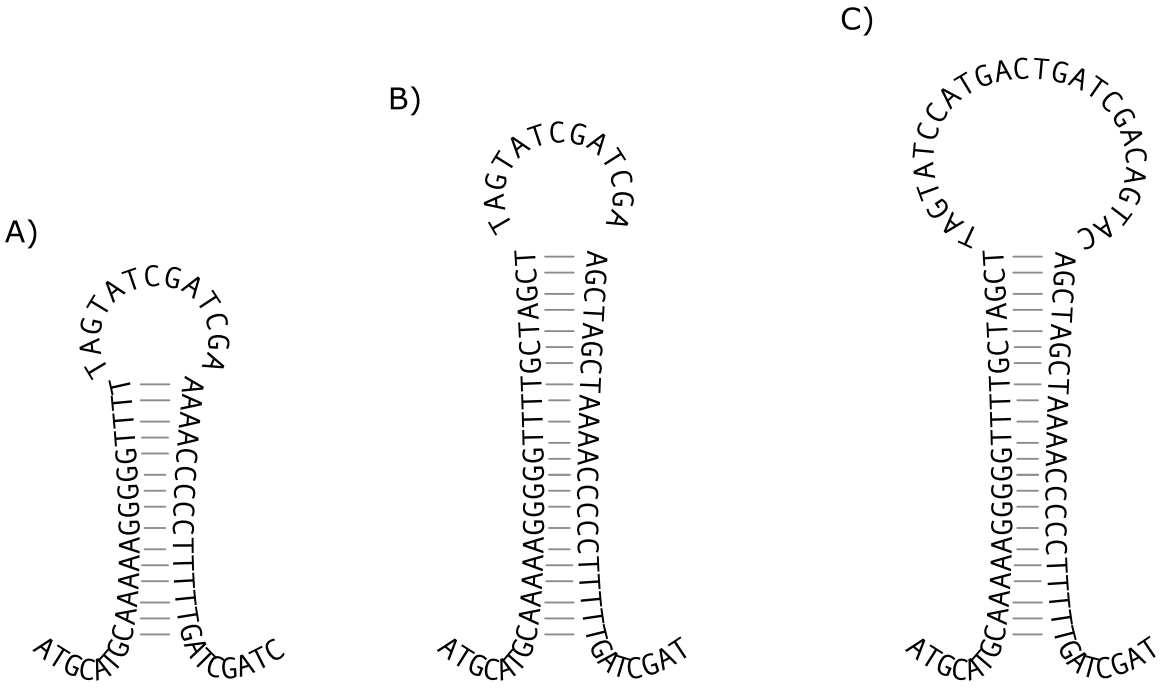


Figure 1-6: Hairpin loops in DNA secondary structures. A hairpin loop is a secondary structure in a sequence containing two regions that are reverse complements of each other. These regions form the "stem" of the hairpin loop. The figure shows three hairpin loops in which the stem size gets progressively larger. Also notice that, the bulbous region — the "loop" — which is not paired with any other region, can be of arbitrary size. The structures play an important part in the regulation of DNA transcription and, for RNA, in the process of translation.

grammar:

$$G = (\{\alpha, \beta, S\}, \{\mathsf{A}, \mathsf{G}, \mathsf{T}, \mathsf{C}\}, P, S), \tag{1.14}$$

where P is given by

$$P = \begin{cases} S \to \alpha \\ \alpha \to \mathsf{A}\alpha\mathsf{T} \mid \mathsf{T}\alpha\mathsf{A} \\ \alpha \to \mathsf{G}\alpha\mathsf{C} \mid \mathsf{C}\alpha\mathsf{G} \\ \alpha \to \beta \mid e \\ \beta \to \mathsf{A}\beta \mid \mathsf{T}\beta \mid \mathsf{G}\beta \mid \mathsf{C}\beta \mid e \end{cases} \tag{1.15}$$

The grammar shown in Equation 1.14 can describe any hairpin loop in which the stem consists of one or more complementary bases and the loop consists of zero or more bases. For example, consider the following derivation of the grammar.

$$\begin{aligned} S &\Rightarrow \alpha \\ &\Rightarrow \mathsf{A}\alpha\mathsf{T} \\ &\Rightarrow \mathsf{AG}\alpha\mathsf{CT} \\ &\Rightarrow \mathsf{AGG}\alpha\mathsf{CCT} \\ &\Rightarrow \mathsf{AGGC}\alpha\mathsf{GCCT} \\ &\Rightarrow \mathsf{AGGCT}\alpha\mathsf{AGCCT} \\ &\Rightarrow \mathsf{AGGCT}\beta\mathsf{AGCCT} \\ &\Rightarrow \mathsf{AGGCTA}\beta\mathsf{AGCCT} \\ &\Rightarrow \mathsf{AGGCTAAGCCT} \end{aligned} \tag{1.16}$$

This derivation produces a sequence that can form a hairpin structure with a stem size of five base pairs and a loop of a single base pair.

The grammar shown in Equation 1.14 is more complex than the grammar used to model the Abraham Lincoln motif (Equation 1.10 on page 27), because there are a long–range dependencies in the sequences. That is, a particular base produced by the grammar in Equation 1.14 is guaranteed to be complementary to a base on the other side of the sequence. In contrast, the productions used to model the Abraham Lincoln motif produced a set of simple derivations in a left–to–right order. Indeed, even more complicated grammars can describe still more long–range, complex interactions between the characters in a sequence.

Hierarchy of restricted grammars

Linguists classify grammars into four increasingly complicated groups based on the format of their productions. A grammar is

1. right–linear, or type–3, if each production in P is of the form $A \to \mathsf{x}B$, where A and B are in N and x is any string in $\Sigma*$;

2. context–free, or type–2, if each production in P is of the form $A \rightarrow \alpha$, where A is in N and α is in $(N \cup \Sigma)*$;

3. context– sensitive, or type–1, if each production in P is of the form $\alpha A \beta \rightarrow \delta \gamma \Gamma$, where A is in N, γ is non–null, and α, β, δ, and γ are in $(N \cup \Sigma)*$;

4. unrestricted, or type–0, if it adheres to none of these restrictions.

This classification system is referred to as the Chomsky hierarchy [53]. Each of these grammars defines a corresponding class of language, which is the set of all sequences that can be produced using a particular type of grammar.

Right–linear, or type–3 grammars are also called "regular" grammars and are the simplest type of grammar. These grammars are called right–linear because derivations of these grammars are produced stepwise from left to right, never growing from the center of the sequence as in the derivation shown in Equation 1.16. As I will show in Section 1.4 on page 33, despite their simplicity, regular grammars are the most frequently used motif model in bioinformatics.

Context–free grammars are the next most complicated class of grammatical model. Indeed, the hairpin grammar shown in Equation 1.14 on the preceding page is a context–free grammar. This type of grammar is characterized by "nested" dependencies (see Figure 1-7). The dependencies are nested in the sense that derivations of the grammar "grow" from the center, due to the structure of the productions.

Context–sensitive grammars and unrestricted grammars are the most complex classes of grammatical models. As shown in Figure 1-7 on the following page, context–sensitive grammars are characterized by long–range dependencies that are "crossing." Derivations of these grammars can typically "grow" from anywhere inside the sequence. For example, consider the following grammar that describes a card player arranging a deck of cards:

$$G = (\{\gamma, \beta, S\}, \{\clubsuit, \heartsuit, \spadesuit\}, P, S), \tag{1.17}$$

where P is given by

$$P = \begin{cases} S \rightarrow \gamma \\ \gamma \rightarrow \clubsuit\heartsuit\spadesuit \mid \clubsuit\gamma\beta\spadesuit \\ \spadesuit\beta \rightarrow \beta\spadesuit \\ \heartsuit\beta \rightarrow \heartsuit\heartsuit \end{cases} \tag{1.18}$$

This grammar is one of the most simple context–sensitive grammars. As well, it serves to illustrate that sequential data are not restricted to characters *per se*. Indeed, in Chapter 3 on page 89, I will extend the definition of sequential data to include ordered arrangements of multidimensional real–valued data sampled from a continuous distribution. Returning to the

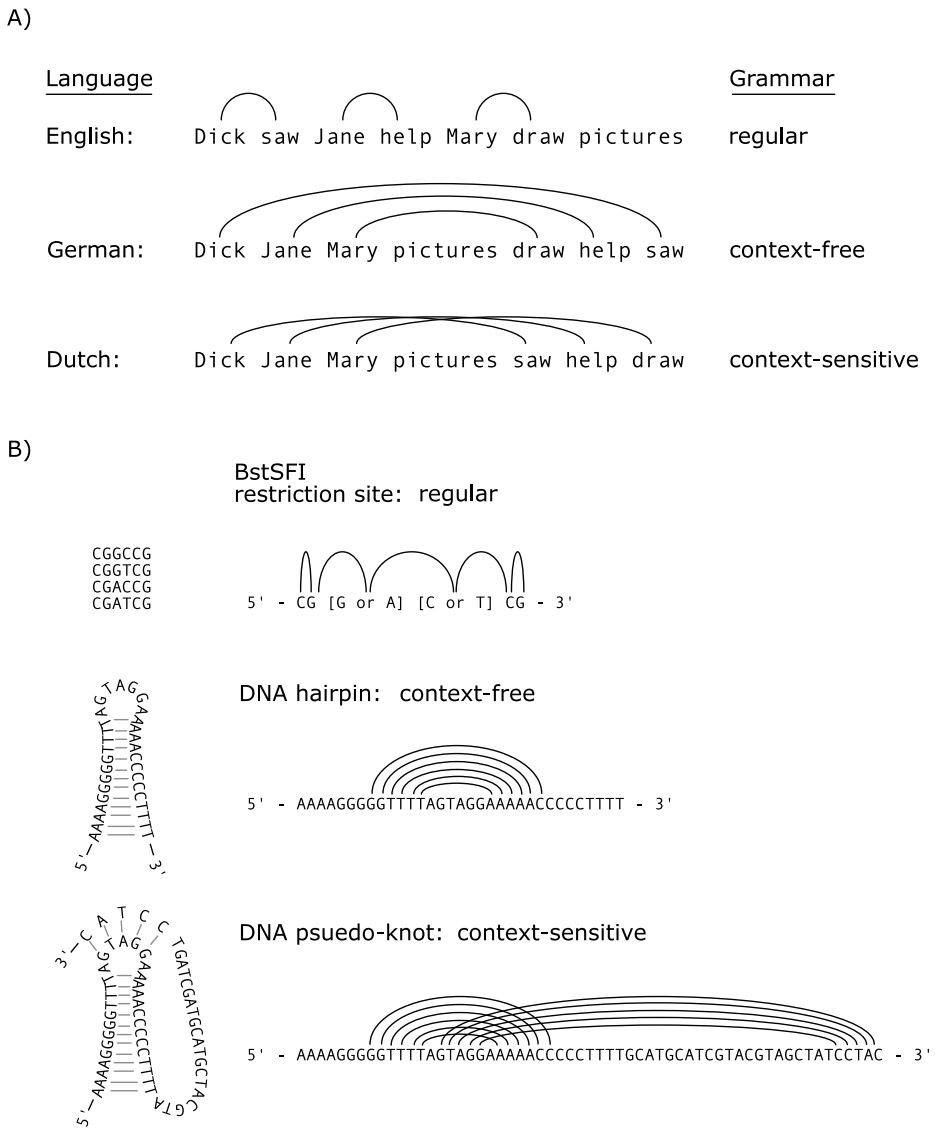


Figure 1-7: Noun–verb dependencies in various languages and their biological analogues. Part A) shows the sentence "Dick saw Jane help Mary draw pictures" translated grammatically into German and Dutch. That is, the words in the sentence are rearranged to reflect the rules of grammar in these two languages, but the sentence is not translated *per se*. As shown, the English version of the sentence has a relatively simple dependency structure between the nouns and verbs that can be modeled using regular grammars. In contrast, German and Dutch require more complicated grammatical models [43, 134, 228]. Part B) shows the biological analogue of the three sentences in Part A). Typically, restriction sites can be modeled using regular grammars, whereas complex DNA secondary structures require context–free or context–sensitive grammars [209]. In the first example, the arches are used to represent a "must be followed by" dependency. In the second two examples, they represent a "must be complementary to" dependency.

current example, consider the following derivation of this grammar:

$$
\begin{aligned}
S &\Rightarrow \gamma \\
&\Rightarrow \clubsuit\gamma\beta\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\gamma\beta\spadesuit\beta\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\gamma\beta\spadesuit\beta\spadesuit\beta\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\spadesuit\beta\spadesuit\beta\spadesuit\beta\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\beta\spadesuit\spadesuit\beta\beta\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\beta\spadesuit\beta\spadesuit\beta\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\beta\spadesuit\beta\beta\spadesuit\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\beta\beta\spadesuit\beta\spadesuit\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\beta\beta\beta\spadesuit\spadesuit\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\heartsuit\beta\beta\spadesuit\spadesuit\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\heartsuit\heartsuit\beta\spadesuit\spadesuit\spadesuit\spadesuit \\
&\Rightarrow \clubsuit\clubsuit\clubsuit\clubsuit\heartsuit\heartsuit\heartsuit\heartsuit\spadesuit\spadesuit\spadesuit\spadesuit
\end{aligned}
\tag{1.19}
$$

Notice that the derivation bears much similarity to the hairpin loop example shown in Equation 1.16 on page 30. However, as I showed earlier, hairpin loops can be described with a context–free grammar, which is more simple than the grammar used in the current playing card example. What distinguishes the two is the size of the "loop," the series of hearts in this example. Here, any derivation of the grammar has exactly the same number of clubs as it does hearts and spades. That is, if there are n clubs, there must be n hearts followed by n spades as below.

$$
\underbrace{\clubsuit\clubsuit\clubsuit\clubsuit\clubsuit\ldots\clubsuit\clubsuit}_{\text{exactly } n \text{ clubs}}\overbrace{\heartsuit\heartsuit\heartsuit\heartsuit\heartsuit\ldots\heartsuit\heartsuit}^{\text{exactly } n \text{ hearts}}\underbrace{\spadesuit\spadesuit\spadesuit\spadesuit\spadesuit\ldots\spadesuit\spadesuit}_{\text{exactly } n \text{ spades}} \tag{1.20}
$$

In contrast, the hairpin loop example introduced earlier was allowed to have an arbitrary number of intervening nucleotides. The extra restriction in this case can be thought of as a three–way dependency between the first clubs card, the first hearts card, and the first spades card. The same is true for the second cards in the succession, resulting in crossing dependencies, much like the Dutch example in Figure 1-7 on the preceding page. The moral of this example is that subtle changes in the structures that need to be modeled can have a profound effect on the appropriate choice of grammars.

Regular grammars and regular expressions

Building regular grammars and regular expressions

For many applications in bioinformatics and computer science, regular grammars are an appropriate motif model and more complicated context–free or context–dependent grammars are not required. For example, most compilers make wide use of regular grammars to interpret programming languages, such as C, C++, or Java. That is, these programming languages are regular languages in the mathematical sense — they have a rigid structure and lack long–range

dependencies.

Similarly, there are many phenomena in biology that can be modeled using regular grammars. For example, restriction enzymes, used for cutting DNA and RNA, typically recognize a set of motif instances that are easily modeled using regular grammars (see Figure 1-7 on page 32, part B).

In such cases, regular grammars are a convenient tool for two reasons. First, it is computationally simple to determine whether or not a string is a derivation of the given grammar, i.e. if the string is in the language of the grammar. This is not the case for more complicated grammars. In general, the computational complexity of this task rises rapidly for more complicated grammars and can take arbitrarily long for unrestricted grammars. Second, regular grammars can be represented compactly using a form called a regular expression. Consider the BstSFI restriction sites shown in Figure 1-7, reproduced below.

$$\begin{array}{l} \texttt{CGGCCG} \\ \texttt{CGGTCG} \\ \texttt{CGACCG} \\ \texttt{CGATCG} \end{array} \tag{1.21}$$

The sequences are described by the following regular grammar:

$$G = (\{\alpha, \beta, \gamma, S\}, \{\mathsf{A}, \mathsf{G}, \mathsf{T}, \mathsf{C}\}, P, S), \tag{1.22}$$

where P is given by

$$P = \begin{cases} S \to \mathsf{CG}\alpha \\ \alpha \to \mathsf{A}\beta \mid \mathsf{G}\beta \\ \beta \to \mathsf{C}\gamma \mid \mathsf{T}\gamma \\ \gamma \to \mathsf{CG} \end{cases} \tag{1.23}$$

This regular grammar can be represented much more succinctly in the following regular expression: `CG[AG][CT]CG`. The regular expression should be read as "any string starting with a `C` and a `G`, followed by either an `A` or a `G`, followed by either a `C` or a `T`, that ends with a `CG`." The term `[AG]` is called a bracketed expression and is used to indicate a production rule in which multiple characters are allowed. For example, the bracketed expression `[ATGC]` would indicate that any of the four nucleotides is permitted.

In order to introduce more complex features of regular expressions, consider the motif describing the short hematopoietin receptor family in Figure 1-8 on page 36. The motif is described by the following regular expression.

```
[LIVF].........[LIV][RK].(9,20)WS.WS....[FYW].
```
(1.24)

In this regular expression the individual characters represent amino acids (see Figure A-1 on page 174 in the Appendix). Here, the first bracketed expression `[LIVF]` indicates that leucine, isoleucine, valine, or phenylalanine are equally acceptable. The term "`.`" is called a "wild–

card" and indicates that any amino acid is acceptable. Or, in the general case, that any of the characters in Σ are acceptable. (Recall that Σ is the set of terminal symbols for a grammar.) The next special term in Equation 1.24 is "`.(9,20)`." This term indicates that the wild–card should be repeated for between nine and 20 places. For example, the regular expression consisting only of the term "`KR(2,4)`" has the following derivations: `KRR`, `KRRR`, `KRRRR`. Note that the strings `KR` and `KRRRRR` are not derivations of the grammar.

Because it is a more compact representation, regular grammars are usually recorded in regular expression form. In contrast, more complex grammars cannot be represented as a simple series of characters and symbols. This ease with which they can be communicated has been one of the factors promoting the widespread use of regular expressions — it would be inconvenient to discover a new protein motif and not be able to record the motif in an easily interpretable form for publication.

The regular expression formalisms presented here, such as the bracketed expression and the wild–card, are not exhaustive. There are many more terms that increase the richness of regular expressions, such as the Kleene star, "`*`", which means "zero or more of the preceding expression" and the "`+`" symbol, which means "one or more of the preceding expression." For an exhaustive treatment of regular expressions, the reader is referred is referred to publications by Sipser [233] and Friedl [84].

Matching regular grammars and regular expressions

Thus far, I have described how regular expressions be used to model a set of motif instances. However, a very common task is to then use a regular expression to look through new, longer sequences for "matches," i.e. subsequences of a given sequence that are derivations of the grammar that the regular expression encodes. For example, consider the following regular expression: `A[KR].Q[LV]C`. We would like to know if there are any derivations of this grammar within the sequence shown below.

`FLGARRQLCVVFKLAAKFQVCSKAKWQLCVFPAVFGKV` (1.25)

A simpleminded approach to this problem is to start with a beginning of the sequence, at letter `F`, and ask whether or not a derivation of the grammar could start that position. Obviously, any derivation of the grammar must begin with an `A`, so the answer is "no." Moving on to the first `A` in the sequence, we see that it is followed by a `K`, which is allowed by the grammar, and that the `K` can be followed by any character, etc. Following this procedure reveals three matches of the regular expression in the sequence, which are underlined below.

`FLGARRQLCVVFKLAAKFQVCSKAKWQLCVFPAVFGKV` (1.26)

In general, algorithms designed to match regular expressions against sequences or other kinds of text use an approach that is, at its core, the same as the simpleminded approach above. One such algorithm and piece of software is described in Section 4.2 on page 127.

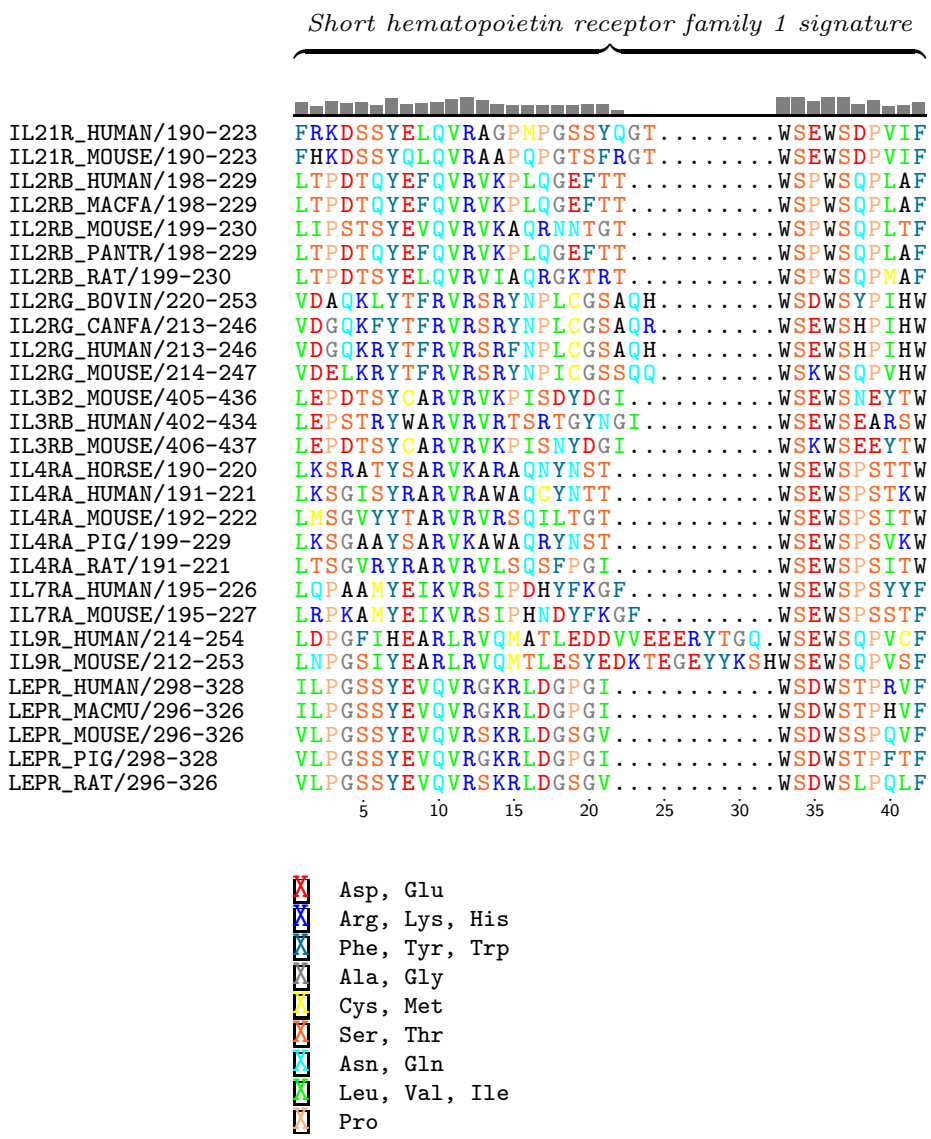


Figure 1-8: Regular grammar describing the short hematopoietin receptor family 1 signature [118]. These proteins are mostly receptors for interleukin cytokines. They are selectively expressed in lymphoid tissues and are typically membrane–bound [190]. The region shown in the figure is characterized by the regular expression `[LIVF].........[LIV][RK].(9,20)WS.WS....[FYW]`. This motif is required for proper protein folding, efficient intracellular transport, and cell–surface receptor binding. The motif is relatively sensitive for the receptor family; however, it misses the rodent thymic stromal lymphopoietin protein receptors, which are in the same family. Furthermore, the motif is not as specific as it could be — as shown above, the motif matches five receptors for the leptin obesity factor, which are not in the same family. Notice that the bar at the top shows the degree of conservation at each position; the amino acids are colored to reflect their physiochemical properties; and, the bracketed expressions, such as `[LIV]`, tend to group together amino acids with similar physiochemical properties.

Position weight matrices

Building position weight matrices

Despite their utility, regular grammars and regular expressions are not suitable for modeling all kinds of motifs. As I showed earlier, regular grammars cannot describe long–range, nested, or crossing dependencies between characters. However, there are also motifs where these dependencies do not exist and yet regular expressions are not accurate models.

Consider the collection sequences shown in Figure 1-9 on the following page. This collection comprises numerous 3' splice sites from the fission yeast *Schizosaccharomyces pombe*. Each sequence is seven nucleotides in length and straddles the intron/exon boundary in a gene. After transcription, these sites will form a "branch point" allowing the introns to be excised from the pre–RNA to form the mature mRNA.

Notice that to sensitively describe these sequences using a regular expression, we would use `[ATGC][ATGC][CT]T[ATG]A[CT]`. This motif will match all of the instances, but it could also match many more: based on the number of bracketed expressions, this regular expression would match 192 unique sequences.

Notice too that each column of the aligned instances shown in Figure 1-9 has a particular "preference" for one kind nucleotide. For example, all but 10 of the sequences have a thymine at the last position. But, in the motif `[ATGC][ATGC][CT]T[ATG]A[CT]`, either cytosine or thymine is allowed in the last position, without any preference. Obviously, this regular expression would be more specific if we labeled the last bracketed expression with these preferences, i.e. "either cytosine or thymine, but with a seven–fold preference for the thymine."

Incorporating such preferences into the grammatical framework requires only minor changes. Recall from the definition on page 30 that a grammar is regular (or right–linear or type–3) if each production in P is of the form $A \rightarrow \mathtt{x}B$, where A and B are in N and $\mathtt{x}$ is any string in $\Sigma*$. A similar set of restrictions defines a position weight matrix, which is a grammar in which each production in P is of the form $A \xrightarrow{p_i} \mathtt{x}B$, where A and B are in N and $\mathtt{x}$ is any **character** in Σ, and p_i is the probability of production i. As well, $\sum_i p_1$ must equal one for all of the productions on which A is on the left side. In loose terms, the position weight matrix, or PWM, can be thought of as a probabilistic regular expression. Using this new structure, the regular expression `[ATGC][ATGC][CT]T[ATG]A[CT]` can be written as a PWM grammar,

$$G = (\{S, \alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta\}, \{\mathtt{A}, \mathtt{T}, \mathtt{G}, \mathtt{C}\}, P, S), \tag{1.27}$$

```
yeast1        AGCTGAC
yeast2        GACTAAT
yeast3        GGCTAAT
yeast4        CATTAAC
yeast5        CTCTAAC
yeast6        TTTTAAC
yeast7        TACTAAC
yeast8        TACTAAT
yeast9        TACTAAC
yeast10       TGCTAAC
yeast11       TATTAAC
yeast12       TTCTAAC
yeast13       TTCTAAC
yeast14       TGCTAAC
yeast15       TTCTAAC
yeast16       CACTAAC
yeast17       TGCTAAC
yeast18       GACTAAC
yeast19       TATTAAC
yeast20       CGCTAAC
yeast21       AACTAAC
yeast22       TACTAAT
yeast23       AACTAAC
yeast24       TACTAAC
yeast25       TACTTAC
yeast26       TACTAAT
yeast27       TACTAAT
yeast28       TTCTAAC
yeast29       TTCTAAC
yeast30       CGCTGAC
yeast31       TACTAAT
yeast32       TACTAAT
yeast33       TACTGAC
yeast34       TACTAAC
yeast35       TACTGAC
yeast36       TGCTAAC
yeast37       AACTAAC
yeast38       TTCTAAC
yeast39       GACTAAC
yeast40       AACTAAC
yeast41       TTCTAAT
yeast42       TTCTAAC
yeast43       TATTAAC
yeast44       TACTGAC
yeast45       TTCTAAC
yeast46       TACTAAT
yeast47       TCCTAAC
yeast48       TACTAAC
yeast49       TACTAAC
yeast50       TTCTAAC
yeast51       GACTAAC
yeast52       TATTAAC
yeast53       TACTGAC
yeast54       TGCTAAC
yeast55       TACTAAC
yeast56       TTTTAAC
yeast57       TTCTAAC
yeast58       TACTAAC
yeast59       TTCTAAC
yeast60       TACTAAC
yeast61       TTCTAAC
yeast62       TACTAAC
yeast63       TACTAAC
yeast64       TACTAAC
yeast65       TACTAAC
yeast66       TATTAAC
yeast67       TACTAAC
yeast68       GACTAAC
yeast69       GACTAAC
yeast70       TACTAAC
yeast71       TACTTAC
yeast72       TTCTAAC
yeast73       TACTAAC
yeast74       TACTAAC
yeast75       TTTTAAC
consensus     agcTgAc
```

Figure 1-9: Yeast 3' splice sites. The figure shows the 3' splice junction of 75 introns from the fission yeast *Schizosaccharomyces pombe*. The coloring of the sequences is used to show the degree of conservation. Notice that certain columns, such as the final column, have strong preferences for certain nucleotides. The final column has a strong preference for cytosine; however, thymine is allowed occasionally. The consensus sequence at the bottom shows the most common nucleotide at each position. If the nucleotide is not perfectly conserved it is shown in lowercase. This kind of motif is poorly modeled using regular grammars. Instead, position weight matrices preserve much of the "preference information."

where P is the set of productions below.

$$\mathsf{P} = \begin{cases} \mathsf{S} \rightarrow \alpha \\ \alpha \xrightarrow{0.067} \mathsf{A}\beta \\ \alpha \xrightarrow{0.773} \mathsf{T}\beta \\ \alpha \xrightarrow{0.093} \mathsf{G}\beta \\ \alpha \xrightarrow{0.067} \mathsf{C}\beta \\ \beta \xrightarrow{0.627} \mathsf{A}\gamma \\ \beta \xrightarrow{0.240} \mathsf{T}\gamma \\ \beta \xrightarrow{0.120} \mathsf{G}\gamma \\ \beta \xrightarrow{0.013} \mathsf{C}\gamma \\ \gamma \xrightarrow{0.120} \mathsf{T}\delta \\ \gamma \xrightarrow{0.880} \mathsf{C}\delta \\ \delta \xrightarrow{1.000} \mathsf{T}\epsilon \\ \epsilon \xrightarrow{0.893} \mathsf{A}\zeta \\ \epsilon \xrightarrow{0.027} \mathsf{T}\zeta \\ \epsilon \xrightarrow{0.080} \mathsf{G}\zeta \\ \zeta \xrightarrow{1.000} \mathsf{A}\eta \\ \eta \xrightarrow{0.133} \mathsf{T} \\ \eta \xrightarrow{0.867} \mathsf{C} \end{cases} \tag{1.28}$$

Equation 1.15 can be represented much more compactly as a frequency matrix

$$\mathsf{f} = \begin{pmatrix} 0.067 & 0.627 & 0.000 & 0.000 & 0.893 & 1.000 & 0.000 \\ 0.773 & 0.240 & 0.120 & 1.000 & 0.027 & 0.000 & 0.133 \\ 0.093 & 0.120 & 0.000 & 0.000 & 0.080 & 0.000 & 0.000 \\ 0.067 & 0.013 & 0.880 & 0.000 & 0.000 & 0.000 & 0.867 \end{pmatrix} \tag{1.29}$$

in which each row corresponds to a single character in Σ and each column corresponds to a single non–terminal character in N (where Σ is disjoint from N, as usual). So, in Equation 1.29, the rows correspond to A, T, G, and C; and the columns correspond to α, β, γ, δ, ϵ, ζ, and η, where S was omitted.

Notice that a derivation of the grammar in Equation 1.27 on page 37 is necessarily also a derivation of the regular grammar `[ATGC][ATGC][CT]T[ATG]A[CT]`, and vice versa. As such, the two grammars describe the same language, or the set of all derivations. But, because the productions of Equation 1.27 are weighted by probability, certain derivations are more

probable than others. The degree to which one derivation of the grammar is more probable than another is characterized by the derivation's log–odds score. To compute the log–odds score, first requires a log–odds matrix, Θ, where

$$\Theta_{ij} = \log_2 \left(\frac{f_{ij}}{q_j} \right). \tag{1.30}$$

The calculation of the frequency and log–odds matrices for the 3' yeast splice sites is shown in Table 1.3. Here, Θ is given by

$$\Theta = \begin{pmatrix} -1.907 & 1.326 & \varnothing & \varnothing & 1.837 & 2.000 & \varnothing \\ 1.629 & -0.059 & -1.059 & 2.000 & -3.229 & \varnothing & -0.907 \\ -1.421 & -1.059 & \varnothing & \varnothing & -1.644 & \varnothing & \varnothing \\ -1.907 & -4.229 & 1.816 & \varnothing & \varnothing & \varnothing & 1.794 \end{pmatrix}, \tag{1.31}$$

where $\varnothing$ is used to indicate values that are undefined because $f_{ij} = 0$ and $\log_2 0$ is undefined. Given the log–odds matrix form of a PWM, the score of any derivation of the PWM is computed merely by looking up values in Θ. Consider the sequence AGCTGAC, which is both a derivation of the grammar shown in Equation 1.27 on page 37 and the first of the sequences shown in Figure 1-9 on page 38. The log–odds score for this sequence is

$$\begin{aligned} \text{score} &= \Theta_{0,0} + \Theta_{2,1} + \Theta_{3,2} + \Theta_{1,3} + \Theta_{2,4} + \Theta_{0,5} + \Theta_{3,6} \\ &= -1.907 - 0.059 + 1.816 + 2.000 - 1.644 + 2.000 + 1.794 \\ &= 4.000. \end{aligned} \tag{1.32}$$

Notice that the score for a sequence that is not a derivation of the grammar is undefined, or, effectively $-\infty$. Table 1.3 on the facing page shows the calculation of the frequency matrix, log–odds matrix, and the scoring of example sequences for this PWM. Also, a small program for calculating a PWM from a set of sequences is provided in Section A.2.1 on page 176 of the Appendix.

The "strength" of a PWM motif is measured by a quantity called its entropy. The motif entropy is the sum of the entropies of each column, or position in the motif. This entropy of a given column in a PWM is a measure of the disorder, or the randomness of the distribution of letters. The column entropy is measured in bits and is given by

$$h_i = -\sum_j f_{ij} \log_2 f_{ij} \tag{1.33}$$

where, f_{ij} is the frequency matrix as shown in Figure 1.3 on the facing page. The entropy of the whole motif is just the sum of the entropies of the columns:

$$H = -\sum_i \sum_j f_{ij} \log_2 f_{ij}. \tag{1.34}$$

Typically, the entropy is measured relative to the background entropy. As above, the back-

Table 1.3: The construction of a position weight matrix from the collection of sequences shown in Figure 1-9 on page 38. Part A) shows the number of nucleotides of each type that occur in each of the seven positions of the aligned sequences. For example, in the first position, there are 58 thymines. Part B) shows the frequency matrix f, where each $f_{ij} = (c_{ij} / \sum_j c_{ij})$. Part C) shows the log–odds matrix Θ, where each $\Theta_{ij} = \log_2(f_{ij}/q_j)$ and q is the vector of background frequencies for the nucleotides. Part D) shows the scoring of three different sequences. To compute the score for a sequence, the corresponding nucleotide at each column is looked up in Θ and the columns are summed together.

A) Count Matrix (c_{ij}):

A	5	47	0	0	67	75	0
T	58	18	9	75	2	0	10
G	7	9	0	0	6	0	0
C	5	1	66	0	0	0	65

⇓

B) Frequency Matrix (f_{ij}):

A	0.067	0.627	0.000	0.000	0.893	1.000	0.000
T	0.773	0.240	0.120	1.000	0.027	0.000	0.133
G	0.093	0.120	0.000	0.000	0.080	0.000	0.000
C	0.067	0.013	0.880	0.000	0.000	0.000	0.867

⇓

C) Log–odds Matrix (Θ_{ij}):

A	-1.907	1.326	∅	∅	1.837	2.000	∅
T	1.629	-0.059	-1.059	2.000	-3.229	∅	-0.907
G	-1.421	-1.059	∅	∅	-1.644	∅	∅
C	-1.907	-4.229	1.816	∅	∅	∅	1.794

⇓

D) Example sequence scoring:

query1	T	A	C	T	T	A	C
Σ	1.629	1.326	1.816	2.000	-3.229	2.000	1.794
				= 7.335			
query2	T	T	C	T	A	A	C
Σ	1.629	-0.059	1.816	2.000	1.837	2.000	1.794
				= 11.017			
query3	G	T	A	T	A	A	T
Σ	-1.421	-0.059	∅				
				= ∅			

ground entropy of a single column is

$$h_i^\circ = -\sum_j q_j \log_2 q_j \tag{1.35}$$

where, q_j is the *a priori*, background frequency of the letter denoted by index j. In the case of identically distributed nucleotides, each q_j is 0.25; i.e. $q_A = 0.25$, $q_C = 0.25$, $q_T = 0.25$, and $q_G = 0.25$. The background entropy of the entire motif is

$$H^\circ = -\sum_i \sum_j q_j \log_2 q_j. \tag{1.36}$$

The difference between the background entropy and the motif entropy is referred to as the information content, I, of the PWM. Using Equations 1.33– 1.36 above, the information content can be calculated as

$$\begin{aligned} I &= H^\circ - H \\ &= -\sum_i \sum_j q_j \log_2 q_j - \left(-\sum_i \sum_j f_{ij} \log_2 f_{ij} \right) \\ &= \sum_i \sum_j f_{ij} \log_2 \left(\frac{f_{ij}}{q_j} \right). \end{aligned} \tag{1.37}$$

Notice that the information content of the motif is minimized when the nucleotide distribution for each column is exactly the background distribution of nucleotides. That is, when $f_{ij} = q_j$ for all i, the $\log_2(f_{ij}/q_j)$ terms are zero. This makes sense intuitively: if the PWM describes the background distribution, the motif can obviously not be distinguished from the background and therefore contains no information. In this same case, the entropy of the motif is maximized and is equal to the background entropy.

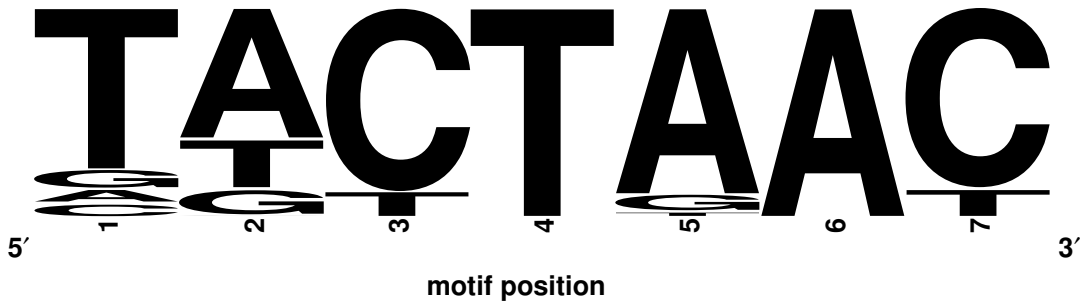
PWMs are commonly represented by two varieties of schematics: pictograms and sequence logos. An example of each of these is shown in Figure 1-10 on the next page. A pictogram is essentially a visualization of the frequency matrix representing a PWM, whe, whereas A sequence logo is a pictogram that is scaled to reflect the information content at each position in the PWM.

Matching position weight matrices

Thus far, I have shown how a PWM can be used to model a set of motif instances and how a derivation of a PWM grammar can be scored. PWMs can also be used to search in new, long sequences for regions of the sequence that appear to match the motif. This is accomplished by "sliding" the PWM over the length of the sequence to look for subsequences that have high log–odds scores.

Consider searching the sequence TAGCTGACTGAC. To slide the PWM over this sequence is equivalent to evaluating the score of each seven nucleotide substring: TAGCTGA, AGCTGAC, GCTGACT, CTGACTG, TGACTGA, and GACTGAC. These are ∅, 4.0, ∅, ∅, ∅, and 5.871. That is, there are two matches for the PWM, one stronger than the other. This method can be

A) Pictogram of the PWM in Table 1.3 on page 41



B) Logo of the PWM in Table 1.3 on page 41

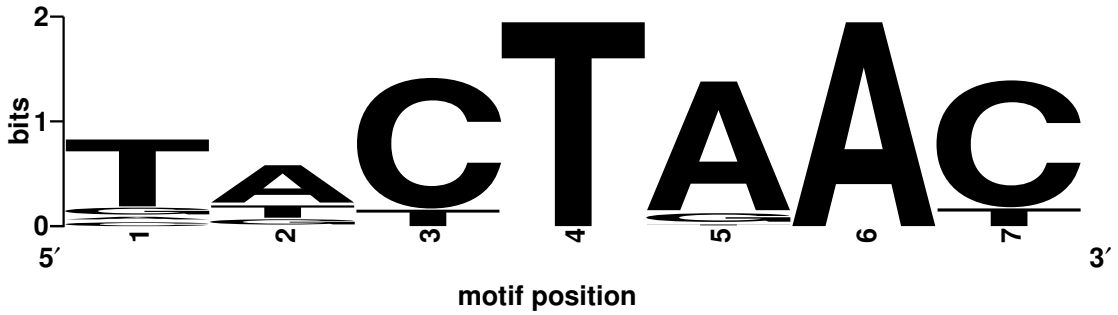


Figure 1-10: Yeast 3' splice site pictogram and logo. Part A) shows the PWM in Table 1.3 on page 41 represented as a pictogram. At each position in the motif, the height of the nucleotides is scaled in proportion to their frequency in f, with the more frequent nucleotides always placed on top. The pictogram clearly shows that positions four and six are perfectly conserved, whereas the other positions are distributed between many nucleotides. Part B) shows a sequence logo representation of the same PWM. The sequence logo is a pictogram in which the height of each column is scaled in proportion to the information content contributed by that position to the motif (see Equation 1.37 on the preceding page). Taller columns have a nucleotide distribution that deviates strongly from the background distribution. In this sequence logo, the background distribution is arbitrarily set to equal *a priori* probability of each nucleotide. As such, the maximum information content in any column is two bits, which is achieved only in the two perfectly conserved positions of the motif.

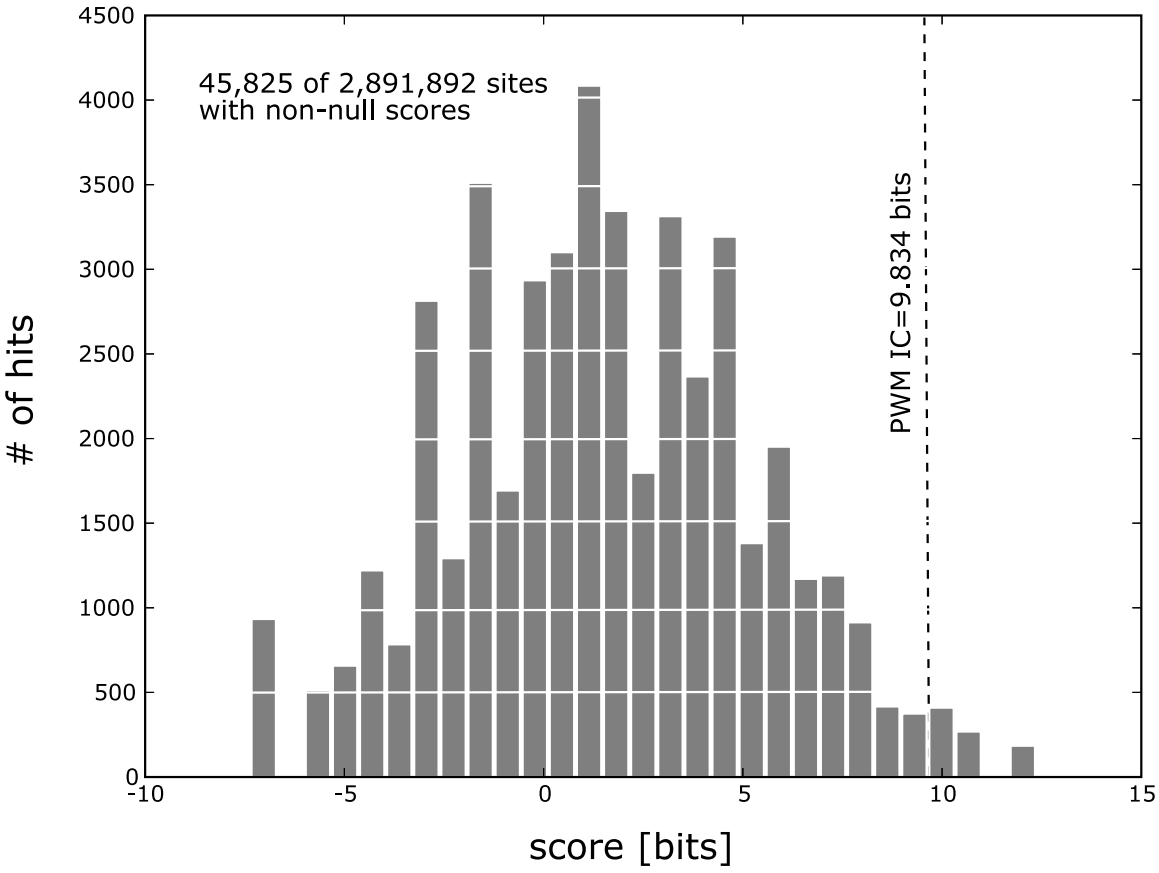


Figure 1-11: Distribution of log–odds scores obtained by searching the PWM in Table 1.3 on page 41 against chromosomes 1–4 of *Saccharomyces cerevisiae*. Of the nearly 3 million possible sites, only 46,000 had non–null scores. As the figure shows, the score distribution is roughly Gaussian. The dashed line indicates the information content of the PWM. Scores above this line are generally considered strong matches. PWMs are more specific than regular grammars, because the threshold above which a match is considered "true" can be varied. In contrast, with a regular grammar is either a match or not a match: there is no variable threshold.

used to search for a PWM in much larger sequences as well. For example, Figure 1-11 (page 44) shows the distribution of scores obtained by searching the PWM in Table 1.3 on page 41 against chromosomes 1–4 of the *Saccharomyces cerevisiae* genome.

1.5 Tools for motif discovery

Introduction

In general, the goal of motif discovery is to derive a set of grammars that sensitively matches a set of given sequences. This is the inverse of many of the examples in the previous section. That is, imagine a case in which you are given a set of derivations and then asked what kind of grammar could have produced the derivations? This is what is called grammar induction in the

computational linguistics literature and is equivalent to guessing the grammar of an unknown language given a few sentences in the language.

In bioinformatics, this task is usually presented in a slightly more difficult form. To illustrate this, consider a hypothetical challenge in which a colleague hides derivations of the regular grammar `[KR]QTRP.[RT]K` in a set of sequences that consist otherwise of random characters. You are presented with the sequences and asked what grammar was hidden therein.

```
AJDFIOASODVIKQTRPXYKIIWEJSJ
JKQTRPCRKXUCIQWEMFIOAKLGS
ADUHFIKACRQTRPXKMSKDAFIOAS
```
(1.38)

Without any prior knowledge, this task is nearly impossible. The colleague could have hidden a regular grammar that consisted solely of `S`, which would be undetectable since there are many characters that occur in all of the sequences. Further, he could have hidden "`....`", in which case there would be no evidence in the sequences. This conundrum is closely related to one of the three tenets of motif discovery developed in Section 1.2 on page 21: the answer to any motif discovery question is invariably dependent on at least some prior knowledge about what forms a motifs may take.

Suppose then that the colleague says the motif is at least five characters long, is a regular grammar, and all the derivations of the grammar look "pretty similar." Given this information, a logical approach would be to look for subsequences of five characters that look relatively similar and occur in all three sequences. After diligently scanning the sequences, you can find two sets of three that seemed to fit this description: {`FIOAS`, `FIOAK`, `FIOAS`} and {`KQTRP`, `KQTRP`, `RQTRP`}. Knowing the answer, it is obvious that we are on the right track; however, again we are at an impasse. It would be easy to write a regular expression describing either of these sets. But, *a priori* it is impossible to tell which set may be the correct answer. The first set has a `K` that is mismatched with a `S`, whereas the latter set has a `K`–`R` mismatch. If these were amino acid sequences we could say that lysine, `K`, is more similar to arginine, `R`, than it is to serine, `S`. Therefore, we might choose the latter set. This decision is related to the remaining two tenets of motif discovery developed in Section 1.2: the answer to any motif discovery problem will always depend on a predefined metric of similarity and a method for grouping together similar objects.

In the following sections, I review a number of approaches for problems such as the example given above, focusing on the two most common classes of approach: those that use regular expression motif models and those that use PWMs. All of these approaches, without exception, always require some degree of intelligent guidance by the user that can be reduced to the three tenets discussed above. In general, motif discovery tools that do not have such requirements have made assumptions on the user's behalf.

Teiresias and other regular expression–based tools

Because they are convenient from both a computational perspective and from the perspective of communicating results, regular expressions are the most common form of motif model used in bioinformatics. Table 1.4 on page 47 shows a list of publications introducing motif discovery

tools in this class. Within the field, these algorithms are commonly referred to as "motif driven" or "pattern driven" algorithms [41]. At their most basic conceptual level, all of these approaches work by first enumerating possible patterns and then checking for the patterns in the sequence set [41].

There are three principal characteristics that distinguish between the various algorithms shown in Table 1.4, which are as follows:

1. The regular expression class complexity.

2. The completeness of the returned motif set. That is, does the algorithm return *all* patterns present in the input sequences?

3. The motif *maximality*. For instance, in the two strings "KYLEJ" and "KYLEL", the motif "KYLE" is maximal, whereas "KYL" is not, because we could add an "E" without decreasing the number of times it occurs. In essence, maximality is a proxy for specificity.

The most important of these distinguishing features is the complexity of the regular expression class that an algorithm returns. No motif discovery tool can search for the "universe" of regular expressions. Recall from the previous section that "`...`" and other types of motifs will always be present, and therefore such a result is meaningless. Furthermore, enumerating regular expressions is NP–complete [90, 161], meaning that, in general, the runtime of a motif discovery tool will increase exponentially with the size of the sequence set it is given. As I showed in the previous section, the answer to any motif discovery problem will always require some *a priori* knowledge of the kinds of motifs that might be found, and simply specifying that the grammar is regular is not enough. Accordingly, most motif discovery tools restrict themselves to finding a particular subclass of regular expressions. This motif class determines the form of each pattern, p_i that we find. Below, a few motif classes, commonly used in biological sequence analysis, are enumerated in order of increasing complexity [82]:

- $p_i \in \Sigma^*$: This is the class of all "solid" patterns, for example "KAGTPT" and "TAGCGGGAT".

- $p_i \in (\Sigma \cup \{.\})^*$: This is the class of all patterns that can have "wildcard" positions, which are denoted by ".", for example "K.G.PT" and "TA...GGAT". The wildcard means that any character from the alphabet will suffice in that position.

- $p_i \in (\Sigma \cup R)^*$: This is the class of all patterns that can have "bracketed" expressions, for example "K[ADG]G[KQ]PT" and "TA[GA][TC]GGAT". The bracketed expression "[TC]" means that either "T" or "C" will suffice in that position. In this notation, R represents the set of characters in the brackets, for example $R = \{TC\}$ or $R = \{GA\}$.

- $p_i \in (\Sigma \cup .)^*$: This is the class of "flexible" patterns. For example "K.(1,3)G.(2,5)PT", where ".(2,5)" means that anywhere between two and five wildcards can exist at that position, that is .(2,5) can be any one of {.., ..., ..., ...}.

In general, the more complex these patterns are, the more expressive the languages will be that we find. However, with increasing complexity, the computational difficulty of the motif discovery problem increases drastically [90, 161].

Table 1.4: Motif discovery tools using regular expressions or similar models. This list is not intended to be exhaustive; however, it includes many of the well–known motif discovery tools used in bioinformatics. Early methods tended to use consensus strings or simple word counting approaches, i.e. counting the occurrences of "n–mers" such as the 4–mer ATGC. Words that are statistically over–represented are called motifs. Later approaches used more complex regular expressions, cf. Rigoutsos and Floratos [207].

Authors	Year	Citation
Queen et al.	1982	[202]
Galas et al.	1985	[87]
Mengeritsky and Smith	1987	[169]
Staden	1989	[237]
Neuwald and Green	1994	[179]
Jonassen et al.	1995	[132]
Wolferstetter et al.	1996	[270]
Sagot et al.	1997	[215]
Rigoutsos and Floratos	1998	[82, 207]
van Helden et al.	1998	[252, 253]
Jacobs Anderson and Parker	2000	[128]
Marsan and Sagot	2000	[167]
Pevzner and Sze	2000	[195]
Bussemaker et al.	2000	[47]
Kiełbasa et al.	2001	[138]
Horton	2001	[123]
Keich and Pevzner	2002	[137]
Eskin and Pevzner	2002	[78]
Buhler and Tompa	2002	[46]
Sinha and Tompa	2002	[232]
Price et al.	2003	[199]
Sinha	2003	[231]
Danilova et al.	2003	[64]
Ganesh et al.	2003	[88]
Liang et al.	2004	[151]
Fogel et al.	2004	[83]
Pavesi et al.	2004	[191]
Hernandez et al.	2004	[114]
Markstein et al.	2004	[166]
Frith et al.	2004	[86]
Sumazin et al.	2005	[240]

Also, for some of these tools, it is possible to guarantee the completeness of the set of returned patterns. That is, a particular tool may guarantee that all regular expressions meeting particular characteristics are discovered. However, this guarantee comes at the price of increased time and space complexity. That is, the set of all possible patterns is very large and can take a large space to enumerate and a long time to search through. As such, many motif driven algorithms use heuristics to limit the space of patterns that are searched.

Here, I will focus on the Teiresias algorithm as a representative regular expression–based motif discovery tool. Notably, Teiresias is the basis for much of the work in this thesis, particularly in Chapters 1 and 2. A more detailed description of Teiresias is available elsewhere [82, 207].

Given a set of sequences $S = \{s_0, s_1, \ldots, s_n\}$, and integers L, W, and K, Teiresias finds all patterns involving at least L non–wildcard characters that occur at least K times and have a fraction of non–wildcard characters of at least L/W. This set of patterns is called $\mathcal{C}$, where $\mathcal{C} = \{p_1, p_2, \ldots, p_m\}$ and each $p_i \in \Sigma(\Sigma \cup \{.\})\Sigma$. This is the set of all regular expressions that begin and end with a character, but may have an arbitrary number of wild–cards and characters in the middle subject to the L and W restriction, for example, `AXG`, `A.G`, `K..R.G`, etc. For each motif p_i in $\mathcal{C}$, Teiresias returns an offset list $\mathscr{L}(p_i)$ that specifies each sequence–position combination where the motif occurs (cf. Figure 1-13 on page 51).

The support of a motif is equal to the number of its occurrences (or, equivalently, "instances" or "embeddings"), $|\mathscr{L}(p)|$. Essentially, L defines the minimum size of patterns in which we are interested, and L/W defines the minimum specificity (the fewer wildcards, the more specific a motif). The four distinguishing characteristics of the Teiresias algorithm are as follows:

1. All *maximal* patterns are reported (see below for a definition of "maximal").

2. Only the maximal patterns are reported.

3. Running time depends only on the number of patterns present in the data, that is it is *output sensitive*.

4. Patterns can be arbitrarily long.

The most important characteristic of Teiresias is that it returns the *complete* set of maximal patterns. And, because of the manner in which these patterns are handled internally by Teiresias, the algorithm runs very quickly.

In the Teiresias parlance, a maximal motif is a regular expression which has the following properties:

1. The motif cannot be made more specific without producing a motif with fewer embeddings (i.e., without $|\mathscr{L}(p)|$ decreasing); and

2. The motif is not missing any instances, i.e. $\mathscr{L}(p)$ includes the locations of all instances of the motif.

These two criteria can be summarized qualitatively by stating that a maximal motif is not "missing" any locations and is as wide as possible, and thus it is as specific and sensitive as possible. Here, "specific" has a particular meaning: a pattern p_i is more specific than p_j if p_j can be transformed into p_i by substituting one or more wild–cards for a character, or by appending

wild–cards and characters to either side of p_j. For example, `CH.MEN..N` is less specific than all of the following regular expressions: `CHEMEN..N`, `CH.MEN..NE.R`, and `CH.MEN.IN`. Necessarily, if a pattern p_i is more specific than a pattern p_j, then

$$|\mathscr{L}(p_i)| \leqslant |\mathscr{L}(p_j)|. \tag{1.39}$$

Teiresias works in two phases: scanning and convolution. During the scanning phase, Teiresias enumerates all "elementary motifs" with exactly L characters and at most $W - L$ wild–cards (see Figure 1-12 on the following page). Elementary motifs are short regular expressions that can be stitched together to form longer regular expressions that are more specific, using the definition of specificity above. For example, as shown in Figure 1-12, the sequences

```
KDWVQKRK
 CWCQKRK
WDQKRKNP
```

have five motifs with 1) exactly $L = 3$ characters, 2) no more than $W - L$ wild–cards for every window of $L = 3$ characters, and that 3) occur at least three times: `W.QK`, `QKR`, `QK.K`, `KRK`, and `Q.RK`. These are the elementary motifs.

In the convolution phase, the elementary motifs are stitched together to see if more specific motifs can be found. The process of convolution is defined as follows:

$$\begin{aligned} p_k &= p_i \oplus p_j \\ &= \begin{cases} p_k p_i' & \text{if } \operatorname{suffix}_L(p_i) = \operatorname{prefix}_L(p_j), \\ \varnothing & \text{otherwise.} \end{cases} \end{aligned} \tag{1.40}$$

In the equation above $\operatorname{prefix}_L(p_i)$ is the sub–pattern at the beginning of p_i with exactly $(L - 1)$ characters. Similarly, $\operatorname{suffix}_L(p_i)$ is the sub–pattern at the end of p_i with exactly $(L - 1)$ characters. For example:

$$\begin{aligned} \operatorname{prefix}_3(\texttt{W.QK}) &= \texttt{W.Q} \\ \operatorname{suffix}_3(\texttt{W.QK}) &= \texttt{QK} \\ \operatorname{prefix}_3(\texttt{QKR}) &= \texttt{QK} \\ \operatorname{suffix}_3(\texttt{QKR}) &= \texttt{KR}\,. \end{aligned} \tag{1.41}$$

To illustrate this, consider the following examples:

$$\begin{aligned} \texttt{DF.A.T} \oplus \texttt{A.TSE} &= \texttt{DF.A.TE} \\ \texttt{L.XF.A.MM} \oplus \texttt{A.MSE} &= \texttt{L.XF.A.MME} \\ \texttt{WX.N.N} \oplus \texttt{N.PSE} &= \varnothing. \end{aligned}$$

If two motifs can be convolved — i.e. $p_i \oplus p_j \neq \varnothing$ — then the offsets of the new, longer

```
>seq 0
KDWVQKRK
>seq 1
CWCQKRK
>seq 2
WDQKRKNP
```

⇓

Teiresias

L/*W*/K = 3/4/3

⇓

Elementary motifs:

motifs →	W.QK	QKR	QK.K	KRK	Q.RK
offset #0	KD**W**V**QK**RK	KDWV**QKR**K	KDWV**QK**R**K**	KDWVQ**KRK**	KDWV**Q**K**RK**
	(0,2)	(0,4)	(0,4)	(0,5)	(0,4)
offset #1	C**W**C**QK**RK	CWC**QKR**K	CWC**QK**R**K**	CWCQ**KRK**	CWC**Q**K**RK**
	(1,1)	(1,3)	(1,3)	(1,4)	(1,3)
offset #2	**W**D**QK**RKNP	WD**QKR**KNP	WD**QK**R**K**NP	WDQ**KRK**NP	WD**Q**K**RK**NP
	(2,0)	(2,2)	(2,2)	(2,3)	(2,2)

Figure 1-12: Scanning phase of Teiresias. During the scanning phase, Teiresias enumerates all elementary motifs with exactly L characters and at most $W - L$ wild–cards. Using the input sequences above, Teiresias finds five such elementary motifs as shown in the table: F.AS, AST, AS.S, STS, and A.TS. The offset list for each of these is shown in the table. In the next phase of the algorithm, these elementary motifs are convolved together to form the final, maximal motifs.

```
>sequence 0
MSKNIVLLPGDHVGPEVVAEAVKVLEAVSSAIGVKFNFSKHLIGGASIDAYGVPLSDEALEAAKK
>sequence 1
MSKQILVLPGDGIGPEIMAEAVKVLELANDRFQLGFELAEDVIGGAAIDKHGVP
>sequence 2
MKFLILLFNILCLFPVLAADNHGVGPQGASGVDPITFDINSNQTGPAFLT
```

⇓

Teiresias:

$L/W/K = 5/8/2$

⇓

Final motifs:

motif	location (seq,pos)	
GPE..AEAVKVLE	(0,13)	(1,13)
IGGA.ID..GVP	(0,42)	(1,42)
MSK.I..LPGD..GPE	(0,0)	(1,0)
A.D.HGV	(1,46)	(2,17)

Figure 1-13: Pattern discovery with Teiresias. Here we have three protein sequences and we use Teiresias to find all patterns involving at least $L = 5$ non–wildcard characters that occur at least $K = 2$ times and have a fraction of non–wildcard characters of at least $L/W = 5/8$. These are called $5/8/2$ patterns and there are three such patterns in this set of sequences. Along with each motif is an offset list $\mathscr{L}(p_i)$ that specifies each sequence–position combination where the motif occurs. For motif p_3 ="A.D.HGV" the associated offset list is $\mathscr{L}(p_3) = \{(1,46),(2,17)\}$, indicating that this motif occurs twice: once in sequence 1 at position 46 and once in sequence 2 at position 17.

regular expression, p_k are given by

$$\mathscr{L}(p_k) = \{(x,y) \in \mathscr{L}(p_i) |\ \exists (x,z) \in \mathscr{L}(p_j) \text{ such that } z - y = \mathscr{W}(p) - \mathscr{W}(\text{suffix}_L(p))\}. \tag{1.42}$$

If $|\mathscr{L}(p_k)| < K$ then the motif does not have sufficient support and is discarded. Conversely, if $|\mathscr{L}(p_k)| = |\mathscr{L}(p_i)|$ then p_i is not a maximal motif. Or if $|\mathscr{L}(p_k)| = |\mathscr{L}(p_j)|$ then p_j is not a maximal motif. But, if

$$|p_i \oplus p_j| < K \ \forall\, j, \tag{1.43}$$

then p_i is maximal.

Obviously, by convolving each elementary motif with every other elementary motif, i.e. by repeating $p_k = p_i \oplus p_j$ for all i and j, the maximal motifs can be discovered. Teiresias uses an intelligent method of sorting the elementary motifs that does not require doing the all–by–all comparison and yet still guarantees that all the maximal motifs are discovered. The set of these maximal motifs are then returned to the user as in Figure 1-13.

The broad applicability of Teiresias has been shown in numerous studies. In particular, the algorithm has been very successful in multiple sequence alignment [189], motif dictionary

building [208], and gene finding in microbial genomes. In the work here, we will expand upon these studies in our application of the Teiresias motif discovery engine to practical problems of interest to the biology community, cf. Chapter 2.

Gibbs sampler and other position weight matrix–based tools

As described in Section 1.4 on page 37, PWMs can be much more specific than regular expressions for modeling a set of motif instances. But, this motif model also presents some unique difficulties. Recall that, as described in Section 1.5 on page 45, most regular expression–based motif discovery tools are "pattern driven" in the sense that, at some level, they rely on enumerating possible regular expressions and then determining which of those has a significant support within a given set of sequences. A similar approach does not work for PWMs because the set of production probabilities (see Equation 1.28 on page 39), or equivalently the target frequencies in the f matrix (see Equation 1.29 on page 39), are sampled from a continuous distribution. Therefore, the set of possible PWMs cannot be enumerated *a priori* because they are effectively infinite.

Most motif discovery tools that use PWM models skirt this issue by taking a more focused approach. Instead of returning a large set of motifs, as is common for regular expression–based tools such as Teiresias, PWM–based tools usually return either one or a small set of motifs. Table 1.5 on the facing page shows a list of publications introducing motif discovery tools that use PWMs. Most of these tools use a procedure whereby they are initialized with a random PWM and progressively optimize the PWM to maximize its sensitivity and specificity for the input sequences. However, some of the algorithms, such as Mitra–PSSM, which was proposed by Eskin [77], work in a much different fashion, somewhat similar to some of the regular expression–based tools described in the previous section.

Here, I will describe the algorithm by Lawrence et al. [147], which is generally referred to as the Gibbs sampler. This algorithm is the basis for many of the other algorithms shown in Table 1.5. As such, it is somewhat indicative of the class has a whole. The Gibbs sampler is a Markov chain Monte Carlo, or MCMC method [155, 170]. The Monte Carlo aspect of the method refers to optimization routine by which the PWM is successively refined. This routine is a Markov chain in the sense that the new, refined PWM depends only on the previous, unrefined PWM.

The Gibbs sampler is shown schematically in Figure 1-14 on page 55. The input to the algorithm is a set of sequences $S = \{s_0, s_1, \ldots, s_n\}$ and an integer $\mathscr{W}(p)$, which is the width of the motif p that we are trying to "discover." (Obviously, p is assumed to be represented by a PWM.) The Gibbs sampler assumes that the motif occurs exactly once in each sequence in S; however, more recent alterations of this basic framework allow for multiple instances in a single sequence or for sequences to be missing an instance. Here, I described the most simple case based on the original manuscript by Lawrence et al..

As shown in Figure 1-14, the Gibbs sampler has five major steps.

1. Choose random starting locations for the motive in all but one of the given sequences.

2. Use these sites to compute a PWM.

3. Score the sequence that was left out in step 1 over its entire length.

Table 1.5: Motif discovery tools using position weight matrices or similar models. As discussed in the text, PWMs are more specific than regular expressions; however, in general, there are fewer algorithms utilizing this motif model. Most of the later tools shown in the table are geared towards finding binding sites for regulatory proteins upstream of sets of co–regulated genes. Of these publications, the seminal manuscript is that by Lawrence et al. [147].

Authors	Year	Citation
Stormo and Hartzell	1989	[238]
Lawrence et al.	1993	[147]
Liu	1994	[154]
Bailey and Elkan	1994	[19]
Leung et al.	1996	[149]
Goffeau	1998	[99]
Hertz and Stormo	1999	[115]
Workman and Stormo	2000	[273]
Hughes et al.	2000	[124]
GuhaThakurta and Stormo	2001	[101]
Bi and Rogan	2004	[35]
Raphael et al.	2004	[204]
Eskin	2004	[77]
Siddharthan et al.	2005	[229]
Liu et al.	2005	[156]
Leung and Chin	2005	[148]
Zhong et al.	2005	[282]
Tharakaraman et al.	2005	[243]
Down and Hubbard	2005	[70]
Macisaac et al.	2006	[159]

4. Choose a site within the sequence probabilistically, based on the scores of each possible site, i.e. choose sites that have higher scores with higher probability.

5. Recompute the PWM using the site selected in step 4 and leaving out a different, randomly selected sequence. Then, go to step 1 and repeat until the PWM no longer changes significantly.

Most of the other PWM–based motif discovery tools listed in Table 1.5 use an approach that is similar to the Gibbs sampler. In general, these tools excel at finding motifs in DNA sequences such as cis–regulatory binding sites. (See Tompa et al. [249] for an excellent review of this problem and a demonstration of the power of PWM–based tools.) Other motif discovery tools use different optimization procedures than the Gibbs sampler, which are slight variations on the MCMC method, such as simulated annealing [142] or expectation maximization. Most of these refinement procedures guarantee that the algorithm will converge to a maximum [92]; however, it is not guaranteed that a maximum is globally optimal. The optimization can become trapped in a local optimum, which is called "slow–mixing" of the Markov chain. New procedures that avoid this are an active area of study [155].

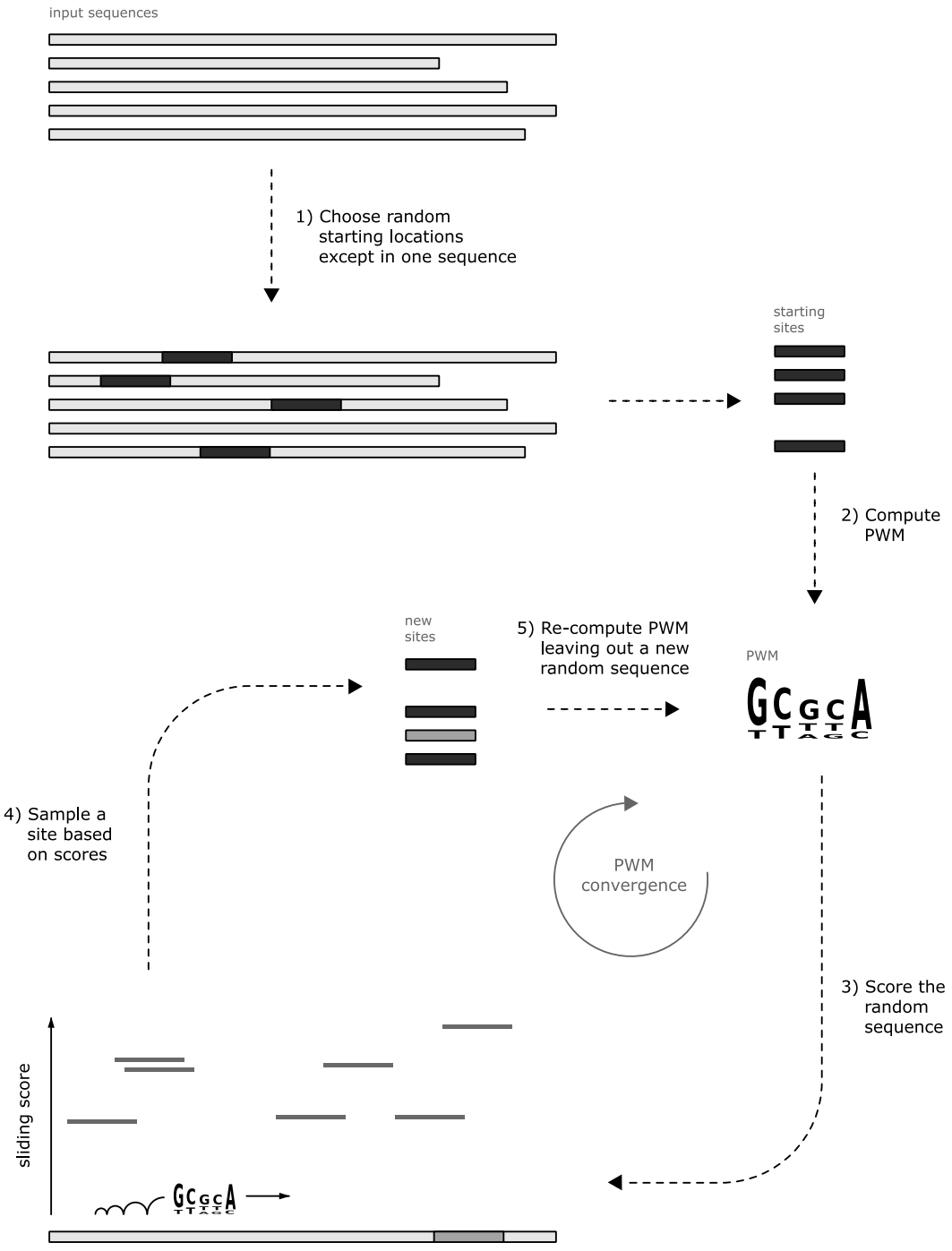


Figure 1-14: Schematic of the Gibbs sampling algorithm. As the figure shows, the Gibbs sampling method is an iterative algorithm that progressively refines a position weight matrix starting from a random PWM. If the input sequences contain a very strong motif, Gibbs sampling tends to converge very quickly upon it. However, in its original manifestation [147], the method was not able to find motifs that either occurred multiple times in a single sequence, or were found in some sequences but not others. In general, most motif discovery tools using PWMs bear a great deal of similarity to the original Gibbs sampling method.

Chapter 2

Design of antimicrobial peptides

2.1 Introduction

In the previous chapter, I introduced grammars as a generalized method for modeling motifs in sequences of characters. In addition, I presented a detailed look at the Teiresias motif discovery tool. In this, the second chapter of my thesis, I show how Teiresias can be used to derive sets of regular grammars describing a particular class of protein sequences — antimicrobial peptides. In what follows, I present a general background on antimicrobial peptides and then provide a rationale for why these peptides are particularly well–suited for being modeled using regular grammars. I detail the construction of an annotation tool for finding new antimicrobial peptides and validating the general hypothesis that regular grammars can be used as a sensitive and specific indicator of antimicrobial function in peptide sequences. Next, I describe the preliminary design of synthetic antimicrobial peptides using an evolutionary approach, which, although ultimately inconclusive, provided motivation for a more focused design. The final section of this chapter describes a more focused design approach, detailing the successful construction of numerous novel peptides with strong antimicrobial activity against a wide spectrum of bacteria.

The research described in this chapter is drawn largely from a publication that is in preparation in collaboration with Christopher Loose, Isidore Rigoutsos, and Gregory Stephanopoulos. (Some experimental work in Section 2.4 on page 71 was also performed by Gyoo Yeol Jung.) Throughout this chapter, the use of the pronoun "we" refers to this group of authors.

2.2 Motivation

Antimicrobial peptides are small proteins that attack and kill microbes. These peptides are effectors of the innate immune system: the phylogenetically ancient first line of defense against pathogen assault [141, 213]. Antimicrobial peptides are ubiquitous amongst multicellular eukaryotes and found in diverse contexts including frog skin [230], scorpion venom [172], and human sweat [219].

There is a growing interest in antimicrobial peptides, due largely to the proliferation of multi–drug resistant pathogen strains [50]. These strains are resistant to one or more common antibiotics such as penicillin, tetracylin, or vanocomycin. In the United States alone, the cost of

treating and preventing infections by these pathogens is estimated to be many billions of dollars annually [184]. In the arms race against microbes, mankind is losing — only a single new class of antibiotics was developed in the past 30 years [182, 260]. However, there is mounting evidence that antimicrobial peptides are less likely to induce bacterial resistance and will make a strong contribution to our therapeutic arsenal [91, 279, 280].

Human antimicrobial peptides, such as the defensins and cathelicidins, help to maintain a passive defense against pathogens in the environment. A malfunction of these peptides leads to severely immunocompromised phenotypes. For example, a deficiency of the LL-37 cathelicidin leads to morbus Kostmann, a congenital neutropenia characterized by recurrent bacterial infection and short life–expectancy [40, 201]. In addition, the pathogenesis of cystic fibrosis (CF) is indirectly caused by antimicrobial peptide impairment [89]. CF patients have a defective Cl^- ion channel in the pulmonary airway epithelia that causes unusually high salt concentrations. The salt disrupts the function of the epithelial defensins, leading to chronic infections and ensuing respiratory failure [234, 280]. More severe phenotypes have been produced in loss–of–function animal models. For example, Wilson *et. al.* [268] showed that mice with depressed defensin activity required a 10–fold lower dose of the *S. typhimurium* pathogen to produce a fatality. In contrast, gain–of–function mice expressing human enteric defensin HBD–5 have a markedly increased resistance to *S. typhimurium* assault [218].

In addition to their more publicized antibiotic capabilities, antimicrobial peptides appear to be important in a variety of other diseases. For example, the antimicrobial peptides of *Anopheles gambiae*, the malaria mosquito, are upregulated after malaria (*Plasmodium berghei*) infection [58] and, in some cases, are capable of killing the ookinetes of the parasite [24, 257]. Antimicrobial peptides are also indicated in a resistance to the AIDS–causing virus, HIV. Long–term HIV nonprogressors display elevated levels of α–defensins that inhibit the proliferation of the virus [281]. Finally, a limited class of antimicrobial peptides may form the basis for novel cancer treatments [74, 140]. For example, the antimicrobial peptide tachyplesin can repress the growth of cancerous tumors both *in vitro* and *in vivo* [52].

The many disease–relevant behaviors of antimicrobial peptides are a result of their ability to broadly distinguish eukaryotic cells from pathogenic invaders. There are two features that give the peptides this ability: a net positive charge and an amphipathic 3–D structure [75, 94]. These features endow the peptide with an affinity for negatively charged outer leaflet of the bacterial cytoplasmic membrane (see Figure 2-1 on the next page). This affinity leads to permeablization of the bacterial membrane, which is the basis for the bactericidal activity of antimicrobial peptides. Although this mode of action is common to almost all antimicrobial peptides, there are many diverse primary sequences that can produce this behavior. These sequences form a handful of conserved families, the most common of which are the α–helical and β–sheet antimicrobial peptides [251].

Figure 2-2 on page 61 shows the structure of aurein–1.2, an archetypal alpha helical antimicrobial peptide from the Australian Southern bell frog [261]. Alpha helical AmPs form the largest single family of AmPs. They are particularly common in amphibians because species such as frogs tend to inhabit wet and warm ecological niches that are conducive to the proliferation of bacteria. (See Figure 2-3 for a phylogenetic tree of the more than 400 amphibian AmPs.) Alpha helical AmPs tend, in general, to have a amphipathic structure in which positively charged residues are segregated on to a particular side of the longitudinal axis of the helix. Negatively charged or neutral residues tend to be isolated on the opposite side from the posi-

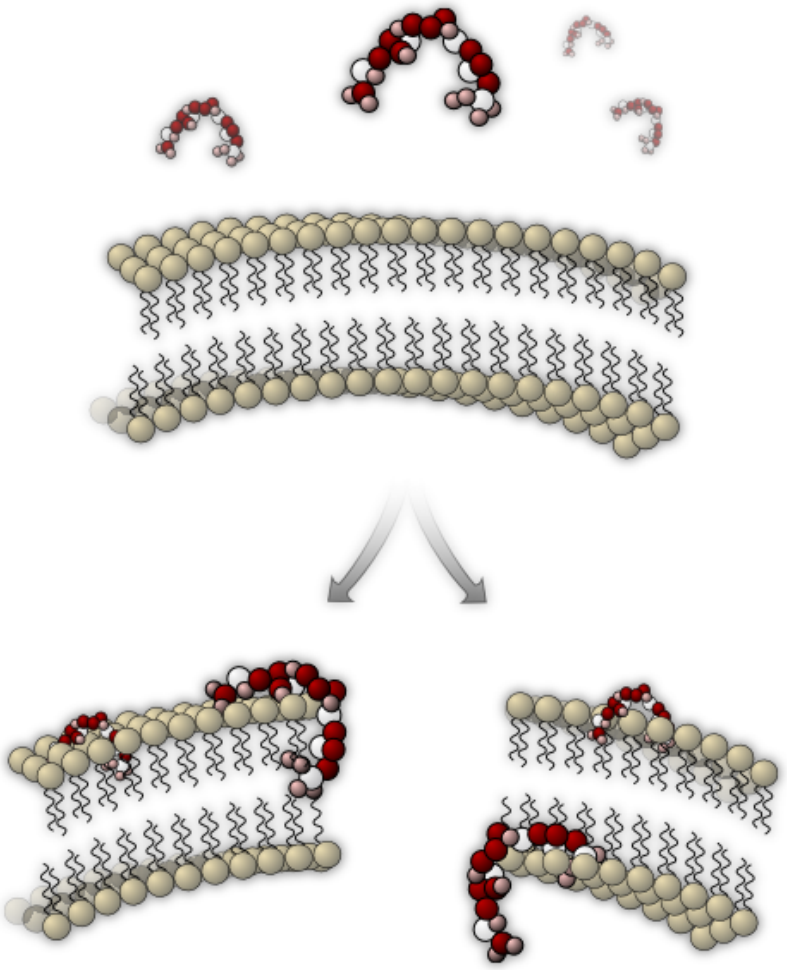


Figure 2-1: Antimicrobial peptide action. In the figure above, amphipathic antimicrobial peptides with net positive charges are attracted via electrostatic forces to the negatively charged outer–leaflet of the microbial membrane (step 1) [280]. This membrane is either the lipopolysaccharide layer or the peptidoglycan layer of gram–negative and gram–positive bacteria, respectively [75]. In addition, the β–1,3–glucan in fungal membranes and the phosphoglycan of certain parasites can give membrane characteristics that are exploited by certain classes of antimicrobial peptides. The peptides cover and lyse the membrane via either a "barrel stave" or "carpet" mechanism (step 2) [226]. Although some antimicrobial peptides are hemolytic, in general, they are not damaging to multicellular organisms because 1) the negatively charged phosphatydilserines of their outer leaflet are sequestered on the cytoplasmic side of the membrane, and 2) the membranes are stabilized by cholesterols [280].

tively charged residues. Evidence suggests that this confirmation allows the peptide to position itself judiciously relative to the bacterial membrane, facilitating entry of the peptide into the membrane and, ultimately, membrane disruption [280].

The characteristic membrane–attack of antimicrobial peptides is the primary rationalization of the peptides' propensity to not induce bacterial resistance to the same degree as small molecule pharmaceuticals. That is, because the peptides leverage a pervasive polygenic trait of bacteria, the structure of the cell wall, it is "expensive" for the bacteria to evolve a resistance [279, 280]. For this reason, many companies are developing therapeutics based on antimicrobial peptides, many of which are in phase III FDA trials [102]. Even more encouraging, some peptides show strong *in vitro* bactericidal activity against pathogen strains that have developed a resistance to multiple conventional antibiotics [91, 239, 246].

2.3 A grammatical approach to annotating AmPs

Our preliminary studies of natural AmPs indicated that their amphipathic structure gives rise to a modularity among the different AmP amino acid sequences. The repeated usage of sequence modules — which may be a relic of evolutionary divergence and radiation — is reminiscent of phrases in a natural language, such as English. For example, the grammar `Q.EAG.L.K..K` (the "`.`" is a "wildcard", which indicates that any amino acid will suffice at that position in the grammar) is present in over 90% of cecropins, an AmP common in insects. Based on this observation we modeled the AmP sequences as a formal language — a set of sentences using characters from a fixed alphabet, in this case the alphabet of amino acid one–letter symbols [134].

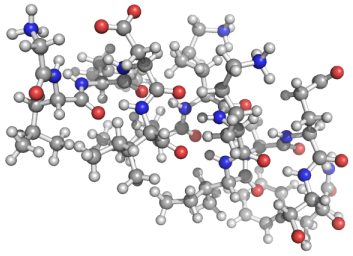
We conjectured that the "language of AmPs" could be described by a set of regular grammars and that these grammars, in turn, could be used to annotate and design novel AmPs. As discussed in Chapter 1, regular grammars are, in essence, simple rules for describing the allowed arrangements of characters. These grammars, such as the cecropin grammar mentioned previously, are commonly written as regular expressions and are widely used to describe patterns in nucleotide and amino acid sequences [118, 225].

To find a set of grammars describing AmPs we used the Teiresias pattern discovery tool [207] (see Section 1.5 on page 45 to discover an exhaustive, maximal set of regular grammars in a collection of antimicrobial peptides assembled from a variety of sources.

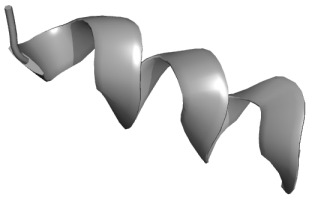
2.3.1 Collecting a database of antimicrobial peptides

Our collection of known antimicrobial peptides was taken principally from two databases: the Antimicrobial Sequences Database (AMSDb) [250] and SwissProt/TrEMBL [22]. The AMSDb contains about 750 antimicrobial peptides, all of which are a subset of SwissProt/TrEMBL. Some of the entries in the AMSDb are sequence fragments that are derived from larger precursors via post–translational modification. We discarded these peptides unless the reported antimicrobial fragment comprised at least 80% of the length of its parent sequence. From the remaining entries, we selected all that were from eukaryotic organisms, including the complete length of the parent peptides in our database.

A) Ball and stick



B) Ribbon



C) Peptide wheel view

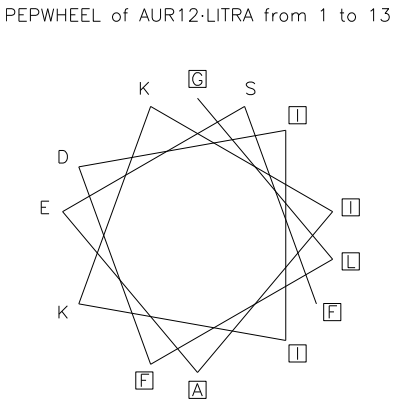


Figure 2-2: The structure of aurein–1.2 [261]. Aureins constitute a large family of secreted proteins originally isolated from the skin of frogs. This particular structure was isolated from the Australian Southern bell frog. The peptide conforms to the classic amphipathic alpha helical structure and has wide–spectrum antimicrobial activity. Part A) shows a ball–and–stick representation of the structure in which nitrogen atoms are colored blue and oxygen atoms are colored red. Part B) shows the same structure using a cartoon representation that clearly shows the alpha helix. Finally, part C) shows a helical wheel projection in which uncharged residues are boxed in order to highlight the segregation of charges on the helix. Graphics created using PyMol (DeLano Scientific, San Carlos, CA, USA).

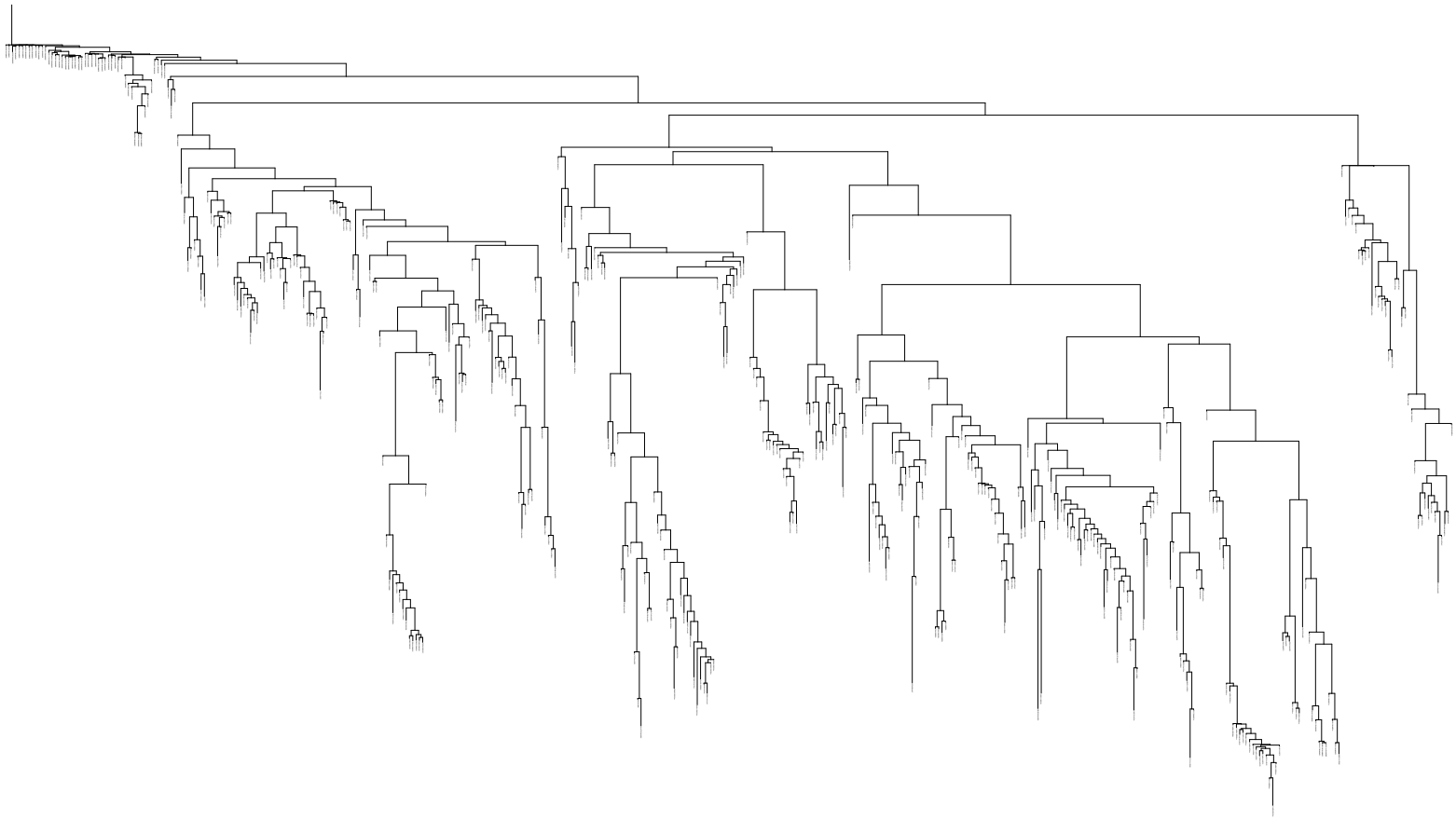


Figure 2-3: A phylogenetic tree of amphibian AmPs. Because they tend to inhabit environments that are conducive to the growth of bacteria, amphibians are under evolutionary pressure to develop numerous and varied AmPs. This tree shows the degree of sequence similarity for over 400 AmPs isolated from amphibian sources. As the figure shows, amphibians have many families of AmPs, some of which are only distantly related, including the aureins, bombinins, bombesins, and brevinins.

Table 2.1: Common antimicrobial peptide families

acaloleptin	achacin	adenoregulin	alpha–defensin
androctonin	andropin	apidaecin	attacin
aurein	azurocidin	bactenecin	bactericidin
bactinecin	beta–defensin	bombinin	bombolitin
buforin	buthinin	caerin	caltrin
cathelin	cecropin	ceratotoxin	citropin
clavanin	coleoptericin	corticostatin	crabrolin
defensin	demidefensin	dermaseptin	dermcidin
diptericin	drosocin	drosomycin	enbocin
formaecin	gaegurin	gallinacin	gloverin
granulysin	hadrurin	heliomicin	hemiptericin
hemolin	hepcidin	histatin	holotricin
hymenoptaecin	hyphancin	indolicidin	lebocin
macin	maculatin	maximin	metalnikowin
metchnikowin	misgurin	moricin	myticin
mytilin	mytimycin	nk–lysin	penaeidin
permatin	phormicin	phylloxin	pleurocidin
polyphemusin	ponericin	protegrin	pseudin
pyrrhocoricin	ranalexin	ranatuerin	rhinocerosin
royalisin	rugosin	salmocidin	sapecin
sarcotoxin	sillucin	spingerin	styelin
tachycitin	tachyplesin	temporin	tenecin
termicin	thanatin	tricholongin	zeamatin

SwissProt/TrEMBL is a database of about 120 thousand heavily annotated sequences. Included in the annotations are keywords grouping proteins into functional categories. For our initial database of antimicrobial peptides we extracted all the eukaryotic sequences matching the keywords "antibiotic", "fungicidal", or "defensin". These sequences were added to the peptides we collected from the AMSDb.

Using the sequences that we extracted from AMSDb and SwissProt/TrEMBL, we made a list of common antimicrobial peptide names — such as "defensin" or "tenesin" — and collected sequences from SwissProt/TrEMBL matching these names. From the name–matched sequences, we manually selected those eukaryotic sequences that had literature evidence of antimicrobial activity but were not explicitly labeled as such in SwissProt/TrEMBL. These sequences, together with the sequences from AMSDb and the first set from SwissProt/TrEMBL, formed our initial database of antimicrobial peptides. In the following section, we describe how these sequences were used, via a homology–based bootstrapping method, to find even more antimicrobial peptides within SwissProt/TrEMBL.

2.3.2 Finding more antimicrobial peptides

A minority of the antimicrobial peptides within SwissProt/TrEMBL, were not found using either the keyword or "common–name" searches. To find these sequences we used two approaches in parallel. First, we used a FastA sequence alignment based approach. Second, we used a grammar matching–based approach with TEIRESIAS. Both of these approaches are detailed below and summarized in Figure 2-4.

Seeding the peptide database through similarity searching

Starting with our initial database of sequences (S_o in Figure 2-4) from SwissProt/TrEMBL and AMSDb, we aligned each sequence in S_o against the entire SwissProt/TrEMBL database. If a sequence in SwissProt/TrEMBL aligned with a sequence from S_o with 80% or greater pair–wise identity over the length of both sequences, the new sequence was marked as a possible antimicrobial peptide. Of the marked sequences, we selected those that were from eukaryotic organisms and had literature evidence of antimicrobial activity. These sequences are shown as S_o^F in Figure 2-4, meaning sequences found using FastA in the first iteration.

Seeding the peptide database through use of grammar discovery

In the second stage of our bootstrapping method, we used TEIRESIAS to find grammars that could be used to search for antimicrobial sequences in SwissProt/TrEMBL. As shown in Figure 2-4, from the S_o sequence set, we derived two separate grammar sets ($C_i^{8/2/2}$ and $C_o\ |_{m-8/2/2}^{6/15/2}$), which we combined together. This combined set, ϕ, was processed (a detailed description of this processing is in the appendix) to increase the selectivity and sensitivity of the grammars for antimicrobial peptide sequences. Finally, in each sequence from SwissProt/TrEMBL, we searched for instances of grammars from ϕ'. If 80% of the amino acids in a peptide from SwissProt/TrEMBL were contained within instances of grammars from ϕ' the peptide was marked. Of the marked sequences, we selected those sequences that were from eukaryotic organisms and had literature evidence of antimicrobial activity, calling these sequences S_o^T.

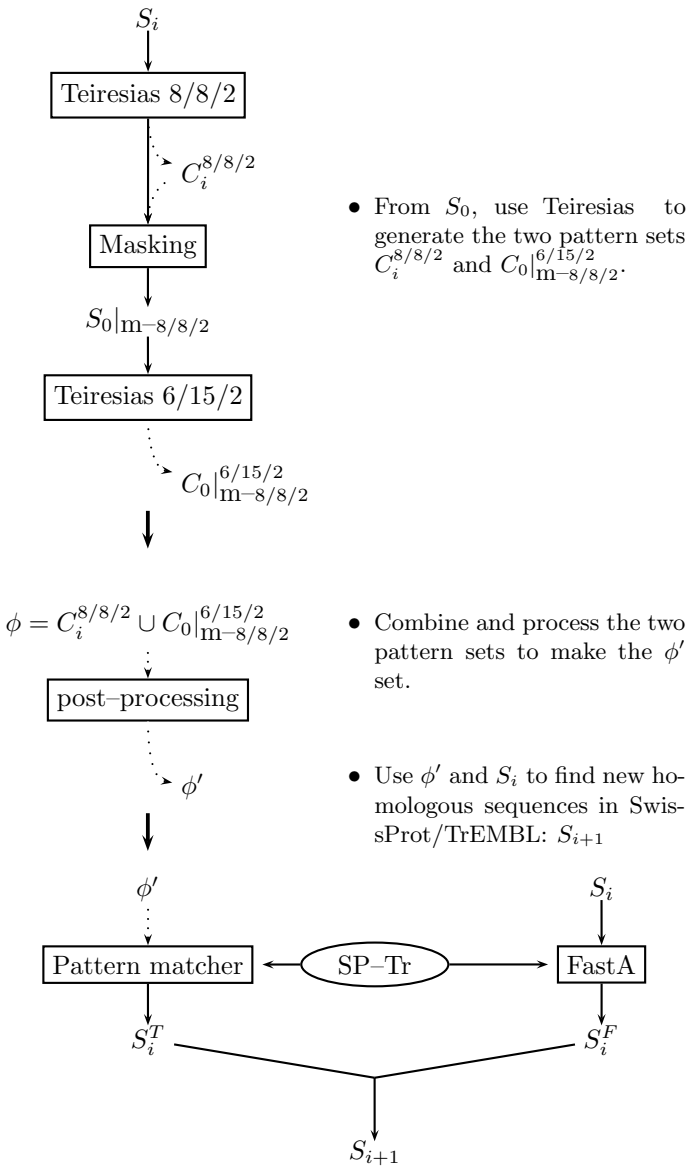


Figure 2-4: A schematic of the bootstrapping method used to collect antimicrobial sequences from SwissProt/TrEMBL. On the left, using TEIRESIAS, we computed an 8/8/2 grammar set ($C_i^{8/2/2}$) from the initial set of sequences, S_o. These grammars were masked from S_o to make $S_o\ |_{m-8/8/2}$, from which the 6/15/2 grammar set, $C_o\ |^{6/15/2}_{m-8/2/2}$, was found using TEIRESIAS again. The two grammar sets were combined and processed (see Appendix) to make ϕ'. This final grammar set was used to find more antimicrobial sequences in SwissProt/TrEMBL. On the right side of the schematic, sequences from S_o were aligned against SwissProt/TrEMBL to find new antimicrobial sequences.

Iterating the bootstrapping method

The new antimicrobial peptides found using FastA and TEIRESIAS, S_o^F and S_o^T respectively, were added to the initial database, S_o, to make S_1. Next, a bootstrapping method was repeated on the S_1 sequence set to make larger and larger sets (S_2, S_3,...) until no more antimicrobial peptides in SwissProt/TrEMBL could be found. This process is shown in Figure 2-4 and detailed below.

1. **Finding Highly Conserved grammars**
   First we found all the highly conserved (8/8/2) grammars in S_o. These grammars are substrings in S_o that are repeated exactly, that is, grammars without any wild–cards or bracketed expressions. Let these grammars be called $C_o^{8/2/2}$, meaning 8/8/2 grammars from the first iteration. In order to simplify the grammar discovery process for the next step, the sequence set S_o was masked [1] with the $C_o\,|^{8/2/2}$ grammars to make the $S_o\,|_{m-8/8/2}$ sequence set.

2. **Finding Loosely Conserved grammars**
   Using the $S_o\,|_{m-8/8/2}$ sequences, we found all 8/15/2 grammars, which we will call $C_o\,|^{6/15/2}_{m-8/2/2}$. These grammars are more loosely conserved than the $C_o^{8/2/2}$ grammars and are typically greater in number.

3. **Post–Processing the grammars**
   Let the union of the two grammar sets computed above be $\phi_o = C_o^{8/2/2} \cup C_o\,|^{6/15/2}_{m-8/2/2}$. We would like to match grammars in ϕ_o against SwissProt/TrEMBL to find any remaining unknown antimicrobial sequences. But, to gain greater specificity and sensitivity, we first processed the grammars in ϕ_o to a make a grammar set $\phi'o$.

   (a) For every grammar in ϕ_o, we de–referenced each wild–card character that could be expressed as a bracketed expression with no greater than four characters. That is, in the grammar "K.T", the "." might be replaced with "[AG]" if, for each instance of "K.T", only "A" and "G" are found in the wild–card position. If more than four characters were needed in the bracketed expression, we left the wild–card character instead.

   (b) For each of the altered grammars in ϕ_o we decomposed the grammar into a set of smaller, redundant grammars by using a sliding window of ten non–wild–card characters. So, a grammar such as "[FWY]FK.[GQ][KRQ]CPDAY" would be decomposed into three distinct grammars: "S[RKM][FWY]FK.[GQ][KRQ]CPD", "[RKM][FWY]FK.[GQ][KRQ]CPDA", and "S[RKM][FWY]FK.[GQ][KRQ]CPDAY", each ten non–wild–card amino acids in length.

   (c) From this new, redundant ϕ_o we kept only those grammars that were statistically significant. These are grammars that have a log–odds probability less than or equal to -30.

[1] "Masking" is described in detail in Rigoutsos and others [208]. In brief, by masking a grammar, we tag each instance of a grammar except for the instance in the longest sequence in which the grammar is found. Tagged regions are then excluded from further grammar discovery processes.

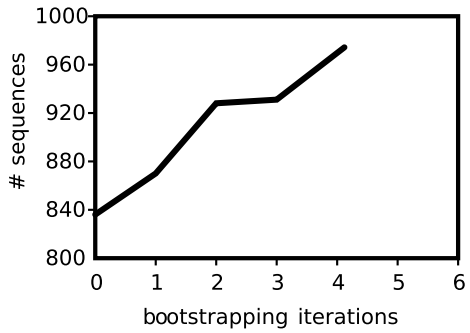


Figure 2-5: A plot of the progress of the bootstrapping method. The figure shows that our antimicrobial sequence database grew from 836 sequences to 931 sequences in 3 iterations.

Let these the processed ϕ_o grammar set be called ϕ'_o.

The final sequence set, from the last iteration, became our database of known antimicrobial peptide sequences and the final ϕ' became our antimicrobial grammar database.

2.3.3 Antimicrobial sequence and grammar databases

The initial database of antimicrobial peptides collected from AMSDb and SwissProt/TrEMBL contained a total of 836 sequences. Starting with these sequences, the bootstrapping method described previously went through 3 iterations until no more sequences were found in SwissProt/TrEMBL. The last sequence set, S_3, which contained a total of 931 sequences, was used as our antimicrobial sequence database and is available on–line at `http://cbcsrv.watson.ibm.com/Tspd.html`. The final grammar set (ϕ' from the last bootstrapping iteration) contained a total of 241,642 grammars covering the sequence space of the final sequence database.

2.3.4 Annotator design and validation

Together, these ~200K grammars describe the "language" of the AmP sequences. In this linguistic metaphor, the peptide sequences are analogous to sentences and the individual amino acids are analogous to the words in a sentence. Each grammar describes a common arrangement of amino acids, similar to popular phrases in English.

Given an arbitrary sequence of amino acids, it is possible that some parts of the sequence are "matched" by one or more of the grammars in our database. For example, the white mustard plant AmP Afp1 (Genbank accession no. P30231) contains the amino acid sequence fragment `CICYFPC`, which matches the grammar `CICY[FVK]PC` from our database. (As discussed in Section 1.4 on page 33, the bracketed expression `[FVK]` indicates that, at the fifth position in the grammar, either phenylalanine, valine, or lysine is equally acceptable.) Based on this match, we would say that the Afp1 fragment is "grammatical."

Using the antimicrobial grammar database, we created an on–line tool for annotating antimicrobial peptides by determining the degree to which a query sequence is grammatical. (This tool is available online at `http://cbcsrv.watson.ibm.com/Tspd.html`.) A user–supplied input sequence is annotated by generating grammar–based alignments of the input against sequences in our database of known antimicrobial sequences (S). This alignment takes

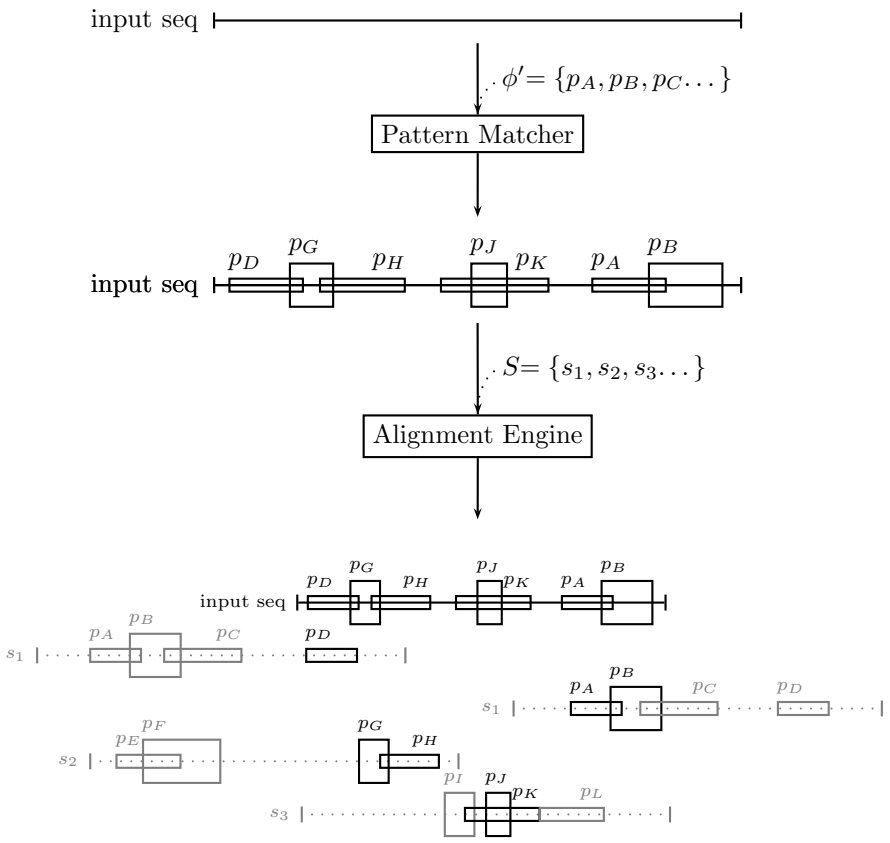


Figure 2-6: A schematic of our grammar–based alignment method. In the figure, a user–supplied input sequence is searched for instances of grammars ($p_A, p_B, p_C \ldots$) that occur in the set of known antimicrobial sequences (S). Grammars that occur in both the input sequence and sequences in S are then used to create alignments. As indicated in the figure, the input sequence shows homology to s_1 in two distinct regions, so both possible alignments are shown. See Figure 2-7 on the facing page for an example of how these grammar–based alignments appear in practice.

place in two distinct steps. First, we search the input sequence for instances of grammars from the antimicrobial grammar database (the final ϕ'). Second, for each contiguous stretch of shared grammars between the input sequence and a sequence from S, an alignment is produced. Figure 2-6 show a schematic of the alignment process.

Since it is possible that, for an arbitrary sequence, only a portion of the sequence is matched by one of our grammars, we developed a heuristic metric Z, which is the degree to which a query sequence is grammatical. To calculate Z, we assign a local score along the backbone of a query sequence that is equal to the number of grammars, or fractions of grammars with at least 10 amino acids, that have matches over the length of the query sequence. The total score for the sequence, Z, is the fraction of the sequence's length that is covered by at least one grammar (see Figure 12-9). For example, a hypothetical sequence `LFLTAIDRYIAAA` — which is matched by `LFLTAI[ID][TR][VY]I`, but no other grammars in our database — would get a score of 10/13 since the first 10 positions in the sequence are covered by the match.

In order to annotate and design synthetic AmPs, we created a software tool to calculate the

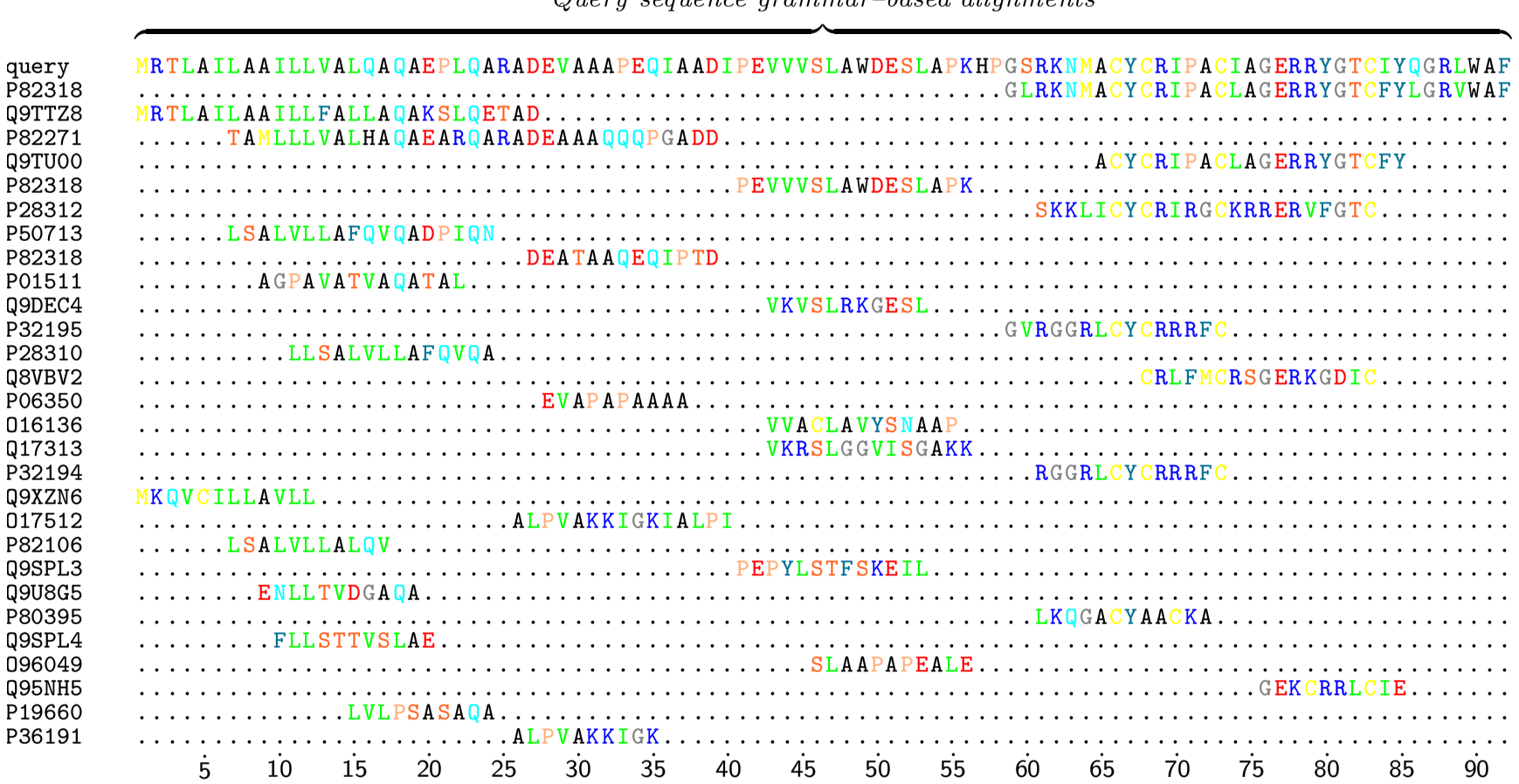


Figure 2-7: Example results of the grammar–based alignment method. The figure shows the annotation of a query sequence after it has been searched exhaustively for grammars in our database of ~200K grammars describing the "language" of AmPs. This alignment was generated using the methodology detailed in Figure 2-6 on the preceding page. Each of the sequences below the query is an AmP sequence from our database, S. The sequences are aligned against the query because they share grammars in common at those particular loci. However, different sequences may share very in degrees of conservation, even if they have the same grammar. For example, see Figure 2.2 on the following page.

Table 2.2: Motif conservation for the query shown in Figure and the motif L[VQH][ALV][KLPQ][AS][EAF][APQS][ALRV]QA.

QUERY	LQAQAEPLQA			
P17534	LVLLAFQVQA	40.00%	Cryptdin-related protein 4C-1	Mus musculus (Mouse).
P19660	LVLPSASAQA	30.00%	Bactenecin 5 precursor (BAC5)	Bos taurus (Bovine).
P19661	LVLPSASAQA	30.00%	Bactenecin 7 precursor (BAC7)	Bos taurus (Bovine).
P28309	LVLLSFQVQA	30.00%	Cryptdin-2 precursor	Mus musculus (Mouse).
P28310	LVLLAFQVQA	40.00%	Cryptdin-3 precursor	Mus musculus (Mouse).
P28311	LVLLAFQVQA	40.00%	Cryptdin-4 precursor	Mus musculus (Mouse).
P28312	LVLLAFQVQA	40.00%	Cryptdin-5 precursor	Mus musculus (Mouse).
P32195	LVVPSASAQA	30.00%	Protegrin 2 precursor (PG-2)	Sus scrofa (Pig).
P33046	LVVPSASAQA	30.00%	Indolicidin precursor	Bos taurus (Bovine).
P49930	LVVPSASAQA	30.00%	Antibacterial peptide PMAP-23	Sus scrofa (Pig).
P49931	LVVPSASAQA	30.00%	Antibacterial peptide PMAP-36	Sus scrofa (Pig).
P49932	LVVPSASAQA	30.00%	Antibacterial peptide PMAP-37	Sus scrofa (Pig).
P50704	LVLLAFQVQA	40.00%	Cryptdin-6/12 precursor	Mus musculus (Mouse).
P50705	LVLLAFQVQA	40.00%	Cryptdin-7 precursor	Mus musculus (Mouse).
P50707	LVLLAFQVQA	40.00%	Cryptdin-9 precursor	Mus musculus (Mouse).
P50708	LVLLAFQVQA	40.00%	Cryptdin-10 precursor (Fragmen	Mus musculus (Mouse).
P50711	LVLLAFQVQA	40.00%	Cryptdin-13 precursor	Mus musculus (Mouse).
P50712	LVLLAFQVQA	40.00%	Cryptdin-14 precursor (Fragmen	Mus musculus (Mouse).
P50713	LVLLAFQVQA	40.00%	Cryptdin-15 precursor	Mus musculus (Mouse).
P50714	LVLLAFQVQA	40.00%	Cryptdin-16 precursor	Mus musculus (Mouse).
P51525	LVVPSASAQA	30.00%	Prophenin-2 precursor (PF-2) (	Sus scrofa (Pig).
P82270	LHAQAEARQA	70.00%	Theta defensin-1, subunit A pr	Macaca mulatta (Rhesus macaque).
P82271	LHAQAEARQA	70.00%	Theta defensin-1, subunit B pr	Macaca mulatta (Rhesus macaque).
P82318	LQAQAEPLQA	100.00%	Neutrophil defensins 1, 3 and	Macaca mulatta (Rhesus macaque).
Q01524	LQAKAEPLQA	90.00%	Defensin 6 precursor (Defensin	Homo sapiens (Human).

score Z for a query sequence and to classify the sequence as either likely to have antimicrobial activity — if its Z–score is above a certain threshold — or not. To determine this threshold, we trained the tool on a subset of sequences from our AmP database as follows. We randomly selected 90% of the natural AmP sequences and generated a Teiresias grammar set, using the same Teiresias parameters that were used to generate our ~200K grammar set. This smaller grammar set was used by our software to classify the remaining 10% of our AmP database, which was hidden among 10% of the non–AmP sequences from Swiss–Prot/TrEMBL (~78K sequences). This experiment was repeated 300 times, with different random sets, to determine the best Z–score. We found that, at an Z–score threshold of 0.73, the software tool will correctly classify both the AmP and non–AmP sequences with 99.95% accuracy.

2.4 Preliminary strategy for the design of novel AmPs

2.4.1 Sequence design

As I showed in the previous section, the Z–score annotation metric is both sensitive and selective for existing AmPs. We hypothesized that this metric could be used equally well to design unnatural sequences that would have antimicrobial activity. In this section, I describe our preliminary strategy for designing these novel, unnatural sequences. *Notably, the experimental data presented in this section were later discovered to not be reproducible due to experimental complications. See Section 2.4.3 on page 76. The data are presented here for the insight they lend to our more focused, and successful, strategy for designing AmPs, which is described in Section 2.5 on page 80.*

Based on the annotation results described in the previous section, we ran a computer simulation to create novel amino acid sequences with high Z–scores, but with minimal homology to natural AmPs. This simulation, shown schematically in Figure 2-8 on the next page, used the Z–score as a fitness function for the *in silico* directed evolution of these novel sequences. To begin, we created a randomized database of 100K progenitor sequences of uniform length with the same amino acid composition (i.e., the same percentage of each amino acid type) as our AmP database. Each of these sequences was allowed to have 4 mutated "children," which were each 100 PAM (point accepted mutations) evolutionary units away from the parent. (The implied rates of mutation from the Blosum–50 matrix were used to make the mutations at the amino acid level [109].) These children, each of which differed from their parent sequence by at least one amino acid, were added to the total population of sequences. In order to avoid generating sequences that were similar to natural AmPs, the population was purged of any sequences that had 6 or more consecutive amino acids in common with any natural AmP sequence. Finally, the remaining sequences were scored using our annotation software. From the population, the sequences with the top 100 Z–scores were propagated to the following round, and the entire process was repeated.

Using the strategy described above, we allowed many populations of sequences to evolve, each with a different sequence length, which remained constant during the simulation. We stopped each simulation after 3,000 rounds of mutation and selection, by which point we found that populations of small sequence length would have converged to $S = 1$. For longer length populations, all the sequences typically reached at least $S = 0.73$ and all tended to be closely related to each other. We chose three sequences of lengths 20, 31, and 63 amino acids

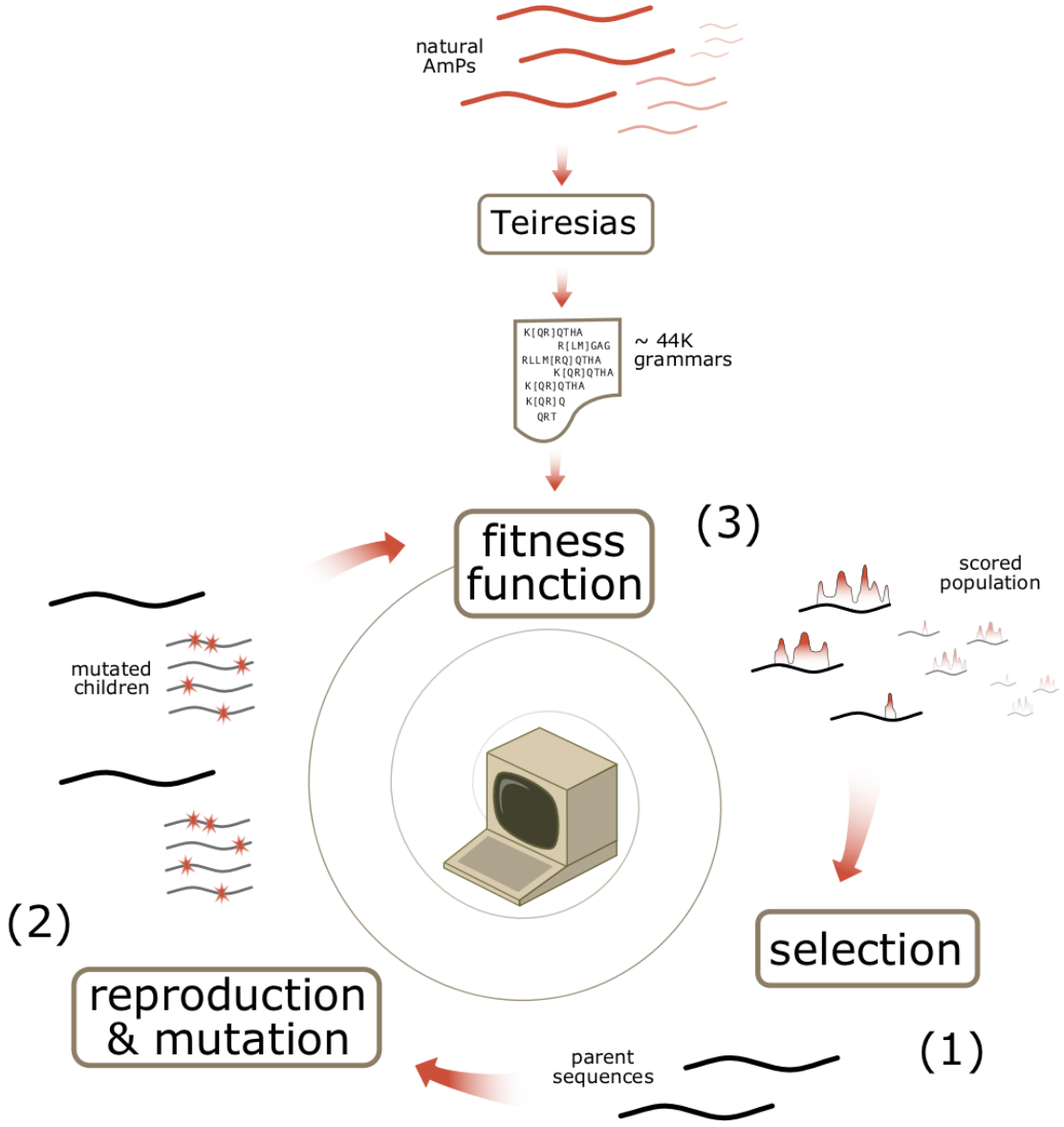


Figure 2-8: A schematic of the *in silico* directed evolution strategy. Position (1) shows the starting point: the database of 100K parental sequences. Each of these sequences has 4 mutated children (2) and the entire population is scored using the Z–score and our database of grammars from natural AmPs. From the scored population (3), the top 100 sequences are chosen and become the parental sequences for the next iteration. In addition to the directed evolution simulation, we considered other methods for generating sequences with high Z–scores. However, we chose this approach because it naturally allows for sequences of arbitrary length and the possibility that grammars may overlap in the designed amino acid sequences.

to test experimentally for antimicrobial activity: sequences synth–1, 2, and 3 in Table 2.3 on the following page.

Using NCBI Blast [11] (blastp) with the default parameters, we compared these sequences to the entire NCBI NR sequence database. The Blast results showed that none of the three sequences has significant homology to *any* known protein (E–value $\leqslant$10), including the naturally–occurring AmPs. (More extensive similarity searching using PSI–Blast and E–value thresholds up to $\leqslant$50 also failed to detect similarity to any natural AmPs.) This is possible because each grammar can be written in a large number of ways. For example, the 10–residue grammar `[LV][GA]K[TN][FL]AGHML` occurs in 3 natural AmPs, but there are 16 possible 10–residue sequences that match this grammar. Since our sequences are built from tiled grammars, the synthetic sequences can quickly deviate from the naturally populated sequence space such that it is impossible to detect similarity using sequence alignment tools (see Figure 2-9 on page 75).

For each of the three synthetic peptides, we also designed a set of shuffled sequences, which we hypothesized would have no antimicrobial activity. These "negative" peptides are shown along with the three synthetic peptides in Table 2.3. The negative peptides have the same amino acids as the synthetic sequences (and thus, the same molecular weight, charge, and pI); however, the order of the amino acids was shuffled so that the negative sequences each have an Z–score of zero.

2.4.2 Peptide synthesis and validation

Using an approach described elsewhere [168], we synthesized all 8 of the peptides shown in Table 2.3. For each peptide we created a translation template consisting of three parts: green fluorescent protein (GFP) with a T7 promoter, an enterokinase recognition site (ERS), and the AmP to be tested. We synthesized the protein–product of each template in an *E. coli*–derived *in vitro* translation system with continuous exchange [139]. The resulting peptides were proteolytically cleaved with enterokinase and the yield of AmP in the translation mixture was measured via GFP fluorescence using a 1:1 molar equivalence between the AmP and GFP concentrations.

We characterized the antimicrobial activity of each synthetic AmP using a broth microdilution assay described previously [12]. The top section of Table 2.4 shows the activities of the synthetic peptides against four bacterial species: *Bacillus cereus*, *Corynebacterium glutamicum*, *E. coli*, and *Citrobacter rodentium*. (See also Figure 2-10 on page 77.) These data suggested that all three synthetic peptides had antimicrobial activity. Furthermore, none of the negative, "un–grammatical" sequences had any activity. Thus, it appeared that the activity of the designed peptides was not an artifact of molecular weight, charge, or pI. Instead, the activity appeared correlated to the Z–score, suggesting that higher order sequence features are responsible for antimicrobial activity. (*These data were later shown to be not reproducible. Later experiments showed that the sequences in Table 2.4 did not have detectable levels of antimicrobial activity under a more stringent assay. See Section 2.4.3 on page 76.*)

The bottom of Table 2.4 shows the measured activities of synth–1 variants that were synthesized chemically — the peptides were purchased in 70% minimum purity from Invitrogen (Carlsbad, CA) — instead of by our *in vitro* method. As shown, the activity profiles for these peptides appeared to match their *in vitro*–synthesized counterparts, suggesting that the antimi-

Table 2.3: The preliminary design synthetic antimicrobial peptides used in this study For each synthetic AmP we also designed two sequences ("negative" a and b), which have the same amino acid composition as the synthetic peptide but have an S–score of zero. The table also shows statistics relavant to AmPs, which were calculated using the EMBOSS software package[206]. Note that synth–3 has only one negative version. Also, the peptides synth–1, 2, and 3 were the *only* peptides designed using our grammatical approach that were synthesized and tested experimentally.

Peptide	S	Size	Charge	pI	Sequence
synth–1:		20	4.5	11.92	
synth–1	1				NKVKKPLTGAHRLLFTFLFV
negative–1a	0				VVLKLLFFKFNLPHKTRTAG
negative–1b	0				LVLTFLFATPKLNGRVKKFH
synth–2:		31	10.0	11.28	
synth–2	1				MKKIKKEAGKNILKLAPKEVAAKKSKKSPTK
negative–2a	0				PAAGESKVKANKKKAKILPTMKLKKEIKKKS
negative–2b	0				SEASLKAKIKKIAMKKVTKGKAKNKPKLPEK
synth–3:		63	3.0	10.41	
synth–3	0.92				MKDKNSTGPLLSALLLAVTAGGSPVAAAPWNPFAAILKAALQIAGAAEPKEVTAKKGPTKADA
negative–3a	0				GWAGLVAETAIADKMSLKAAGEPPNQNDGAVLKTPPKAAASAKPLGAAKTLAFISPVTLALAK

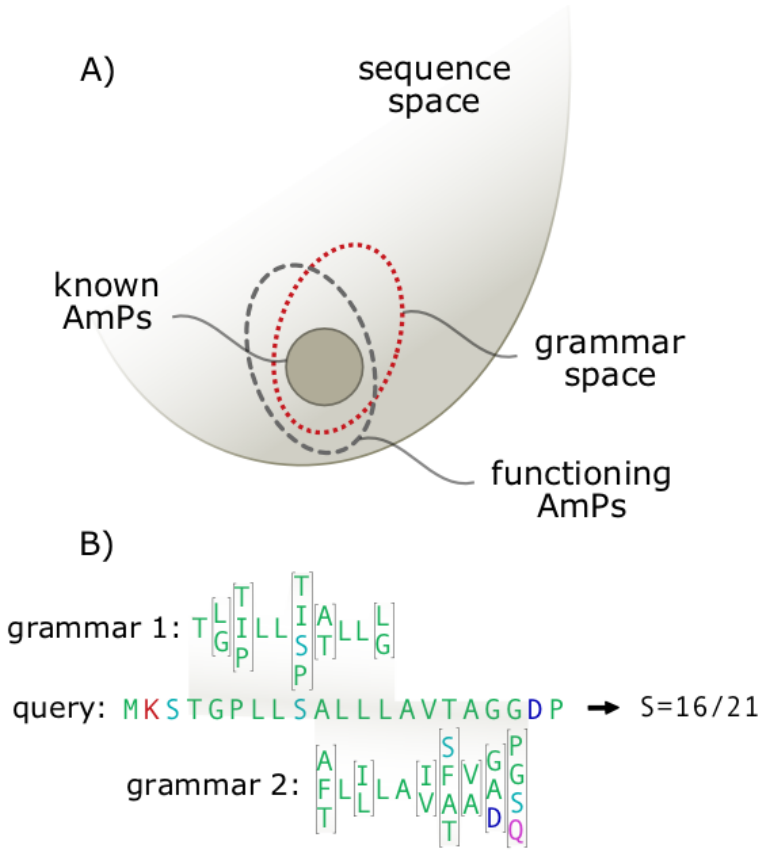


Figure 2-9: The AmP design space. Part A shows the sequence space surrounding the set of natural AmPs. The "sequence space" is the combinatorially large set of all possible sequences. Even for a 20–residue peptide like synth–1 (see Table 2.3) this space is huge: $20^{20} \approx 10^{26}$ sequences. (For comparison, there are about 10^{22} stars in the known universe.) Our linguistic model focuses the search space to the "grammar space," but allows a deviation from natural AmP sequences. This allows us to design peptides that show no significant homology to any naturally occurring sequences, but have the desired function. Part B shows a subsequence of the synth–2 peptide. Above and below the subsequence are grammars that match the sequence in a tiled arrangement. For each bracketed expression, any of the amino acids listed in the bracket will suffice.

crobial activity was not a relic of the translation mixture. (We also used the chemically synthesized copy of synth–1 to validate the size of our *in vitro* synthesized copy; see Figure 32-11.) Furthermore, we found that luciferase (a luminescent protein with no antimicrobial characteristics), when synthesized via our *in vitro* method, had no activity. Thus, we were confident that the translation mixture had no innate antimicrobial activity that may have produced spurious results.

For each peptide/organism combination, we measured a minimum inhibitory concentration (MIC) at which 80% of colony growth was inhibited (see Table 2.5). Many of the peptides appeared to exhibit strong bacteriostatic activity ($MIC_{80} \leqslant 8$ μg/mL). For example, all of the peptides seemed highly active against *B. cereus*. However, with the exception of synth–3, all of the AmPs appeared specific to gram–positive bacteria. Such specificity is a common characteristic of natural AmPs. For example, the insect cecropins are usually specific to gram–positive bacteria [226]; whereas, the honey bee AmP apidaecin is active only against gram negative bacteria [51]. In general, the underlying reasons for the variations in the susceptibilities of different bacterial species is unknown [280].

We selected the synth–1 family of peptides (*synth–1, *negative–1a, and *negative–1b in Table 2.3) to characterize more thoroughly. These peptides were tested at concentrations up to 50 μg/mL (a typical MIC for moderately active naturally–occurring AmPs) against the gram–negative bacteria. These additional experiments suggested that that *synth–1 was active against *C. rodentium* at 50 μg/mL, preventing 99% of the bacterial growth with an MIC_{80} of roughly 40 μg/mL. However, *synth–1 was not active against *E. coli* at this concentration. As expected, the *negative–1a and *negative–1b peptides did not have any activity against either *C. rodentium* or *E. coli* at 50 μg/mL.

In addition, we tested the synth–1 family of peptides for cooperativity with the polymyxin B nonapeptide, which is known to permeabilize the outer membrane of gram–negative bacteria. We found that the nonapeptide did not sensitize *C. rodentium* or *E. coli* to *synth–1, *negative–1a, or *negative–1b, suggesting that the outer membrane may not be the limiting factor in the activity of synth–1 against gram–negative bacteria.

Finally, we measured the activity of the synth–1 family of peptides against human erythrocytes (see Figure 2-12 on page 78). Our results show that *negative–1a was moderately hemolytic and suggest an HM_{50} of approximately 60 μg/mL. *Synth–1 and *negative–1b were less active against erythrocytes, with HM_{50} concentrations (by extrapolation) of roughly 260 and 290 μg/mL, respectively.

2.4.3 Later experimentation

As mentioned briefly in the preceding sections, the experimental data suggesting that synth–1 had to antimicrobial activity, were later shown not to be reproducible. Specifically, the *synth–1 peptide was shown to have no antimicrobial activity up to 50 μg/mL. These experiments implied that the data for all of the synthetically generated peptides discussed in the previous section were suspect, with the exception of the data on the hemolytic potential of the peptides. Based on these new findings, we revisited the AmP design problem and developed a more focused approach on the assumption that the original evolutionary methodology would not succeed. In particular, the lack of activity by the *synth–1 peptide indicated that perhaps the Z–score was an inadequate metric for designing AmPs, despite its power for annotation.

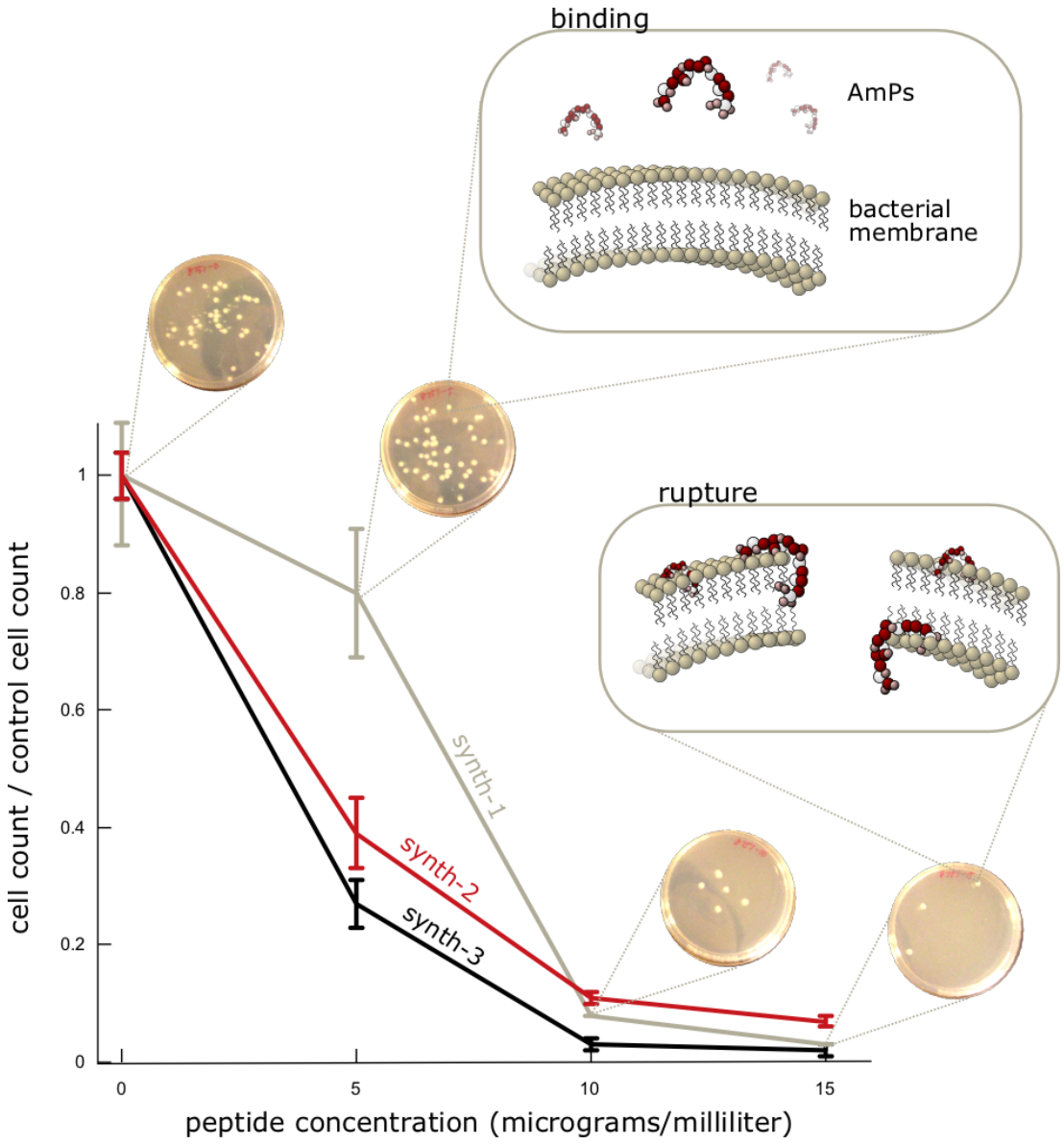


Figure 2-10: Bacteriostatic activity of the three synthetic AmPs against *B. cereus*. The breakouts show photographs of the colonies, which decreased in number with increasing peptide concentration. The inlaid schematic shows the generally accepted mechanism of AmP action: the electrostatic affinity for the outer–leaflet of the bacterial membrane leads to binding and rupture of the cell [226]. (*These data were later shown to be not reproducible. Later experiments showed that these peptides had undetectable levels of antimicrobial activity under a more reliable antimicrobial assay. See Section 2.4.3 on the preceding page.*)

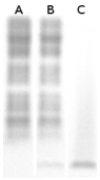


Figure 2-11: SDS–PAGE gel showing the synth–1 *in vitro* translation product (lane B). Lane A shows the translation mixture with no peptide and lane C shows the *synth–1 peptide, which was produced via solid phase synthesis and validated by mass spectroscopy.

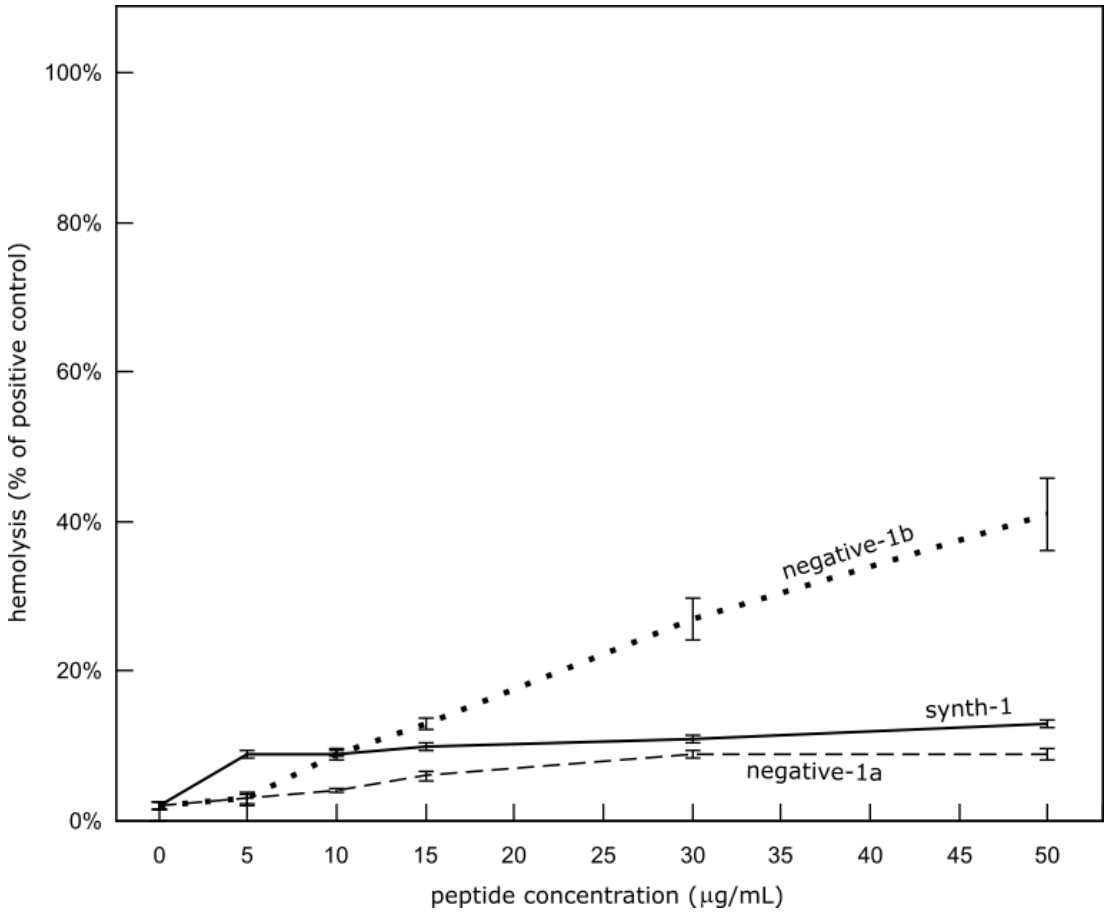


Figure 2-12: Activity of the synth–1 family of peptides against human erythrocytes determined using a procedure described elsewhere [153]. The ordinate shows the degree of hemolysis relative to 50 μg/mL of melittin, which causes complete hemolysis.

Table 2.4: Antimicrobial activity of the synthetic peptides against a variety of bacteria. Each entry in the table shows the relative viability of the bacteria: the ratio of the cell count at a particular concentration of AmP to the cell count at 0 μg/mL. The entries in dark gray show high viability (low antimicrobial action) and the white entries show low viability (high antimicrobial action). The names prepended with a "*" are peptides that were chemically synthesized rather than produced via *in vitro* translation.

species →	*B. subtilis*				*C. glutamicum*				*E. coli*				*C. rodentium*			
peptide conc. (μg/mL) →	0	5	10	15	0	5	10	15	0	5	10	15	0	5	10	15
synth–1:																
synth–1	1.00	0.80	0.08	0.03	1.00	0.73	0.21	0.15	1.00	1.00	1.00	0.93	1.00	0.99	0.97	0.98
negative–1a	1.00	0.96	1.00	1.00	1.00	1.00	0.93	1.00	1.00	1.00	0.99	1.00	1.00	0.99	1.00	0.98
negative–1b	1.00	0.98	1.00	1.00	1.00	1.00	0.96	0.99	1.00	1.00	0.98	0.99	1.00	1.00	1.00	1.00
synth–2:																
synth–2	1.00	0.27	0.03	0.02	1.00	0.28	0.21	0.12	1.00	1.00	1.00	0.99	1.00	0.99	0.99	0.97
negative–2a	1.00	1.00	1.00	1.00	1.00	1.00	0.96	0.91	1.00	0.98	0.99	1.00	1.00	0.95	0.99	0.95
negative–2b	1.00	1.00	1.00	1.00	1.00	0.92	0.95	0.98	1.00	0.98	0.97	0.97	1.00	0.98	0.94	0.99
synth–3:																
synth–3	1.00	0.39	0.11	0.07	1.00	0.40	0.26	0.18	1.00	1.00	1.00	0.99	1.00	1.00	0.98	0.10
negative–3a	1.00	0.96	0.98	0.98	1.00	1.00	0.96	0.96	1.00	1.00	1.00	0.99	1.00	1.00	0.96	0.96
Chemically synthesized:																
*synth–1	1.00	0.72	0.14	0.14	1.00	0.56	0.16	0.11	1.00	0.89	0.92	0.86	1.00	0.90	0.90	0.90
*negative–1a	1.00	1.00	0.84	0.87	1.00	1.00	1.00	1.00	1.00	0.92	0.90	0.94	1.00	1.00	0.98	1.00
*negative–1b	1.00	0.92	0.98	0.96	1.00	1.00	0.96	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00

Table 2.5: Minimum inhibitory concentration of the preliminary design synthetic AmPs against a variety of bacteria. In the table MIC_{50} is the concentration of peptide, in μg/mL, required to inhibit 50% of the bacterial growth. A "-" indicates that the MIC is greater than 15 μg/mL.

	synth–1		synth–2		synth–3	
	MIC_{50}	MIC_{80}	MIC_{50}	MIC_{80}	MIC_{50}	MIC_{80}
Gram–positive:						
B. subtilis	7.5	10	3.5	6	4.5	8.5
C. glutamicum	4	13.5	3.5	11	4	10
Gram–negative:						
E. coli	-	-	-	-	-	-
C. rodentium	-	-	-	-	12	15

2.5 Focused design of AmPs

2.5.1 Derivation of highly conserved AmP grammars

In the previous section, I described a strategy for designing novel AmPs that have a strong emphasis on sensitivity. That is, much effort was expended collecting a database of AmP sequences that was exhaustive so that the set of grammars derived from that database would be exhaustive as well (see Section 2.3 on page 60). The annotation experiments suggest that this strategy is sensitive for discovering novel AmPs; however, the lack antimicrobial activity by *synth–1 suggests that perhaps this approach (and the metric Z) is not selective. This lack of selectivity may be rooted in the exhaustive database of AmP sequences, which contains many sequences spanning a wide range of activities. That is, there are some AmP sequences in the database that have very low activity and some with very high activity. In addition, many of the sequences in this database are in precursor form. For the sequences, the precursor undergoes a series of post translational modifications before yielding a mature, active antimicrobial peptide. The regions of the proteins that are cleaved off, or otherwise not responsible for the antimicrobial activity of the peptide are essentially "noise" in the derived set of ~200K grammars derived in the previous section.

In order to focus instead on specificity, not sensitivity *per se*, we decided to use a database of well–characterized eukaryotic AmP sequences from the Antimicrobial Peptide Database (APD) [263]. The APD is unique in that it is the only database of antimicrobial peptides that restricts the sequences it catalogs to only those for which there is a large body of experimental data confirming the activity of each AmP. Furthermore, the AmPs listed in the APD are mature in the sense that they are not precursor proteins. Therefore, we know with high confidence that each sequence in the APD has antimicrobial activity and that we are unlikely to be training on sequences that are not responsible in some part for this activity.

As in Section 2.4 on page 71, we used the Teiresias pattern discovery tool to derive regular grammars that occur commonly in the set of 526 well–characterized eukaryotic AmP sequences from the APD. Using these APD sequences, we ran the Teiresias pattern discovery tool with

the following settings: L = 6, W = 6, and K = 2 (a detailed description of the Teiresias input parameters and associated tools is available in Section 1.5 on page 45). The resulting grammar set was masked from the input sequences and the process was repeated using L = 7, W = 15, K = 5 with the following amino acid equivalency groups `[[AG], [DE], [FYW], [KR], [ILMV], [QN], [ST]]`. The equivalency groups mean that Teiresias will consider any two characters in the same group to be exactly equivalent. Thus, in the groups above alanine is treated exactly as glycine. In effect, using equivalency groups allows us to find motifs that are more weakly conserved, but that have similar chemistries. (As I will show in Chapter 3 on page 89, Teiresias is unable to use a more fine–grained metric for the similarity between two amino acids. That is, Teiresias can only use equivalency groups to say "equivalent" or "not equivalent" but it cannot use metrics such as "alanine is five arbitrary units in a way from glycine, which is 10 arbitrary units away from leucine.)

As I discussed in Section 1.5 on page 45, Teiresias outputs its grammars in regular expression format, using wildcards. To make the grammars more selective, we de–referenced each wildcard in the grammars to a bracketed expression, using the same procedure described in Section 2.3 on page 60. That is, we replaced each wildcard with the set of amino acids implied by the grammar's offset list. Finally, as in Section 2.3, to allow partial matches as short as 10 amino acids, we divided each grammar into sub–grammars using a sliding–window of size 10, resulting in 1551 grammars of length ten.

By design, these 1551 are sensitive for the AmP sequences from the APD. That is, these sequences from the APD are likely to be matched by the grammars. However, the grammars are not necessarily specific for the APD AmPs. That is, non–AmP sequences may also be matched by the grammars. As discussed above, in our revised strategy, we used the APD sequences to enhance specificity. Here, we reinforce this specificity by eliminating noninformative grammars.

To select only those AmP grammars that are both sensitive and selective, we searched each of the grammars against the nearly exhaustive set of all known AmPs that was assembled in Section 2.3 on page 60. These sequences consisted of the ~750 AmPs from the AMSdb [250], which were supplemented with an additional ~200 antimicrobial peptides from Swiss–Prot/TrEMBL that were not included in the AMSdb. In addition, we searched each of the grammars against sequences from Swiss–Prot/TrEMBL that were not AmPs. Using these two searches, we eliminated grammars that were not at least 80% selective for AmPs. That is, at least 80% of the matches for a single grammar had to come from the set of all known AmPs.

The resulting, final set of 684 ten amino acid grammars was used as the basis set of grammars to design the unnatural AmPs. As before, we say that these 700 grammars describe the "language" of the AmP sequences and any sequence that is matched by one of the grammars is, at least in part, "grammatical."

2.5.2 Design of synthetic AmP sequences

To design unnatural AmPs, we combinatorially enumerated all grammatical sequences based on the set of ~700 grammars. First, for each grammar, we wrote out all possible grammatical amino acid sequences. So, for example, for the grammar `[IVL]K[TEGDK]V[GA]K[AELNH][VA][GA]K` produced 600 sequences, where 3*5*2*5*2*2 = 600, due to the option of choosing one of many amino acids at each bracketed position. There are roughly 3 million such 10–mers that correspond to antimicrobial patterns. Then we wrote out all possible 20 amino acid sequences for

which each window of 10 amino acids is found in the set of 3 million 10–mers.

This process is somewhat analogous to the convolution step of Teiresias. That is, we have essentially "stitched" small grammatical sequences together to form longer grammatical sequences. For example, the grammatical 10–mer IKTVAKEVGK would be stitched together with any other 10–mer beginning with the nine amino acid sequence KTVAKEVGK. In this way, the small set of ~700 grammars can give rise to a tremendous number of 20 amino acid sequences.

From this set, we removed any 20–mers that had six or more amino acids in a row in common with a naturally occurring AmP. There are roughly 12 million such 20–mers, each of which is a "tiling" of ten 10–mers.

In the last section (see page 71), I described a metric, Z, for scoring sequences against a database of grammars. Recall that Z essentially is a measure of what fraction of a query sequence is matched by grammars from the database of AmP grammars. However, this metric was not dependent upon how many grammars matched the query sequence. That is, there was no weighting of grammars that were particularly common in AmPs relative to grammars that may have only occurred once or twice. Consistent with the approach in the previous section, this metric is sensitive, rather than specific. In our new, more focused approach, we developed a different metric Q, which is the degree to which a given 20–mer is grammatical. This score is computed by making a sequence dot plot matrix [162] (see Figure 2-13 on the facing page). In the dot plot, the columns represent the positions, 1–20, of the query 20–mer and the rows represent the concatenated sequences of the ~1000 naturally occurring AmPs. A dot is placed in the matrix wherever a grammar matches both a naturally occurring AmP and the 20–mer. Then score Q is then just the number of dots in the matrix. That is, the score Q is the area shown at the bottom of Figure 2-14 on page 84. As shown in the figure, the score Q is more indicative of how homologous a query sequence is to the naturally occurring AmPs than the score Z developed in the previous section. (Rather than the area under the curve, the score Z is just the fraction of the query that is matched by grammars.)

In order to choose a representative set of sufficiently different synthetic sequences to test experimentally, we clustered the 12 million sequences using the Mcd–hit software [150] at 70% identity. From these clusters, we chose 42 high scoring sequences to test experimentally. These sequences have varying degrees of similarity to naturally occurring AmPs, as determined by sequence alignment. Notably, from each cluster, we took the highest scoring synthetic sequence based on the Q metric. These 42 sequences are shown in the left–hand side of Table 2.6 on page 86.

For each of the 42 synthetic peptides, we also designed a shuffled sequence, in which the order of amino acids was rearranged randomly such that the sequence did not match any grammars. These shuffled peptides are shown in the right–hand side of Table 2.6 on page 86. Necessarily, these peptides had the same amino acid composition as their synthetic counterparts and thus, the same molecular weight, charge, and pI: bulk physiochemical factors often correlated with antimicrobial activity. We hypothesized that because the shuffled sequences were "ungrammatical" they would have no antimicrobial activity, despite having the same bulk physiochemical characteristics. In addition, we selected 9 peptides from the APD as positive controls (Cecropin P1, Cecropin Melittin Hybrid, Cecropin–A Magainin 2 Hybrid, Melittin, Magainin 2, Hepcidin, Pyrrhocoricin, Ranalexin, and Parasin) and six 20–mers selected randomly from the middle of non–antimicrobial proteins as negative controls.

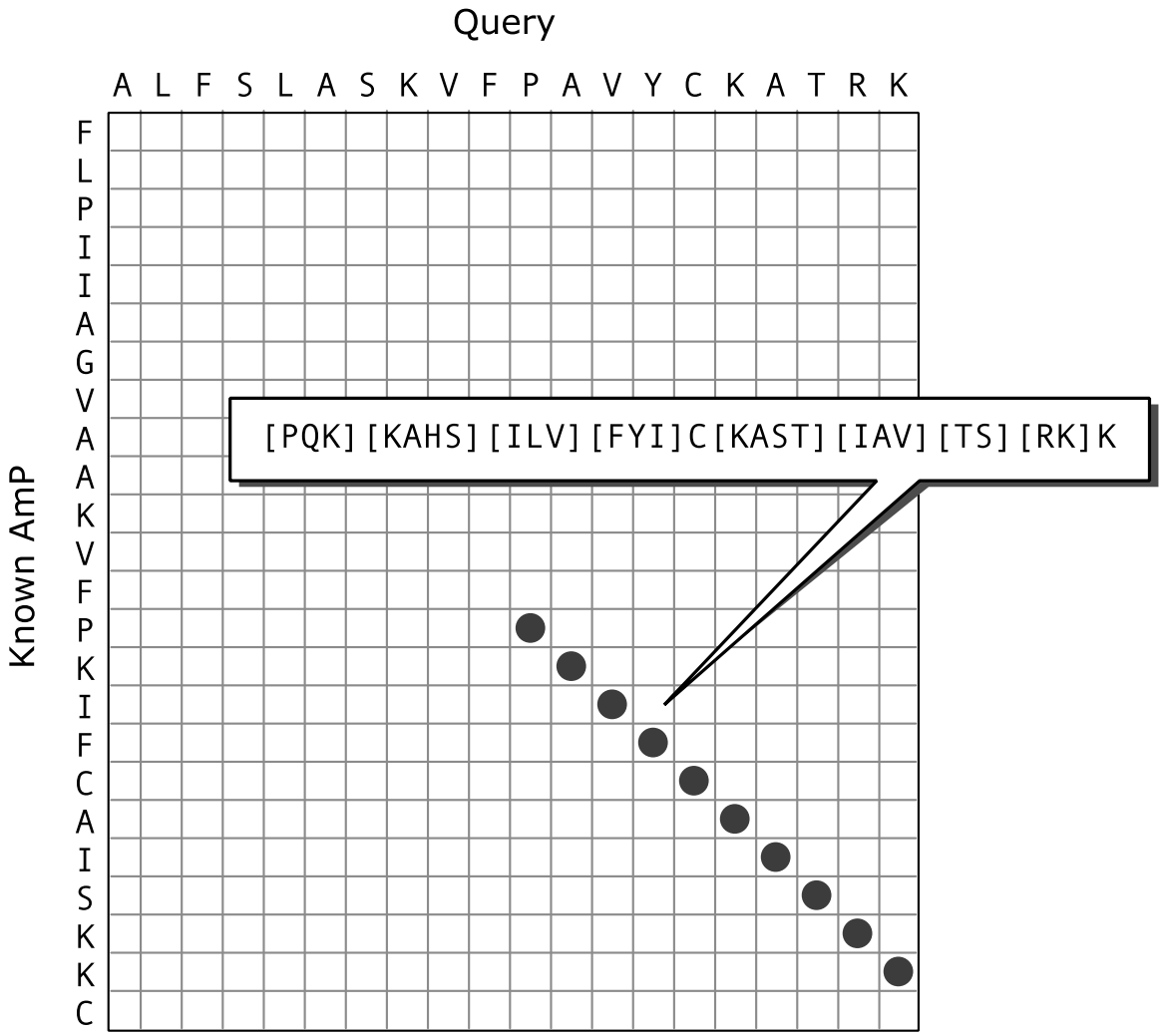


Figure 2-13: An example grammar–based dot plot. The figure shows a query sequence all the top and a single known AmP sequence on the vertical axis along the side. The matrix has an entry for each pair of amino acids between the two sequences. The breakout shows a grammar from our database that matches both of the sequences. Note that both sequences begin a grammar with a proline residue; however, the grammar is not entirely conserved. The next residue in a grammar differs in each of the two sequences. To calculate our scoring metric Q we compute many grammar–based dot matrices using this same approach. For example, see Figure 2-14 on the next page. Activity of rationally designed AmPs.

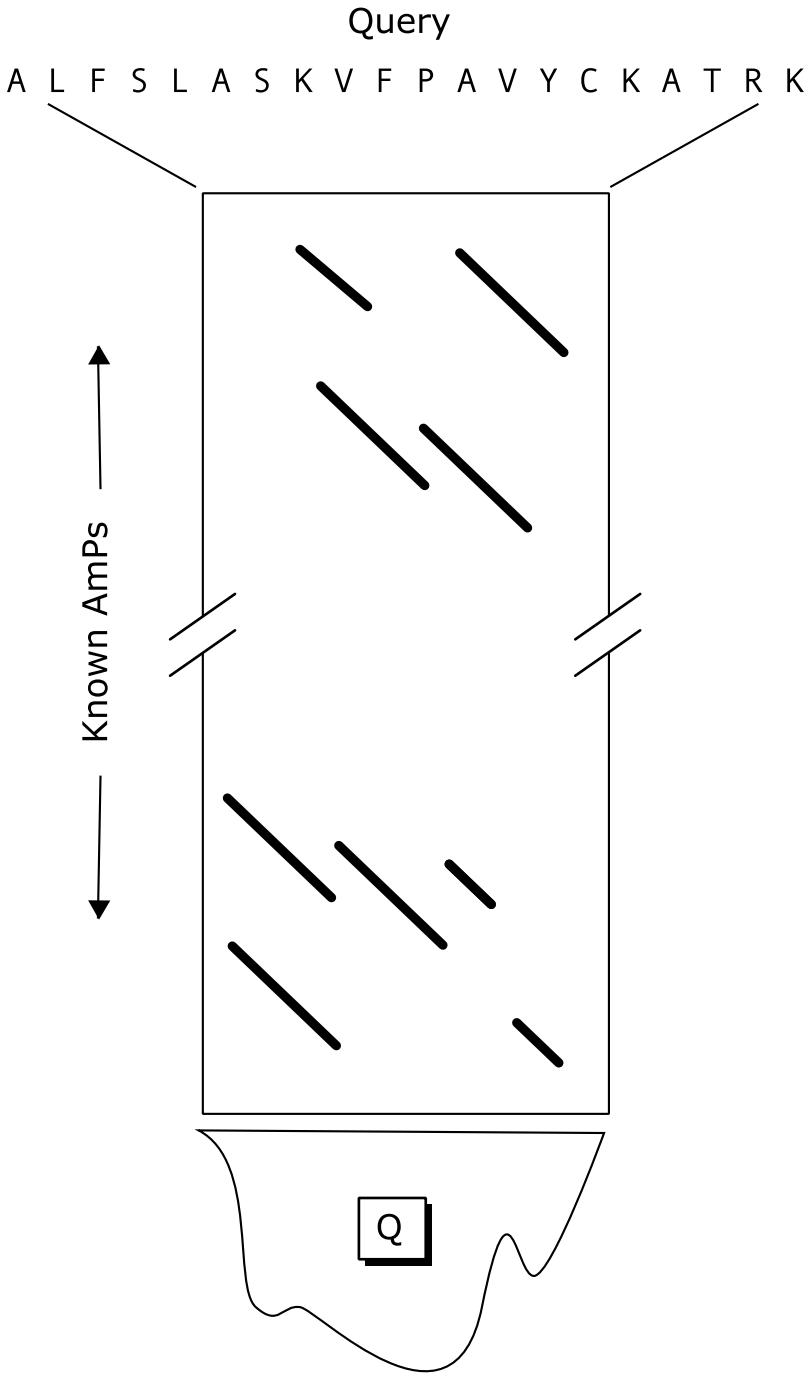


Figure 2-14: An example grammar–based dot plot for computing Q. The figure shows a "zoomed out" view of many dot matrices concatenated together. (See the dot matrix shown in Figure 2-13 on the page before.) At the top of the matrix is the query sequence, which is the synthetic, hypothetical AmP that is to be scored. Along the vertical axis lay the concatenated sequences of the set of ~900 known AmPs. The diagonal streaks show places where a grammar matches both the query sequence and a known AmP. At the bottom, to score Q is shown. The score is the area under the curve and is simply a tally of the total number of dots in the dot matrix. War, equivalently, the total length of all streaks in the figure. In this sense, the score Q has a greater emphasis on specificity than did the score Z, which was merely be extent of the query sequence covered by grammars.

2.5.3 Assay for antimicrobial activity

Each of the peptides shown in Table 2.6 on the following page was synthesized using solid–phase, Fmoc chemistry on an Intavis Multipep Synthesizer (Intavis LLC, San Marcos, CA) at the MIT Biopolymers Lab. Mass spectrometry was used to confirm the accuracy of the synthesis — typical purities obtained with the synthesizer were >85%.

We characterized the activity of each synthetic AmP using a broth microdilution assay described elsewhere [274]. This assay measures the MIC at which the peptide inhibits growth of the target organism. The assay is based on the NCCLS M26A and the Hancock assay for cationic peptides (Hancock, NB, Canada). Briefly, serial dilutions of peptides in 0.2% Bovine Serum Albumin and 0.01% Acetic acid were made at 10x the desired testing concentration. Target bacteria were grown in Mueller Hinton Broth (BD, Franklin Lakes, NJ) to OD600 between 0.1 to 0.3 and diluted down to $2 - 7 \times 10^5$ cfu/mL in fresh MHB, as confirmed by plating serial dilutions. Five µL of the peptide dilutions was incubated with 45 µL of the target in sterile, capped, polypropylene strip tubes for 16–20 hours. The minimum concentration that prevented growth based on visual inspection of OD was defined as the MIC. When desired, the samples that did not grow were streaked on an MHB agar plate to see if the peptide was bacteriocidal.

Recombinantly produced standards for Cecropin P1, Cecropin Melittin Hybrid, Melittin, Magainin 2, and Parasin were purchased from the American Peptide Company (Sunnyvale, CA). In antimicrobial assays, four of the five recombinant peptides had identical activities to the chemically synthesized versions from MIT biopolymers, with the last being one dilution different (Cecropin P1).

2.5.4 Results and conclusions

Table 2.6 on the next page shows the MICs of synthetic peptides against *B. cereus* and *E. coli*, as representative gram positive and gram negative bacteria. (Two of the designed and 4 shuffled peptides were insoluble). Of the 40 soluble designed peptides, 18 had activity against at least one of the bacterial targets at 256 µg/ml or less. Only 2 of the soluble shuffled peptides displayed activity. Thus, the activity is not an artifact of molecular weight, charge, or pI.

Of the the negative controls — 6 peptides randomly selected from the middle of non–antimicrobial proteins from Swiss–Prot/TrEMBL — none had activity. Six of the nine naturally–occurring AmPs in the positive control group show activity and one was insoluble.

Two of the designed peptides, D28 (FLGVVFKLASKVFPAVFGKV) and D51 (FLFRVASKVFPALIGKFKKK), inhibited *B. cereus* growth at 16 µg/mL, which is close to the MICs of the strong positive controls melittin and cecropin–melittin hybrid (8 µg/mL). (Here we use the letter "D" to distinguish a designed peptide from its shuffled equivalent with the same number.) Peptides with gram positive activity are particularly exciting because of the prevalence of drug–resistant nosocomial *S. aureus* and the threat of bioterror agents such as *B. anthracis*, or anthrax. Therefore, we assayed the seven designed peptides that had gram positive activity, including the highly active D28 and D51 peptides, against the Smith Diffuse strain of *S. aureus* and the Sterne strain of *B. anthracis*. As shown in Figure 2-15 on page 88, all seven peptides had activity against both bacteria, whereas only one of the seven shuffled controls had activity. Moreover, two designed peptides, D28 and D51, had activity against *Bascillus antrhracis*

Table 2.6: Antimicrobial activity of rationally designed and shuffled peptides. Each entry shows the minimum inhibitory concentration in μg/mL. "+" = MIC greater than 256 μg/mL. ++ = MIC greater than 128 μg/mL, not sufficiently soluble to test at 256 μg/mL.

Peptide	Sequence	*E. coli*	*B. subtilis*	Shuffled Sequence	*E. coli*	*B. subtilis*
1	ALFSLASKVVPSVFSMVTKK	+	+	MVVFSVPKFKSTVAKLLSSA	+	+
2	VVFRVASKVFPAVYCTVSKK	128	+	TAKVVVFVSFSYVVPKKRAC	+	+
5	FLFGLASKVFPAVYCKVTRK	64	256	FLPVLVKVFRYSKKTAAGCF	++	64
6	LSAVGKIASKVVPSVIGAFK	+	+	GVSSPIVAVKFKGAVASLIK	+	+
7	PVIGKLASKVVPSVFSMIKR	+	+	SRVPLKSPVKIVGSKVMIFA	+	+
9	GLMSLVKDIAKLAAKQGAKQ	256	+	GLKKDALQSIVKKAQLAAMG	+	+
15	SALGRVASKVFPAVYCSITK	+	+	LYSPTCVKAAVSRFIGKVSA	+	+
22	LGALFRVASKVFPAVISMVK	256	64	SVPSVGAVLFFKRAAVMKLI	+	+
23	ALGKLASKVFPAVYCTISRK	128	+	KYGPALVIAVKKSCSLTFRA	+	+
24	GFIGKLASKVVPSVYCKVTG	128	+	GGSTLGVFVKKSKACVIVPY	Not soluble	
25	PVVFSVASKVVPSLISALKR	+	+	KSPFVLVVSSRVAAVIKSLP	+	+
28	FLGVVFKLASKVFPAVFGKV	64	16	GVSVAGAKKVKVLFVFPFLF	+	+
29	PAVFKIASKVVPSVYCKVSR	128	+	KVYVVKIAVPCFPKSARSVS	+	+
30	GALFGLASKVFPAVFGAFKK	256	+	KVVLFGAAGAKLFKASFFGP	Not enough material	
31	SAVGKLASKVFPAVFSMVTK	+	+	FMKVLAVFGSVVTSAPKASK	+	+
33	VKDLAKFIAKTVAKQGGCYL	++	++	ALVYAGIKKTAFLKVQKCDG	+	+
34	GVVGKLASKVVPSVFGSFTK	+	+	SVKPVGSSVVKGTALVKFFG	+	+
35	LPVVFRVASKVFPALISKLT	+	256	KVFIATLVVSSFLLAKPPRV	+	+
36	SAVGSVASKVVPSLISKVTK	+	+	STVKVASKLAVVVSPISKGS	+	+
39	MKSIAKFIAKTVAKQGAKQG	+	+	AKKAQKSGAQTIVKIFAKGM	+	+
42	LPAVFKLASKVVPSVFGLVK	+	+	VVAKKFFVLVKGLAPVLSPS	+	+
43	SFVFKLASKVVPSVFSALTR	256	256	ASPTVFRSSVFLSLFVVAKK	+	+
44	SVIGKIASKVVPSVYCAISK	+	+	IASAVPVCVKGKISKSYISV	+	+
45	PVVGRVASKVFPAVIGLVKK	+	+	VKRAGKGVAVVPSPLFKIVV	+	+
51	FLFRVASKVFPALIGKFKKK	64	16	RKVAPALIKSFVFLFKFKKG	+	+
55	LSFVGRVASKVVPSLISMIK	256	+	SSSIPIKMVLVRALVFVKSG	+	+
56	SALGRLASKVVPAVIGKVTT	+	+	TLVGVVAKLVATKIGSSPRA	+	+
57	LGVVGSLASKVVPAVISKVK	+	+	PKVVGLSIVVVKAKVSSALG	+	+
62	LPAVFKLASKVFPAVYCKAS	128	+	PSLLYKAKAVFCKPSAVAVF	++	++
63	LPVLFKLASKVFPAVFSSLK	256	64	VSVKKVLPFAPLKSLLSFAF	256	256
65	VVGRVASKVVPSLIGLFTTK	+	+	FKVVISKPGLSVRVGTALVT	++	++
69	SVVFGVASKVVPSVIGKVKT	+	+	VFSVKGGKPSVVIKVVVAST	+	+
75	FLPFVGRIASKVVPSVIGKV	+	+	SKFPLAGIFSVPGVKRVVVI	+	+
77	GKKLAKTIAKEVAKQGAKFA	64	+	VIAFAKTKEAKAKLKGQAKG	+	+
81	PFVGRVASKVVPSVYCAITR	Not soluble		PAVYKSIVGFSPVARVTVCR	Not soluble	
82	FVGSLASKVVPSVFGAIKTK	+	+	KTVPVVLKASIKVSSAGFGF	+	+
83	LPVVFKIASKVVPSVISKIT	+	+	KIVKVITVKSISPASLVPVF	++	++
84	GAVFGVASKVVPSVFSAIKK	+	+	SVKVAKSVIPSAVFAGGKVF	+	+
85	FVGGVASKVVPSVYCKVSKK	+	+	KVGKGSYPCSFVKVVAKVSV	+	+
88	VVFKLASKVVPSVYCTITKK	256	+	VKTKCSVPAVVYILVKTFKS	+	+
96	GALFSLASKVVPAVIGLIKK	256	+	LPVLFSSAIAKVGIKLGAKV	+	+

Table 2.7: Antimicrobial activity of rationally designed and shuffled peptides against *S. aureus* and *B. anthracis*. Each entry shows the minimum inhibitory concentration in μg/mL. "+" = MIC greater than 256 μg/mL. ++ = MIC greater than 128 μg/mL, not sufficiently soluble to test at 256 μg/mL.

Peptide	Sequence	*S. aureus*	*B. anthracis*	Shuffled Sequence	*S. aureus*	*B. anthracis*
28	FLGVVFKLASKVFPAVFGKV	8	16	GVSVAGAKKVKVLFVFPFLF	+	+
51	FLFRVASKVFPALIGKFKKK	16	16	RKVAPALIKSFVFLFKFKKG	128	256
22	LGALFRVASKVFPAVISMVK	64	64	SVPSVGAVLFFKRAAVMKLI	+	+
63	LPVLFKLASKVFPAVFSSLK	128	128	VSVKKVLPFAPLKSLLSFAF	+	+
5	FLFGLASKVFPAVYCKVTRK	256	128	FLPVLVKVFRYSKKTAAGCF	+	+
43	SFVFKLASKVVPSVFSALTR	256	128	ASPTVFRSSVFLSLFVVAKK	+	+
35	LPVVFRVASKVFPALISKLT	256	128	KVFIATLVVSSFLLAKPPRV	+	+

at 16 μg/mL, which is equivalent to the activity of cecropin–melittin hybrid, a strong natural peptide.

Also, D28 was synthesized by MIT biopolymers 4 separate times and the resulting peptides had consistent activities against both *E. coli* and *B. cereus*.

In an attempt to generate strong, synthetic AmPs, we optimized our best candidate, peptide D28, using a heuristic approach. We created 44 variants of D28 by introducing mutations that were selected to increase positive charge, increase hydrophobicity, remove an interior proline residue, and improve segregation of positive and hydrophobic residues based on a helical projection. 16 of the 44 D28 variants showed improved activity against *E. coli* or *B. cereus*. All of the D28 variants with improved activity against *B. cereus* included a mutation at an internal proline, either to lysine or glycine. D28 and six of its variants were assayed for bacteriocidal activity, and all had activity within a 2–fold dilution of their MIC. One variant had MICs of 16 μg/mL against *E. coli* and 8 μg/mL against *B. cereus* (relative to 64 and 16 μg/mL, respectively, for D28).

We suspect that our linguistic approach to designing synthetic AmPs is successful due to the pronounced modular nature of naturally–occurring AmP amino acid sequences. As we have shown, this approach can be used to rationally expand the AmP sequence space without using structure–activity information or complex folding simulations. The peptides designed in this work are different from previously designed synthetic AmPs [246, 251] in that they bear limited homology to any known protein, which may be desirable for AmPs used in clinical settings. Some critics argue that widespread clinical use of AmPs that are too similar to human AmPs will inevitably elicit bacterial resistance, compromising our own natural defenses and posing a threat to public health [29]. We hope that this approach will help to expand the diversity of known AmPs well beyond those found in nature, possibly leading to new candidates for AmP–based antibiotic therapeutics. Our designed AmPs show some degree of homology with natural AmPs because the grammars are based on native sequences. Peptide D28, for example, was matched by grammars derived from 11 natural AmPs including brevinin, temporin, and ponericin. However, Smith–Waterman alignments of our designed peptides against all natural AmPs in the Swiss–Prot/TrEMBL database reveal that the degree of homology is, by design (see Methods), limited. In particular, our two most active peptides, D51 and D28, have 50 and 60% sequence identity with the nearest natural AmP, respectively. Peptide D51 has 6 semi–

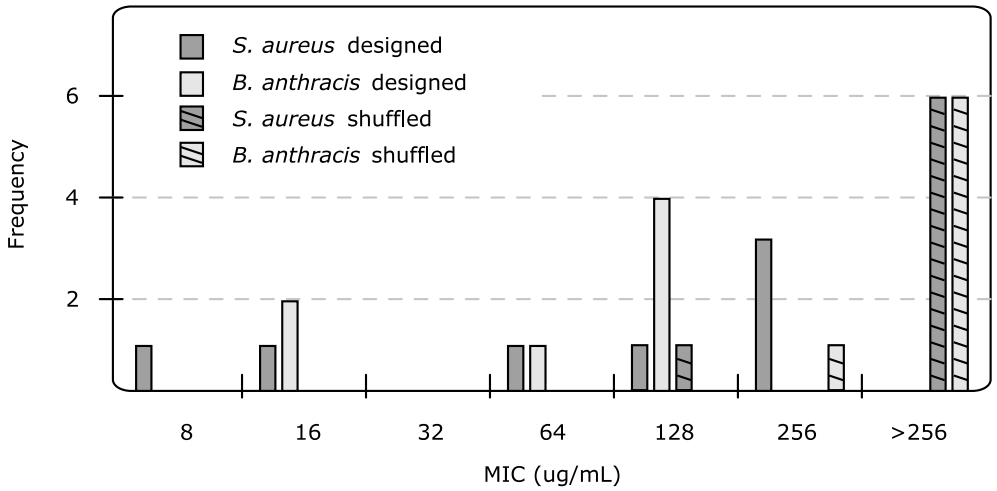


Figure 2-15: Activity of rationally designed AmPs against *S. aureus* and *B. anthracis*. The figure shows that shuffled peptides (the hashed bars) tend to be grouped on the right side of the plot, indicating that they have little or no antimicrobial activity. Only one of the shuffle peptides shows activity; however, it appears twice on the plot, once at 128 μg/mL against *S. aureus* and once act 256 μg/mL against *B. anthracis*. In contrast, all of the designed peptides show some degree of activity. The most highly active peptide is that the left–hand side of the plot.

conservative and 4 nonconservative substitutions relative to its closest neighbor, Ponericin W5. Our linguistic design approach may be most valuable as method for rationally constraining a sequence–based search for novel AmPs. Diverse leads generated by our algorithms may be optimized using approaches described in the literature [117]. But, the linguistic approach described here has a number of limitations. First, sequence families that are poorly conserved on an amino acid level would not benefit from this approach. Second, we suspect that the small size of AmPs is helpful. Due to the simple nature of regular grammars, they would be less useful for designing larger proteins and, in particular, proteins with complex tertiary or quaternary structures.

Chapter 3

A generic motif discovery algorithm

3.1 Introduction

In the previous chapter, I described the use of regular grammars for modeling the primary sequences of antimicrobial peptides. In that work, I showed that our specific approach to the design of novel AmPs yielded peptides with strong antimicrobial activity. However, recall that, in order to achieve specificity with some degree of sensitivity, the grammars had to be split into tiled 10 amino acid windows for increased sensitivity and then compared against a database of non–AmP sequences in order to increase specificity by throwing out uninformative grammars. This is because, as discussed in Chapter 1 on page 15, regular grammars are inherently more "coarse grained" then other models such as position weight matrices. Thus, to design AmPs, we had to use large sets of redundant, overlapping regular grammars. In such situations, the underlying sequence information might be better modeled by a position weight matrix or many other kinds of models.

In this chapter, I present a GEneric MOtif DIscovery Algorithm (Gemoda) for sequential data. Gemoda is a motif discovery tool very similar to Teiresias; however, Gemoda's output motifs are representation–agnostic: they can be represented using regular expressions, position weight matrices, or any number of other models. In addition, Gemoda can be applied to any dataset with a sequential character, including both categorical data such as protein and amino acid sequences, and real–valued data such as the price of a stock as a function of time. As I show in the following sections, Gemoda deterministically discovers motifs that are maximal in composition and length. As well, the algorithm allows any choice of similarity metric for finding motifs. I demonstrate a number of applications of the algorithm, including the discovery of motifs in amino acids sequences, a new solution to the (l,d)–motif problem in DNA sequences, and the discovery of conserved protein sub–structures.

The research described in this chapter is drawn largely from two publications:

- M. Styczynski, K. Jensen, I. Rigoutsos, & G. Stephanopoulos. "An extension and novel solution to the (l,d)-motif challenge problem." *Genome Inform Ser Workshop Genome Inform.* 2004;15(2):63–71; and

- K. Jensen, M. Styczynski, I. Rigoutsos, & G. Stephanopoulos. "A generic motif discovery algorithm for sequential data," *Bioinformatics* 22:21-28 (2006).

Throughout this chapter, the use of the pronoun "we" refers to the authors of these manuscripts.

3.2 Motivation

As discussed in Chapter 1 on page 15, motif discovery encompasses a wide variety of methods used to find recurrent trends in data. In bioinformatics, the two predominant applications of motif discovery are sequence analysis and microarray data analysis. Less common applications include discovering structural motifs in proteins and RNA [119, 174].

Motif discovery in sequence analysis typically involves the discovery of binding sites, conserved domains, or otherwise discriminatory subsequences. There are many publicly–available tools, a large number of which are listed in Section 1.5 on page 44, each of which is quite adept at addressing a specific subclass of motif discovery problems. Some of the commonly–used tools for motif discovery in nucleotide and amino acid sequences include MEME [19], Gibbs sampling [147], Consensus [115], Block Maker [113], Pratt [132], and Teiresias [207]. Newer, less-widely used tools include Projection [45], MultiProfiler [136], MITRA [78], and ProfileBranching [199]. This list is not intended to be exhaustive; however, it is indicative of the wealth of options available for solving such problems (see also Tables 1.4 & 1.5 in Chapter 1 on pages 47 & 53, respectively).

All of the existing motif discovery tools for nucleotide and amino acid sequences can be classified on a spectrum ranging from exhaustive tools using simple motif representations to non–exhaustive tools using more complex representations. The majority of the tools can be found at the extreme ends of the spectrum, with tools that exhaustively enumerate regular expressions (or single consensus sequences) at one end and probabilistic tools, based on position weight matrices (PWMs), at the other. This partitioning of tools is due to a computational trade–off: more descriptive motif representations such as PWMs frequently make exhaustive searches computationally infeasible.

One of the primary motivation for this work is the modeling of cis–regulatory sequences. We found that regular expressions are poor representations of binding sites and that, instead, these were better captured with PWMs. From a biological perspective, this makes more sense — the k_D of binding between the trans and cis factors are probabilistic, not deterministic. Thus, in order to model these sites using regular grammars or regular expressions, one must, in general, use combinations of patterns in an effort to piece together the information that would be contained within a PWM from many regular expressions.

Consider the following example. The LexA regulon consists of 9 gene sequence that are regulated by a single protein trans factor. The binding site of this trans factor is found in 8 of these sequences. Using the Teiresias motif discovery tool, with parameters $L = 10, W = 20, K = 5$ (see Section 1.5 on page 45) returns the following patterns

```
5  4  CTGTATAT.....CAG 0 355 0 376 4 298 6 326 7 363
5  5  CTGTAT....A..CAG 0 376 1 322 4 298 6 326 7 363
5  5  ACTGTA.....A..CAG 0 375 1 321 3 358 4 297 7 362
5  5  CTGTA.AT..A..CAG 0 376 3 359 4 298 6 326 7 363
6  5  CTGTA.AT.....CAG 0 355 0 376 3 359 4 298 6 326 7 363
5  5  ACTGT.T....A..CAG 0 375 1 321 4 297 5 307 7 362
5  5  ACTGT...T..A..CAG 0 375 3 358 4 297 5 307 7 362
5  5  CTGT.T.T..A..CAG 0 376 4 298 5 308 6 326 7 363
```

where the above grammars have been left and the native output form of Teiresias. The numbers on the right hand side indicate the offset list for each grammar. So, collectively, these patterns hit 7 of the 8 sequences; however, none of the patterns individually hits more than 5 sequences. Basically, this is because regular expressions don't capture such sites well.

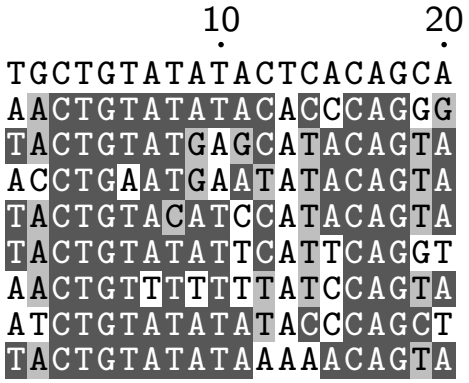
And this chapter, I described Gemoda: a motif discovery tool that has many of the strengths of Teiresias, but can find motifs that are best represented as PWMs. The details of Gemoda are discussed in the later sections of this chapter. But here, for motivation, consider the following output from the Gemoda all over them. If a user tells Gemoda to find all patterns in the LexA sequences such that, on a pairwise basis, each window of 20 nucleotides in each instance contains at least 10 nucleotides in common to each other instance, and the pattern occurs in at least 8 sequences; Gemoda returns only a single pattern:

```
0 353  TGCTGTATATACTCACAGCA
0 374  AACTGTATATACACCCAGGG
1 320  TACTGTATGAGCATACAGTA
2 230  ACCTGAATGAATATACAGTA
3 357  TACTGTACATCCATACAGTA
4 296  TACTGTATATTCATTCAGGT
5 306  AACTGTTTTTTTATCCAGTA
6 324  ATCTGTATATATACCCAGCT
7 361  TACTGTATATAAAAACAGTA
```

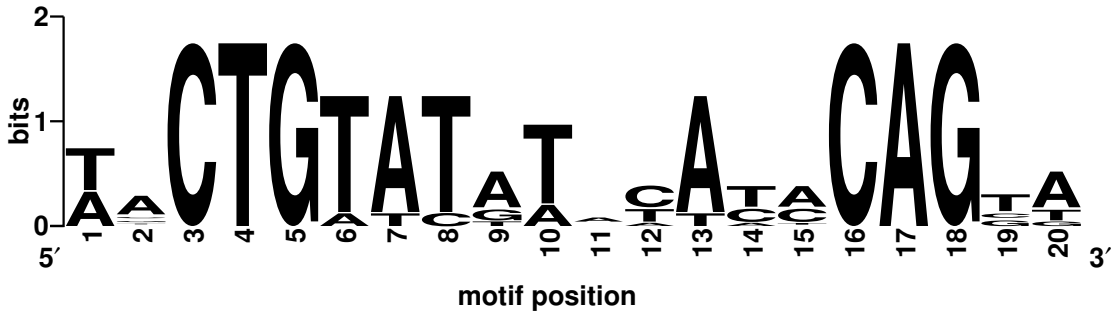
where, instead of one pattern per line, each line represents one of the offsets and the numbers on the left–hand side are, collectively, the offset list. Notice that here, only a handful of the positions within the pattern are fully conserved . However, most of the positions have "preferences." For example, the seventh position is mostly A. This pattern can be expressed as a PWM, has in Figure 3-1 on the next page, thus preserving these preferences in the matrix probabilities. Notably, this pattern is exactly the experimentally determined motif.

Depending on the task at hand, a specific type of motif discovery tool may be more useful than others. For example, the PWM–based tools excel at finding *cis*–regulatory binding elements [249], whereas the regular expression–based tools are well–suited to finding conserved domains in large protein families [208]. Generally, it can be difficult to know *a priori* which motif discovery tool will be right. Accordingly, there is an unmet need for motif discovery tools that can use a variety of motif models.

A)



B)



C)

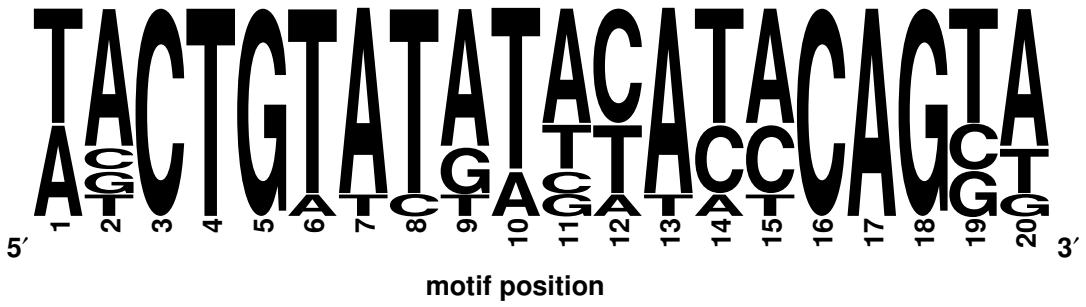


Figure 3-1: Alignment representing the LexA cis–regulatory binding site. Part A) of the figure shows the aligned sequences colored to indicate the degree of conservation. Part B) of the figure shows a sequence logo representing the information content of a PWM computed from the alignment of the motif instances. Part C) of the figure shows a sequence logo, wherein the height of each letter is proportional to its frequency, rather than to the information content it in codes as is the case in part B).

3.3 Algorithm

Gemoda was designed to meet the demand for complex motif representations, like PWMs, while still being exhaustive. The philosophical underpinnings of the Gemoda algorithm can be traced back to Teiresias [207]; Winnower [195]; the algorithm by [163]; and a variety of algorithms for association mining [277, 278]. In particular, Gemoda shares some of its logical steps with the Teiresias algorithm while incorporating a more flexible definition of "similarity" and allowing motif representations other than regular expressions.

The principle difference between Teiresias and most frequent itemset mining algorithms is that Teiresias acts on categorical sequential data, usually biosequences or integers. Most frequent itemset mining tools use market basket data sets, for example, a collection of products that a customer bought. Patterns in market basket data can be used to predict what other products a customer might buy (this is how Amazon.com works). The difference between categorical sequential data and market basket data (both are stochastic in that they consist of discrete values sampled from some real space) is that the former is ordered, whereas the latter is an unordered set. For similarly sized datasets, this makes sequential pattern discovery much easier. However, typically sequential datasets, such as biosequences or time–series stock data, are much larger. For example, a person may only purchase a few products from Amazon; however, gene sequences can consist of may thousands of characters.

Gemoda's design goals can be summarized as follows: *exhaustive discovery* of all *maximal motifs* in a way that allows flexibility in *motif representation*, incorporation of a variety of *similarity metrics*, and the ability to handle diverse *sequential data types*. Each point of emphasis can be explained as follows:

- **Exhaustive discovery:** Gemoda's combinatorial nature provides an algorithmic guarantee that all motifs meeting certain criteria are deterministically discovered.
- **Maximal motifs:** Gemoda returns only motifs that are maximal in both length and composition with respect to the similarity and clustering functions.
- **Motif representation:** The motifs discovered by Gemoda are reported as short multiple sequence alignments (in the case of motif discovery in nucleotide and amino acid sequences) and can be modeled using regular expressions, PWMs/PSSMs, Markov models, or any other representation.
- **Similarity metrics:** Any criterion, ranging from sequence alignment scores to geometric functions, may be used to compare sequences.
- **Sequential data types:** The nature of Gemoda's computations is not unique to any specific type of data, and thus can be used on any data with a sequential character — that is, data in which there is a natural left–to–right order, such as a sequence of nucleotides or amino acids. In the most general sense, sequential data also include real–valued series data, such as a stock price or the ordered (x, y, z) triplets of an alpha–carbon trace in a protein structure.

The algorithm has three distinct phases: comparison, clustering, and convolution. During the comparison phase, short overlapping windows in the data set are compared. During

clustering, these windows are grouped together to form elementary motifs. Finally, during convolution, these motifs are "stitched" together to form maximal motifs (see Figure 3-2 on the facing page). In the following sections, we give some brief definitions and nomenclature, then describe each of the algorithm's three phases in detail. Finally, we illustrate a few applications of Gemoda.

3.3.1 Preliminary definitions and nomenclature

The input to Gemoda is a set of sequences of data points $S = \{s_1, s_2, \ldots, s_n\}$, where sequence s_i has length W_i. So, for example, the j^{th} member of the i^{th} sequence is denoted by $s_{i,j}$. Each $s_{i,j}$ is a primitive, or atomic unit, for the data that is being analyzed. For time–series data, $s_{i,j}$ may be a point sampled from $\mathbb{R}^K$ (with K arbitrary), whereas for a DNA sequence it would be one of the characters $\{\texttt{A,T,G,C}\}$.

To demonstrate this notation and how he can be used to represent real–valued sequential data, rather biosequences consider the following example. Say we have two small peptides and we are interested in their structural properties. For each amino acid, we have a two–dimensional feature vector. The first feature is the hydrophobicity index [14] and the second is the size of the amino acid: 1 if it is over the 50^{th} percentile and 0 otherwise. The two peptides are `AIKDWR` and `DIHV`. Our two sequences are then

$$\begin{aligned} \text{seq–0} &= \begin{pmatrix} 0.61 & 2.22 & 1.15 & 0.46 & 2.65 & 0.60 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \\ \text{seq–1} &= \begin{pmatrix} 0.46 & 2.22 & 0.61 & 1.32 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \end{aligned}$$

such that

$$\begin{aligned} s_{0,0,0} &= 0.61 \\ s_{1,0,0} &= 0.46 \\ s_{1,1,1} &= 1 \\ s_{0,3,1} &= 0 \\ s_{1,2,0} &= 0.61, \end{aligned}$$

and so on.

Typically, one seeks motifs of a minimal, domain–dependent length. We denote this minimum length by L (similar to Teiresias) and we define a matrix A of size $N \times N$, where $N = \sum_{i=1}^{n}(W_i - L + 1)$. That is, A is a matrix with one row and one column for each window of size L in our entire sequence set. For example, the 10^{th} window of size L in the 5^{th} sequence would be expressed as $s_{5,10:10+L-1}$, where "$10 : 10 + L - 1$" denotes "position 10 through position $10 + L - 1$, inclusive." To keep track of which window corresponds to which index in A, we define the one–to–one function $\mathscr{M}(s_{i,j:j+L-1}) \mapsto q \in [1, N]$. (For simplicity, we define $(s_{i,j}+1)$ to be $s_{i,j+1}$, unless $s_{i,j+1}$ does not exist, in which case $(s_{i,j}+1)$ is undefined.) Similarly, $\mathscr{M}^{-1}(q) \mapsto (s_{i,j:j+L-1})$ such that $i \in [1, n]$ and $j \in [1, W_i - L + 1]$.

We also define a similarity function $\mathscr{S}(s_{i,j:j+L-1}, s_{q,z:z+L-1})$, that takes as arguments two

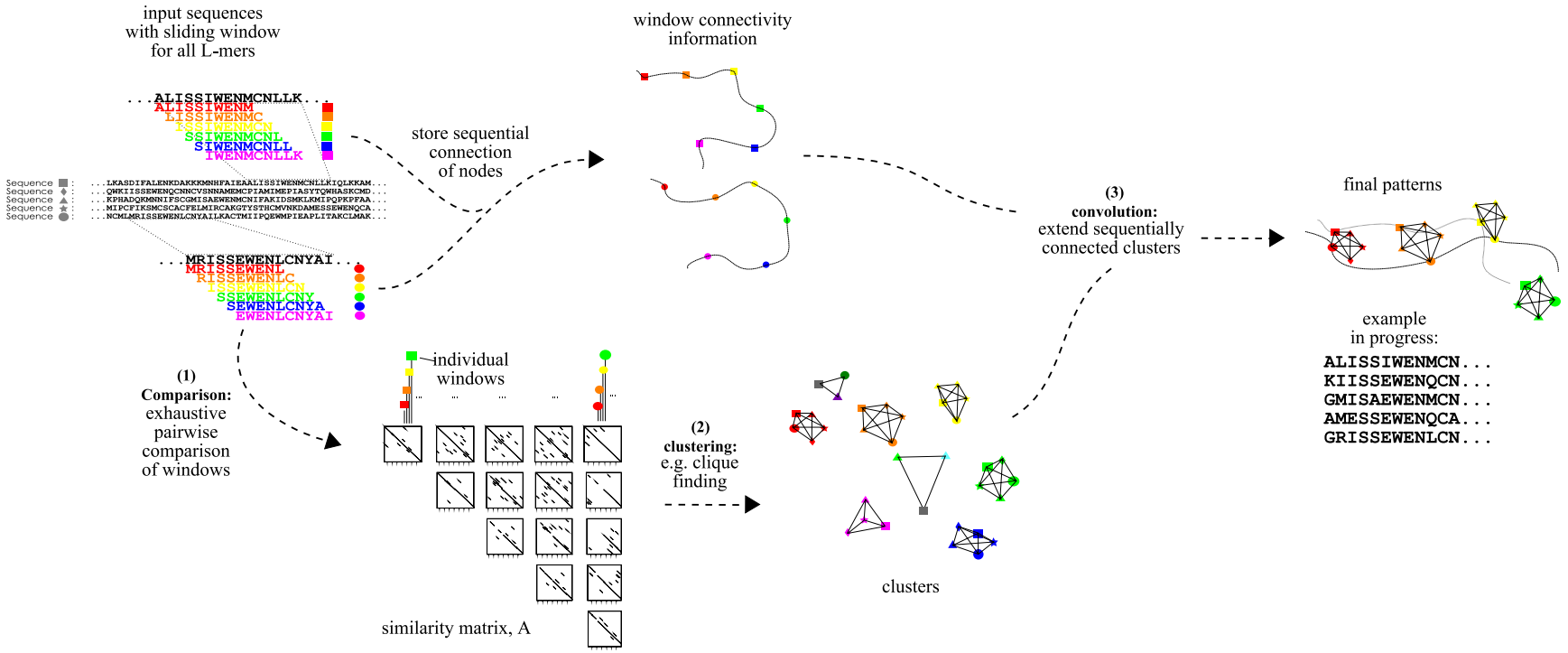


Figure 3-2: A sketch showing the flow of the Gemoda algorithm for an example input set of protein sequences. The various colors in the input sequences are used to indicate the sequential ordering of the L–residue windows. The various shapes are used to indicate a particular window's sequence of origin. (1) In the comparison stage, each window is compared to each other window on a pair–wise basis. Here we show the similarity matrix, A, where the values in the matrix have been thresholded. Those pairs of windows in A that have a similarity score above the threshold are colored black. Note that the graph looks very similar to a standard dot plot. (2) In the clustering phase, groups of windows are clustered together. Here, we show the clusters as cliques, or maximal fully–connected subgraphs in the thresholded matrix A. (3) Finally, these clustered are "stitched" together in the convolution phase using the sequential ordering of the windows to reveal the maximal motifs. A similar process applies for any kind of sequential data analyzed by Gemoda.

arbitrary windows and returns a real–valued number indicating the level of similarity between the two windows. In the most simple case, $\mathscr{S}$ may use the identity matrix to count how many DNA bases two windows have in common; for real–valued data, the function may return the sum–of–squares error between two windows or any other measure of similarity.

We define a motif p as a data structure with two features: a width $\mathscr{W}(p)$ and a list of locations in the data where the motif occurs, $\mathscr{L}(p)$. A motif has the property that the locations in $\mathscr{L}(p)$ meet some predefined clustering requirements (discussed below) based on the similarity function $\mathscr{S}$ for each window of length L within the motif. The support of a motif is equal to the number of its occurrences (or, equivalently, "instances" or "embeddings"), $|\mathscr{L}(p)|$.

We say a maximal motif is a motif which has the following properties:

1. The motif's width cannot be extended in either direction (left or right) without producing a motif with fewer embeddings (i.e., without $|\mathscr{L}(p)|$ decreasing); and
2. The motif is not missing any instances, i.e. $\mathscr{L}(p)$ includes the locations of all instances of the motif.

These two criteria can be summarized qualitatively by stating that a maximal motif is not "missing" any locations and is as wide as possible, and thus it is as specific and sensitive as possible.

Given these explanations and definitions, we can now detail the computations involved in each phase of the Gemoda algorithm. A simple natural–language example illustrating how each phase proceeds is included in the supplementary materials.

3.3.2 Comparison phase

In the comparison phase of the Gemoda algorithm, the sequences are divided into overlapping windows of size L which are then compared to each other in a pairwise manner to produce a similarity matrix, A (see Figure 3-2 on the page before). Formally, $A_{i,j}$ is equal to $\mathscr{S}(\mathscr{M}^{-1}(i), \mathscr{M}^{-1}(j)) = \mathscr{S}(s_{i,j:j+L-1}, s_{q,z:z+L-1})$.

A is then, quite simply, a similarity matrix for all N windows based on the similarity function $\mathscr{S}$. In most cases, $\mathscr{S}$ is commutative (and the A matrix is symmetric); however, this is not a requirement.

Consider the following example. Say we have two DNA sequences — seq–0 = AATTGGCC and seq–1 = GATAGGA — and that we are interested in patterns that are at least $L = 5$ bases long. Also, here, we will consider the sequences as just a series of characters, that is, a one–dimensional feature vector. We will define $\mathscr{F}(A, B)$ to be the Hamming distance: the number of mismatches between string A and string B.

There are 7 windows of size 5 in the sequences:

$$\begin{aligned}
\mathscr{M}^{-1}(0) &= s_{0,0:4} \\
&= \mathsf{AATTG} \\
\mathscr{M}^{-1}(1) &= s_{0,1:5} \\
&= \mathsf{AATTG} \\
&\vdots \\
\mathscr{M}^{-1}(6) &= s_{1,2:6} \\
&= \mathsf{TAGGA}.
\end{aligned}$$

The members of the matrix A are computed as follows:

$$\begin{aligned} A(0,0) = \mathscr{F}(\mathsf{AATTG}, \mathsf{AATTG}) &= 0 \\ A(0,1) = \mathscr{F}(\mathsf{AATTG}, \mathsf{ATTGG}) &= 2 \\ &\vdots \\ A(2,5) = \mathscr{F}(\mathsf{TTGGC}, \mathsf{ATAGG}) &= 3 \\ &\vdots \\ A(6,6) = \mathscr{F}(\mathsf{TAGGA}, \mathsf{TAGGA}) &= 0. \end{aligned}$$

The matrix A is then

$$A = \begin{bmatrix} 0 & 2 & 5 & 5 & 2 & 3 & 4 \\ - & 0 & 3 & 5 & 3 & 1 & 4 \\ - & - & 0 & 2 & 5 & 3 & 2 \\ - & - & - & 0 & 5 & 5 & 3 \\ - & - & - & - & 0 & 4 & 4 \\ - & - & - & - & - & 0 & 4 \\ - & - & - & - & - & - & 0 \end{bmatrix},$$

where the $-$ is used because the matrix is symmetric.

Obviously, depending on the type of sequential data being analyzed, the similarity function should be changed accordingly. However, any kind of data can always be used to produce a generic similarity matrix A, which is the input to the next phase of the algorithm. From this point onward, the algorithm data–agnostic in the sense that subsequent phases act only on A and $\mathscr{M}$ — they are independent of the specific data that produced these structures.

3.3.3 Clustering phase

The purpose of the clustering phase is to use the similarity matrix A to group similar windows into clusters. These clusters will become "elementary motifs" from which the final, maximal motifs will be constructed in a manner similar to the Teiresias algorithm.

We define a clustering function $\mathscr{C}(A) = c^L = \{c_1^L, c_2^L, \ldots, c_Z^L\}$ where each c_i^L is a set of indices in A and $c_i^L[q]$ is the q^{th} member of c_i^L. Note that $\mathscr{C}$ can be any function; common clustering functions include hierarchical clustering, k–nearest–neighbors clustering, and many others. We call each c_i^L an "elementary motif" of length L. We note that a clustering function may assign each node (window) to one or more groups. In the latter case, each c_i^L may have a non–null intersection with any c_j^L. That is, a single window may appear in an arbitrarily large number of clusters.

3.3.4 Convolution phase

The purpose of this phase is to "stitch together" the elementary motifs to generate the final, maximal motifs [207]. For the purposes of Gemoda (and consistent with the above concept

of convolution), we say that a motif h of width $\mathscr{W}(h) > L$ meets the similarity criterion if for each window of length L completely within the motif, all instances participate in a cluster together based on $\mathscr{S}$ and $\mathscr{C}$. In this manner, we can piece together longer continuous motifs from smaller motifs that all meet the similarity criterion over windows of length L.

Next we define the "directed intersection" of two elementary motifs, $c_i^L \curvearrowright c_j^L = c_r^{L+1}$, where c_r^{L+1} is the set of those indices q in c_i^L such that $\mathscr{M}(\mathscr{M}^{-1}(c_i^L[q]) + 1)$ is in c_j^L. That is, c_r^{L+1} is the set of indices in c_i^L that are located, in the sequences S, one position earlier than the indices in c_j^L. c_r^{L+1} is then a motif of length $L + 1$.

We define the operation "$\sqsubset$" as follows: $c_i^L \curvearrowright c_j^L \sqsubset c^{L+1}$ is true if the set of indices $c_i^L \curvearrowright c_j^L$ is a subset or a superset of the indices in any member of c^{L+1}. This operation compares a convolved motif of length $L+1$ to all previously–convolved motifs of length $L+1$ to identify significant overlap: if the list of locations in the proposed motif is a superset or subset of the list for any other motif, the result of this operation is true. With this step, Gemoda can identify and eliminate redundant and non–maximal motifs.

If $c_i^L \curvearrowright c_j^L \sqsubset c^{L+1}$, then all super– or sub–sets of the proposed convolved motifs are removed from c^{L+1}; these windows are then taken together with the proposed motif, and the union of those sets of windows is returned to c^{L+1}.

Our objective is to find all the maximal motifs in the sequence set using the elementary patterns. We do this by performing $c_i^k \curvearrowright c_j^k$ for all i and j at each length $k \geqslant L$ until c^k is empty ($|c^k| = 0$). We then define the set of maximal motifs comprising c^k for all k as P, the final set of motifs that are returned to the user. This simple induction scheme guarantees that all (and only) the maximal motifs are in P given appropriate clustering functions (see supplementary materials).

3.4 Implementation

3.4.1 Choice of clustering function

Gemoda can use any clustering function; however, as the size of the input sequence set increases, storing the matrix A can become practically difficult. In these cases, it can be easier to store true/false values in A, where the value is true if the similarity score between two windows is better than a user–defined threshold g. The matrix A can then be viewed as an unweighted, undirected graph with a vertex for each window and edges between those nodes with pairwise similarity scores better than g (see Figures 3-2 on page 95 and 3-10 on page 113). When constructed as such, we have found that clustering functions based on finding either cliques[1] or connected components (maximal disjoint subgraphs) can be effective for motif discovery in diverse applications.

In the case where the clustering function $\mathscr{C}(A)$ is chosen such that each c_i^L is a clique in the g–thresholded A matrix, the Gemoda algorithm has a guarantee of compositional and length maximality, relative to the threshold g. That is, Gemoda will discover all motifs where each pair of instances has a similarity score better than g over every window of size L, there are no

[1] We define a clique as a maximal, fully–connected subgraph. It may be alternatively defined without the requirement for maximality, thus making the clusters we discuss "maximal cliques". We use the former definition for the sake of brevity and clarity when discussing the maximality of extending motifs.

"missing" instances having this property, and the motif cannot be extended either to the left or right (see inductive proof in the supplementary material).

Clique enumeration is NP–complete [90, 248]; however, in practice this complexity is usually not an issue because the density (the ratio of the number of edges to the number of vertices) of graphs is usually low for datasets of nucleotide or amino acid sequences (with reasonable choice of g).

In the case where the clustering function $\mathscr{C}(A)$ is chosen such that each c_i^L is a maximal disjoint subgraph in the g–thresholded A matrix (i.e., c^L represents the connected components of A), the computational complexity for the clustering phase is significantly less than for clique–based clustering. As well, in the case where Gemoda is applied to nucleotide and amino acid sequences, the motifs from this connected components method may be more intuitive than motifs found using clique–based clustering.

The space and time usage of this implementation is not unreasonable. In most cases, memory usage is not a limiting factor. For instance, the peak memory usage for a large sequence set containing $65,000$ characters is 1 GB, within the reach of many personal computers. Furthermore, the upcoming examples given in this work can all be done in reasonable times. The amino acid sequence example and protein structure example take at most tens of seconds on an average desktop PC, while the hardest of the DNA sequence examples takes two hours. These times are more than reasonable given the exhaustive guarantees provided by the algorithm.

3.4.2 Summary of user–supplied parameters

The input to Gemoda is a set of sequences (categorical or real–valued), a window length, a similarity function, and a clustering function. Various clustering functions may require other parameters. For example, the clique–finding and connected components clustering algorithms discussed above require both a threshold parameter g and, optionally, a minimal support parameter k. Other parameters can be easily incorporated into various clustering functions, such as a "unique support" parameter p that limits returned motifs to those that occur in at least p different sequences.

3.4.3 Availability

We have written open source programs implementing the Gemoda algorithm that are publicly available at the following URL: `http://web.mit.edu/bamel/gemoda`. The software includes a number of "helper" applications for interoperability with common bioinformatics tools. For example, applications are included that allow users to model Gemoda's output motifs (in the case of nucleotide or amino acid sequences) as PSSMs — using the pftools package available via the Prosite database [118] — or as hidden Markov models, using the popular HMMer software [72].

The implementation is distributed in two variants, each with a different comparison stage of the algorithm. The gemoda–s variant is for motif discovery in FastA–formatted text strings, typically nucleotide or amino acid sequences. The gemoda–r variant is used for motif discovery in sets of multi–dimensional, real–valued sequences. The gemoda–s variant is distributed with a number of similarity functions based on various nucleotide and amino acid substitution matrices. The gemoda–r variant is distributed with similarity functions based on the root mean

square deviation, with options for optimal translation and rotation.

The Gemoda software is written in the C programming language and is described in detail in Chapters C and B in the Appendix (page 379). The code is segmented in such a way as to allow the extension of the algorithm to varieties of sequential data that were not anticipated by the authors. Furthermore, where possible the code was crafted to be "object–oriented like" for maximum readability. The software makes extensive use of the GNU Scientific Library [1] and the popular Basic Linear Algebra Subprograms (BLAS) [37, 68, 69] to speed–up computationally intensive operations associated with the discovery of motifs in three–dimensional protein structures and other real–valued data.

3.4.4 Motif Significance

Each pair of nodes in a similarity graph can be described with two different quantities: $\eta_{i,j}$, the number of neighboring nodes (including each other) that the two nodes have in common, and $\chi_{i,j}$, the number of consecutive windows starting from each of those nodes that are connected to each other. For instance, if window 1 is similar to windows 1, 10, 25, and 36, and window 10 is similar to windows 1, 10, 25, and 37, then these two nodes have three neighbor nodes in common and $\eta_{1,10} = 3$. If window 1 is similar to 10, 2 is similar to 11, and 3 is not similar to 12, then there are two consecutive similar windows and $\chi_{1,10} = 2$.

By analyzing each node as above, we can accumulate a matrix of graph statistics, Φ, such that

$$\phi_{i,j} = |\{(x, y) : \eta_{x,y} = i, \chi_{x,y} = j, 0 \leqslant x, y \leqslant N\}| \tag{3.1}$$

(where the vertical bars indicate the cardinality of the set, or the number of ordered pairs) and

$$\Phi_{i,j} = \sum_{a=i}^{\infty} \sum_{b=j}^{\infty} \phi_{a,b} \tag{3.2}$$

These statistics can then be used in the following calculation for $p_{rel}(q, r)$, the relative likelihood of an output motif of length q and support r given the calculated similarity matrix:

$$p_{rel}(q, r) = \binom{N}{r} \left[\prod_{i=0}^{r-2} \left(\frac{\Phi_{i,1}}{\Phi_{i,0}} \right)^{r-i-1} \right] \left(\frac{\Phi_{r,q-L+1}}{\Phi_{r,1}} \right) \tag{3.3}$$

In this equation, the combinatorial factor represents the number of different ways that windows can be sampled in groups of r, the cumulative product represents the necessary conditions for the formation of a clique of length L, and the last factor represents the likelihood of extending a clique of support r to be length q. In this way, the relative likelihood measure attempts to represent the expected number of motifs of length q and support r that would occur at random given the calculated similarity matrix. Notably, this significance is based solely on the similarity matrix A, and so it can be used for either categorical or real–valued sequence data clustered with the clique–finding method.

3.4.5 Proof of exhaustive maximality

When using clique–finding as the clustering function, each elementary pattern of length L is a clique in our similarity graph. That is, the elementary pattern is a set of windows that are all similar on a pairwise basis and there is no other window that can be added to the set.

When the algorithm enters the convolution stage, it starts by convolving each length L elementary motif with all of the others. An elementary motif that is *non–maximal* can be convolved with another elementary motif to yield a motif at level $L + 1$ that has the same cardinality. All such motifs are marked as non–maximal. Those elementary motifs that remain unmarked cannot be extended on either side without losing support; since they are cliques we know they cannot be made greater in cardinality. Thus, all such unmarked cliques of length L can be labeled as maximal motifs and saved for output. In this way, we know that only maximal motifs will be returned to the user, and all such motifs will be returned.

When the "$\sqsubset$" operation is performed on two elementary motifs of length L that are being convolved, it ensures that no identical motifs of length $L + 1$ exist and that no motif of length $L + 1$ is a subset of any other. Additionally, since we have exhaustively compared a complete list of elementary motifs, and all such motifs are cliques with maximum cardinality, we are certain that all possible comparisons between motifs are being made. That is, no unique motifs of length $L + 1$ could be created that are not subsets of motifs created by our exhaustive comparison. Finally, it is important to note that the result of convolving any two cliques will always be a clique. We know this because we take the set of all instances that can be extended (so the subgraph is maximal) and because all instances that are extended were pairwise similar in both windows being convolved (thus meeting our definition of similarity over multiple windows).

Thus, since Gemoda exhaustively generates *all possible* cliques of length $L + 1$, and every added motif of length $L + 1$ is maximal in support, we then know with certainty that c^{L+1} is an exhaustive list of motifs, or cliques, of length $L + 1$. The induction step is then trivial, as setting L equal to $L + 1$ at each step gives an exhaustive list of cliques just as when we started with c^L. This allows for a continual guarantee of exhaustiveness and maximality in output. The obvious termination condition for the algorithm is when $|c^i| = 0$. The pseudocode sketch in 3-3 on the following page faithfully encapsulates the inductive algorithm described above.

3.4.6 Two simple examples

To demonstrate exactly how the algorithm works, we now provide two simple, natural–language examples along with a step–wise narrative of the Gemoda algorithm and demonstrations of how the examples would be run using the software implementation of the Gemoda algorithm provided by the authors and described in Chapter B on page 183.

Example 1: Consider two sequences, ABCDEFG and ABCEFDG, that would be represented with the following Fasta–formatted file:

```
> Sample 1
ABCDEFG
> Sample 2
ABCEDFG
```

```
begin
  n := 0
  while |c^n| ≠ 0 do
      for i := 0 to |c^n| step 1 do
          ismaximal := true
          for j := 0 to |c^n| step 1 do
              f := c_i^n ↷ c_j^n
              if |f| ≠ 0
                if f ⊏ c^{n+1} = false
                    c^{n+1} := c^{n+1} ∪ f
              else
                    choosemaximal(f, c^{n+1})
                fi
                if |f| = |c_i^n|
                    ismaximal := false
                fi
              fi
          od
          if ismaximal = true
              P := P ∪ c_i^n
          fi
      od
      n := n + 1
  od
end
```

Figure 3-3: Pseudo–code for the Gemoda convolution. The figure shows the recursive algorithm used during the convolution stage of Gemoda. The algorithm produces only maximal motifs and discards motifs that are not maximal in support at each level. Subsequent levels progress to motifs of larger length. As discussed in the text, when Gemoda uses a clique–finding clustering phase, the convolution phase guarantees that the algorithm is both maximal *and* exhaustive.

Using a window of length 3, a minimum similarity of 1, a clique–finding clustering method, and the similarity function defined as the identity matrix (the same function described by the reviewer), the command–line argument (for the software implementation of Gemoda provided by the authors) would look something like this:

```
$ gemoda-s -i testSeqs -l 3 -g 1 -k 2 -m identity_aa
```

Given this command, Gemoda finds the maximal motif `ABC..FG`. How this happens is illustrated in Figure 3-4 on the following page.

Windows 3 and 8 have their first letter in common, allowing them to meet the similarity threshold. Windows 4 and 9 have their last letter in common, allowing them to meet the similarity threshold allowing the motif to extended past the letter `D`. In the case of a 2–clique as in this problem, convolution reduces graphically to following diagonal "streaks" of similarity that are not on the main diagonal. This streak is evident in part b of the figure.

Giving the above–mentioned input data and parameters to Gemoda, we get back not only the motif that can be represented as `ABC..FG`, but also two other motifs that may not have been readily obvious. The complete output of Gemoda is as follows:

```
pattern 0:      len=7   sup=2   signif=1.000000e+00
    0    0       ABCDEFG
    1    0       ABCEDFG

pattern 1:      len=5   sup=2   signif=5.000000e+00
    0    1       BCDEF
    1    2       CEDFG


pattern 2:      len=5   sup=2   signif=5.000000e+00
    0    2       CDEFG
    1    1       BCEDF
```

These additional motifs are due to the low similarity threshold; one letter of similarity is sufficient to make three consecutive windows all meet the threshold.

Now consider the same sequences with $g = 2$. As described in eariler, a motif of width $\mathscr{W} \geqslant L$ must meet the clustering and similarity requirements for each pair of L–length windows that is completely within the motif. In this example, since the third and forth pairs of aligned windows, `cde & ced` and `def & edf`, do not meet the criterion of $g = 2$ for a similarity function based on the identity matrix, they are not in elementary motifs that can be convolved. This is illustrated in the following diagram.

```
      abc
      :::   --- pair 1: 3/3            ---
      abc                               |---- maximal motif #1
       bcd                              |
       ::    --- pair 2: 2/3            ---
       bce
```

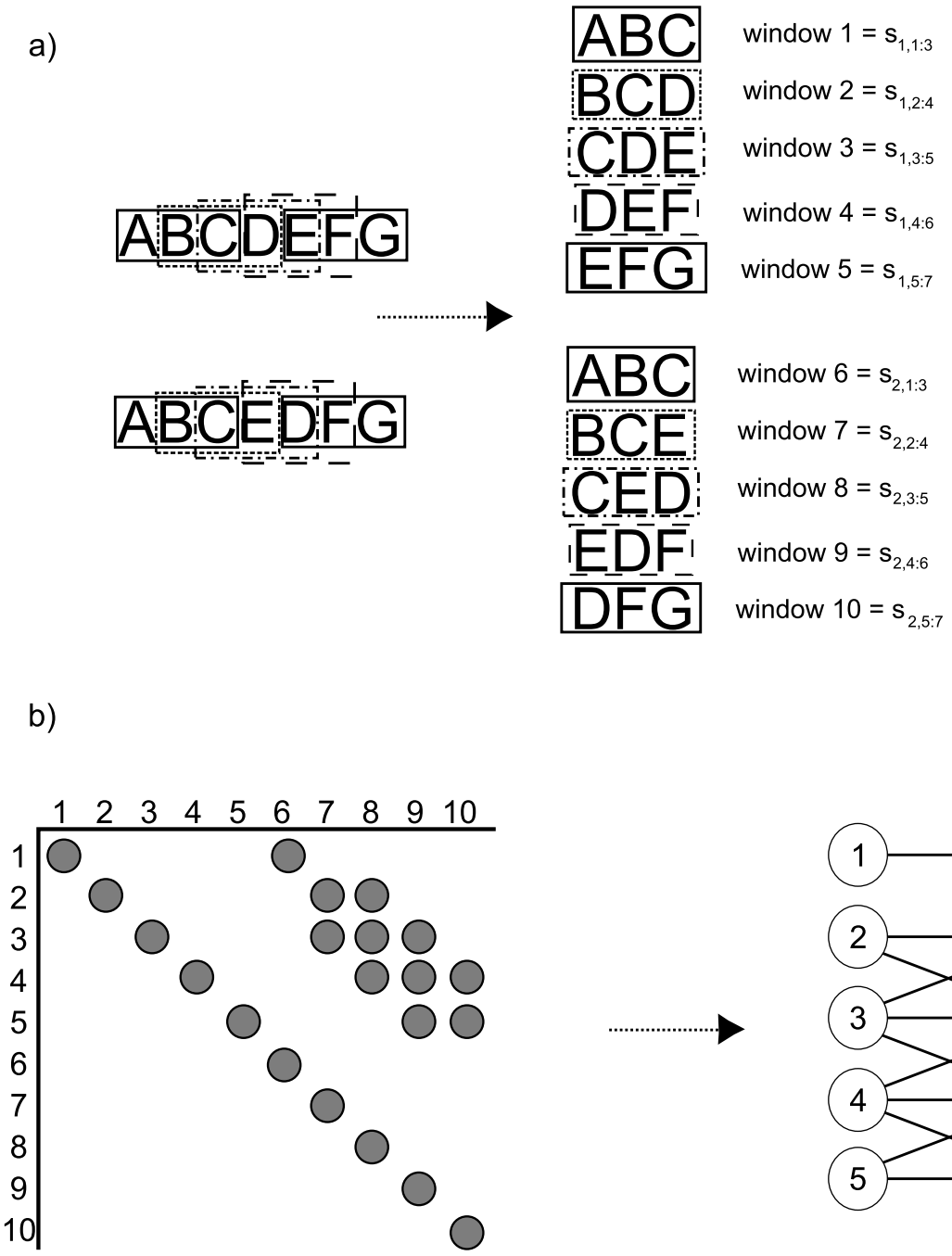


Figure 3-4: A natural language example illustrating the steps that Gemoda takes. In a), we see the three words, or sequences, being broken into overlapping windows of three letters each. Gemoda would then compare each of these windows to each other using either of the similarity metrics described in the text. In b), we see the resulting similarity matrix and how it looks when drawn as a graph. In the matrix, two nodes are similar by the identity metric if there is a dot at their intersection. Making each window a vertex and connecting vertices with an edge if the windows are similar, we obtain the graph on the right.

```
cde
:     --- pair 3: 1/3
ced
 def
   :   --- pair 4: 1/3
 edf
  efg
   ::   --- pair 5: 2/3            ---- maximal motif #2
  dfg
```

As shown, the first two pairs of $L = 3$ length windows, which surpass the $g = 2$ threshold, form elementary motifs and are convolved together. However, because the third pair does not meet the criteria (and thus form an elementary motif) it is not convolved. A similar logic applies to the final two windows. Thus, the final, convolved, maximal motifs in this problem are `abc.` and `.fg`, and `abc..fg` is not a maximal motif motif (with $L = 3$, $g = 2$).

Example 2: Suppose we have a set of three words,

```
MOTIF
MOTOR
POTION
```

and we would like to find the motifs that some of these words share in common. Further, suppose that we are only interested in motifs that are at least four letters long and for which at least three of the four letters are "similar" between the windows. In this example, each word is a sequence, and the parameter L is 4. Thus, there are 7 possible windows that are taken sequentially from the three input sequences, numbered as shown in figure 3-5.

If we choose a similarity function based on the identity matrix with a threshold of three — that is, for two windows to be similar, at least three letters must be the same — then we find that only the following pairs of windows are similar: $(1, 3)$, $(1, 5)$, and $(2, 6)$. Importantly, we note that though window 1 is similar to both windows 3 and 5, windows 3 and 5 are not similar to each other.

If, on the other hand, we choose a similarity function based on a matrix that distinguishes only between vowels and consonants — that is, any vowel is considered similar to any other vowel, and the same goes for any consonant — we would see different results for the same threshold value. In this case, we would find the following set of similarities: $(1, 3)$, $(1, 5)$, $(3, 5)$, $(2, 4)$, $(2, 6)$, and $(4, 6)$.

Given these similarity matrices for the different similarity functions, we can now cluster the graphs. Using the similarity matrix from the identity function, a clique–finding algorithm would find no cliques larger than size 2; that is, the only cliques that exist are the pairs of similar nodes. Since window 3 (`MOTO`) is not similar to window 5 (`POTI`), they cannot be in the same cluster.

However, if we use the similarity matrix produced by the weaker vowel/consonant function, we will find exactly two cliques of size 3: $\{1, 3, 5\}$ and $\{2, 4, 6\}$. Though there exist pairs of

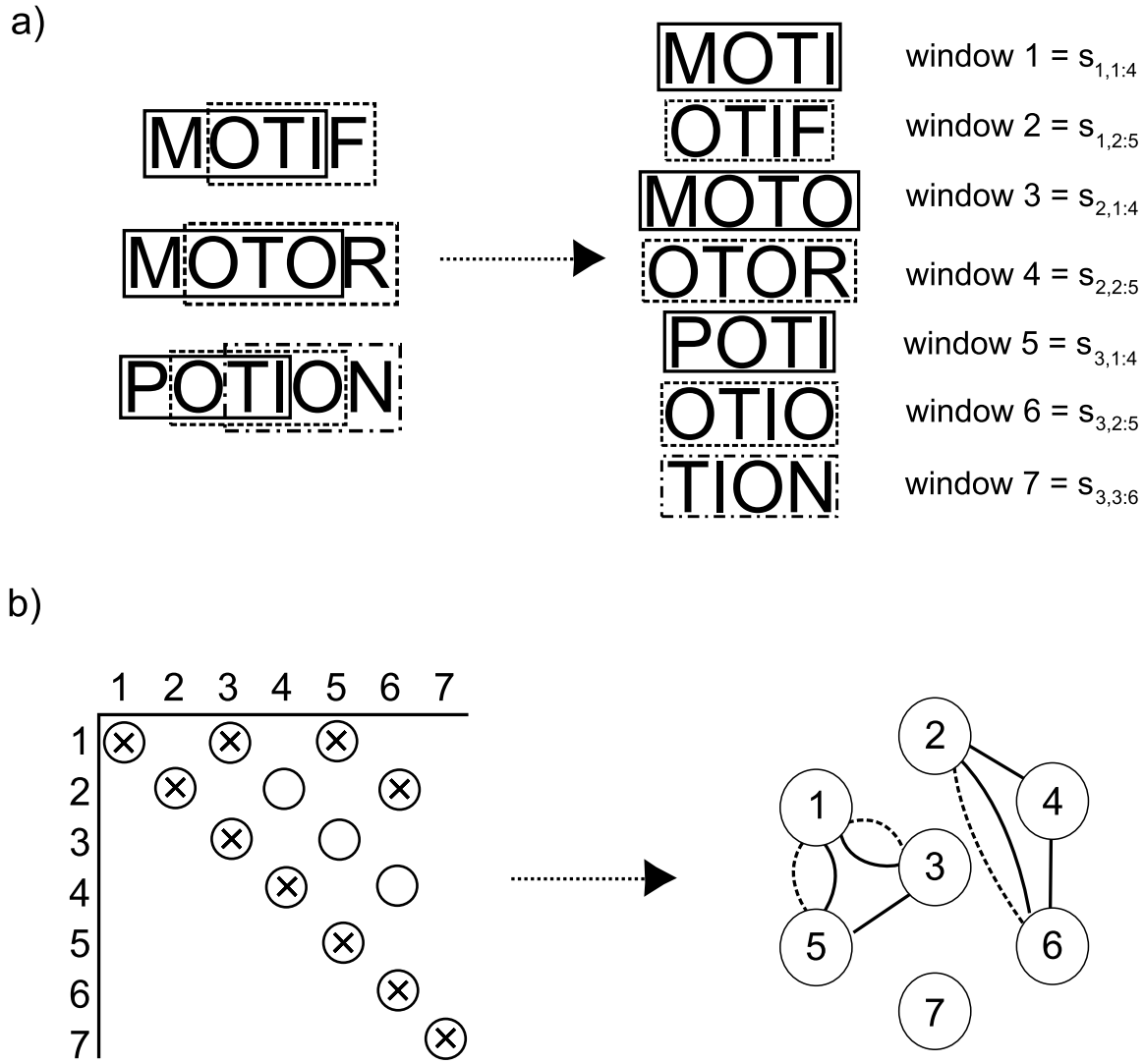


Figure 3-5: A second natural language example illustrating the steps that Gemoda takes. In a), we see the three words, or sequences, being broken into overlapping windows of four letters each. Gemoda would then compare each of these windows to each other using either of the similarity metrics described in the text. In b), we see the resulting similarity matrix and how it looks when drawn as a graph. In the matrix, two nodes are similar by the identity metric if there is an "X" at their intersection, while they are similar by the vowel/consonant metric if there is an "O" at their intersection. Making each window a vertex and connecting vertices with an edge if the windows are similar, we obtain the graph on the right. Dotted lines indicate similarity by the identity metric, while solid lines indicate similarity by the vowel/consonant metric. In this representation, it is clear what the results of both clique–finding and commutative clustering methods will be.

nodes that are similar, none of them is a clique because they are not maximal — that is, each individual pair of nodes that is similar (e.g., $(1, 3)$) can have another node added to its set (5) without violating the pairwise similarity constraint, so only the larger set is a clique.

We also note that applying a connected components clustering function to the matrix created by the identity function would give still different results. In the connected components clustering function, the fact that windows 3 and 5 are not similar would not prevent them from being in the same motif; the function finds all disjoint subgraphs and defines them as the motifs. The motifs for such a case would be $\{1, 3, 5\}$ and $\{2, 6\}$, which we will call motifs c_0^L and c_1^L, respectively.

Finally, we perform the convolution step. Using the last set of motifs described (with connected components clustering and the identity similarity function), we perform the convolution operation on each ordered pair of motifs; in this case, it means performing $c_0^L \curvearrowright c_1^L$, $c_1^L \curvearrowright c_0^L$, $c_1^L \curvearrowright c_1^L$, and $c_0^L \curvearrowright c_0^L$. For the first operation, we find the windows immediately after each of the windows in c_0^L, which is the set $\{2, 4, 6\}$. The intersection of this set with motif c_1^L is the convolved motif of length $L + 1$, which is $\{2, 6\}$; we can call this c_0^{L+1}. In performing $c_1^L \curvearrowright c_0^L$ and $c_1^L \curvearrowright c_1^L$, we note that no windows exist "after" windows 2 and 6, because their respective sequences end. In this case, the first set to be intersected is null, so the intersection is null. The final self–convolution operation also yields a null set. We now have only one motif for the new round of convolution, c_0^{L+1}. Performing $c_0^{L+1} \curvearrowright c_0^{L+1}$ results in a null set, meaning that there are no more motifs. At this point, we terminate convolution. It is worth noting that c_0^L is returned as a maximal motif because window 4 cannot be extended, but c_1^L is not because all of its instances were convolved in one direction.

Thus, we get different sets of motifs for different similarity and clustering functions. For identity similarity and clique–finding clustering, the final list of motifs is {{`MOTIF`, `POTIO`}, {`MOTI`, `MOTO`}}. For identity similarity and connected components clustering, the final list of motifs is {{`MOTIF`, `POTIO`}, {`MOTI`, `M`
For vowel/consonant similarity and either clustering method, the final list of motifs is {{`MOTIF`, `MOTOR`, `POTIO`}}.

3.5 Application

In this section, we demonstrate Gemoda's capability by presenting several sample applications. Specifically, we address motif discovery in amino acid sequences, in nucleotide sequences, and in protein structures.

As discussed previously, the clustering and convolution stages of the Gemoda algorithm are generic — they are independent of the nature of the input data. However, the comparison stage is data–specific. In what follows, we discuss how the comparison stage is changed for each kind of data and outline the types of results Gemoda is capable of finding.

3.5.1 Motif discovery in amino acid sequences

To use Gemoda to find motifs in amino acid sequences, the comparison stage needs to reflect the notion of "similarity" for amino acid sequences. Specifically, we choose a window comparison function $\mathscr{S}$ that returns a sequence alignment score, such as the bit–score from an amino acid scoring matrix (e.g., the popular Blosum matrices [109]).

Here, we demonstrate how Gemoda can be used for motif discovery in amino acid sequences

by "discovering" known protein domains in the (ppGpp)ase family of enzymes. These eight enzymes catalyze the hydrolysis of guanosine 3',5'–bis(diphosphate) to guanosine 5'–diphosphate (GDP) and are classified by the Enzyme Commission (EC) number 3.1.7.2 [21].

We used Gemoda to identify motifs in these eight (ppGpp)ase enzymes using the Blosum–62 scoring matrix as the basis of our similarity function $\mathscr{S}$ and the clique–based clustering function described previously. Specifically, we sought motifs that occurred in all eight sequences, were at least 50 residues long, and had a pairwise bit–score of at least 50 bits over a window of 50 residues.

The sequences for this example are distributed with the source code for the software implementation of Gemoda written by the authors (see Chapter B on page 183). Using the software, this example would be run as follows, assuming the protein sequences are in a file called "spot.fa":

```
$ gemoda-s -i spot.fa -l 50 -g 50 -k 8 -m BLOSUM62
```

With these parameters, Gemoda discovers four motifs in this set of eight sequences; the longest motif, with a length of 103 amino acids, is shown in Figure 3-7 on page 110 as an alignment of the regions that correspond to instances of this motif (see also Figure 3-10 on page 113). A comparison with the known protein domains in the NCBI Conserved Domain Database (version 2.02) [165] reveals that this motif captures the RelA_SpoT domain (CDD PSSM–id 15904).

The remaining three motifs are not present in the CDD database. However, further inspection using the tools available from the PFAM database [25] revealed that they composed the left, middle, and right regions of the HD domain [13]. In the SpoT enzymes, this domain has a number of insertions and deletions that give rise to gaps such that Gemoda identified and reported individually the left, middle, and right regions of conservation of the HD domain.

In this example, the Blosum–62 matrix was chosen as the similarity metric because it is optimized for detecting distant homologs. The Gemoda input parameters $L = 50$ and $g = 50$ were chosen to enforce a one–bit–per–base score, which should rise above random "noise" since, by design, the expected bit–score for two aligned amino acids is negative for the Blosum set of scoring matrices.

In order to test the sensitivity of these results to noise, we conducted an experiment to determine the degree to which these (ppGpp)ase motifs could be found if obscured by noise caused by adding random spurious sequences to the 8 enzyme sequences. We found that, with the Gemoda input parameters described above and using random sequences selected from Swiss–Prot (Release 45.0) [22], the target motifs could be detected in an 8–fold majority of spurious sequences.

3.5.2 Motif discovery in protein structures

The detection of 3–dimensional motifs in sets of protein structures is another problem type that Gemoda can address. Often, homologs that are related through a distant lineage show little to no sequence similarity, particularly at the nucleotide level [73]. However, these homologs

```
>sp|O67012|SPOT\_AQUAE - Aquifex aeolicus.
MSKLGEVSLEEDLEKLLSHYPQHAEEIQRAYEFAKEKHGEQKRKTGEPYIIHPLNVALKLAELGMDHETIIAALLHDTLEDTDTTYEEIKERFGERVAKLVEGVTKIGKIKYKSEQAENYRKLILATAE
DPRVILLKLSDRLDNVKTLWVFREEKRKKIAKETMEIYAPLAHRLGVWSIKNELEDWAFKYLYPEEYEKVRNFVKESRKNLEEYLRKYVIPKVRKELEKYGIEAEIKYRSKHYYSIWEKTRRKGIRLED
VHDILGVRIIVNTVPECYTVLGIIHSLFRPVPGKFKDYISLPKPNLYQSLHTTVIADKGKLVEFQIRTWEMHERAEKGIASHWAYKEGKNPSDAGVYSWLRELVESIQGSTNPSEVLENLKSNLFFEEV
FVFTPKGDLVVLPKGSTPVDLAYKIHTEVGNHCAGAKSNGRIVPLNYELKSGDVVEIITNPNKSPSYEWLSFVKTSRARNKIKQFLKKQERERYLSEGKRILERIREKLGLSHEDLINKIRERVRFDTE
EELLLALGKRKISSANLIKLIFPKKKEEKEERRGSSTVFLEDLSNIKHEVAKCCKPIPGDEILGVITRTKGLVLHEKSCSNLKNVLRLNPEKVKEVQLQASGYFQTDIRVVASDRIGLLSDITKVISES
GSNIVSSMTNTREGKAVMDFTVEVKNKEHLEKIMKKIKSVEGVKICKRLYH

>sp|O51216|SPOT\_BORBU - Borrelia burgdorferi (Lyme disease spirochete).
MIQAYEIAHLIKINDLEKARNIFKKTVENTYKDEFERKSIFKALEIAEQLHYGQYRESGEPYIIHPIMVSLFLAKFQLDFKATIAGLLHDVLEDTNVEKEEIVKEFDEEILSLIDGVTKIHDLHNKTRS
IKEANTISKMFFAMTHDIRIIIIKLADKLHNMTTLSYLPKNRQDRIAKDCLSTYVPIAERLGISSLKTYLEDLSFKHLYPKDYKEIKNFLSETKIEREKKLYKGKLSIEKELQKSGIEAEITVRSKHFY
SIFRKMQTRTNKLTQIFDTLGIRIICKKQKECYEILEIVHRVWKPIPGRLKDYIASPKENKYQSLHTTVRIPEDNQLIEIQIRTEEMDRIANYGVAAHWIYKEQIELKADDLSFINRIKKWQQESANKS
QYSMNDIHKELLNTFIYVYTPEGEVVELPFGSNSIDFAYIIHTDIGDQALYAKINGKISSITKPLKNEQIVEIFTSKDSKPDVIWLNSVRTKKARSKIRSWLNKNDNTIFVDNNIIAYLVGANKEQRKL
FSLFKSYTKTKIKRIAIDPECSPTTGEDIIGIIHKDEIIVHNENCQKLKSYKKPQLIEVEWEATPTRKVHHIILLLKELKGIFSYLENIFTLNDVRLISEKIEDCGNGHGITNIIVSSNAKNITKIISA
LKENPNILQIMQIEEDIKNYDN

>sp|P17580|SPOT\_ECOLI - Escherichia coli, Escherichia coli O157:H7, and Shigella flexneri.
MYLFESLNQLIQTYLPEDQIKRLRQAYLVARDAHEGQTRSSGEPYITHPVAVACILAEMKLDYETLMAALLHDVIEDTPATYQDMEQLFGKSVAELVEGVSKLDKLKFRDKKEAQAENFRKMIMAMVQD
IRVILIKLADRTHNMRTLGSLRPDKRRRIARETLEIYSPLAHRLGIHHIKTELEELGFEALYPNRYRVIKEVVKAARGNRKEMIQKILSEIEGRLQEAGIPCRVSGREKHLYSIYCKMVLKEQRFHSIM
DIYAFRVIVNDSDTCYRVLGQMHSLYKPRPGRVKDYIAIPKANGYQSLHTSMIGPHGVPVEVQIRTEDMDQMAEMGVAAHWAYKEHGETSTTAQIRAQRWMQSLLELQQSAGSSFEFIESVKSDLFPDE
IYVFTPEGRIVELPAGATPVDFAYAVHTDIGHACVGARVDRQPYPLSQPLTSGQTVEIITAPGARPNAAWLNFVVSSKARAKIRQLLKNLKRDDSVSLGRRLLNHALGGSRKLNEIPQENIQRELDRMK
LATLDDLLAEIGLGNAMSVVVAKNLQHGDASIPPATQSHGHLPIKGADGVLITFAKCCRPIPGDPIIAHVSPGKGLVIHHESCRNIRGYQKEPEKFMAVEWDKETAQEFITEIKVEMFNHQGALANLTA
AINTTTSNIQSLNTEEKDGRVYSAFIRLTARDRVHLANIMRKIRVMPDVIKVTRNRN

>sp|P43811|SPOT\_HAEIN - Haemophilus influenzae.
MIARDAHEGQFRSSGEPYITHPVAVASIIAQLHLDHEAVMAALLHDVIEDTPYTEEQLKEEFGASVAEIVDGVSKLDKLKFRTRQEAQVENFRKMILAMTRDIRVVLIKLADRTHNMRTLGSLRPDKRR
RIAKETLEIYCPLAHRLGIEHIKNELEDLSFQAMHPHRYEVLKKLVDVARSNRQDLIERISQEIKVRLENSGIFARVWGREKHLYKIYQKMRIKDQEFHSIMDIYAFRVIVKNVDDCYRVLGQMHNLYK
PRPGRVKDYIAVPKANGYQSLQTSMIGPKGVPVEVHIHTEDMEQVAEMGITAHWVYKENGKNDSTTAQIRVQRWLQSLVEIQQSVGNSFEFIENVKSEFFPKEIYVFTPKGRIVELPMGATAVDFAYAV
HSDVGNTCVGVTVEHKPYPLSKALESGQTVNIITDPNAHPEVAWLNFVVTARAKTRIRHYLKQRCEEDAVKLGEVELNVALQPHNLGDFSIQQIRTVLDALALSSLDELLREIGLGNQSASMIAHQFVG
VPLESANTKNLEFESKILTIAPMQVGKTQFAQCCHPILGDPIVGCCTEKNTVVVHHQHCASLKNACRQSLAKWDNVQSAVNFEAELQIEILNEQNALLSLMTAISASESSLQNIWTEELENNLLLVILQ
VCVKDIKHLANIVHRIKGITGVVNVKRNINEL

>sp|P47520|SPOT\_MYCGE Probable - Mycoplasma genitalium.
MATIQEIECDFLAKIAQKFTNAEIELINKAFYHAKTWHENQKRLSGEPFFIHPLRTALSLVEWNMDPITICAGLLHDIIEDTDQTEANIAMIFSKEIAELVTKVTKITNESKKQRHLKNKKENLNLKSF
VNIAINSQQEINVMVLKLADRLDNIASIEFLPIEKQKVIAKETLELYAKIAGRIGMYPVKTKLADLSFKVLDLKNYDNTLSKINKQKVFYDNEWDNFKQQLKKILAQNQIEYQLESRIKGIYSTYKKLT
VHEQNISKIHDLFAIRLITKSELDCYHILGLIHLNFLIDSKYFKDYIASPKQNLYQSIHTTVRLKGLNVEIQIRTQQMDNVSKFGLASHWIYKEQKEGLLAPALQLNYLVTKQKHSHDFLKRIFGTDII
KINVSASHEPNVIKQINVDSNNKLLDIAFENYPKQFAKLTKIEIDGVEINSFDTSVENEMLIEFYFGKNNNLKSKWIRYMNNPIYREKVKKSLAKLAKSGRYSELAFYEKELGEKQLKLASETEIQKRL
NTLRIKKMSDYLALIECTNFTNDEHLLFLAKNNDKWNKLTKPLKFAFSKVVFHNSYFEQIEGIFITKIVIEPCCSKIPDMPEQVTGILTKNILSVHRYGCKNLQNKKQLKIIPLYWNIQQLKLKPRKFR
SYININGVWSEKTINKICQTIINGDGYIEKIIPKINKQKDEFDLNITLFVNNYQQLLTLMDQITTKNISFSWKYL

>sp|P75386|SPOT\_MYCPN Probable - Mycoplasma pneumoniae.
MFYNWLKLYKFSKMATLVEIERDFLQKTAQKFAPEVVALITKALDYSKKWHGEQKRLSGEPFFIHPLRTALRLVEWNMDSNTVCAGLLHDIIEDTQVTEADLTAIFGKEITDLVVKVTKITSESKKQRQ
LNRKKEDLNLKSLVNIAMSSQQEVNALVLKLADRLDNISSIEFLAVEKQKIIAKETLELYAKIAGRIGMYPVKTQLADLSFKVLDPKNFNNTLSKINQQKVFYDNEWGNFKKQLEEMLEQNQIEYRLES
RIKGIYSTYQKLTFHEQNIAKIHDLFAIRLIVKSELDCYHLLGLIHLNFTVLMKHFKDYIASPKQNFYQSIHTTVRLKGLNVEIQIRTQRMDHVSKYGFASHWIYKEKKEGLLASALQVNYLNSKQMHS
RDFFKRIFGTDIIKVNVSSDNEPNIVKKLNVESNSKLLDIAYELYPKQFNKLEKIKLDGVEVMSFDVTAENEMVIEFCFGKTNNLKRRWLRYMNNHVFRERVKKDLNKLKKAVKYSELPLYEKALEELH
LKLADETQIKQRLNALGIKKLTEFLELIEYPHFPKNEHLYFLASNNQKWRELIKPIKFALSQAVFQNSYFEQIEGIYITKIVIETCCTKIPDMPEQVIGILMKNILRVHLHDCRELANQKQPKIIPLYW
NAHQLKMRPRKFRCQINIRGVWSETTVNKIVQTIIEGDSYLERIIPKIDKQKDEFELNITMFIDNYHQLITIMEQITTKNISYVWKYL

>sp|O34098|SPOT\_SPICI - Spiroplasma citri.
MDRDIKYEEVLAQIKLYIKDEATLKEIQKAYEYAEEKHHGQVRNSGARYIIHPLWTTFFLAQWRMGPKTLIAGLLHDVLEDTPATFEELQELFGIEIANLVEGVTKVSYFAKENRTQIKAQYLRKLYLS
MAKDIRVIIVKLADRLHNLKTIGYLKPERQQIIARESLEIYSAIAHRLGMKAVKQEIEDISFKIINPVQYNKIVSLLESSNKERENTINQKIEELKKILITEKKMSVKVYGRSKSIYSIYRKMNQFGKN
FDDIHDILAVRIITNSVDDCYKVLGFVHQHYTPLNNRFKDYIATPKHNLYQSLHTTIVADDGLIFEVQIRTEEMDELAEQGVAAHWRYKEGENYDIAKKQKDIDERLDIFKRILDLENISVQERDEIQQ
EVYKPDHLMEQIIQNDIFSSLIYVLTPNGKVVTLPFGSTVLDFAYKIHSEIGEKTIGAKINGLFSPISTVLKSGDVVDIKTAATQKPNHSWLVVSKTSSALEKIKKYLKKELVEVTSDAKSVNLEKIKQ
TKSQIEEYIAKKDLKWKLVNSETQLERLHAINFNNIEDFLLDVANDEYTLEEAINLVYLDHETSQNEKILKKLQDKQYKKAQLKDDIIVQGISNIKVVISQCCLPIPYEDITGYVSKAEGIKVHLKTCR
NIQSGDKQDRQVEVSWNEAVCKNKQYDCAIRIEAIDRPALLVDVTKVLSHLNASVQMMSANVSGDLMNLTIKTIIKVSNADRLQQIRSSLLTIPDIKVVERVMM

>sp|P74007|SPOT\_SYNY3 Synechocystis sp. (strain PCC 6803).
MNAVAALPTPTIHTTCAQDIHDIELPQWLEDCLQQWQREIEQGQDETTAPHCLICRAFCFAYDLHAQQRRKSGEPYIAHPVAVAGLLRDLGGDEAMIAAGFLHDVVEDTDISIEQIEALFGEETASLVE
GVTKLSKFNFSSTTEHQAENFRRMFLAMAKDIRVIVVKLADRLHNMRTLDALSPEKQRRIARETKDIFAPLANRLGIWRFKWELEDLSFKYLEPDSYRKIQSLVVEKRGDRESRLETVKDMLRFRLRDE
GIEHFELQGRPKHLYGIYYKMTSQDKAFEEIYDIAALRIIVESKGECYRALSVVHDVFKPIPGRFKDYIGLPKPNRYQSLHTTVLGLTSRPLEIQIRTEEMHHVAEYGIAAHWKYKESGGSENATLTST
DEKFTWLRQLLDWQSDLKDAQEYVENLKQNLFDDDVYVFTPKGEVISLARGATPVDFAYRIHTEVGHHMKGARVNGQWLGVDTRLKNGDIVEIVTQKNSHPSLDWLNFVVTPSARHRIRQWFKRSRRDE
NILRGRELLEKELGKTGLEALLKSEPMQKTAERCNYQNVEDLLAGLGYGEITSNSVVNRLRENNVNNVKNSQSSQEVTLASSPQVHPPTPPATGKDNSPIAGIEGLLYHIAGCCHPLPGEPIMGVVTRG
ARGISIHRQGCHNLEQMDGDRLIPVRWNPNTNNHQTYPVDIVIEAIDRVGVLKDILSRLSDNHINVRNADVKTHLGRPAIISLKIDIHDYQQLLGIMAKIKNMSDVMDLRRVISG
```

Figure 3-6: Guanosine–3',5'–bis(diphosphate) 3'–pyrophosphohydrolase ((ppGpp)ase) (Penta–phosphate guanosine-3'–pyrophosphohydrolase) sequences. These eight enzymes catalyze the hydrolysis of guanosine 3',5'–bis(diphosphate) to guanosine 5'–diphosphate (GDP) and are classified by the Enzyme Commission (EC) number 3.1.7.2 [21].

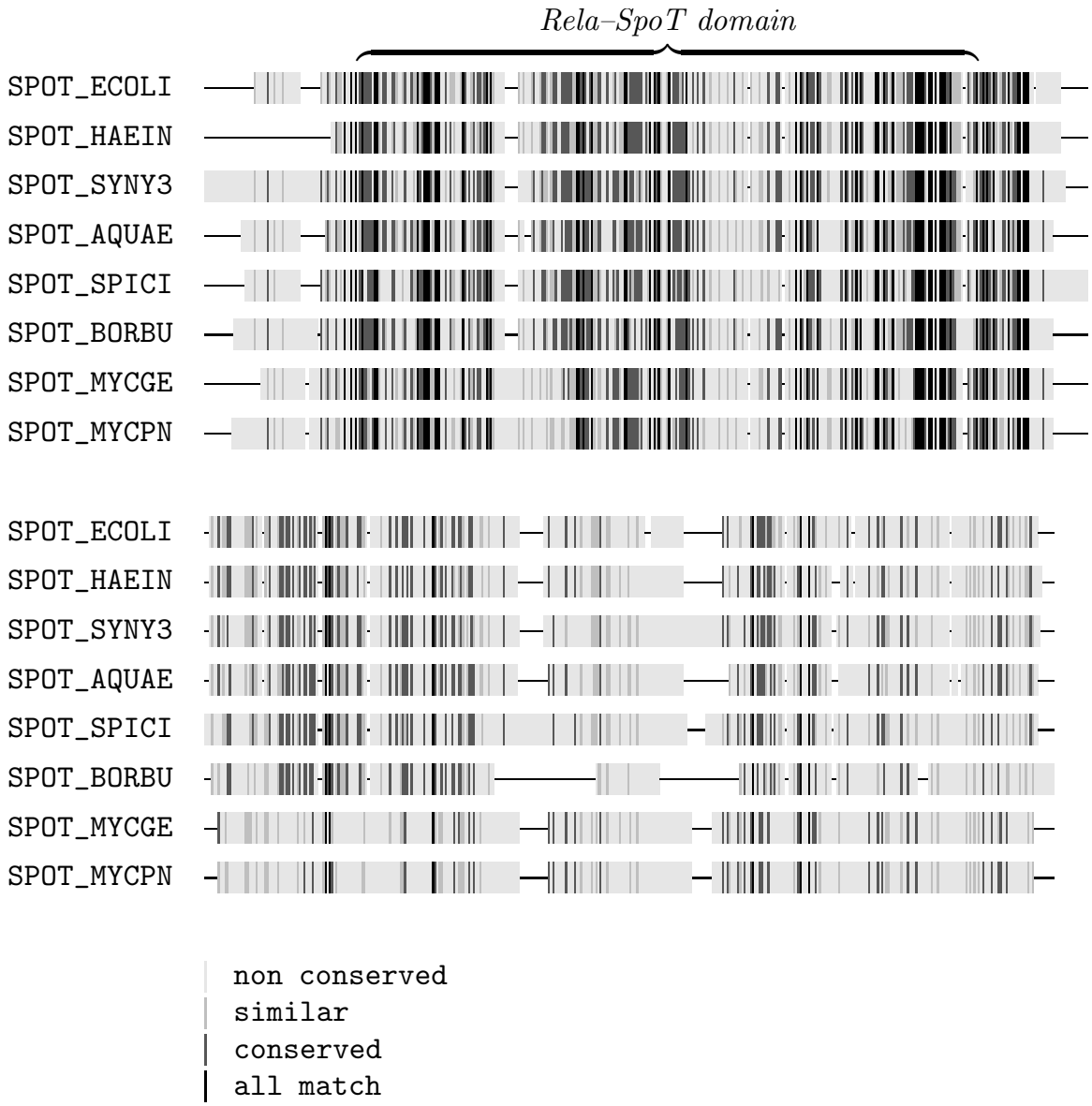


Figure 3-7: The RelA_SpoT motif detected in the 3.1.7.2 enzyme sequences.

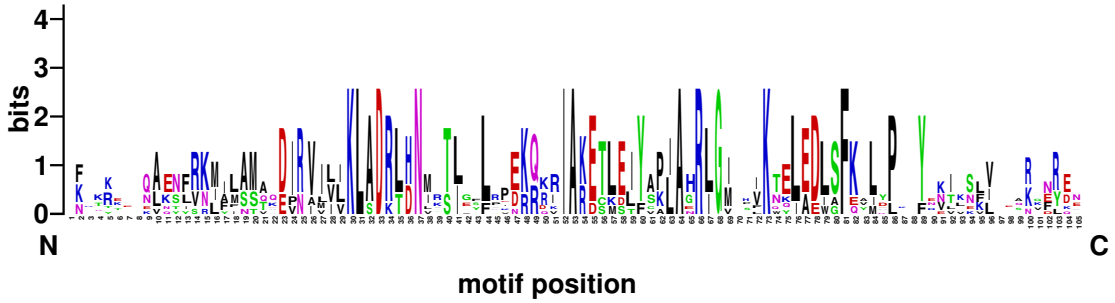


Figure 3-8: Logo representation of the RelA_SpoT motif detected in the 3.1.7.2 enzyme sequences. In this figure, the horizontal axis represents the position in the motif shown in Figure 3-7 on the preceding page, in the vertical axis represents the information content at each position.

frequently show conserved tertiary structures [66], making motif discovery in protein structures often revealing in situations where there appears to be no similarity at a sequence level.

There are a number of well–developed tools for the pair–wise comparison of protein structures or the comparison of a single protein structure to precomputed structural motifs; these have been reviewed elsewhere [73]. Some of the more popular tools include SSAP [185], VAST [160], Dali [120], and Mammoth [187]. The Gemoda algorithm, when used for structural motif discovery, is most similar to the Sarf algorithm [4, 5] and, to a lesser degree, algorithms by [125] and [133]. Conceptually, Gemoda could be thought of as a hybrid of the Sarf and Teiresias algorithms, combining 3–D elementary motif discovery with convolution. To the best of our knowledge, Gemoda is the only tool that can compare an arbitrary number of protein structures simultaneously and produce an exhaustive set of maximal motifs.

To discover motifs in protein structures, Gemoda compares L–residue windows of the proteins' alpha–carbon trace using the minimized RMSD similarity metric (one of many possible metrics for comparing protein sub–structures [144]). Here we use "minimized" to indicate that the protein structures are optimally super–imposed via rigid–body rotation and translation [15, 122]; occasionally this term is implicit. Using the clique–finding clustering algorithm, Gemoda finds motifs that are sets of alpha–carbon traces (in a set of protein structures) that can be super–imposed with an RMSD less than g Å over each window of L residues on a pair–wise basis. Similar to the amino acid and nucleotide applications of Gemoda, these structural motifs are maximal in both length and support.

Here, we demonstrate how the Gemoda algorithm can be used for structural motif discovery by "discovering" the structural homology between the human galactose-1-phosphate uridylyltransferase (PDB id 1HXQ) [266] and fragile histidine triad proteins (PDB id 3FIT) [152], originally reported elsewhere [121]. Using Gemoda, we looked for motifs of at least 30 residues, occurring in at least three chains, that had a pairwise RMSD of 1.5 Å or less (based on superposition of the alpha–carbon backbone) over each window of 30 residues.

This search returns 4 motifs, the longest of which is 66 residues (see Figure 3-12 on page 115). This motif has one embedding in the 3FIT protein and two, in different chains, in the 1HXQ protein. As shown in the figure, the motif is an alpha helix followed by a beta sheet.

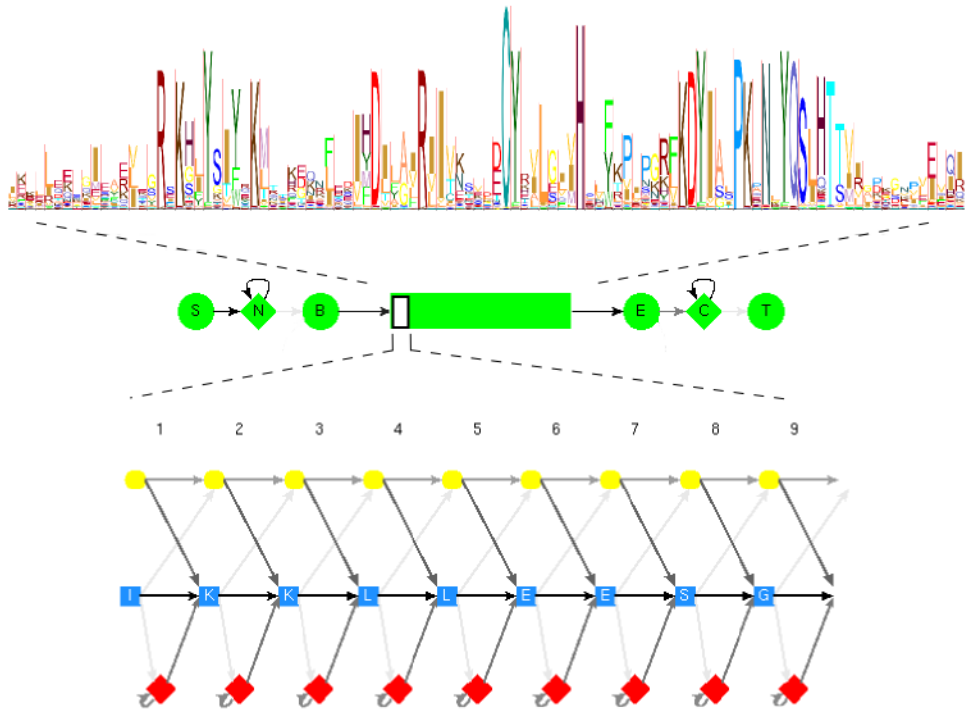


Figure 3-9: Hidden Markov model representation of the RelA_SpoT motif detected in the 3.1.7.2 enzyme sequences. In this figure, the boxes represent the different possible Markovian states at the first few positions in in the motif shown in Figure 3-7 on page 110 [220].

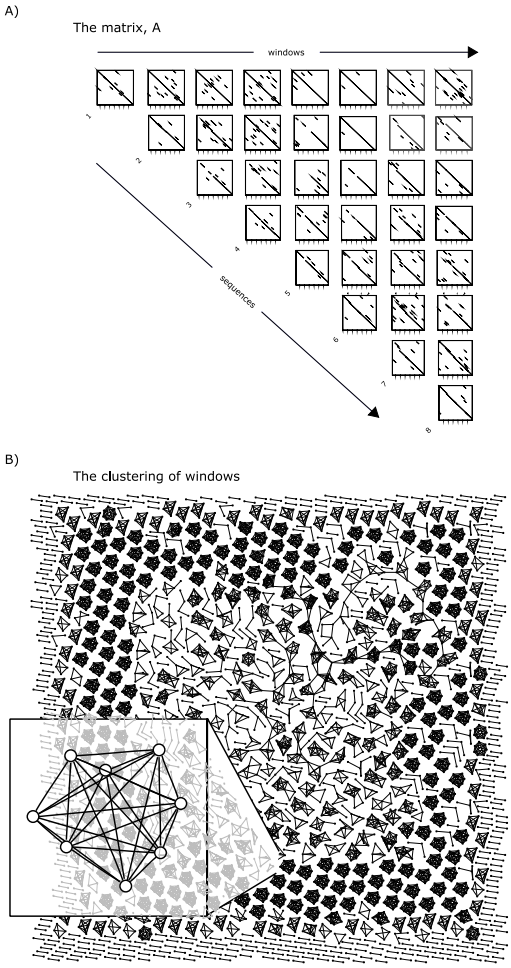


Figure 3-10: The similarity graph for the 3.1.7.2 enzyme example. (A) is the similarity matrix A, which contains one row and column for each window of 50 residues in the set of input sequences. Entries in the matrix have been thresholded such that pairs of windows that can be aligned with a bit–score greater than 20 are given a black dot and all others are white, producing the familiar dot–plot appearance of the matrix. (B) is a graph representation of A. Each vertex represents a window, and two vertices are connected with an edge if they have a black dot in the top image. The breakout shows a clique of size eight, which represents a set of windows that participate in the motif shown in Figure 3-7 on page 110. In general, as the bit–score threshold is lowered, the number of edges in the graph increases, making the clustering stage more computationally intensive. When using clique–based clustering with too small of a threshold, computational expense may make the problem infeasible. At these thresholds the "signal" cannot be distinguished from the "noise." However, with the parameters used in this example, the clustering phase is quite easy, which is intuitive given the number of disjoint subgraphs shown in the bottom image.

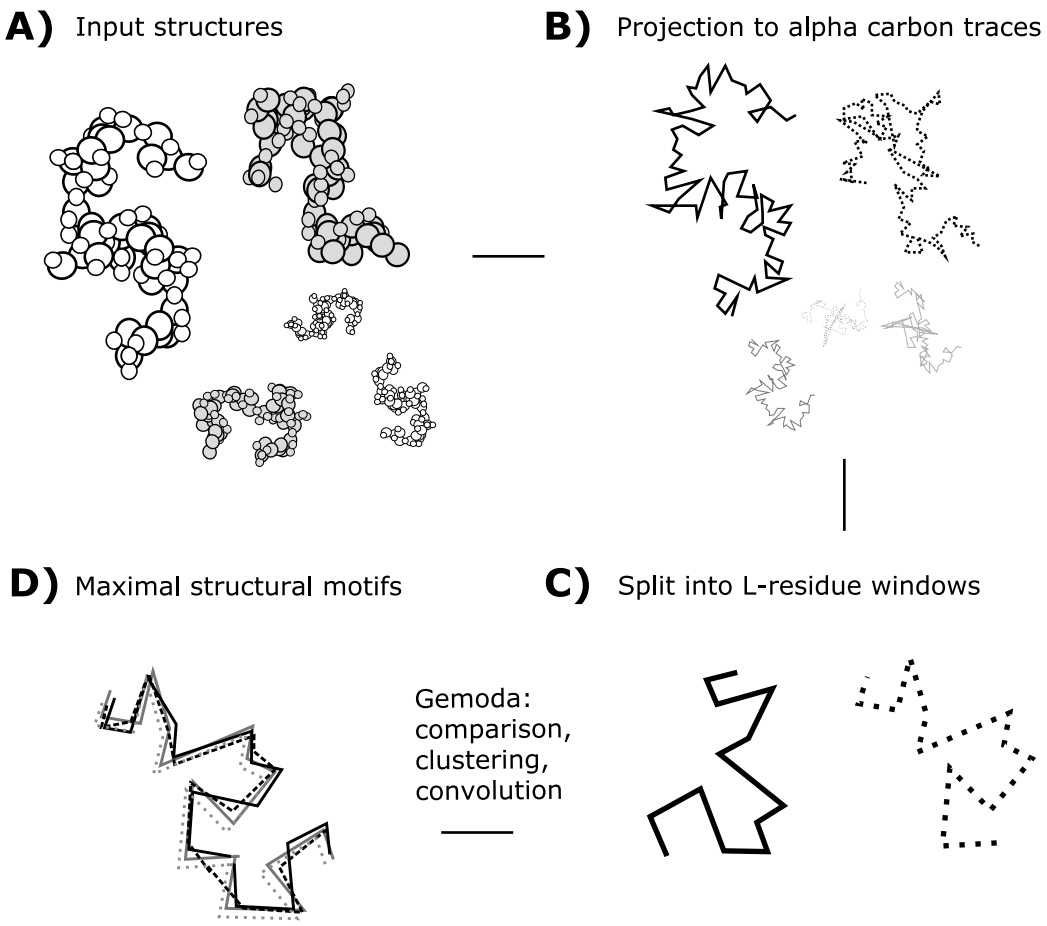


Figure 3-11: Alpha carbon trace projection used by Gemoda

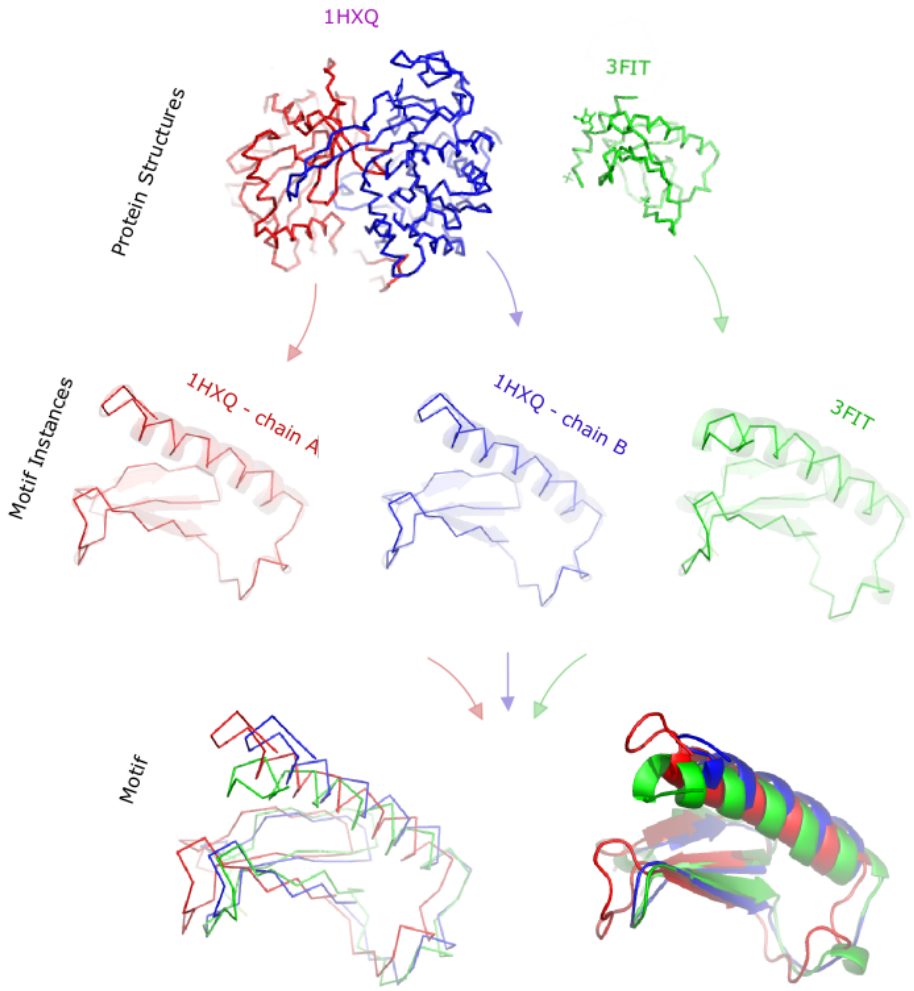


Figure 3-12: A motif showing structural conservation between the human galactose-1-phosphate uridylyltransferase and fragile histidine triad proteins originally reported by Holm and Sander [121]. The motif, as shown here, was "discovered" using the Gemoda algorithm along with three other, smaller, structural motifs that are highly conserved between the two proteins. Notably, the proteins show little sequence similarity over the region displayed in the structural motif above. Graphics created using PyMol (DeLano Scientific, San Carlos, CA, USA). See also Figure 3-13 on the following page.

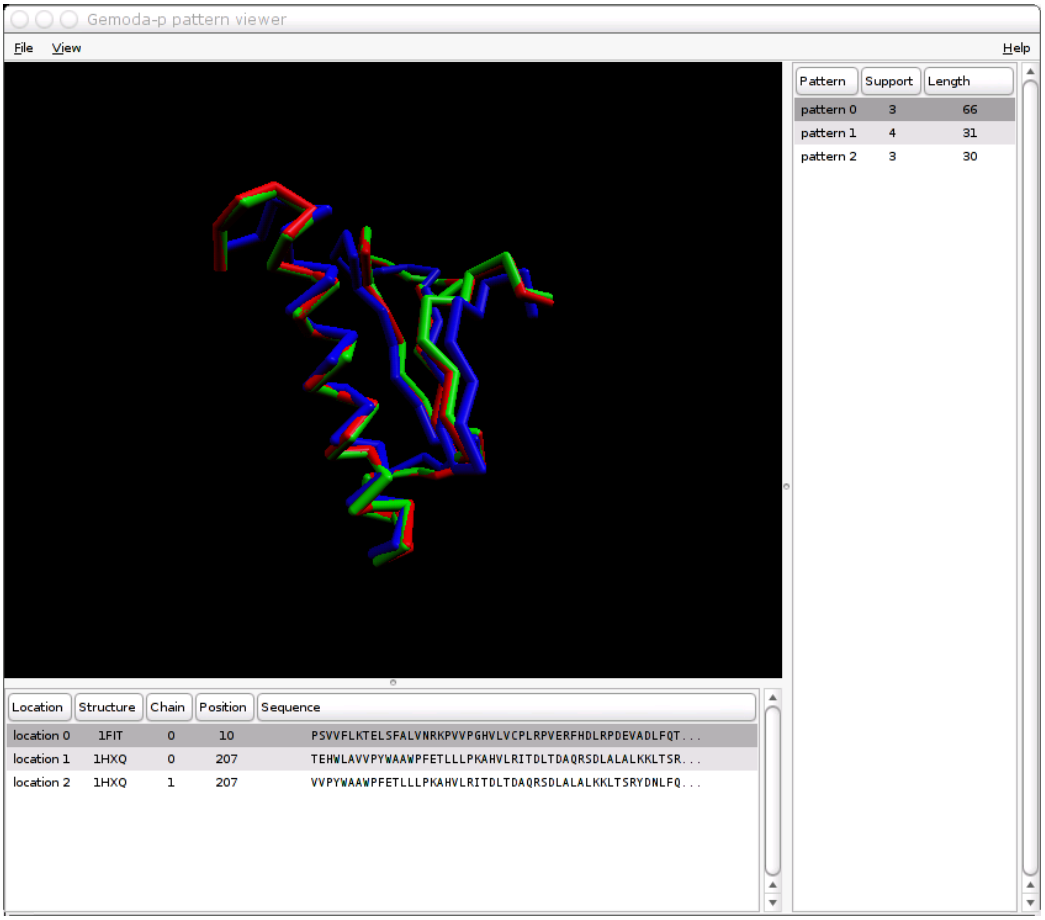


Figure 3-13: The human galactose-1-phosphate uridylyltransferase and fragile histidine triad structural motif (see Figure 3-12 on the preceding page) in Gemoda's 3–D structure viewer, which was written by the authors for viewing output from gemoda–p.

3.5.3 Motif discovery in nucleotide sequences and the (*l*,*d*)–motif problem

Introduction

Four years ago, Pevzner and Sze [195] noted that despite significant advances in pattern discovery, there were still gaping holes in our ability to identify and enumerate frequent patterns in biological sequences. Experimental noise and error were not the only significant issues, as the community was still incapable of solving certain problems with purely synthetic data and no worry of experimental or gross error. One such problem, defined below, was the (*l*,*d*)–motif challenge problem; it exposed the fact that certain motifs, despite having a strong consensus and being rather unlikely to occur at random in independent and identically distributed (i.i.d.) sequences, are extremely hard for most motif discovery algorithms to locate. The reason that these motifs are hard to locate is that even though they may deviate very little from a consensus sequence, their pairwise deviation tends to be rather large. Other false pairwise similarities are thus extremely likely to occur at random elsewhere in the dataset, and this random noise obscures the true motif's signal. Pevzner and Sze [195] presented two algorithms that looked towards solving this problem; Buhler and Tompa [45] followed suit by presenting a more effective algorithm. However, the problem is still not completely solved per se; difficulties exist in obtaining the correctly refined motifs and instances even for this simplified model of biology. In addition, though existing algorithms move towards solving this simplified problem, they are not nearly as helpful in addressing the biological realities that computational biologists face.

The original (*l*,*d*)–motif problem [195] can be paraphrased as follows:

> Within a set of random DNA sequences with i.i.d nucleotides, a parent motif of length l is embedded in each sequence in a random location. Each time the motif is embedded, it is mutated in d locations. The (*l*,*d*)–motif problem is to recover the locations of the embeddings, knowing only the parameters l and d and that each sequence contains exactly one instance of the motif.

At first, this seems to be a reasonable simplification of the phenomenon of binding sites and other functional sites in DNA. It is not uncommon to have some ancestral sequence from which each motif occurrence is some short evolutionary distance away. This model accurately captures the difference between instance–instance similarity and instance–ancestor similarity. That is, even though a motif instance may be a very short distance from its ancestor (say, four mutations out of fifteen bases), any two instances of the motif may be significantly different from each other (eight mutations out of fifteen bases). This low degree of instance–instance similarity can occur rather frequently in random i.i.d. nucleotide sequences, thus obscuring the true evolutionary relationship of the motif instances (the signal) with purely random relationships of background nucleotides (the noise) [45, 46].

As discussed by Buhler and Tompa [45], local search methods (such as the common ones mentioned before) using typical initialization strategies encounter an insurmountable amount of noise when searching for some sparse motifs described by the (*l*,*d*)–motif problem. We would ideally like to be able to recover such motifs, since they are expected to occur by chance in every sequence with rather low probability (approximately 10^{-7}) [45, 46].

In a more realistic scenario, a researcher may not know the size l of the motif *a priori*. Instead, it is more likely that she would know the evolutionary distance between motif instances,

i.e. the rate of mutation d/l. It is also unrealistic to mutate the embedded motif *exactly* d times; rather, the researcher is more likely to be interested in motifs that are d or fewer mutations away from each other. That is, in a real–world senario, we would more likely have a reasonable estimate of the upper limit d/l of the mutation distance between embedded motifs. There may also be multiple, different motifs in the dataset. Finally, as experimental data are commonly rife with noise, it is likely that some of the sequences may be false–positive candidates for the motif; that is, some sequences may contain no motifs at all.

With these issues in mind, we define an extended (*l,d*)–motif problem as follows:

> Within a set of random DNA sequences with i.i.d. nucleotides, a parent motif of length $\geqslant L$ is embedded zero or more times in each sequence in a random location, such that the motif has been embedded a total of k times in the data set. Also, each time the motif is embedded it is mutated such that there are no more than d mutations over any window of l nucleotides (that is, the rate of mutation is d/l). This process is repeated for any number of parent motifs, each with the same l and d, but possibly different L. The extended (*l,d*)–motif problem is to recover the locations of the embeddings for every parent motif without any *a priori* knowledge of where they might be, but only knowing the parameters l and d.

We will refer to this formulation as the "extended" (*l,d*)–motif problem and the previous formulation as the "restricted" (*l,d*)–motif problem. In what follows, we detail an algorithm for solving both the extended and restricted (*l,d*)–motif problems.

We say that a motif, p, is just a data structure with two features: a width, $\mathscr{W}(p)$, and a list of locations in the data where the motif has been embedded, $\mathscr{L}(p)$. A motif has the property that the locations in $\mathscr{L}(p)$ are all within a Hamming distance of $2d$ from each other over every window of size l.

We will call the Hamming distance function $\mathscr{H}$, where $\mathscr{H}$ takes two windows of size l from our sequence set, and returns a real–valued number equal to the number of characters that differ between the two windows.

Solving the restricted (*l,d*)–motif problem

The input set for the (*l,d*)–motif problem is any arbitrary set of n sequences, each with length W_i nucleotides. Most bioinformatics literature treatments use $W_i = 600$ and $n = 20$. Different versions of this problem have been discussed at length; the most commonly discussed is the (15,4) problem, while the (14,4) and other associated, more difficult problems are also addressed in the literature.

It has been shown before that the most commonly used motif discovery algorithms, including CONSENSUS [115], Gibbs sampling [147], and MEME [19], are unable to solve the restricted (15,4) problem. Algorithms that are capable of solving the restricted (15,4) problem have been presented in the literature. While some of these, including Winnower and SP-STAR [195], are unable to solve the more complicated (14,4) problems, others are able to address this and other, more difficult, problems with some degree of accuracy. These latter algorithms usually leave the deterministic realm, though, and rely on probabilistic methods to find the planted motifs.

On the other hand, our algorithm allows for exhaustive, deterministic solution of these problems. The (*l*,*d*)–motif problem solved by the above–mentioned tools is a degenerate case of the extended problem that our algorithm was designed to solve. Thus, our algorithm is not optimally tuned for solving the restricted (*l*,*d*)–motif problem in the least amount of time. Nonetheless, solving a range of the restricted (*l*,*d*)–motif problems is still a valuable check on the utility of our tool to make sure it can solve at least some of them in a reasonable amount of time. In addition, our exhaustive search allows for one to see how many other false signals are in the data. This can facilitate the assessment of statistical significance of results, certainly an important step in analyzing any proposed signal.

Our algorithm requires three user input parameters: l, g, and k. l is the minimum motif size and the size of the sliding window used for judging similarity between two sequences. g is the similarity threshold for any two windows to be deemed instances of the same motif; in this case, if two windows of length 10 are a Hamming distance of 2 away from each other, g would need to be 8 or less for the windows to be in the same motif. Finally, k is the support, or minimum number of motif occurrences required to report the motif to the user.

It is obvious that any two motifs of length l each being mutated d times from an ancestral sequence can differ at most at $2d$ locations. Thus, at least $(l - 2d)$ locations must be preserved in the motif. This observation lays the foundation for discovery of the hidden motifs. Our algorithm is run with parameters $l = 15$, $g = 7$, and $k = 20$ for the (15,4) problem. The discovery of the motif is then a straightforward combinatorial problem with deterministic discovery of the solution.

It is important to note, however, that our method will solve and return a superset of the restricted (*l*,*d*)–motif problem. That is, any group of d–mutants from a common ancestor can be described as having $(l - 2d)$ identical bases, but not all groups of sequences with $(l - 2d)$ identical bases can be used to synthesize an ancestor from which all group members deviate $\leqslant d$ bases. When there are a large number of "signal" motif members, there is usually sufficient overall deviation to prevent a $\geqslant d$–mutant from joining a motif group. However, at smaller support k, it is more likely to find motif instances that violate the d–mutant constraint. It is not desirable to immediately remove motifs with such members from the output, as they do still meet the constraints imposed by our parameter values; rather, we can use a simple post-processing method to note which motifs have readily obvious ancestors and thus are the most likely candidate signals.

A few interesting observations can be made regarding the complexity of the algorithm and the quality of its solutions. First of all, the time to solution is not affected directly by the length of the motif to be discovered as in many other exhaustive methods. Rather, it is the sparseness or subtlety of the motif (or more accurately, the probability of the pairwise motif similarity occurring randomly) that has the most profound impact on the complexity of the algorithm. The most computationally expensive step is the clique-finding function, which increases in computation time with the number of edges (np–complexity at worst, though on average much better). For varying l and d, as two l-mers sampled randomly from the background are more likely to meet the threshold of similarity defined by l and d, there will be more false edges (similarities) in the graph, and thus the clustering algorithm will take longer. Motifs of widely different length may be (approximately) equally likely in the background distribution if d is set to a certain value for each. In this case, it would take almost exactly the same amount of time to find both motifs in the same input set. Of course, the size of the data set also has a

significant impact on computation time, as for any algorithm; a larger input set causes more false occurrences of a potential motif, and the resulting distance matrix needs more time to be explored by our clique-finding algorithm.

Also, our method does not preclude discovery of more than one instance of a motif in any given sequence. Much like the re–framing of the (*l,d*)–motif problem presented above, this is more reflective of what one expects may happen in a real biological system: motifs of biological significance may occur more than once in a biosequence, and it behooves us to be able to discover all occurrences. In fact, in the original dataset for the (15,4)–motif problem used by Pevzner and Sze [195], there is actually an additional instance of the original motif that occurred completely by chance; this instance was discovered in our solution of the problem. With Gemoda, we can easily identify this instance without any additional work or manipulation. The sequence logo for the planted motif from Pevzner and Sze's initial dataset is shown in Figure 3-14 on page 122; the consensus sequence is GGCTTTGTAGCTAAC. The "accidental" instance of the embedded motif that can be identified using Gemoda is GGATTGATAGCTAAG.

Finally, it is important to note the absolute accuracy of our results. In previous papers presenting algorithms to solve the (*l,d*)–motif problem, a metric called the performance coefficient is used to gauge the accuracy of the algorithms. This is defined as $\frac{K \cap P}{K \cup P}$, where K is the set of $l * s$ nucleotides representing the s motif instances each of length l and P is the set of $l * s$ nucleotides representing the s proposed motif instances of length l. Coefficients above .75 are usually deemed acceptable for these algorithms. Improved algorithms return results with coefficients of about 0.9 or 0.95. Examples of the performance of other algorithms are presented in Table 3.1. Clearly, our algorithm returns all coefficients of 1; that is, it will return the exact location of all motif occurrences. This is a notable improvement over other algorithms that may return approximate motif locations that then need to be verified and slightly adjusted or optimized by hand. In fact, in any given run of PROJECTION (the most accurate of the algorithms in Table 3.1), one will usually find that one or two (or even more) of the returned motif instances are not just imperfectly located, but are false positives.

The computation time of our tool becomes unacceptable as the motifs become degraded beyond the (15,4) problem. This is to be expected for a deterministic algorithm as the probability of the signal reaches a level that causes many pairwise similarities to occur by chance. Since our strategy is generalized and exhaustive, we expect the computation times to be suboptimal. Beyond this table, one would benefit from other probabilistic or heuristic algorithms in order to solve the more difficult (*l,d*)–motif problems in an acceptable period of time. Fortunately, it seems to not be a too frequent occurrence to search for a (18,6)–motif in each of 20 biological sequences, so our algorithm should be of significant utility for common applications.

Solving the extended problem

Of course, in a real biological problem, one does not have nearly the same certainty in the contents of each biosequence as is allowed by the (*l,d*)–motif problem. This becomes evident upon analyzing the situations that the (*l,d*)–motif problem is meant to analyze, the most salient of which being the discovery of transcription factor binding sites. In order to come up with the candidate coregulated sequences, the results of laboratory experiments are analyzed to find which genes are sufficiently coexpressed. However, much of this data is prone to noise. Some genes may not be coexpressed, though they may seem to be due to some experimental aberra-

tion. Of those that are actually coexpressed, they may or may not be coregulated by the same transcription factor; it is a distinct possibility (and quite frequently a reality) that genes appearing to be coexpressed are not bound by any common factor. The same analysis follows for other situations for which the (l,d)–motif problem is an otherwise reasonable approximation: experimental noise prevents certainty that all input sequences are truly.

Other methods meant to be robust enough to solve the restricted (l,d)–motif problem will lose significant advantage in this more realistic, extended set of circumstances. Our algorithm was designed specifically to deal with the issues addressed by the extended challenge problem. It discovers, in a provably exhaustive and deterministic fashion, all motifs described in the extended problem definition. Other algorithms discussed previously in this paper are just not constructed to deal with such uncertainty in motif characteristics; as such, there is little way to accurately compare the performance of ours and other algorithms on the fully extended problem. Thus, it seems intuitive to simplify the extended problem to something more complicated than the restricted (l,d)–motif problem, but for which there is still a useful metric for comparison between ours and other algorithms. What follows are two cases (discussed qualitatively) which demonstrate the specific benefits of our tool for pattern discovery on (15,4) problems beyond the restricted version.

Case 1: An underestimated number of motif instances. One source of difficulty in the extended problem may be the uncertainty as to the exact number of motif instances. For this case, we still restrict ourselves to windows of size l with d mutations from a consensus sequence. However, we allow for uncertainty in the number of motif instances. For this case study, we instruct algorithms to find motifs with instances in at least 15 sequences when in fact there is an instance in every sequence. If an algorithm such as WINNOWER were to search for cliques across 15 sequences when in fact all 20 sequences had a motif instance, it would have a final graph with much more than the single signal that it usually hopes to obtain. PROJECTION's attempts to find 15 instances when 20 actually occur are similarly problem-ridden, returning different candidate motifs on different runs. These results would sometimes have significant overlap with initial planted motif, though at other times would have very little overlap. Most disturbingly, all of these proposed motifs would have approximately the same score, thus making it difficult to discern a truly useful motif from one constructed from background noise. Our algorithm, on the other hand, returned the initially planted motif along with other smaller patterns that still met the criteria for classification as a motif.

Case 2: Zero-or-one motif instances. In this next case, we analyze the impact of there being zero or one motif instances in each sequence. To implement this simplification, we instruct each algorithm to find the exactly 15 motif instances that are implanted across 20 sequences. This makes the problem astonishingly similar to the (l,d)–motif problem, with the exception that not every sequence contains a motif instance. This problem setup is thus significantly more realistic, as one does not expect every sequence to have a motif occurrence in every pattern discovery problem. Of course, this is still a simplification of reality, as one would not expect to know the exact number of motif instances. However, not even this gross simplification can salvage the efficacy of existing algorithms for the discovery of such subtle motifs. A study using PROJECTION found results that rarely approached acceptable levels and more frequently approached performance coefficients expected from purely random guessing. Again, though,

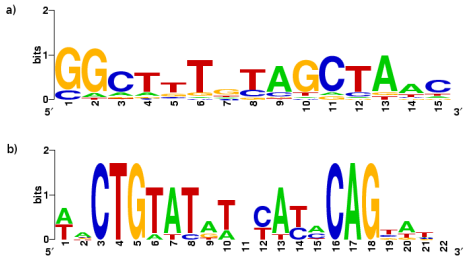


Figure 3-14: The sequence logo for a) the motif implanted in each sequence for the (*l*,*d*)–motif problem and b) the LexA binding site motif generated from the highest–scoring motif returned by Gemoda.

our algorithm solved the problem with only a small increase in computation time over solving the original (*l*,*d*)–motif problem.

Identifying natural cis–regulatory elements

For some regulons in *E. coli* with mild to strong consensus sequences, Gemoda returns results that are similar to or improve upon the results from commonly–used motif discovery tools. For instance, using the set of upstream regions (400 base pairs upstream and 50 base pairs downstream of the translation start site) for the 9 operons believed to be regulated by LexA [217], Gemoda's top–scoring motif was used to generate the sequence logo found in Figure 3-14. This motif closely matches the literature PWM for the LexA binding site and represents 80% of the literature–found binding sites with no false positives. Of course, the difficulty of DNA motif discovery problems varies greatly, and this is only one straightforward example of such problems.

The parameters used for this search were $L = 20$, $g = 10$, and $k = 6$ with the identity matrix scoring scheme and clique–based clustering described above. The length was selected based on the knowledge that the DNA–binding domain of LexA is a helix–turn–helix variant, and so it was likely to be a relatively long motif. The similarity threshold was chosen as one–half of L, which we know from the (*l*,*d*)–motif problem ought to be approximately sufficient to prevent the graph from being too dense (and thus expensive to cluster). The support threshold was chosen to be about two–thirds the total number of sequences, allowing for some noise in the data. Of course, the judicious selection of parameters is an outstanding problem in binding site discovery. It is worth noting that most of these selections were simple or intuitive and that there was some tolerance in the results for slight perturbations in parameters.

Conclusions

The benefit of our proposed algorithm is then obvious: deterministic and provably complete output even in the face of uncertainty in motif characteristics. The motifs could have been longer than 15 bases, could have had fewer mutations, or could have occurred in a variable number of sequences, and our tool would have found them. Its only obvious negative aspect is its computational expense. The restricted (15,4) problem took 6 hours, while the extended problem took 13 hours. Compared to the runtimes of algorithms like PROJECTION, which

can be as low as five minutes for the restricted problem, these runtimes may seem extremely large. In practice, however, this computation time is far from unacceptable; one would not expect to often encounter the need to run motif discovery many times sequentially, particularly if the results being returned to the user are deterministically correct.

Perhaps even more importantly, we have reframed the challenge problem statement in a way that is more biologically meaningful; hopefully this new challenge will inspire other methods that outperform ours in some way. While a deterministic and exhaustive method is always welcome, for some problems it seems that a heuristic approach may provide a good balance between time and accuracy; we look forward to seeing new tools that address our amended problem with sufficient accuracy.

3.6 Discussion

Gemoda makes four contributions. First, the algorithm is generic in that it is equally applicable to any variety of sequential data. Second, Gemoda allows arbitrary similarity metrics. In the examples shown here, we chose relatively simple metrics (scoring matrices and RMSD–base metrics); however, similarity metrics can be easily changed or added. For example, in the case of amino acid sequences, one can easily define hybrid metrics incorporating primary, secondary, and tertiary structure features. In the case of nucleotide sequences, the metric may be changed to incorporate methylation information. The third contribution is that Gemoda returns motifs that are not tied to any particular motif representation. In the case of amino acid sequence motifs, it is easy to model Gemoda's motifs using regular expressions, hidden Markov models, or position–specific scoring matrices. Finally, when used with the clique–finding clustering algorithm, Gemoda returns an exhaustive set of maximal motifs. To the best of our knowledge, Gemoda is the only motif discovery algorithm incorporating the above features.

As mentioned in the introduction, Gemoda integrates the best characteristics from a number of previously published motif and association discovery algorithms. For specific problems, Gemoda's performance can be improved further, though at the expense of generality. For example, a window sampling approach such as that used by Blast [11] would be useful in applications where speed is more important than completeness of results. For protein structure comparisons Gemoda could also be altered to use contact maps like those used by Dali [120]. The convolution stage could also be made faster by using heuristical, non–exhaustive convolution methods. Also, the clustering phase could be expedited by using approximate clique finding methods.

The lack of an underlying model in Gemoda is a major strength, as this facet of the algorithm allows exhaustive enumeration of motifs that is difficult for methods using complex motif representations. In addition, this aspect of Gemoda makes comparing nucleotide sequences just as easy as comparing real–valued data, like gas chromatography–mass spectrometry (GC–MS) datasets, which may follow different motif models (Styczynski *et. al., in preparation*).

One weakness may be that the Gemoda algorithm does not natively employ iterative steps for motif discovery. In that sense, the algorithm is similar to Teiresias [207] and MITRA [78]. However, because it employs a user–defined scoring metric (and clustering function) there is nothing to prevent such iteration *per se*. For example, the output motifs from a run of Gemoda could be used to recompute a refined scoring function. Using amino acid substitution matrices, this would be in the spirit of the method used to compute the Blosum [109] matrices from the

Blocks database [113].

Futhermore, the Gemoda algorithm could be modified to find gapped motifs. Gemoda is capable of finding gapped motifs in which the gap length is fixed and small relative to the size of the flanking conserved regions. However, motifs with larger, variable length gaps cannot be detected natively by Gemoda. In this respect, Gemoda is similar to MEME [19], Teiresias [207], and Block Maker [113]. Other tools, including Consensus [115] and the Gibbs sampler [147], have been altered from their original formulation to account for gaps.

It may be possible to alter the convolution step to allow for large or variable–length gapped motifs. Another option is to look for maximal motifs whose offsets are highly correlated. Our studies indicate that such *post hoc* analysis of Gemoda's output can usually find well–conserved gapped motifs, including those with variable gap lengths, as was the case for the (ppGpp)ase example.

Gemoda's generic nature makes it readily applicable for many problems. In the protein sequence application, Gemoda's exhaustive search using a scoring matrix as a similarity metric identified multiple motifs. It provided an accurate representation of these domains in as much as an eight–fold excess of spurious sequences. In the DNA motif discovery application, Gemoda identified an otherwise unintentional result in a synthetic dataset and satisfactorily described a motif embedded in a genomic dataset. In the protein structure application, Gemoda demonstrated that it can compare multiple arbitrary–dimensional structures simultaneously and return results previously shown in the literature. Gemoda can also be directly applied to other diverse types of sequential datasets, or it can be extended to address problems not yet considered.

Table 3.1: Performance on a range of (l,d)–motif problems with synthetic data. Data from other algorithms are from Buhler and Tompa [46]. GibbsDNA, WINNOWER, and SP-STAR are averaged over eight random instances, while PROJECTION is averaged over 100 random instances. Computation times for our proposed algorithm are averaged over three random instances.

l	d	GibbsDNA	WINNOWER	SP-STAR	PROJECTION	Proposed algorithm	Time
10	2	0.20	0.78	0.56	0.80	1.00	8 min
11	2	0.68	0.90	0.84	0.94	1.00	$<$ 1 min
12	3	0.03	0.75	0.33	0.77	1.00	10.5 h
13	3	0.60	0.92	0.92	0.94	1.00	10 min
14	4	0.02	0.02	0.20	0.71	1.00	$>$ 3 months
15	4	0.19	0.92	0.73	0.93	1.00	6 h
17	5	0.28	0.03	0.69	0.93	1.00	3 weeks

Chapter 4

Other exercises in motif discovery

4.1 Introduction

The previous two chapters of this thesis were focused on unsupervised methods for motif discovery, with an emphasis on grammatical models. Specifically, in Chapter 2 I demonstrated how regular grammars can be used to model and design novel antimicrobial peptides. In Chapter 3, I addressed some of the weaknesses of grammar–based motif discovery tools by developing a new approach that is generic in the sense that it is applicable to many different kinds of sequential data and is model agnostic. In this chapter, I continue this trend away from the core issues of grammar–based motif discovery, to examine many closely related topics and different approaches to motif discovery.

The first section of this chapter describes the development of an efficient tool for matching regular grammars against large databases of sequences. The topic of the second section is the evolution of amino acid scoring matrices over time. As described in Chapter 3, these matrices are the most common metrics of protein similarity and are used widely by motif discovery and sequence alignment programs. The next section describes how these scoring matrices and sequence alignment programs can be co–opted for solving nontraditional bioinformatics problems such as handwriting and voice recognition. Finally, the last two sections are devoted to exercises in motif discovery that do not use grammatical methods, but instead rely heavily on classical machine learning techniques and simple statistical analyses.

4.2 Biogrep: a tool for matching regular expressions

4.2.1 Introduction

As more genomes are sequenced and annotated, increasing numbers of functional DNA and protein sequence motifs (or *patterns*) are being discovered. These motifs can be used to detect remote homologies that are missed by sequence alignment tools such as Blast [10] and FastA [194]. Many databases such as Prosite [118], PRINTS [17], and BLOCKS [108] contain collections of biologically significant patterns that are correlated with the function of protein families and are expressed as regular grammars, or equivalently, regular expressions (see Section 1.4 on page 33). For example, the Prosite the motif `[AG]....GK[ST]` is indicative of

ATP/GTP binding proteins.

Searching for such regular expressions can be an important part of sequence annotation. There are a variety of tools available for pattern–matching, the most common being the "grep" family of Unix tools, including a number of very fast and sophisticated variants such as agrep [275] and NR–grep [178]. Also, there are many excellent bioinformatics–specific pattern–matching tools including Patscan [71], tacg [164], and fuzzpro [206]. However, all of these tools are optimized for searching for single patterns, that is, one–at–a–time.

Biogrep is a pattern–matching tool designed to match large pattern sets (100+ patterns) against large biosequence databases (100+ sequences) in a *parallel* fashion. This makes biogrep well–suited to annotating sets of sequences using biologically significant patterns.

4.2.2 Implementation and results

Biogrep is written in the C programming language using the GNU regular expression [104] and POSIX threads (pthreads) [173] libraries. The program reads query patterns from either a plain text file, one–per–line, or from a Teiresias–formated pattern file [207] (see Section 1.5 on page 45). These patterns are treated as POSIX extended regular expressions and are searched against a user supplied file, which can be either a FastA–formatted biosequence database or any text file.

Table 4.1 shows a comparison of Biogrep with a few common programs. The grep family of pattern matching tools are absent from the table because their run times are extremely long. This is because many of these tools cannot take sets of patterns and have to be used on a per pattern basis. The next best alternative to Biogrep is a simple PERL script split between multiple processors.

Table 4.1: Performance of Biogrep matching all the 1333 patterns in Prosite (release 17.01) against the 782370 protein sequences in Swiss–Prot/TrEMBL [22] (release as of 8 July 2002). Runs were carried out on an IBM p670 eserver running AIX 5L with 8 Power4 processors.

program	# processors	execution time (s)
biogrep	1	8683
biogrep	2	4477
biogrep	4	2266
biogrep	6	1620
perl	1	11780
perl	6	1916
patscan	1	28466

Biogrep has a number of user options, which are described in the documentation that comes with the software. Most importantly, Biogrep can divide the pattern–matching task between a user–specified number of processors using threads. This drastically reduces the user–time required to match large sets of patterns (see Table 4.1). In addition, Biogrep is distributed with detailed documentation, numerous examples, and various helper–scripts for interfacing with

other pattern matching/discovery programs. The Biogrep source code is available at `http://web.mit.edu/bamel/biogrep.shtml`.

4.3 The evolution of updated BLOSUM matrices and the Blocks database

4.3.1 Introduction

As I discussed in Chapter 3, amino acid substitution matrices are a very common way to measure the degree of similarity between protein sequences (see for example Section 3.5.1 on page 107). Indeed, the fidelity of amino acid sequence alignment and motif discovery tools depends strongly on the target frequencies implied by the underlying substitution matrices. The BLOSUM series of matrices, constructed from the Blocks 5 database, is by far the most commonly used family of scoring matrices. Since the derivation of these matrices, there have been many advances in sequence alignment methods and significant growth in protein sequence databases. However, the BLOSUM matrices have never been recalculated to reflect these changes. Intuition suggests that if the Blocks database has changed — by the growth or addition of blocks — that matrices computed after these changes may be different than the original BLOSUM matrices.

Here we show that updated BLOSUM matrices computed from successive releases of the Blocks database deviate from the original BLOSUM matrices. At constant re–clustering percentage, later releases of the Blocks database give rise to matrices with decreasing relative entropy, or information content. We show that this decrease in entropy is due to the addition of large, diverse families to the Blocks database. Using two separate tests, we demonstrate that isentropic matrices derived from later Blocks releases are less effective for the detection of remote homologs, and that these differences are statistically significant. Finally, we show that by removing the top 1% large, diverse blocks, the performance of the matrices can largely be recovered.

This work is part of a manuscript that is currently under consideration. The manuscript was co–authored with Mark Styczynski, Isidore Rigoutsos, and Gregory Stephanopoulos. Throughout this section, the use of the pronoun "we" refers to these authors.

4.3.2 Motivation

Many different scoring matrices have been proposed in the literature, but the BLOSUM series [109] and PAM series [65] of matrices are by far the most widely used. For a review of the many different substitution matrices, the reader is referred to articles by Henikoff and Henikoff [110, 112] and Vogt et al. [258]. Despite the vast array of matrices available, a single matrix, BLOSUM62, has become a *de facto* standard — it is the default matrix for popular pairwise sequence alignment tools such as BLAST [10] and FastA [194] and multiple sequence alignment tools such as Clustal–W [245] and t–coffee [183].

The BLOSUM series of matrices was constructed in 1992 from Blocks 5 [109]: a database of protein blocks, or highly conserved protein regions, derived from families in the PROSITE database [118]. These blocks were used as a training set to derive a set of implied target frequencies that dictate the frequency with which an amino acid of one type should be aligned with an amino acid of another type. The various members of the BLOSUM matrix family — BLOSUM100, BLOSUM62, BLOSUM50, etc. — were made by clustering the sequences in each block at various thresholds, effectively down–weighting similar sequences to create matrices

optimized for aligning more distant homologs.

The Blocks database is itself used for homology searching [111, 197] and other functions [181, 214]. As such, it is periodically updated, with ten major releases in the past ten years and some minor releases. Intuition suggests that these improvements in the Blocks database may make it a better training set for creating scoring matrices. The goal of this manuscript is to show the effects of updates to Blocks on the matrices derived from the database.

When the BLOSUM matrices were initially created and published, it was hypothesized that the use of more protein groups (and thus more blocks) in the matrices' creation would have little effect on the matrix [109]. This was supported by the removal of specific blocks, or even half of the blocks, yielding approximately the same matrices. However, in retrospect it is obvious that the known protein motifs in 1992 are a small fraction of those cataloged in today's databases. Furthermore, it is plausible that motifs discovered "early" were inherently biased due to experimental methods and likely not representative of nature as a whole. It is unclear whether new, more recent blocks would yield identical, similar, or significantly different matrices.

In the following sections, we detail the construction of updated BLOSUM scoring matrices from successive releases of the Blocks database and describe the results of two sequence alignment tests used to evaluate the performance of these matrices.

4.3.3 Methods

Matrix construction

All previous versions of Blocks databases were taken from the Blocks ftp server, `ftp://ftp.ncbi.nih.gov/repository/blocks/unix/`. BLOSUM matrices were constructed using a version of the BLOSUM source code (available from the above FTP server) originally used to prepare the BLOSUM family of matrices, but with some slight modifications and bugfixes reported elsewhere [129]. These changes included fixing integer overflows in multiple locations and fixing the weighting of substitutions between clusters of sequences. For each version of the Blocks database, a full scan of all integer–valued reclustering percentages between 20 and 100 was performed (Figure 4-1). The matrix for each Blocks release with relative entropy closest to the originally reported BLOSUM62 matrix (0.6979) was selected as the representative matrix for that release.

Sequence datasets

Two different database searches were used to judge the ability of each matrix to detect homologs: a search of SWISS–PROT 22 [23] using a set of queries previously determined to reflect "difficult" searches that are able to distinguish the abilities of different matrices [110], and a search of the ASTRAL database [42] using each member as a query. These two different validation strategies have different benefits: the former is historically relevant, as it was a method used to initially demonstrate the superiority of BLOSUM62 to other matrices [109, 110]. The latter is more time–consuming, but it reflects current knowledge of protein homology and allows for the determination of the statistical significance of differences between matrices.

The first method we used for testing matrices was designed to emulate the work by Henikoff and Henikoff [110]. In that work, the 257 PROSITE 9.0 [20] families that were most challenging to detect were used as queries against SWISS–PROT 22 (numbering 25,044 sequences).

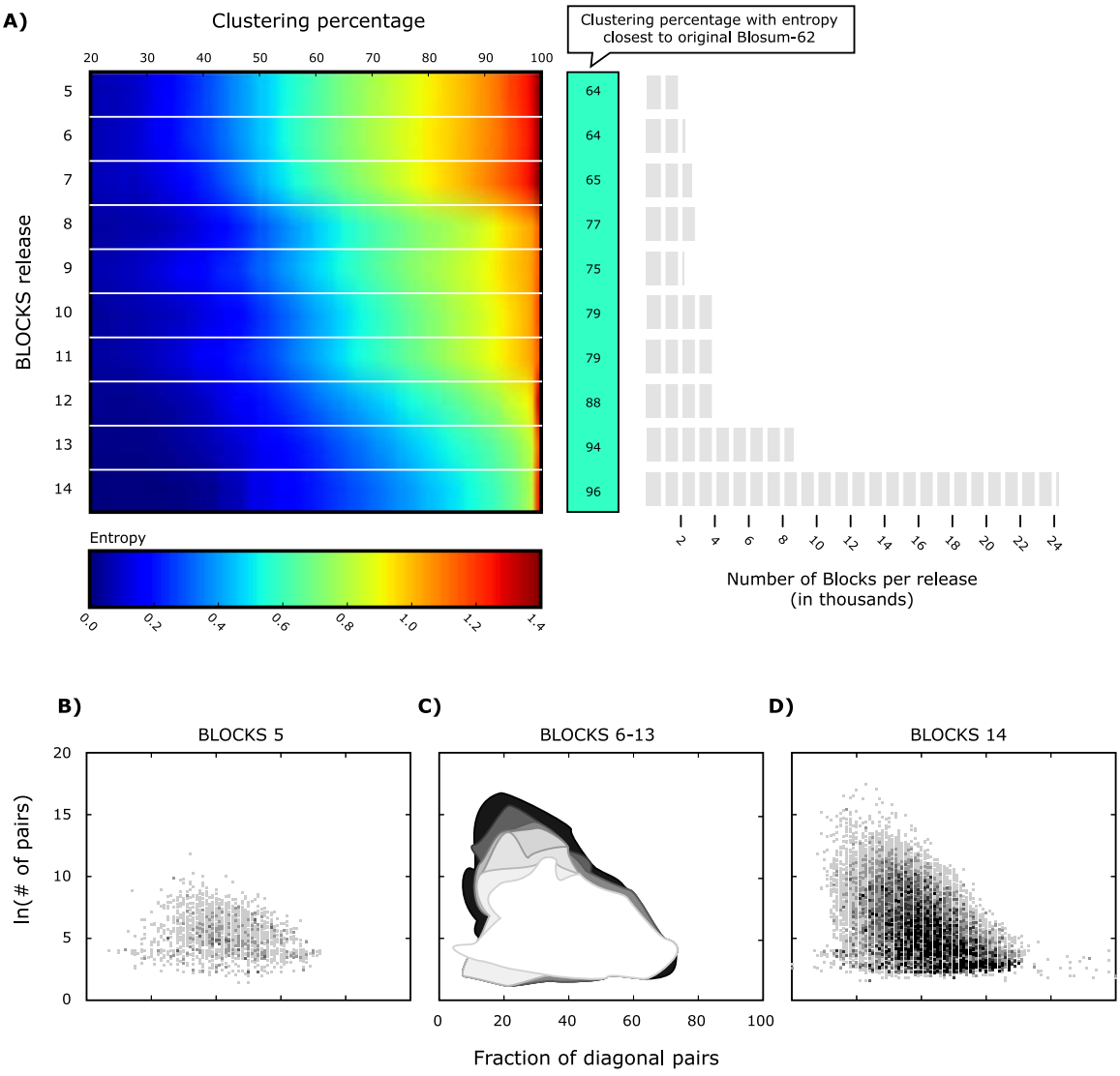


Figure 4-1: Characteristics of the BLOSUM matrices calculated from successive releases of the Blocks database. Panel A) shows the entropy of the scoring matrices computed from various Blocks releases as a function of the clustering percentage used by the BLOSUM algorithm (see methods). Blue colors indicate low entropies and red colors indicate high entropies. Oddly, at constant clustering percentage, matrix entropy decreases with successive Blocks releases (see part B below). The middle part of panel A) shows the clustering percentage which results in the matrix which has an entropy closest to the original BLOSUM62 matrix. The right–most panel shows the number of blocks in each release of the Blocks database. Panel B) of the figure shows a scatter plot in which each block in the Blocks 5 database is represented as a dot. The location of the dot along the x-axis represents the percent of the amino acid pairs contributed by that block that lie along the matrix diagonal — i.e. identical pairs such as A–A, G–G, etc. The location of the dot along the y–axis indicates the total number of amino acid pairs contributed by that block. (Note that the y–axis is in log units and that the matrix was computed at 50% clustering.) Finally, panel D) shows the scatter plot for Blocks 14. Notably, successive releases of the Blocks database incorporated many large blocks comprising distantly related sequences, as shown by the migration of the point clouds towards the upper left quadrant.

For each family, the list of all members was used as true positives.

The second method we used for testing matrices was designed to emulate the work by Price et al. [200]. We used the ASTRAL database [42] as the basis for our more exhaustive experiments for detection of remote homologs. ASTRAL is created based on the SCOP database [175], which classifies proteins based on their function, structure, and sequence into a hierarchical structure of classes, folds, superfamilies, and families. Sequences in the same superfamily can have low sequence similarity, but are likely to have a common evolutionary origin based on their structural and functional features. Because these classifications are made by human inspection, not via automated sequence alignment procedures, it makes a perfect "gold standard" for remote homolog detection tests.

From the full set of ASTRAL genetic domain sequences, we chose the sequence set from which 40% identical sequences had been eliminated. By using this subset, our search focuses on the detection of remote homologs that are more challenging for substitution matrices to discover and thus will differentiate the abilities of the respective matrices to find distant relatives. The sequences were further filtered by pseg [272] for the removal of low–complexity regions. The unfiltered sequence set is available on–line from the ASTRAL database at `http://astral.berkeley.edu/scopseq-1.69/astral-scopdom-seqres-gd-sel-gs-bib-40-1.69.fa`. This non–redundant set numbers 7,290 sequences. Each sequence was extracted from the database one at a time and used as a query for the entire database. Search results in the same superfamily as the query were considered to be true positives.

Search methods

We chose the Smith–Waterman [235] local alignment algorithm for all searches against both databases for its high sensitivity in detecting remote homologs. In particular, we used the ssearch implementation of the Smith–Waterman algorithm by Pearson [192, 193].

For our database searches, we used the ssearch default parameters for unknown matrices, which are a -10 penalty for gap initiation and a -2 penalty for gap extension. We believe that these parameters are reasonable settings; they represent an intermediate ground between the values used in the initial BLOSUM paper (-8/-4) and current commonly–used settings (for instance, the defaults for BLOSUM62 in ssearch are -7/-1, while in BLAST they are -11/-1). Moreover, previous work [100] has shown that while slight performance boosts can be found by optimization of gap penalties, there is frequently a broad maximum of penalty values with approximately equal efficacy. In addition, a sampling of the Kolmogorov–Smirnov statistic values returned by ssearch for searches using our penalty values were well within the acceptable range. This indicates that the distribution of alignment scores is the expected extreme–value distribution and that a significant alteration of the gap penalties is most likely unnecessary. That is, our penalties are neither too forgiving nor too permissive.

Most importantly, the determination of completely optimized sets of matrices and parameters is not the ultimate goal of this work. Rather, the goal of this work is to analyze the BLOSUM matrices as affected by the changing entries in the Blocks database. In this sense, the use of globally optimal parameters for each matrix is not imperative; instead, the consistent use of some average, acceptable parameter values for all matrices provides a level, controlled environment for determining the relative raw ability of each matrix to detect remote homologs. So, though we feel we chose acceptable parameters for our work, it is not of intrinsic importance

to determine the optimal parameters for each matrix.

It is worth noting that other works (particularly the early BLOSUM works that the PROSITE–based testing method is based upon) frequently used BLAST [10] instead of Smith–Waterman to evaluate the quality of scoring matrices. In this work, we chose Smith–Waterman because of its sensitivity and to avoid any artifacts due to the heuristic shortcuts in BLAST.

Evaluation of results

For both sets of database searches, we used the same respective methods for evaluating search results as in previous literature. In the PROSITE–based testing, we used head–to–head comparison of effectiveness in finding family members. For all PROSITE families that were queried, the matrix that found the most true positives was noted. The relative effectiveness of any two matrices was then found by subtracting the number of times that one matrix was more effective from the number of times that the other was more effective. True positives were defined as described previously. The search criterion used was the same as for the previous work [110], as initially described by Pearson [193]: if a true positive appeared before 99.5% of the true negative sequences, it was considered "found".

For ASTRAL–based testing, we used the Bayesian bootstrap method to evaluate the statistical significance of the mean difference in coverage between any two substitution matrices [200]. This method uses coverage vs. errors per query as a means to evaluate the effectiveness of different substitution matrices. Coverage is defined simply as the fraction of true positives found at a given errors per query threshold. True positives were identified as described above.

4.3.4 Results

We began by first assembling the matrices that we would be using in our experiments. As stated in the Methods section, we used a modified version of the original BLOSUM program that incorporated multiple bugfixes. We created a matrix for each integer clustering value between 20 and 100; the results can be seen in panel A of Figure 4-1.

The center of panel A lists the reclustering percentage needed for each Blocks release to produce a matrix with entropy closest to that of the original BLOSUM62 matrix. We used this set of isentropic matrices for our sequence alignment tests. A given matrix's relative entropy reflects the required minimum length of homology in order for it to be distinguished from noise [9]. Merely maintaining (in this case) a reclustering percentage for a time–dependent family of matrices would have little meaning, as changes in entropy could occur that would obscure the effectiveness of the information encoded in the matrix. In this sense, it is only "fair" to compare matrices of the same entropy. Thus, we used matrices with the same relative entropy of BLOSUM62, 0.6979, which is approximately the value previously shown to be most effective for database searches [109]. (Note that, due to the bugfixes mentioned earlier, the BLOSUM matrix computed from Blocks 5 had its entropy analog at a reclustering percentage of 64 rather than 62.) We refer to matrices computed from the "revised" BLOSUM code as RBLOSUM, making the baseline matrix for that family RBLOSUM64.

The right-hand side of panel A in Figure 4-1 shows that the number of blocks in each release increases in an almost monotonic fashion, with the exception of release 9. The general trend is expected, as the PROSITE database that is used to create the blocks would likely have more

families of known homology added in later releases. The decrease in blocks in release 9 remains an anomaly; we speculate that it may have been due to a one–time change in parameters in the creation of the blocks, though we have no way to verify this theory.

Inspecting the heatmap in panel A of Figure 4-1 reveals that, as expected, relative entropy increases with increasing reclustering percentage in any given Blocks release. However, at constant clustering percentage, matrices computed from successive releases of the Blocks database show markedly decreased relative entropy. We hypothesized that this trend was due to changes in the character of blocks in the database. Indeed, panels B–D of Figure 4-1 suggest that the presence of extremely large, diverse blocks may have been the cause of this phenomenon. The scatter plots in panels B–D show point clouds representing all the blocks in a given Blocks release (panel C shows the outlines of these clouds). Each block is represented as a single point at a location that indicates the degree to which the block contributes identical amino acid pairs (x–axis) and the total number of amino acid pairs contributed by the block. The three panels show a trend towards the incorporation of blocks that have many sequences that are only remotely homologous. This trend is manifested in the migration of the point clouds towards the upper left quadrant of each of the three scatter plots.

These panels explain why the reclustering percentage needed to be increased so much in order to create isentropic matrices. As large blocks with more diverse sequences are added to the database, something must be done to offset that diversity in order to obtain an isentropic matrix. Since the highly diverse members of a family (block) will not cluster together, they will have a significant impact on the substitution counts that are used to derive the matrices. In order to offset this impact and steer the entropy of the matrix away from that of the background, it is necessary to increase the re–clustering percentage used to compute the matrices. In this way, blocks containing highly homologous sequences will have greater influence on the substitution counts and steer the matrix closer to the desired counts and information content.

Having assembled a set of isentropic matrices, we then used our two tests — the historical, PROSITE–based test and the statistically rigorous, ASTRAL–based test — to evaluate the effectiveness of updated BLOSUM matrices. By using both of these tests rather than just one, the comparison of updated substitution matrices is grounded in the same metrics as would have been used when the matrices were first published, while providing quantitative statistical results.

We found that, with time, the character and quality of the entries in the Blocks database has changed significantly. Figure 4-2 shows a slightly complex trend that warrants some analysis. The figure shows boxes whose vertical position indicates their relative performance; the further a box is vertically from the Blocks 5 box, the greater the difference in performance between the isentropic matrices derived from those releases (see caption). In early updates of Blocks, the resulting RBLOSUM matrices tended to hover around a certain performance. This is consistent with previous hypotheses [109] that the BLOSUM matrix would not be altered by adding to or subtracting from the Blocks database. The variation could be explained in part by integer rounding; since the desired scores are rounded to the nearest whole number, it is possible that the intended scores for a given matrix are not completely accurately represented by a given BLOSUM matrix. Another possibility is that changing block quality causes these fluctuations; this possibility is further analyzed below. However, the particularly poor performance of Blocks releases from 12 on, and that of release 9, is inconsistent with the initial hypothesis that matrix performance would remain approximately constant.

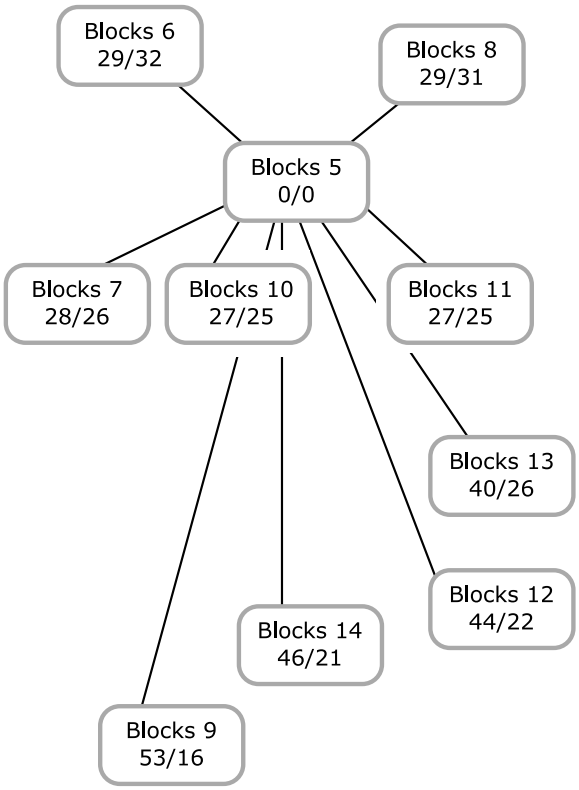


Figure 4-2: The relative performance of updated BLOSUM matrices. This figure is designed to emulate Figure 4 from Henikoff and Henikoff [109]. All matrix performances are compared to the revised BLOSUM62 isentropic analogue derived from Blocks 5, RBLOSUM64. Vertical distance from Blocks 5 indicates relative performance, with matrices above Blocks 5 performing better and those below it performing worse. Comparisons were based on the 257 "difficult" queries in Henikoff and Henikoff [110], derived from PROSITE 9.0 keyed to SWISS–PROT 22. Numbers in each box indicate the number of groups for which RBLOSUM64 from Blocks 5 performed better than and worse than isentropic matrices from other releases. Releases immediately following Blocks 5 seem to cluster around the same level of performance, while later releases (and release 9) have unusually bad performance.

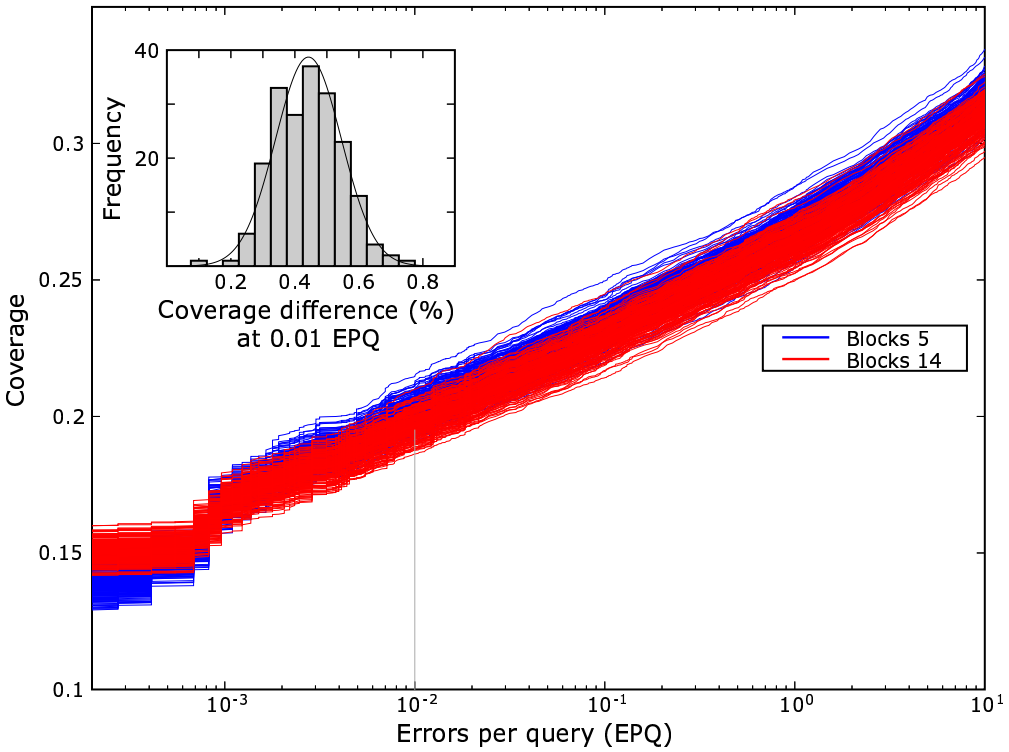


Figure 4-3: A complete set of Bayesian bootstrap replicates, with inset histogram of coverage difference. These data were created using the PSCE software [200]. (See Price et al. [200] for a thorough explanation of Bayesian bootstrapping). Each thin, faintly colored line represents one Bayesian bootstrap run. The thick lines represent the total dataset results. In this case, the two distributions overlap somewhat, but analysis of the data via the inset histogram of coverage difference reveals that the difference in coverage clearly follows a distribution with non–zero mean. These distributions are used to compute the confidence intervals shown in Figure 4-4.

These results are largely consistent with our results from the ASTRAL–based tests. Figure 4-3 is a representative result for a set of Bayesian bootstrapping runs for the ASTRAL–based test (in this case, for releases 5 and 14 of the Blocks database). The lighter, thinner lines track coverage as a function of the allowed errors per query (EPQ) for individual bootstrap runs, while the two thick lines represent the full–database result. Clearly, there is some overlap between the two distributions, but a pairwise comparison of runs (as demonstrated by the inset evaluated at 0.01 EPQ) shows a distinctly non–zero difference between the two distributions. The difference in coverage at a variety of EPQ values can be used as a metric to judge how consistently different the performances of any two matrices are.

This metric is used in Figure 4-4 to show the performance of all updated matrices relative to the baseline RBLOSUM64 matrix computed from Blocks 5. These results correspond quite well to the results in Figure 4-3. That is, releases 7, 8, 10, and 11 perform comparably to 5, release 6 is slightly better, and release 12 is slightly worse, while releases 9, 13, and 14 perform

substantively worse than release 5. These latter releases have statistically significant differences. This agreement suggests that the original test employed by Henikoff and Henikoff [109, 110] was rather effective and efficient in that the results of the test would not have changed much with access to today's larger databases.

4.3.5 Discussion

The reason for the poor performance of RBLOSUM matrices derived from later releases of Blocks remains to be explained. Figure 4-1 suggests that the number of blocks and shifting isentropic clustering percentage are not reasonable explanations. If these were so, one would expect to see either gradually degrading performance (for database size) or significant step changes in performance at releases 8, 12, and 13 (for isentropic clustering percentage). However, there is certainly not a gradual degradation in performance, and there is no significant change in performance at release 8. In addition, any decrease in performance at release 9 disappears for the next two releases.

We hypothesized that two phenomena — the decreased entropy at constant clustering in successive Blocks releases, and the poor performances of these releases — were both caused by the changing character of blocks added in later releases. Specifically, we thought that the trends shown in panels B through D in Figure 4-1 might be responsible for these phenomena.

To test this hypothesis, we sorted the blocks in the Blocks 14 database by the number of off–diagonal (i.e., non–identity) amino acid pairs contributed to the RBLOSUM matrix by each block. We then removed the blocks that were the top 1% of contributors to off–diagonal pairs (243 blocks) and created an isentropic RBLOSUM matrix from this "cleaned" database. Notably, the reclustering percentage required to create an isentropic matrix decreased from 94 to 84 for the cleaned database. The performance of this matrix relative to RBLOSUM64 from Blocks 5 is shown in Figure 4-5. The cleaned version of the Blocks 14 database gives rise to an RBLOSUM matrix that is superior to any of the other matrices we tested, including RBLOSUM64 from Blocks 5 (Figure 4-5) and the original BLOSUM62 from Blocks 5 (data not shown).

The performance of the RBLOSUM matrix created from the "cleaned" Blocks 14 database supports our hypothesis that the addition of large, diverse blocks has had an adverse effect on the performance of updated RBLOSUM matrices. We believe that the decrease in performance may be due to a change in the database that is used to create the Blocks database [107]. Initially, Blocks was based on the PROSITE database. As of release 12 of Blocks, blocks were formed from InterPro groups rather than PROSITE groups. In release 12, only InterPro groups with cross–references to PROSITE groups were used to create blocks. In release 13, this restriction was lifted, and it has remained lifted to the current release of Blocks. We believe that this explains almost all of the trends that we observe in the data. When the Blocks database partially shifted to being based on InterPro, performance first decreased slightly with the addition of sequences that had not previously been included. When the shift was completed, performance degraded significantly. The only unexplainable anomaly is the unusually poor performance of release 9 of Blocks; we believe that can be attributed to the unusually small number of blocks in that release. Again, we speculate this may have been due to some one–time change in parameters, but we have no way to prove or disprove such a speculation.

In conclusion, we see that in some sense, the hypothesis that Henikoff and Henikoff [109]

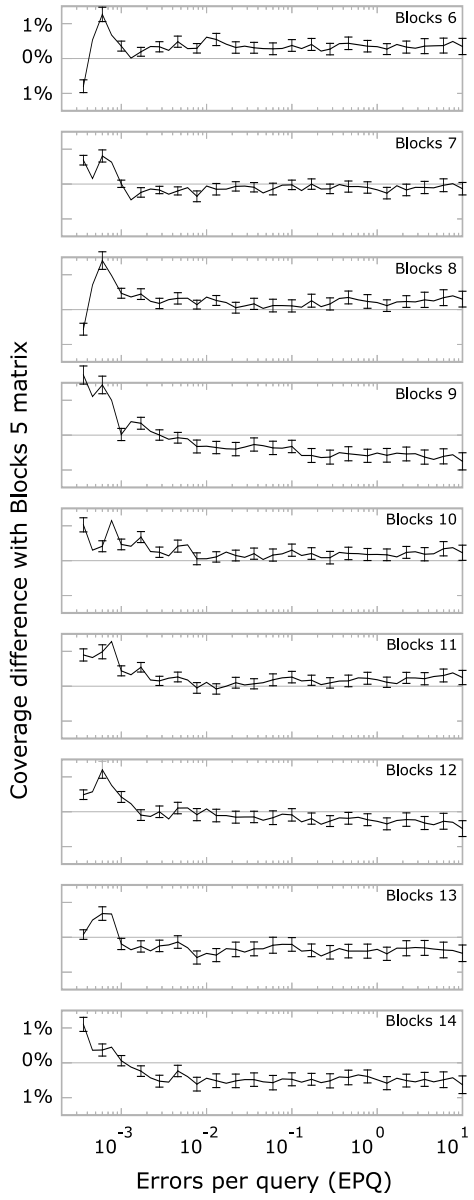


Figure 4-4: Plots of the differences in performance of updated RBLOSUM matrices. Each matrix is compared to the RBLOSUM64 matrix in 200 Bayesian bootstrap replicates to find the mean difference in coverage, and the confidence interval for that coverage, at a specific EPQ rate. These differences are plotted as a function of EPQ rate, with positive values meaning that a given matrix performs better than RBLOSUM64 on the dataset. Error bars represent 95% confidence intervals. At data points where the error bars do not intersect with the origin, the performance difference between the matrices is statistically significant. These results correlate well with, and provide statistical analysis of, the results in Figure 4-2.

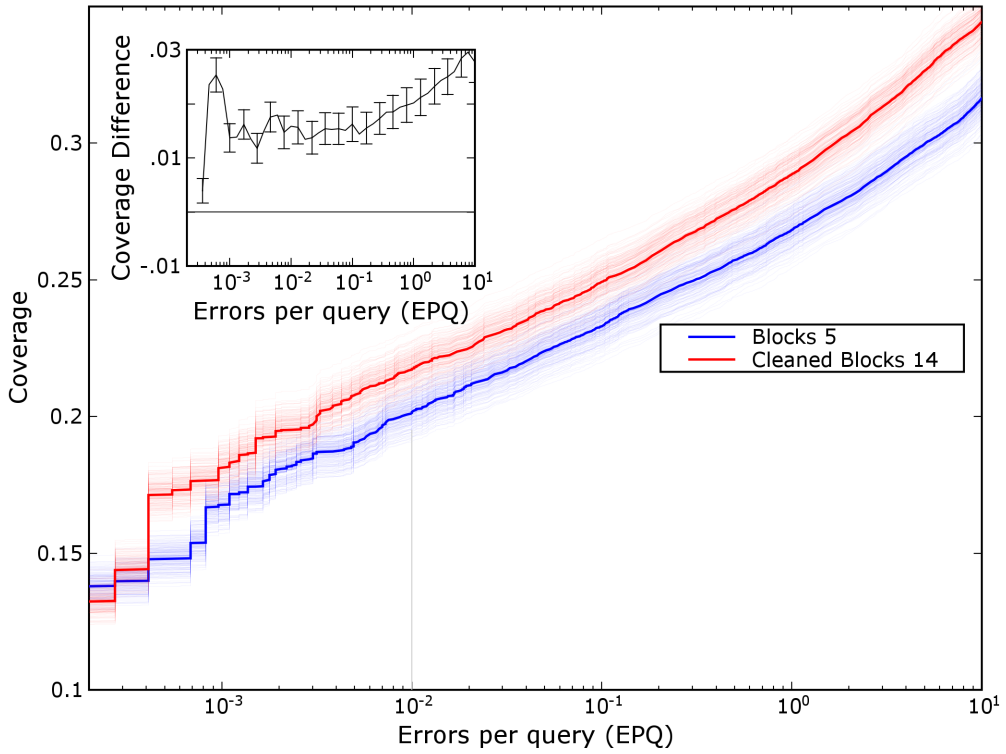


Figure 4-5: Coverage of a cleaned RBLOSUM matrix compared to the RBLOSUM64 matrix. Again, thin, faint lines represent individual bootstrap runs, while the dark line represents the parent dataset. These two distributions are quite distinct, with the cleaned RBLOSUM matrix being significantly more effective than the RBLOSUM64 matrix (and, transitively, all updated RBLOSUM matrices). The inset shows the coverage difference between the two matrices' coverage as a function of errors per query. Error bars represent 95% confidence intervals. Note the different scale from Figure 4-4 and the statistical significance at all EPQ values since no error bar crosses the origin.

initially proposed was true: for releases of the Blocks database based on PROSITE, despite some slight variation, the performance of isentropic RBLOSUM matrices is relatively constant over successive releases. However, since the quality of the blocks added in recent releases has decreased, such is not the case for the matrices derived from the current Blocks database. This suggests that, to the extent that there are "bad" blocks, there may also be "good" blocks, and sensible, judicious selection of these blocks may be a reasonable approach for the creation of amino acid substitution matrices.

4.4 Bioinformatics and handwriting/speech recognition: unconventional applications of similarity search tools

4.4.1 Introduction

Bioinformatics has benefited immensely from tools and techniques imported from other disciplines. Markov models used for gene–finding have their origin in information science, neural networks are imported from machine learning, and the countless clustering methods used for analyzing microarray data are from a wide variety of fields.

Sequence alignment tools are no exception to this trend; however, within bioinformatics, they have reached new levels of speed and sophistication. Tools, such as Blast [10, 11] and FastA [194], are used routinely to search through a database for sequences (DNA or protein) that are similar to a query sequence. Over the years, these tools have been optimized for speed by employing a number of heuristic shortcuts to the dynamic programming algorithms on which they are based. Even searches in very large databases, such as Swiss–Prot/TrEMBL [22] or GenBank [32], take only a few seconds for queries of small to moderate size. This is substantially faster than the time required for a rigorous Smith–Waterman search [264]. In light of the remarkably speed and accuracy that characterize these algorithms, it is intriguing to investigate other applications where similarity search tools might be of material importance. In this work, I present two alternative applications of these fast sequence alignment tools outside the domain of bioinformatics: handwriting recognition and speech recognition. All of the work described in this section is part of a publication appearing in the proceedings of the fourth Singapore–MIT Alliance Programme on Molecular Engineering of Biological and Chemical Systems, which was co–authored with Gregory Stephanopoulos. Throughout this section, the use of the pronoun "we" refers to these authors.

The dynamic handwriting recognition problem is to recognize handwriting from a touch tablet as found on personal digital assistants (PDAs), for example Palm Pilots, or tablet PCs [242]. These writing tablets sample the position of a pen as a function of time to produce a series of (x, y) points that are used by handwriting recognition algorithms to determine which character was written. An excellent review of the most common algorithms is available from Plamodon and Srihari, 2000. These include feature analysis, curve matching, Markov models, and elastic matching, the last of which is based on dynamic programming and is related to both Blast and FastA.

To apply similarity search concepts to the handwriting recognition problem, we represented the path of a PDA pen as a protein sequence by translating the (x, y) points into a string of amino acids. Using the protein representation of handwriting samples, we were able to classify unknown samples with FastA. This is analogous to the problem of protein annotation using similarity searching: given a protein (a written character) of unknown function, we annotated the protein by searching for similar sequences (characters with similar (x, y) paths).

We applied the same sequence alignment approach to speech recognition. Automated phone services, security checkpoints, and computer dictation software employ some form of speech recognition. Common speech recognition methods include feature recognition, neural networks, hidden Markov models, dynamic programming [180] and a variety of other statistical and signal processing algorithms. A good review of these techniques and more is available from Juang & Furui, 2000. For this problem, we represented digital speech recordings as sequences

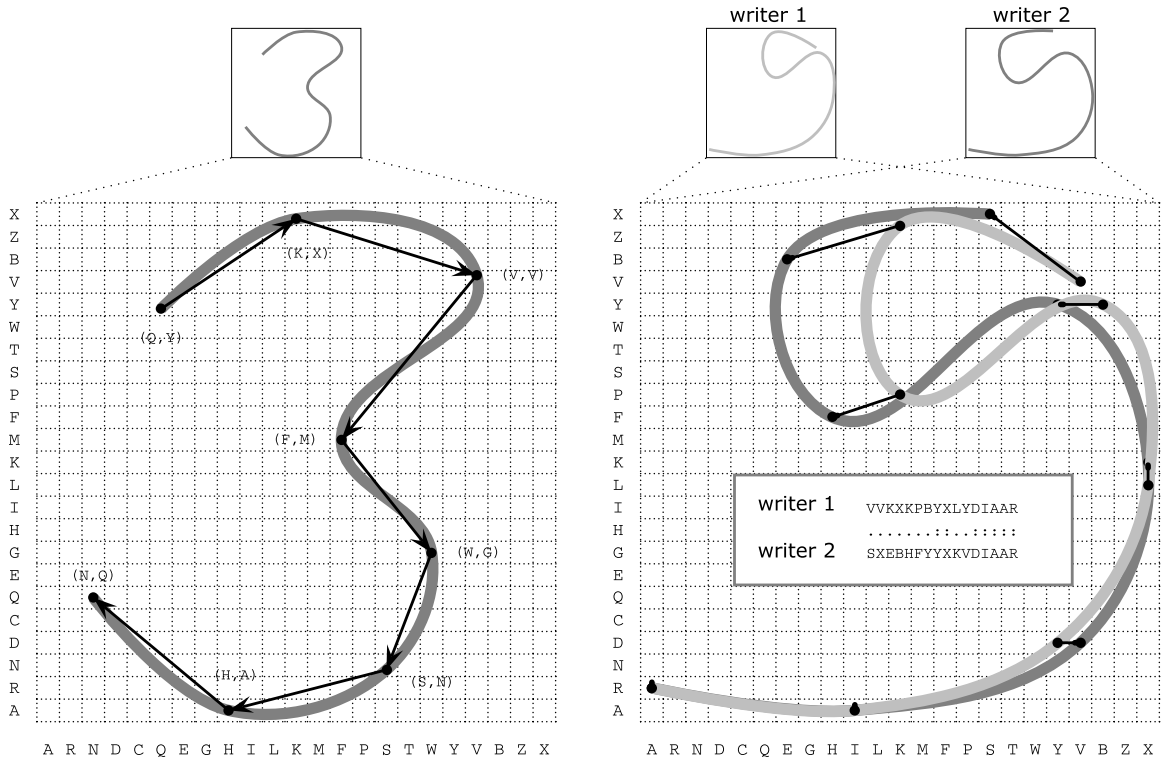


Figure 4-6: Projection of a digit written with a PDA stylus into protein space. Concatenating the set of points gives a protein sequence representative of the digit. In this case, the sequence is QYKXVVFMWGSNHANQ.An alignment of nines from two different writers. The boxes at the top show the input from each writer and the large grid show the superposition of the two handwritten digits. The FastA alignment between the protein representations of the two digits is shown in the center. Two visualizations of the handwriting recognition problem. In both cases the x and y axes are divided into 23 parts corresponding to the columns and rows in an amino acid scoring matrix. The eight sampled points from the digit are cast from x, y space into protein space by assigning amino acid coordinates to each point.

of amino acids, and used a database of annotated recordings to classify unknown recordings.

In the following section, we describe the data sets used for the handwriting recognition and speech recognition problems. Then, we detail how these data were represented using strings of amino acids and how we used FastA to annotate unknown samples in four handwriting and speech recognition experiments. We compare our results to more traditional methods of handwriting and speech recognition and, finally, we discuss ways of improving upon the results and extending sequence alignment to other classification problems.

4.4.2 System and Methods

Handwriting Recognition

For our handwriting recognition experiments, we used data from Alimoglu and Alpaydin, 1996, available in the University of California Irvine repository of machine learning databases [38].

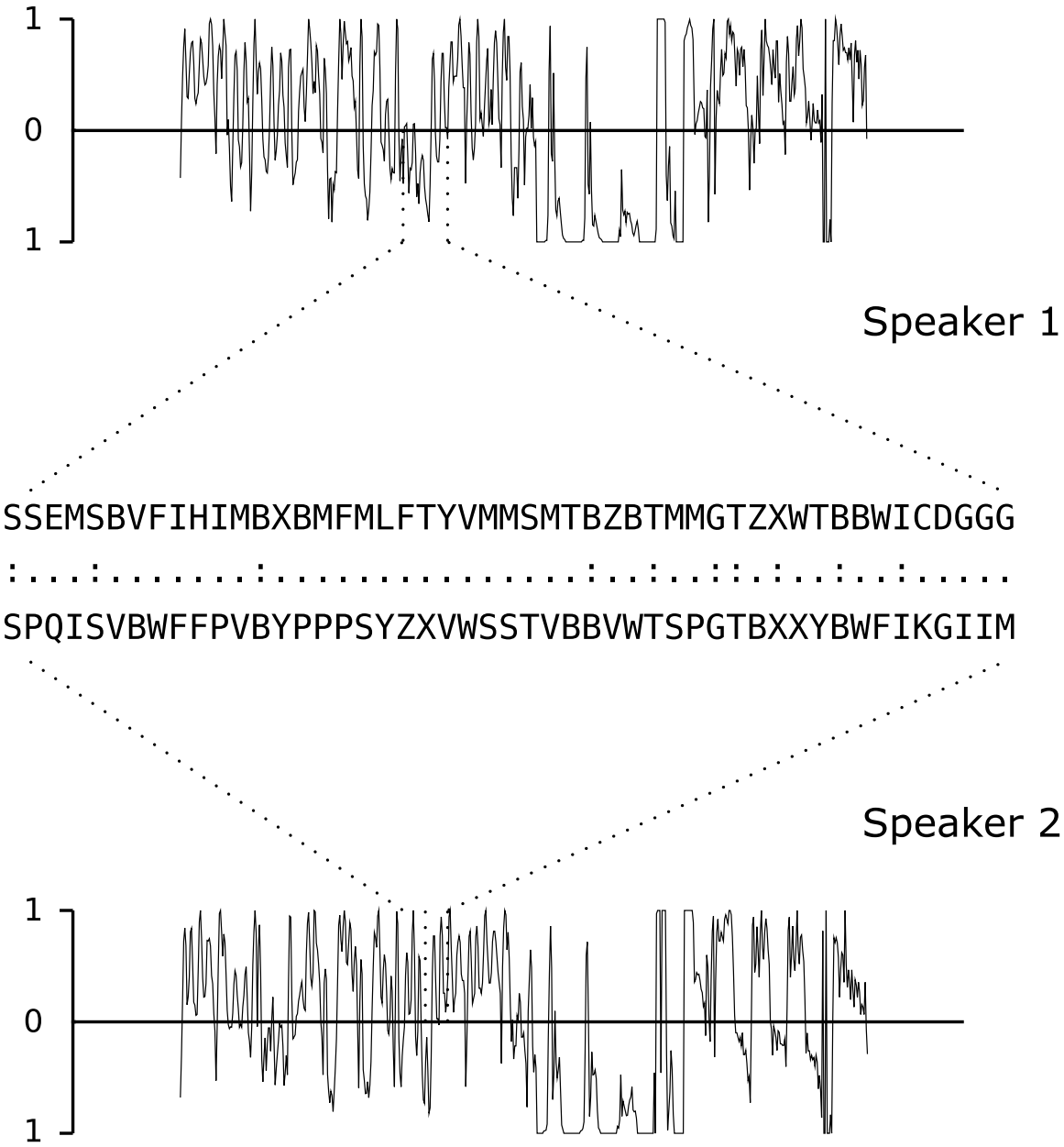


Figure 4-7: An alignment of the spoken–letter "X" recorded from two different speakers. The plots at the top and bottom are recordings for first and second speakers, respectively. The breakout in the center shows a section of the protein projection of each recording and the alignment generated using FastA as described in the text. This example was taken from the first speech recognition experiment. In this case, the bottom recording was the top scoring alignment against the top recording.

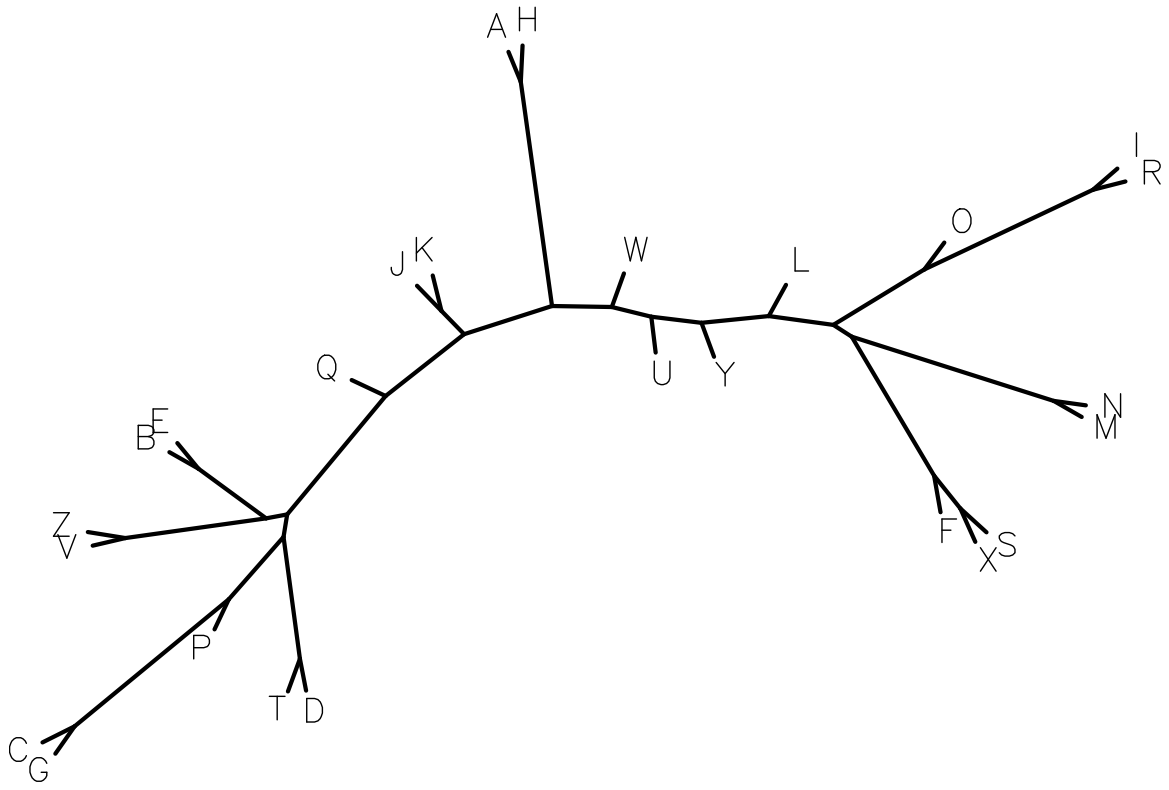


Figure 4-8: A phylogenetic tree of voice–proteins. This tree was created using the Phylip [81] tree drawing program from a multiple sequence alignment of all 26 voice–proteins from a single speaker. The multiple sequence alignment was made using the ClustalW [116] alignment tool, with the scoring matrix in Table 4.3 on page 147. In the tree, similar sounding (homologous) letters are grouped near each other. For example, all the letters containing the /ee/ sound [*B, C, D, E, G, P, T, V, Z*] are clustered on the left side of the tree.

Table 4.2: Results for the handwriting and speech recognition problems described in the text. For each experiment, the misclassification is the percent of sequences in the unknown set for which the digit or letter was not predicted correctly.

Experiment	Classification	Classification in Alimoglu & Alpaydin, 1996
1	97.34%	97.80%
2	99.64%	n/a

(a) Handwriting recognition results.

Experiment	Classification	Classification with clustering	Classification in Dietterich & Bakiri, 1995
1	93.84%	98.91%	96.73%
2	92.61%	98.61%	n/a

(b) Speech recognition results. The second column shows the misclassification using the clustering of all /ee/ sounding letters as described in the text.

These data comprised of 10992 handwritten digits between *0* and *9*, written by 44 writers with each writer submitting 250 digits (8 samples were discarded by the original authors).

Each digit was written with a stylus pen on a touch tablet, which recorded the x and y coordinates of the pen as a function of time. These data were re-sampled such that each written digit was represented by a series of eight (x, y) points, spaced out by a constant arc length over the path of the digit. Then, for each digit, the set of (x, y) points were scaled such that the largest axis, usually the y axis, ranged from 0 to 1. By dividing the number line $[0, 1]$ into 23 "bins" we translated each of these coordinates into a pair of amino acids as shown in Figure 4-6 on page 143. We concatenated these amino acid pairs to obtain a protein sequence representation of each digit: a "digit–protein."

Speech Recognition

For our speech recognition experiments, we used data from Deitterich and Bakiri, 1995, available in the University of California Irvine repository of machine learning databases [38]. This data set consisted of 7797 recordings of individuals speaking one of the letters *A–Z*. A total of 150 speakers each said every letter *A–Z* twice (three recordings were discarded by the original authors). Then, each recording was processed into a set of 617 real–valued attributes in the range $[-1, 1]$. A more detailed description of the database is available from Dietterich & Bakiri, 1995.

By dividing the number line $[-1, 1]$ into 23 bins we translated these real numbers into a series of amino acids. For example, the series "-1.0,-0.55, 0.11, 0.65" was translated to "AQKY". We concatenated these amino acids to make a protein representation of each recording: a

Table 4.3: The scoring matrix used for the handwriting and speech recognition FastA alignments. Each entry of the scoring matrix, s_{ij}, is given by $s_{ij} = 10 - (i\text{-}j)$. That is, matching amino acids are given 10 "points", amino acids that are one off are given 9 points, and so on. This matrix was used in place of the default scoring matrix, Blosum50 [109], for FastA. The scoring matrix was found heuristically. Also, a few experiments indicated that the alignments are relatively insensitive to permutations about the form of s_{ij} given above.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X
A	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
R	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
N	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
D	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
C	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
Q	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7
E	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6
G	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5
H	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4
I	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3
L	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1	-2
K	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	-1
M	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0
F	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1
P	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2
S	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3
T	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4
W	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5
Y	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6
V	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8	7
B	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9	8
Z	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	9
X	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10

"voice–protein".

4.4.3 Results

Handwriting Recognition

We conducted two handwriting recognition experiments. In both experiments part of the digit–protein database was assumed to contain a "known" set of digits that was subsequently used to annotate, or classify, the remaining "unknown" digits. For our first experiment, we used for the known database containing the writing of 30 persons (7494 digits) and an unknown database with the writing of the remaining 14 persons (3498 digits). Using FastA, we searched each sequence from the unknown set in the known set and used the top scoring hits to annotate the unknown digits. Searches were carried out using the scoring matrix shown in Table 4.3 on the page before with FastA version 3.4t11 using the default gap open and extension penalties, and the following options: `-p -Q -d0 -f-8 -g-1 -H -E1000 -b1`. An example alignment of two handwritten nines from different writers is shown in Figure 4-6 on page 143.

For our second experiment, we used 25% (2748 digits) of our digit–protein database, selected randomly, as the unknown set and the remaining 75% (8244 digits) as our known set. Alignments and annotations using FastA were performed as in the first experiment.

The results of the two handwriting recognition experiments are shown in Table 4.2 on page 146. In experiment 1, our results are about the same as the best k–means clustering results of Alimoglu and Alpaydin [6, 7]. This experiment simulates the user–independent handwriting recognition problem: the handwriting of one group of writers was used to classify digits from a different group. In the user–dependent problem, experiment 2, the database of known handwritten digits contains samples from all the writers, on average. Thus, for every unknown handwriting sample, there is often a close match in the database of known samples. As such, the results of experiment 2 are significantly better than those of experiment 1 as shown in Table 4.2 on page 146.

In experiment 1, the average time for each alignment was 0.117 seconds per unknown sequence on a 1 gHz Pentium III processor. This is much shorter than the time required to write the digits. Thus sequence alignment could be used as a "real–time" method for handwriting recognition. This high speed, together with the high accuracy for user–dependent recognition makes sequence alignment good candidate for use on a Tablet PCs, or even PDAs.

Speech Recognition

Using the voice–protein database, we conducted two experiments, analogous to the two handwriting recognition experiments described previously. First, we used a known set consisting of 6238 recordings from 120 speakers and an unknown set with 1559 recordings from the remaining 30 speakers. Second, we used 25% (1949 recordings) of the voice–protein database, selected randomly, as the unknown set and the remaining 75% (5848 recordings) as the known set. Each of the speech recognition alignments was performed using the same scoring matrix and FastA parameters as the handwriting recognition experiments. An example alignment of two voice–proteins is shown in Figure 4-7 on page 144.

The results of the two speech recognition experiments are shown in Table 4.2 on page 146. Experiment 1 is compared to the best Error Correcting Output Code (ECOC) results of Deit-

terich and Bakiri [67], but there was no comparison available for experiment 2. The misclassification for experiment 1 was 6.16%, higher than the ECOC result of 3.27%. However, we observed that most of the errors were due to rhyming letters, and in particular all of the /ee/ sounding characters [*B, C, D, E, G, P, T, V, Z*]. This indicated that these characters were similar on a sequence level, so we constructed a phylogenetic tree of the sequences to study their relationship.

A phylogenetic tree of 26 voice–proteins from a single speaker is shown in Figure 4-8 on page 145. As the figure shows, the protein projections of phonetically similar letters tend to be homologous. Furthermore, letters such as *A* and *H*, which have the /ay/ sound at the beginning, are more closely related to each other than they are to *J* and *K*, which have the /ay/ sound at the end. Because the /ee/ sounding letters all have /ee/ at the end, they are particularly difficult to distinguish from each other. These letters account for a disproportionate majority of the errors in our two experiments. By clustering these letters together such that they are considered the same for classification purposes, the error in experiment 1 was reduced to 1.09%. If the original error was evenly distributed between the classes, the error would have been reduced only to about 5.5%. This suggests that, although string alignment performs poorly for /ee/ sounding characters, it performs well for all other characters.

4.4.4 Conclusions

This work showed that sequence alignment can be a powerful classification tool for problems outside the domain of bioinformatics. In both the handwriting and speech recognition problems, we projected real–valued data into strings of amino acids and used FastA as a classification tool, in a manner analogous to protein annotation. In the case of handwriting recognition, we showed that sequence alignment is a viable alternative to traditional methods, such as k–means clustering, and is fast enough to be used as a real–time recognition method.

There are many ways to improve upon the results we presented here. First, we did not have any explicit training phase for either set of experiments. However, there are at least two sequence alignment parameters which can be trained: the gap open and extension penalties, and the scoring matrix. The optimization of these parameters for protein annotation is well documented [9, 65, 109, 110, 112, 258] and would be similar for alternative sequence alignment applications such as handwriting recognition. Second, intelligent projection of data into strings can greatly improve results. Here, we used bins of equal size to partition the real–valued data into amino acids; however, bins of unequal size may improve the resolution between closely related sequences and improve classification. Finally, more customizable sequence alignment tools would be very useful. These tools should take an arbitrary alphabet (Blast and FastA are restricted to 23 amino acids) and a user–defined scoring matrix (FastA allows user–defined matrices, but Blast does not).

The potential applications of sequence alignment tools outside of bioinformatics are boundless. Tools such as Blast and FastA can be used to quickly classify or search through any data that can be projected into a string of characters. Of course, these methods will work best with data that is of a low dimension. Our experiments with more complex data data, such as color images, suggest that how the data are projected into a string is very important with large number of dimensions. However, for simple types of data, such as customer purchase histories, black and white images, or Internet chat transcripts, we have been able to use sequence alignment as

a quick and effective classification tool.

4.5 Machine learning approaches to modeling the physiochemical properties of peptides

4.5.1 Introduction

In this section, I discuss the modeling of small peptide sequences using non–grammatical models. Most commonly, peptides and protein sequences are represented as a string of letters drawn from the alphabet of characters representing the twenty natural amino acids. Here, I present a series of experiments using a more meaningful representation of amino acids and test the ability of various machine learning techniques to predict peptide function. Specifically, I develop a set of three amino acid representation schemes and test these schemes combinatorially with a set of six machine learning techniques. All of the work described in this section is part of a publication appearing in the proceedings of the fourth Singapore–MIT Alliance Programme on Molecular Engineering of Biological and Chemical Systems, which was co–authored with Mark Styczynski and Gregory Stephanopoulos. Throughout this section, the use of the pronoun "we" refers to these authors.

4.5.2 Motivation and background

Amino acid representations

The most common representation of small peptides are as strings of letters representing the twenty amino acids, e.g. KWRAG, which is the five residue sequence lysine, tryptophan, arginine, alanine, and glycine. Notably, both amino acid names and their corresponding abbreviations are human constructs that carry no information about the underlying physiochemical characteristics of each amino acid. That is, the string KWRAG carries little information in and of itself, without some information about what a K is and how it is different from the other amino acids. In place of such physical descriptions, previous efforts have described the similarity of amino acids based on the tendency for one amino acid to substitute for another in homologous, similarly–functioning proteins across different species [65, 109]. That is, substitutions that are observed in nature can be construed in some sense as indicating similarity between certain amino acids. While such efforts have been extremely useful for tasks such as aligning more distant protein homologs, they typically do not capture enough information to be practically useful in *de novo* design or prediction of protein activity.

Here we experiment with feature vector representations of small peptides using sets of amino acid physiochemical characteristics derived from the AAindex database [135, 176, 247]. The AAindex database lists 453 physiochemical parameters for each of the twenty amino acids. These parameters range from those that are very tangible and intuitive — for example, residue volume, which is AAindex parameter BIGC670101 [36] — to the abstract — for example, the normalized frequency of participation in an N-terminal beta–sheet, which is AAindex parameter CHOP780208 [57]. The parameters were culled from the scientific literature by the AAindex authors and might be considered the universe of what we, as the scientific community, know about each amino acid.

Thus, a very logical way of representing an amino acid is as a feature vector of these 453 attributes. In this sense each type of amino acid has a different feature vector of the same dimen-

sionality. This might be considered the "maximally informative" representation of the amino acids since it incorporates an expansive set of features culled from the literature. Extending this, we could write an amino acid sequence as the concatenation of these vectors. That is, a three residue peptide could be represented as a $3 * 453 = 1359$ feature vector. Intuitively, this representation retains more information than the string representation. Further, we would imagine that the physiochemical representation would be more useful for modeling the function of a peptide sequence, such as its propensity to fold in a certain manner or to react with a certain enzyme.

The representation of amino acids has received some previous attention in the literature. For example, Atchley *et. al.* [16] use the physiochemical parameters from the AAindex to create a low–dimensional projection of the characteristics of each of the twenty natural amino acids. Further, they used this low–dimensional progression to derive metrics of similarity between the amino acids, similar to popular amino acid scoring matrices such as Blosum [109] and PAM [65].

HIV–I Protease

In this work we will use the HIV–I protease as a model system for demonstrating the merits of different physiochemical amino acid representations. Specifically, we show the success of different representations and different machine learning methods at modeling substrate specificity of the protease.

The HIV–1 protease is a proteolytic enzyme encoded by the HIV genome [44]. The protease plays a critical role in viral replication and the development of viral structure [262]. The protease recognizes specific eight–residue sequences in its substrates (see Figures 4-9 and 4-10 on the next page). The protease's natural targets are subsequences of other HIV genes which must be cleaved for the virus to successfully replicate. Accordingly, small molecule inhibitors of the protease are a common therapy for HIV/AIDS [39].

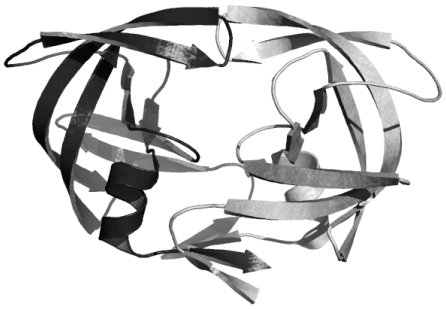


Figure 4-9: Structure of the HIV–I protease, derived from the Protein Data Bank (PDB) [34] entry 7HVP [241]. Over one hundred other structures of the protease have been solved since the first in 1989 and are available from the PDB's website. The protein is a dimer of two 99 amino acid chains. The regions of the protein at the top of the figure, the "flaps," open up and accept a substrate protein, closing behind it. Two aspartate residues in the active site, aided by the presence of water, act to cleave the substrate.

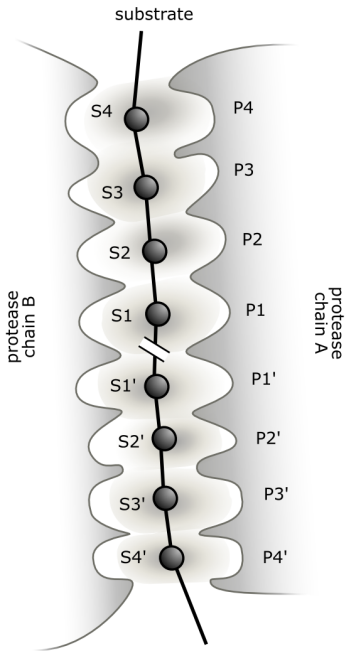


Figure 4-10: Schematic of the HIV–I protease active site. The active site comprises eight binding pockets (P1–P4 and P1'–P4') into which eight residues from the target protein fall. The target protein is cleaved between the S1 and S1' residues. One half of the catalytic unit is made up by chain A of the protease and the other by chain B (see Figure 4-9 on the preceding page).

In addition to the handful of sites that the protease cleaves to facilitate viral development, it can cleave a number of other "non–natural" substrates [28]. These substrates have been the focus of intense experimental study [18, 26, 27, 60]. In a recent manuscript, You *et. al.* collected a comprehensive set of 700+ eight–residue substrates that have been tested for cleavability by the HIV–I protease [276]. In addition, You *et. al.* developed a series of models for the protease's substrate selectivity that, in general, outperform previous computational models [48, 56, 177, 211], which relied on a much smaller dataset [49].

4.5.3 Methods

Amino acid representations and input data set

A set of 746 eight–residue peptides were generously provided by You *et. al.* [276], each with a class: cleavable by the HIV–I protease or not cleavable. In addition, the complete set of 453 physiochemical parameters for each of the 20 naturally occurring amino acids was downloaded from the AAindex database (release 7.0, July 2005).

From these 453 parameters, we removed redundant parameters for which the magnitude of the correlation coefficient with another parameter was greater than 0.80. The remaining 155 independent parameters were kept. Using these parameters, we made three different projections of the 746 experimentally tested protease substrates as detailed below.

Full physiochemical projection In this projection each eight–residue peptide was represented as a 1241–dimensional feature vector: 8 residues with 155 physiochemical features per residue plus the class — cleaved or not cleaved. Of our three representations, this one retains the most information about the peptides.

Feature–selected physiochemical projection Using the "FULL" projection (above) we performed a feature selection routine to select only those features that are most correlated to the class. (Throughout this manuscript, all modeling and feature selection were performed using the Waikato Environment for Knowledge Analysis, or WEKA [269]). Briefly, we evaluated the worth of a subset of features by considering the individual predictive ability of each feature with respect to the cleaved/uncleaved class, along with the degree of redundancy between the features. Using this method, we created a 54–dimensional projection of the peptide substrates (53 features plus the class).

Analysis of this lower–dimensional projection revealed that the features of the outer residues (S4, S4') are relatively unimportant, whereas the central residues (S1, S1') are quite important in determining cleavability. For the S1 position, seven parameters were chosen:

- FASG760102: Melting point [79];
- FAUJ880105: Minimum width of the side chain [80];
- PALJ810111: Normalized frequency of beta–sheet in alpha+beta class [188];
- PRAM900101: Hydrophobicity [198];
- ROBB760107: Information measure for extended without H-bond [210];
- KOEP990101: Alpha–helix propensity derived from designed sequences [143]; and
- MITS020101: Amphiphilicity index [171].

PCA projection of physiochemical properties Using the full, 155–dimensional representation of each of the 20 naturally occurring amino acids, we performed principal component analysis (PCA) to find linear combinations of features that capture the variation between different kinds of amino acids. More formally, PCA, or the Karhunen–Loève transform, is a linear transformation by which the 20 data points in a 155–dimensional space are projected onto a new coordinate system. The system is chosen such that the greatest variance is captured by the first axis, or the first "principal component." Successive principal components (axes) capture progressively less variance. Each component is a linear combination of some of the initial features; given appropriate uniform normalization, the weight of each feature in a given component indicates the relative importance of that feature in defining the component.

Using PCA, we derived 16 principal components that capture 95% of the variance in the amino acids, with the first PC capturing 30% of the variance. The set of 746 peptide 8–mers were projected into a reduced 129–dimensional space: 8 concatenated 16–dimensional residues plus the class of the peptide.

Model creation and classification

For each of the three peptide representations detailed above, we tested the ability of six machine learning techniques to classify the peptides as either cleaved or uncleaved. Each of these models is described below. For each model, we evaluated the performance using 10x10 cross–validation (see Conclusion): for each of ten runs, 10% of the peptide dataset was withheld for testing a classifier trained by the remaining 90% of the peptides. The sensitivity and specificity of each classifier's predictions for all ten of its cross–validation runs can then be combined to determine the percentage of correctly classified peptides. This value is used to quantify the classifier's overall accuracy and facilitates pairwise comparison of models and representation schemes.

Decision tree model Decision trees are simple, intuitive classification schemes that use a series of questions (decisions) to place a sample in a class with low error rate. More specifically, a decision tree is a structure in which the internal branches represent conditions, such as "hydrophobicity index at S3 $>$ 0.52". Following these conditions leads to the leaves of the tree, which are classifications indicating whether the peptide is cleaved or not. Here, we use a particular variant of the decision tree, a C4.5 decision tree [203], which is notable for not being prone to overfitting of input data. An example decision tree from our experiments is shown in Figure 4-11 on page 159.

Logistic regression model A logistic regression is just a non–linear transformation of a linear regression. In this model, each independent variable (the different dimensions of our various projections) are regressed to the class (cleaved or not cleaved). Here we use a variant of logistic regression that leads to automated feature selection and is described elsewhere [146].

Bayesian network model Bayesian network models use directed acyclic graphs to model the joint probability distribution of each class over all input features. That is, the model captures conditional dependencies between the features with regards to how they impact the final classification of each sample. Bayesian networks can be used to find causality relationships, one of many features that make these models particularly well–suited to many applications in computational biology (see, for example, [85, 105, 221]). The method uses a Bayesian scoring metric that ranks multiple models based on their ability to explain data with the simplest possible method. The Bayesian metric is a function of the probability of the model being correct given a set of observed data; this is, in turn, correlated to the model's prior probability and its physical likelihood. For a more detailed explanation of Bayesian networks, see Witten and Frank [269] or Heckerman [106].

Naive Bayes model The naive Bayes model, or "Idiot's" Bayes model [103], is a simple machine learning scheme that assumes *naively* that each feature has an independent effect on the classification of each sample [130]. In the case of the HIV–I protease substrates, this means that the physiochemical characteristics of the S1 residue contribute to the cleavability of the peptide in a way that is independent of the other residues: S1', S2, etc. The resulting network dependencies are less complex than one might otherwise obtain from a Bayesian network model but are frequently useful, particularly for unwieldy datasets or problems with physical characteristics that may warrant the assumption of conditional independence of features.

Support vector machine model with linear basis function The support vector machine (SVM) is a machine learning technique posed as a quadratic programming (QP) problem [31]. The formulation can best be conceptualized by considering the problem of classifying two linearly separable groups of points. The first step is to define the "convex hull" of each group, which is the smallest–area convex polygon that completely contains a group. The SVM approach looks for the best linear classifier (single straight line) between the two groups of points, defined as either the line that bisects the two closest points on each convex hull or the two parallel planes tangent to each convex hull that are furthest apart. These alternative definitions provide two alternative formulations of a convex QP problem; notably, they both reduce to the same problem. (A rigorous mathematical treatment of these qualitative explanations can be found elsewhere [30, 62].) Tried and true methods for solving QP problems can then be used to (relatively quickly) determine the best classifier. This method can be expanded to allow for linearly inseparable cases by altering the optimization problem to account for a weighted cost of misclassification when training the model. There is evidence in the literature that an SVM approach to defining the best classifier is less susceptible to overfitting and generalization error [63, 254, 255].

Support vector machine model with radial basis function The above description of an SVM, despite accounting for the possibility of inseparability, does not address the need for non–linear classifiers. For instance, if the members of one class fall within a well–defined circle and the non–members fall outside of the circle, the above method will perform extremely poorly because it will try to form just one plane to separate the groups [31]. Rather than attempting to fit higher–order curves, it is easier to project the input attributes into a higher–dimensional space in which the groups are (approximately) linearly separable. The higher–dimensional spaces can be characteristic of any desired classifier (e.g., nonlinear terms generated by multiplying attributes or squaring attributes). The same method for computing the best linear classifier is then used. The result is mapped back into attribute space of the appropriate dimensions and constitutes a non–linear classifier. Though one may expect such a process to be prohibitively expensive for data with many attributes, there exists a computational shortcut using "kernel functions" to avoid calculating all possible higher–dimensional feature values. In this work, the basis function for the kernel gives us the ability to detect optimal classifiers that are based upon training points' radius from some center point (as in the above example).

4.5.4 Conclusion

Our results show that the full, 1241–dimensional representation performed the best, followed by the PCA representation and, finally, the representation made via feature selection. (See Figure 4-12 on page 160 and Table 4.6 on page 158 & 4.7 on page 158. In these tables "FULL" is the full physiochemical, 1241–dimensional representation; "CFS" is the feature–selected, 55–dimensional representation; and "PCA" is the 129–dimensional representation created using principal component analysis.)

Of the models tested, results show that logistic regression is the best, followed by (linear basis function) SVMs and Bayesian networks (See Figure 4-12 on page 160 and Table 4.4 on the next page & 4.5 on the facing page.) The single best model/representation combination was the SVM model with radial basis function (SVM–rbf) and the FULL representation. It is

Table 4.4: Machine learning model comparison. Each i, j entry represents the number of representations, out of three, for which the i model performed *worse* than the j model. Here "worse" means that the model had a statistically significant lower performance, based on a two–tailed t–test at the 0.05 confidence level.

	DT	LR	NB	BN	SVM	SVM–rbf
DT	-	2	1	3	2	2
LR	0	-	0	0	0	0
NB	0	3	-	1	2	1
BN	0	1	0	-	1	1
SVM	0	0	0	0	-	1
SVM–rbf	0	2	0	1	2	-

Table 4.5: Machine learning model ranking. Each row shows, for each model, how many other model/representation pairs that model (with any representation) "wins" against. (Thus, the max of the sum of the columns in any row is $18 - 3 = 15$; however, ties are not shown.) Here "win/loss" means that the model had a statistically significant higher/lower performance, based on a two–tailed t–test at the 0.05 confidence level.

total wins	total losses	model
8	0	LR
7	1	SVM
5	3	BN
5	5	SVM–rbf
1	7	NB
0	10	DT

worth noting that though this single combination was the best, the radial basis function SVM itself did not perform consistently well. Though this may not have been expected, it is definitely reasonable per the "No Free Lunch" theorem: no single machine–learning method should be expected to perform the best in all cases [271].

In general, these results suggest that higher–dimensional physiochemical representations tend to have better performance than representations incorporating fewer dimensions selected on the basis of high information content. As such, it seems that as long as the training set is a reasonable size, more accurate classifiers can be constructed by keeping as many significant input attributes as possible. Though methods like principal components analysis help to reduce computational complexity for unwieldy datasets, it is better to avoid feature selection until a supervised method (like the models tested in this work) can determine which features are most important in classifying samples.

Table 4.6: Machine learning representation comparison. Each i, j entry represents the number of models, out of six, for which the i representation performed *worse* than the j representation. Here "worse" means that the representation had a statistically significant lower performance, based on a two–tailed t–test at the 0.05 confidence level.

	FULL	CFS	PCA
FULL	-	0	1
CFS	3	-	4
PCA	2	1	-

Table 4.7: Machine learning representation ranking. Each row shows, for each representation, how many other model/representation pairs that representation (with any model) "wins" against. (Thus, the max of the sum of the columns in any row is $18 - 6 = 12$; however, ties are not shown.) Here "win/loss" means that the representation had a statistically significant higher/lower performance, based on a two–tailed t–test at the 0.05 confidence level.

5	1	FULL
5	3	PCA
1	7	CFS

```
CHOP780207_S2' <= 0.41765
|   FAUJ880105_S1 <= 0.57778
|   |   FASG760102_S1 <= 0.27711: uncleaved (32.0/1.0)
|   |   FASG760102_S1 > 0.27711
|   |   |   QIAN880122_S4' <= 0.81022
|   |   |   |   PRAM900101_S1 <= 0.27463
|   |   |   |   |   MEEJ810102_S4 <= 0.33702
|   |   |   |   |   |   RACS820112_S2 <= 0.58621
|   |   |   |   |   |   |   ZIMJ680101_S1' <= 0.52117
|   |   |   |   |   |   |   |   PRAM820101_S2' <= 0.43367
|   |   |   |   |   |   |   |   |   ROSM880103_S3' <= 0.23077: cleaved (2.0)
|   |   |   |   |   |   |   |   |   ROSM880103_S3' > 0.23077
|   |   |   |   |   |   |   |   |   |   CHOP780207_S4 <= 0.21176: cleaved (2.0)
|   |   |   |   |   |   |   |   |   |   CHOP780207_S4 > 0.21176: uncleaved (11.0/1.0)
|   |   |   |   |   |   |   |   PRAM820101_S2' > 0.43367
|   |   |   |   |   |   |   |   |   RADA880105_S2 <= 0.75274
|   |   |   |   |   |   |   |   |   |   PRAM900101_S1 <= 0.06866: cleaved (10.0/2.0)
|   |   |   |   |   |   |   |   |   |   PRAM900101_S1 > 0.06866: uncleaved (4.0)
|   |   |   |   |   |   |   |   |   RADA880105_S2 > 0.75274
|   |   |   |   |   |   |   |   |   |   QIAN880137_S3' <= 0.5124: cleaved (69.0/3.0)
|   |   |   |   |   |   |   |   |   |   QIAN880137_S3' > 0.5124
|   |   |   |   |   |   |   |   |   |   |   RACS820112_S2 <= 0.43103: cleaved (2.0)
|   |   |   |   |   |   |   |   |   |   |   RACS820112_S2 > 0.43103: uncleaved (4.0/1.0)
|   |   |   |   |   |   |   ZIMJ680101_S1' > 0.52117: cleaved (248.0/7.0)
|   |   |   |   |   |   RACS820112_S2 > 0.58621
|   |   |   |   |   |   |   RACS820103_S4 <= 0.43007
|   |   |   |   |   |   |   |   CHAM830104_S3' <= 0
|   |   |   |   |   |   |   |   |   RADA880105_S2 <= 0.75274: uncleaved (5.0/1.0)
|   |   |   |   |   |   |   |   |   RADA880105_S2 > 0.75274: cleaved (2.0)
|   |   |   |   |   |   |   |   CHAM830104_S3' > 0: cleaved (11.0)
|   |   |   |   |   |   |   RACS820103_S4 > 0.43007: uncleaved (6.0)
|   |   |   |   |   MEEJ810102_S4 > 0.33702
|   |   |   |   |   |   GARJ730101_S4' <= 0.01426: uncleaved (9.0)
|   |   |   |   |   |   GARJ730101_S4' > 0.01426
|   |   |   |   |   |   |   CHAM830104_S3' <= 0
|   |   |   |   |   |   |   |   QIAN880102_S4 <= 0.57143: uncleaved (7.0/1.0)
|   |   |   |   |   |   |   |   QIAN880102_S4 > 0.57143: cleaved (3.0)
|   |   |   |   |   |   |   CHAM830104_S3' > 0: cleaved (9.0)
|   |   |   |   PRAM900101_S1 > 0.27463
|   |   |   |   |   GEIM800106_S1' <= 0.94
|   |   |   |   |   |   RACS820102_S3 <= 0.81522
|   |   |   |   |   |   |   FAUJ880108_S2' <= 0.4375: uncleaved (31.0/1.0)
|   |   |   |   |   |   |   FAUJ880108_S2' > 0.4375: cleaved (4.0/1.0)
|   |   |   |   |   |   RACS820102_S3 > 0.81522: cleaved (6.0)
|   |   |   |   |   GEIM800106_S1' > 0.94: cleaved (9.0)
|   |   |   QIAN880122_S4' > 0.81022
|   |   |   |   MITS020101_S1 <= 0.35354
|   |   |   |   |   ZIMJ680101_S1' <= 0.82085: uncleaved (20.0)
|   |   |   |   |   ZIMJ680101_S1' > 0.82085
|   |   |   |   |   |   RACS820102_S3 <= 0.3587: uncleaved (4.0)
|   |   |   |   |   |   RACS820102_S3 > 0.3587: cleaved (5.0)
|   |   |   |   MITS020101_S1 > 0.35354: cleaved (2.0)
|   FAUJ880105_S1 > 0.57778
|   |   QIAN880137_S3' <= 0: cleaved (3.0)
|   |   QIAN880137_S3' > 0: uncleaved (37.0/1.0)
CHOP780207_S2' > 0.41765
|   ZIMJ680101_S1' <= 0.58306: uncleaved (145.0/2.0)
|   ZIMJ680101_S1' > 0.58306
|   |   PRAM900101_S1 <= 0.27463
|   |   |   FAUJ880105_S1 <= 0.57778
|   |   |   |   FAUJ880105_S1 <= 0: uncleaved (2.0)
|   |   |   |   FAUJ880105_S1 > 0
|   |   |   |   |   RACS820103_S3 <= 0.72378
|   |   |   |   |   |   WILM950104_S2 <= 0.44834: uncleaved (5.0)
|   |   |   |   |   |   WILM950104_S2 > 0.44834
|   |   |   |   |   |   |   PRAM820101_S2' <= 0.77041: cleaved (8.0)
|   |   |   |   |   |   |   PRAM820101_S2' > 0.77041: uncleaved (4.0/1.0)
|   |   |   |   |   RACS820103_S3 > 0.72378: cleaved (9.0)
|   |   |   FAUJ880105_S1 > 0.57778: uncleaved (4.0)
|   |   PRAM900101_S1 > 0.27463: uncleaved (12.0)
```

Figure 4-11: The decision tree calculated for the CFS, a 54–dimensional representation of the 8–mer peptides. The branch points are in the form PARAMETER_RESIDUE. For example, `CHOP780207_S2'` represents the AAindex parameter `CHOP780207` (normalized frequency of participation in a C–terminal non–helical region) at the S2' residue. Values for all AAindex parameters are normalized to 1 across all amino acids. The tree shows various questions about a peptide that, when followed, lead to a set of conclusions. For example, if a given peptide has `CHOP780207_S2 <= 0.41765` and `FAUJ880105_S1 > 0.57778` and `QIAN880137_S3 > 0` then the peptide is classified as uncleaved. As shown in the table, 37 of the 746 known peptides are correctly classified by this scheme and only one is incorrectly classified.

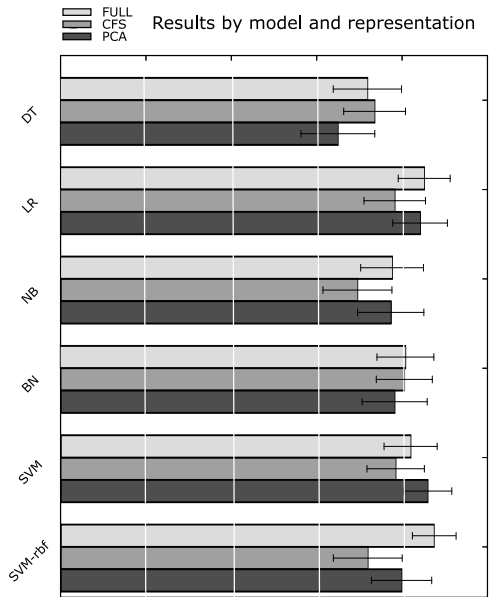


Figure 4-12: Classification results for all amino acid representations and model types. The three different amino acid representations are shown in shades of gray: "FULL" is the full physiochemical, 1241–dimensional representation; "CFS" is the feature–selected, 55–dimensional representation; and "PCA" is the 129–dimensional representation using created using princple component analysis (see text). Error bars show the standard deviation over the 10x10 cross–validation test (100 samples per representation/model combination with a total of 1800 tests.) The best performing model was the SVM with radial basis function (SVM–rbf in the figure) with the full 1241–dimensional feature vector representing each eight–residue sequence. Averaged over all representations, the logistic regression model is best (see Table 4.4 on page 157). The poorest performing model is the decision tree (DT) with the 129–dimensional feature vector created using the PCA projections created as described in the text. In general the full 1241–dimensional representation performed the best, followed by the PCA representation and finally the CFS representation, which was created by a feature selection process.

4.6 Identifying functionally important mutations from phenotypically diverse sequence data

4.6.1 Introduction

In the previous section, I departed from the use of grammar–based models of sequences and explored statistical modeling approaches. This section continues this line of work, but is focused on the identification of important mutations in nucleotide sequences, rather than global, physiochemical characteristics of small peptides. In particular, in this section I present a simple statistical method for parsing out the phenotypic contribution of a single mutation from libraries of functional diversity that contain a multitude of mutations and varied phenotypes. This work is part of a publication that is in press at *Applied and Environmental Microbiology*, which was co–authored with Hal Alper, Curt Fischer, and Gregory Stephanopoulos. Throughout this section, the use of the pronoun "we" refers to these authors.

4.6.2 Motivation

The engineering of functional nucleic acid sequences and other biomolecules is frequently hampered by a limited understanding of how specific mutations at a genotype level are manifested in the phenotype. For some well–studied, large protein families, these relationships can be inferred; however, such cases are rare. In the absence of these relationships, we resort to strategies that explore the genotype space in a random manner, such as directed evolution.

In many cases, directed evolution of genes and other functional DNA loci is an effective approach to sample the sequence space in search of biomolecules with desirable properties [98, 236]. However, the most successful examples employ a selectable fitness criterion that allows for high-throughput screening of the mutational space: sampling a large enough space eliminates the need to make rational mutations. For many proteins or functional nucleic acids, it may not be possible to link a desired phenotype with a selectable criterion, fit for high-throughput screening. In the absence of such a criterion, clonal populations of mutants must be assayed individually for the phenotype of interest. This scenario might be called "assay–based" directed evolution, a situation in which the upstream mutagenesis has a higher throughput than the downstream characterization. In this scenario, there is a premium on information linking mutational changes to their phenotypic manifestations. Further, there is a strong incentive to "learn from" the (relatively small) mutational spectra of these mutants to determine sequence-phenotype interactions, and to use this information rationally in subsequent rounds of mutagenesis.

Here, we present a simple statistical method for analyzing a mutational spectrum to parse out the phenotypic manifestation of individual mutations, even when they are masked by the presence of many other mutations. Because assay-based directed evolution does not employ any pre–screening or selection of clones, as is the case when a selectable marker is available, mutants are expected to have a range of phenotypes, including both increased and decreased fitness. Here, we demonstrate our method by identifying mutations in a library of mutagenized PL–λ promoters [8] that result in either increased or decreased promoter activity and we show how to quantify the statistical confidence in these mutation-phenotype linkages

The central premise of our method is that mutations that have no effect on mutant phenotype should partition randomly, following a multinomial distribution, between phenotypic classes. For example, consider a hypothetical experiment in which we mutagenize a protein that can fluoresce in one of three colors: red, blue, or green. After generating a library of 1000 mutants, each bearing many point mutations, our assay reveals that 600 have the red phenotype, 300 are blue, and 100 are green. If a particular point mutation has no effect on the color, then we expect that, by chance, mutants containing this modification will be distributed between the red, blue, and green classes in a ratio of 6:3:1. That is, the mutation should not be correlated to any particular phenotypic class. More rigorously, we say that the mutations are multinomially distributed between the three classes with background frequencies 0.6, 0.3, and 0.1.

Multinomial statistics and related combinatorial statistics commonly arise in the analysis of naturally–occurring mutational diversity [2, 196]. For example, similar statistical analyses have been used to find functional gene domains [157], important structural RNA sites [131], and genomic loci with an overabundance of single nucleotide polymorphisms (SNPs) [259]. Here we apply multinomial statistics to the analysis of an artificially generated mutational landscape to parse out critical residues controlling phenotypic behavior. We show that, based on this information, mutants with sets of individual mutations can be made, and we suggest that this can be used as a method for improving directed evolution experiments by incorporating sequence information.

In what follows, we detail the construction of numerous PL–λ promoter variants, which were generated by error–prone PCR such that each mutant incorporated many point mutations. The activity of these promoters was assayed using flow cytometry to measure the fluorescence of a GFP reporter gene. We show how our statistical analysis revealed the phenotypic manifestation of numerous mutations. Finally, we present a validation of our method by constructing point mutations for several of the identified mutations and combinations of sites using site-directed mutagenesis. These mutations, we show, have the predicted effect on the promoter phenotype, even when removed from the background of other mutations.

4.6.3 Materials and Methods

Strains and Media

E. coli DH5α (Invitrogen) was used for routine transformations as described in the protocol. Assay strains were grown at 37°C with 225 RPM orbital shaking in M9-minimal media (11) containing 5 g/L D–glucose and supplemented with 0.1% casamino acids. All other strains and propagations were cultured at 37°C in LB media. Media was supplemented with 68 μg/ml chloramphenicol. All PCR products and restriction enzymes were purchased from New England Biolabs and utilized Taq polymerase. M9 Minimal salts were purchased from US Biological and all remaining chemicals were from Sigma-Aldrich.

Library Construction

Nucleotide analogue mutagenesis was carried out in the presence of 20 μM 8–oxo–2'–deoxyguanosine (8–oxo–dGTP) and 6–(2–deoxy–β–D–ribofuranosyl)–3,4–dihydro–8H–pyrimido–[4,5–c][1,2]oxazin–7–one (dPTP) (TriLink Biotech), using plasmid pZE–gfp(ASV) kindly provided by M. Elowitz

as template [158] along with the primers PL_sense_AatII, TCCGACGTCTAAGAAACCATTATTATC and PL_anti_EcoRI, CCGGAATTCGGTCAGTGCGTCCTGCTGAT. Ten and 30 amplification cycles with the primers mentioned above were performed. The 151 bp PCR products were purified using the GeneClean Spin Kit (Qbiogene). Following digestion with AatII and EcoRI, the product was ligated overnight at 16°C and transformed into library efficiency *E. coli* DH5α (Invitrogen). About 30,000 colonies were screened by eye from minimal media–casamino acid agar plates and 200 colonies, spanning a wide range in fluorescent intensity, were picked from each plate. Selected mutants were sequenced using primers PL_Sense_Seq, AGATCCTTGGCGGCAAGAAA and PL_Anti_Seq, GCCATGGAACAGGTAGTTTTCCAG.

Library Characterization

About 20 µL of overnight cultures of library clones growing in LB broth were used to inoculate 5mL M9G medium supplemented with 0.1% w/v casamino acid (M9G/CAA). The cultures were grown at 37°C with orbital shaking. After 14 h, roughly the point of glucose depletion, a culture sample was centrifuged at 18,000 g for 2 minutes, and the cells were resuspended in ice–cold water. Flow cytometry was performed on a Becton–Dickinson FACScan as described elsewhere [8], and the geometric mean of the fluorescence distribution of each clonal population was calculated.

Mean and standard deviation were calcuated from the FL1–H distribution resulting after gating the cells based on a FSC–SSC plot. A total of 200,000 events were counted to gain statistical confidence in the results

Construction of designed promoters

Promoters with specific nucleotide changes were created using overlap–extension PCR and primers specifically designed to incorporate these changes. Primers were designed to divide the promoter region into thirds, and the proper primers were assembled piecewise in a PCR reaction consisting of 95°C for 4 minutes, 10 cycles with an annealing temperature of 44°C, followed by 30 cycles of PCR with an annealing temperature of 60°C, and a final extension for 3 minutes at 72°C. Fragments were gel extracted using 2.5% agarose gels and Qiagen MERmaid spin kit. The isolated fragment was then linked with the final primer using the same PCR and extraction procedures. Finalized fragments were then digested using EcoRI and AatII and ligated into the digested plasmid backbone. Sequencing was performed to verify correct constructs.

4.6.4 Results

Generation of mutant library

Previously, we reported on the development of a promoter library generated through the random mutagenesis of the sequence space [8]. In that work, library diversity was created through error–prone PCR of the PL–TET01 promoter, a variant of the PL–λ promoter [59], which was placed upstream of a gfp gene. The promoter region contains two tandem promoters PL–1 and PL–2, each of which contains -10 and -35 sigma factor binding sites [95–97]. Furthermore, the promoter contains, at approximately the same location, an UP element that binds

C–terminal domain of the alpha subunit and a binding site for integration host factor (IHF). In addition, the PL–TET01 promoter has two tetO2 operators from the Tn10 tetracycline resistance operon [158].

Mutants in the library were analyzed using flow–cytometry to measure the single–cell level of expression of GFP as a proxy for the activity of the mutagenized promoters. (A detailed schematic of the experimental procedure is shown in Figure 4-14 on page 166.) Promoters that had roughly log–normal fluorescence distributions (no obvious tails in the distribution or bimodal distributions) were sequenced and, from that set, those mutants that contained deletions or insertions were removed. The final set comprised 69 mutant promoters, with well–behaved fluorescence distributions (single distribution with a low standard deviation) , that only contained transition and transversion mutations. Notably, our error–prone PCR method introduces predominantly transitions and not transversions, except in rare cases.

Identification of critical sites

Returning to the red, blue, green example introduced earlier, each of these N hypothetical mutants can be classified into one of M mutually–exclusive and collectively–exhaustive phenotypic classes — $P_1, P_2, \ldots, P_M$ — such that there are $n_1, n_2, \ldots, n_M$, mutants in each class and $\sum n_i = N$. Consider a subset of mutants B of size X, where $X < N$, comprising mutants with a particular mutation. If the mutation does not influence the phenotype of the mutants, we would expect, by chance, that there would be $x_i = X(n_i/N)$ mutants of type P_i. In general, the probability that the set $x_1, x_2, \ldots, x_M$ will take on the particular set of values $y_1, y_2, \ldots, y_M$ is

$$\Pr(x_1 = y_1, x_2 = y_2, \ldots, x_M = y_M) = \binom{X}{y_1, y_2, \ldots, y_M} \prod_{i=1}^{M} \frac{n_i}{N} \tag{4.1}$$

where $\sum y_i = X$. In this equation, the term

$$\Pr(x_1 = y_1, x_2 = y_2, \ldots, x_M = y_M) = \binom{X}{y_1, y_2, \ldots, y_M} \prod_{i=1}^{M} \frac{n_i}{N} \tag{4.2}$$

is the so–called multinomial coefficient, which can be equivalently written

$$\binom{X}{y_1, y_2, \ldots, y_M} = \frac{X!}{y_1!, y_2!, \ldots, y_M!}. \tag{4.3}$$

The coefficient is the number of ways sets of size $y_1, y_2, ..., y_M$ could be chosen from a set of size X. (For example, in the case $X = 6, M = 2, y_1 = y_2 = 3$, the coefficient is 20 because there are 20 different ways to choose two subsets of size three from a set of six.)

The probability that q or more (where $q < X$) of the B mutants would be seen in a particular class, P_i, by chance is

$$\Pr(x_i \geqslant q) = \sum_{k=q}^{X} \binom{X}{q} \left(\frac{n_i}{N}\right)^k \left(1 - \frac{n_i}{N}\right)^{N-k}. \tag{4.4}$$

Equivalently, this is the p–value for seeing q of the B mutants in class P_i. The lower the

Figure 4-13: Structure of the PL–TET01 promoter. There are a numerous functional sites on the PL–TET01 promoter that are known to effect the rate of complex formation between the promoter and RNA polymerase [95–97]. The promoter region contains two tandem promoters PL–1 and PL–2, each of which contain -10 and -35 sigma factor binding sites. Furthermore, the promoter contains, at approximately the same location, an UP element that binds C–terminal domain of the alpha subunit and a binding site for integration host factor (IHF) a global regulator of gene expression in *E. coli*. The IHF site acts to bend the promoter region, brining the alpha–CTD binding site in sufficient proximity to the beta subunit of the RNA polymerase. In addition, the PL–TET01 promoter has two tetO2 operators from the Tn10 tetracycline resistance operon [158].

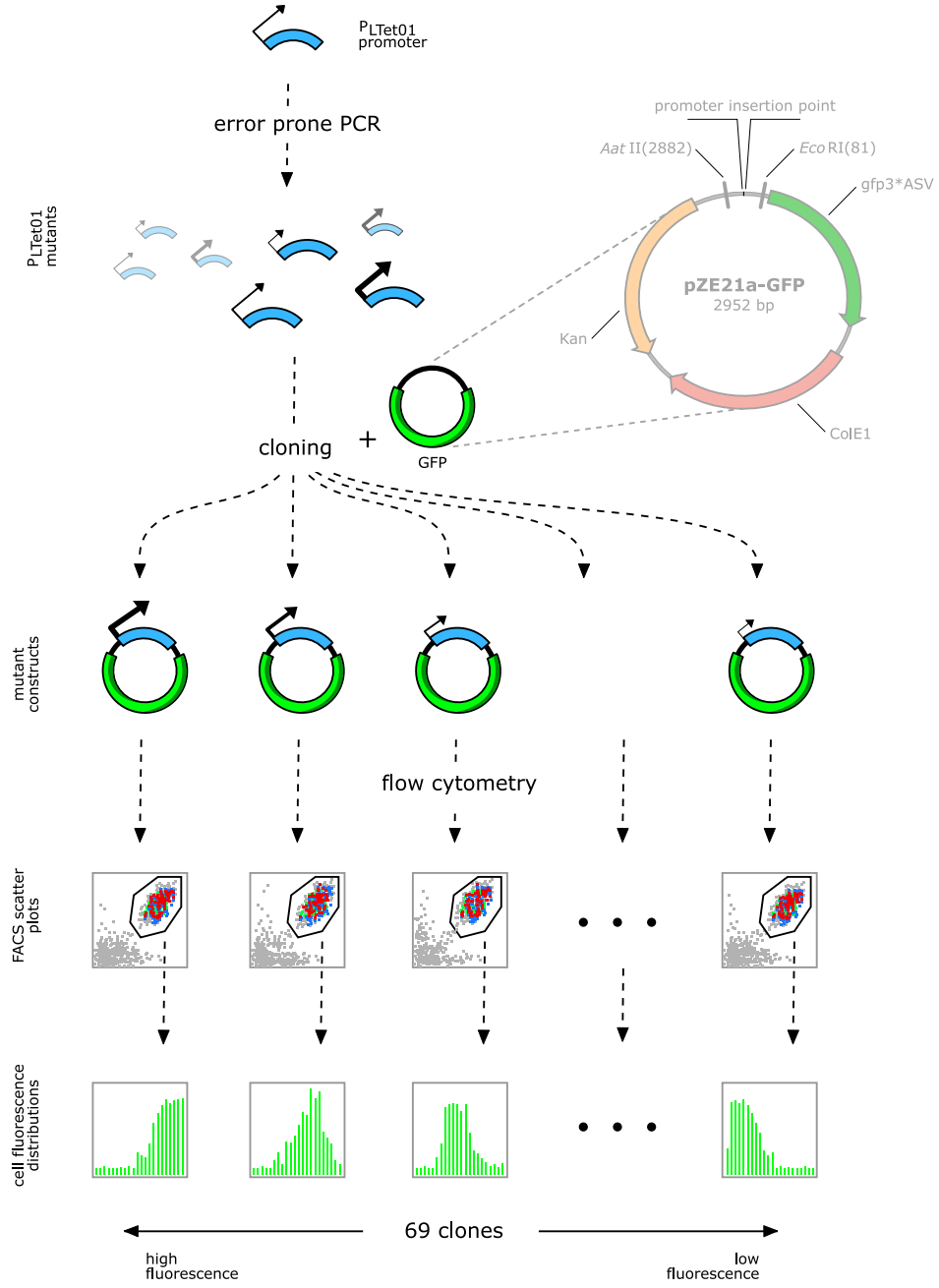


Figure 4-14: Schematic of the experimental procedure. A variant of the constitutive bacteriophage PL–λ promoter (PL–TETO1) was mutated through error–prone PCR to create mutated fragments of promoters. These fragments were then ligated into plasmid constructs and used to drive the expression of *gfp* in *E.coli*. These cells were then analyzed using flow cytometry to quantify the fluorescence of GFP and output capacity of the promoter.

p–value, the more confident we are that the B mutation is correlated with the P_i phenotype.

For this study, we divided the mutants into two phenotypic classes based on their fluorescence (i.e. $M = 2$): the top 50th percentile and the lower 50th percentile. Figure 4-15 on the following page shows a detailed schematic of the statistical analysis, which is greatly simplified in this case because there are only two phenotypes. As shown in the figure, applying our statistical method to the sequence data resulted in the identification of seven nucleotide positions that are correlated with one of the two phenotypic classes in a statistically significant manner. The figure should be read clockwise from the top–left, progressively showing the fluorescence distribution, mutation distribution, statistical distribution of mutations, and finally, the identified important positions in part D in the lower left. In quadrant A, the vertical axis shows the mutant number, where the mutants are sorted in descending order by their relative fluorescence. In general, the single-cell fluorescence distribution for each mutant strain was log-normal distributed. The horizontal axis shows the mean of the log relative fluorescence for each mutant strain, where the error is the standard deviation of this distribution. Reading to the right from quadrant A into quadrant B reveals the point mutations present in each mutant. For each location in a mutant (where location is indicated on the horizontal axis) that was changed via the error-prone PCR, a black dot is indicated. With only a handful of exceptions, all of these changes are base transitions rather than transversions, so the sequence of each of the 69 clones can be inferred from the WT sequence shown in quadrant D. Reading down from quadrant B into quadrant C shows how mutations at a particular location partition between the two classes of mutants: the top and bottom 50th percentiles. Sites that have no effect on the fluorescence phenotype should partition equally between the two classes, i.e. they should follow a binomial distribution with $p = 0.5$. Sites that deviate from this distribution are labeled with a dot and are colored either green or red, corresponding to the apparent effect of a mutation at the site. For these sites, p–values are indicated, where this value is the probability of seeing a distribution at least as skewed to one side. Sites that were subsequently tested experimentally (see below) are indicated with an asterisk, where the color of the asterisk denotes the expected effect of a mutation at the site. We chose a range of sites to test experimentally, from those with high-confidence (low p–value) positive effects, to those with low-confidence (p–value ~ 0.5) negative effects (see Table 4.8 on page 171). These sites are also shown in quadrant D, which contains the WT nucleotide sequence of the promoter region that was subjected to mutation.

Site–directed mutagenesis of predicted sites

We selected 8 sites in the promoter region to test whether their phenotypic effects, as predicted by the statistical method, agreed with their observed effects when the mutations were introduced individually, without the background of other mutations. These 8 mutated positions are shown in Table 4.8 on page 171 and labeled in Figure 4-15 on the next page, parts C&D. The sites were chosen to span a range of characteristics. The -8 site was predicted to have a negative effect on promoter strength with high confidence, i.e. it was statistically significant (see Table 4.8). The -10, -28, and -123 sites were predicted to have negative effects, but had moderate p–values and, thus, medium–to–low confidence. Sites -14 and -21 were predicted to have positive effects with high confidence. The sites -82 and -96 were chosen because they had p–values of exactly 0.5. Notably, there are two ways that a position could have produced an insignificant p–value (i.e. a p–value close to 0.5): the mutation could partition equally between the two classes,

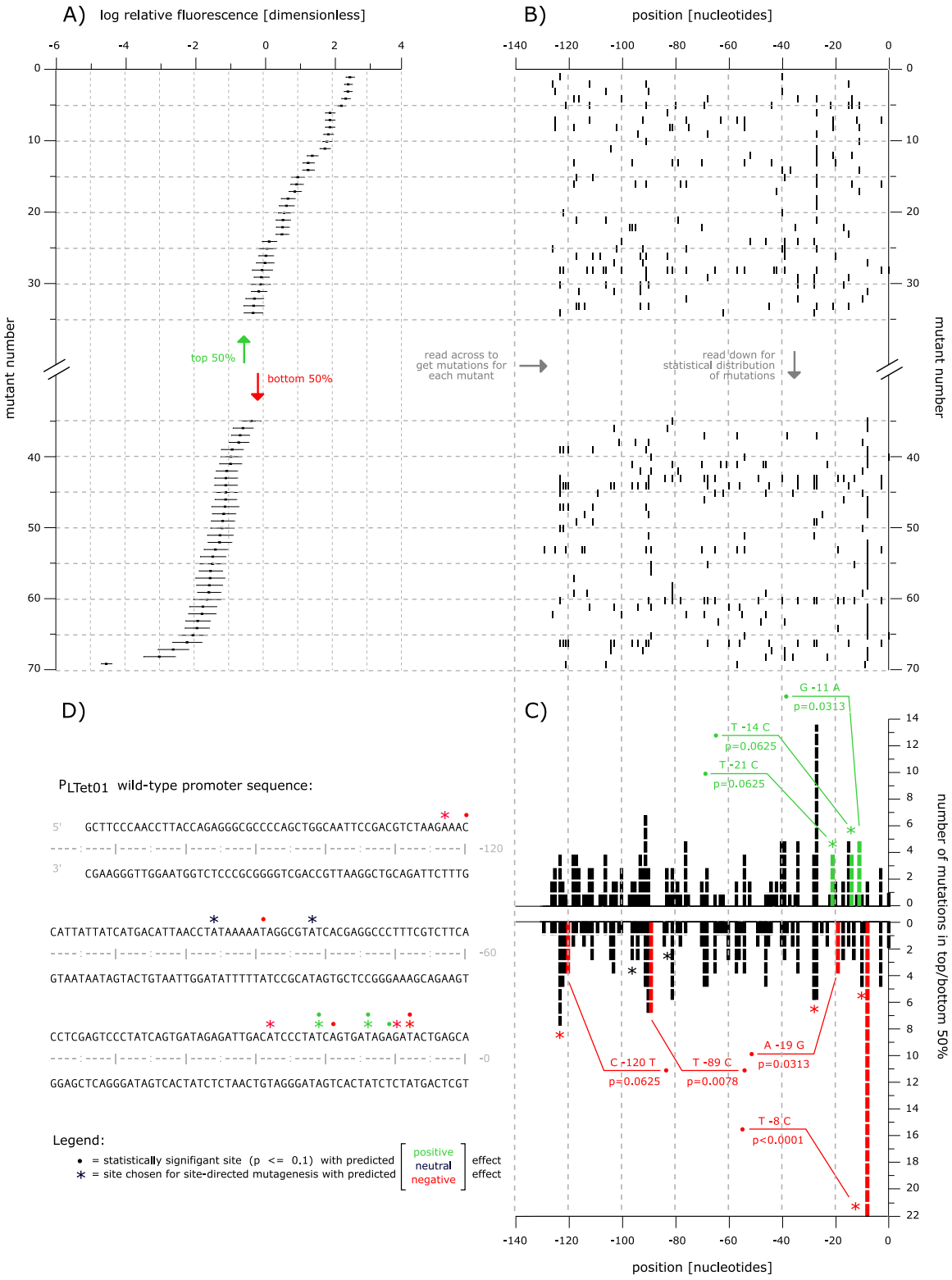


Figure 4-15: Statistical distribution of mutations and their effects on mutant fluorescence. See text for a description.

or the mutation could have been observed very few times. Mutations at both the -82 and -96 sites were observed relatively few times and seemed to partition between the top–50th percentile and bottom–50th percentile classes with equal frequency. Thus, in the absence of a statistically significant correlation, we predicted they would have no effect on the phenotype. (These observations are summarized in Table 4.8 on page 171.)

For each of the sites listed in Table 4.8 on page 171, we created mutant strains incorporating transition SNPs at the specified location. Each of these mutants were analyzed using flow-cytometry to test the single-cell level of expression of GFP using the same protocols as for the parent mutant library. The fluorescence results for each mutant are shown in Table 4.8 on page 171 in the right-most columns. In addition, for certain combinations of sites in Table 4.8 on page 171, we created double and triple mutants (see Table 4.9 on page 172).

4.6.5 Discussion

As shown in Table 4.8 on page 171, the statistical method correctly predicts the phenotypic effects of 7/8 of the individual mutations that were tested. Furthermore, the phenotypic effects of the mutations with statistically significant p–values were correctly predicted. For these mutations, we showed that the effect of an individual mutation on the phenotype can be parsed out from a mutational spectrum, even when the effect is obscured by a background of other mutations.

It is interesting to note that while most of the statistically significant mutations are near the sigma factor binding sites, two are located further upstream of this region. The -123 site, which was not statistically significant, but was tested experimentally, showed that such distal sites are participating in the regulation of transcription.

There are a few caveats to the use of our statistical method. First, the method assumes independence between mutations. That is, we assume mutated sites cannot interact. As shown in Table 4.9 on page 172, 4/6 of the combination–mutations had the predicted effect. The two combination–mutants that had unintuitive phenotypes could be a result of interaction between sites. (Notably, the -82,-14,-21 triple mutant appeared to have a high fluorescence by visual inspection in a rich medium pre–culture; however, quantification of GFP activity by flow cytometry revealed consistently low measurements in the minimal medium used.)

The second caveat is that the method can require a significant number of mutants for each position: for a position to be statistically significant in our particular experiment, at least 4 observations were required. (This would be true for any two–phenotype mutational spectra, where each phenotype occurs with equal prior probability.) The number of observations required scales roughly with the number of mutation types. Our mutagenesis method introduced only transitions, not transversions, which allowed us to treat each site as "mutated" or "not mutated" without loss of information. The method can by applied to cases in which all four nucleotides are present; however, roughly 4 times as many observations would be required to make a statistically significant correlation between a particular nucleotide (at a single position) and a phenotype. Finally, the statistical method presented here is only applicable to situations in which the method used to introduce sequence diversity does not also introduce deletions or insertions. Ignoring relatively small insertions or deletions in the analysis would not significantly bias the results of identifying critical residues (data not shown). However, rigorously, alterations would be needed to differentiate between deletions and mutations in our statistical

framework. In such cases, more complex models could be adapted, such as those used to describe the distribution and effects of naturally–occurring mutations over a fitness landscape for populations under positive and negative selective pressures [186, 212].

Despite its caveats, this method has a significant advantage when compared to deducing critical mutations using sequence data from only the best performing mutants. Intuitively, if we were to ignore the bottom–50th percentile in Figure 4-15 part C on page 168, we may mistakenly identify sites as associated with high fluorescence that are, in fact, evenly distributed between the two classes. That is, having sequence data for multiple phenotypes allowed us to determine, with quantifiable confidence, the effect of each individual mutation in a way that discounts artifacts of the mutagenesis method, such as a bias for mutagenizing particular loci.

Table 4.8: Summary of site–directed mutagenesis loci. The selected sites, which span a range of p–values and predicted activities, were each mutated and assayed for fluorescence levels individually (see Figure 4-15 on page 168. As shown in the table, all sites but the -96 site were in the phenotypic class predicted by our statistical method.

Site	Predicted activity	P–value	Observations	Confidence	Relative Fluorescence	Log Relative Fluorescence	Agreement?
-8	Low	<0.0001	22	High	0.036	-3.32	Yes
-10	Low	0.1094	6	Med	0.011	-4.52	Yes
-14	High	0.0625	4	High	1.428	0.35	Yes
-21	High	0.0625	4	High	1.585	0.46	Yes
-28	Low	0.3770	10	Low	0.756	-2.58	Yes
-82	No effect	0.5000	2	Low	0.926	-0.08	Yes
-96	No effect	0.5000	5	Med	0.046	-3.08	No
-123	Low	0.1938	12	Med	0.087	-2.45	Yes

Table 4.9: Summary of double and triple mutants constructed by site–directed mutagenesis.

Sites	Predicted activity	Relative Fluorescence	Log Relative Fluorescence	Agreement?
-14, -21	High	1.924	0.65	Yes
-14, -82	High	0.954	-0.04	No
-21, -82	High	1.433	0.36	Yes
-96, -123	Low	0.274	-1.43	Yes
-82,-14,-21	High	0.140	-1.97	No
-8,-10,-28	Low	0.018	-4.03	Yes

Bibliography

[1] *GNU Scientific Library Reference Manual - Second Edition*. Network Theory Ltd., 2003. 100, 183

[2] W. T. Adams and T. R. Skopek. Statistical test for the comparison of samples from mutational spectra. *J Mol Biol*, 194(3):391–6, Apr 1987. 162

[3] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey, 1979. ISBN 0-13-914556-7. 26

[4] N. N. Alexandrov. SARFing the PDB. *Protein Eng*, 9(9):727–732, Oct 1996. 111

[5] N. N. Alexandrov and D. Fischer. Analysis of topological and nontopological structural similarities in the PDB: new examples with old structures. *Proteins*, 25(3):354–365, Aug 1996. 111

[6] F. Alimoglu and E. Alpaydin. Methods of combining multiple classifiers based on different representations for pen–based handwriting recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*, Istanbul, Turkey, 1996. 148

[7] F. Alimoglu and E. Alpaydin. Combining multiple representations and classifiers for pen–based handwritten digit recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Ulm, Germany, 1997. 148

[8] H. Alper, C. Fischer, E. Nevoigt, and G. Stephanopoulos. Tuning genetic control through promoter engineering. *Proc Natl Acad Sci U S A*, 102(36):12678–83, Sep 2005. 161, 163

[9] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J Mol Biol*, 219:555–65, 1991. 134, 149

[10] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215:403–10, 1990. 127, 130, 134, 142

[11] S. F. Altschul, T. L. Madden, A. A. S. Zhang, J., Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–402, 1997. 73, 123, 142

[12] D. Amsterdam. *Susceptibility testing of antimicrobials in liquid media*, pages 52–111. Antibiotics in Laboratory Medicine. Williams & Wilkins, Baltimore, MD, 4th edition, 1996. 73

[13] L. Aravind and E. V. Koonin. The HD domain defines a new superfamily of metal-dependent phosphohydrolases. *Trends Biochem Sci*, 23(12):469–472, 1998. 108

[14] P. Argos, J. K. Rao, and P. A. Hargrave. Structural prediction of membrane-bound proteins. *Eur J Biochem*, 128(2-3):565–575, Nov. 1982. 94

[15] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987. ISSN 0162-8828. 111

[16] W. R. Atchley, J. Zhao, A. D. Fernandes, and T. Drüke. Solving the protein sequence metric problem. *Proc Natl Acad Sci U S A*, 102(18):6395–400, May 2005. 152

[17] T. K. Attwood, P. Bradley, D. R. Flower, A. Gaulton, N. Maudling, A. L. Mitchell, G. Moulton, A. Nordle, K. Paine, P. Taylor, A. Uddin, and C. Zygouri. PRINTS and its automatic supplement, prePRINTS. *Nucleic Acids Res*, 31:400–2, 2003. 127

[18] P. Bagossi, T. Sperka, A. Fehér, J. Kádas, G. Zahuczky, G. Miklóssy, P. Boross, and J. Tözsér. Amino acid preferences for a critical substrate binding subsite of retroviral proteases in type 1 cleavage sites. *J Virol*, 79(7):4213–4218, Apr 2005. URL `http://www.hubmed.org/display.cgi?uids=15767422`. 153

[19] T. L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol*, 2:28–36, 1994. 53, 90, 118, 124

[20] A. Bairoch. Prosite: a dictionary of sites and patterns in proteins. *Nucleic Acids Res*, 19 Suppl:2241–2245, Apr 1991. URL `http://www.hubmed.org/display.cgi?uids=2041810`. 131

[21] A. Bairoch. The ENZYME database in 2000. *Nucleic Acids Res*, 28(1):304–305, Feb 2000. 108, 109

[22] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res*, 28(1):45–48, Jan 2000. 60, 108, 128, 142

[23] A. Bairoch and B. Boeckmann. The swiss-prot protein sequence data bank. *Nucleic Acids Res*, 20 Suppl:2019–2022, May 1992. URL `http://www.hubmed.org/display.cgi?uids=1598233`. 131

[24] C. Barillas-Mury, B. Wizel, and Y. S. Han. Mosquito immune responses and malaria transmission: lessons from insect model systems and implications for vertebrate innate immunity and vaccine development. *Insect Biochem Mol Biol*, 30(6):429–442, June 2000. 58

[25] A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy. The Pfam protein families database. *Nucleic Acids Res*, 32 Database issue:138–141, Feb 2004. 108

[26] Z. Beck, L. Hervio, P. Dawson, J. Elder, and E. Madison. Identification of efficiently cleaved substrates for HIV-1 protease using a phage display library and use in inhibitor development. *Virology*, 274(2):391–401, Sep 2000. URL `http://www.hubmed.org/display.cgi?uids=10964781`. 153

[27] Z. Beck, Y. Lin, and J. Elder. Molecular basis for the relative substrate specificity of human immunodeficiency virus type 1 and feline immunodeficiency virus proteases. *J Virol*, 75(19):9458–9469, Oct 2001. URL `http://www.hubmed.org/display.cgi?uids=11533208`. 153

[28] Z. Beck, G. Morris, and J. Elder. Defining HIV-1 protease substrate selectivity. *Curr Drug Targets Infect Disord*, 2(1):37–50, Mar 2002. URL `http://www.hubmed.org/display.cgi?uids=12462152`. 153

[29] G. Bell and P.-H. Gouyon. Arming the enemy: the evolution of resistance to self-proteins. *Microbiology*, 149(Pt 6):1367–1375, Jun 2003. 87

[30] K. Bennett and E. Bredensteiner. Duality and gemoetry in svms. In P. Langley, editor, *Proc. of 17th international Conference on Machine Learning*, pages 65–72. Morgan Kaufmann, 2000. 156

[31] K. P. Bennett and C. Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000. URL `citeseer.ist.psu.edu/bennett03support.html`. 156

[32] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman , J. Ostell, B. A. Rapp, and D. L. Wheeler. GenBank. *Nucleic Acids Res*, 28:15–8, 2000. 142

[33] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Res*, 34(Database issue):D16–20, Jan 2006. 15, 17

[34] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Res*, 28(1):235–242, Feb 2000. 12, 152

[35] C. Bi and P. K. Rogan. Bipartite pattern discovery by entropy minimization-based multiple local alignment. *Nucleic Acids Res*, 32(17):4979–91, 2004. 53

[36] C. C. Bigelow. On the average hydrophobicity of proteins and the relation between it and protein structure. *J Theor Biol*, 16(2):187–211, Aug 1967. 151

[37] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of Basic Linear Algebra Subprograms (BLAS). *ACM Transactions on*

Mathematical Software, 28(2):135–151, June 2002. ISSN 0098-3500. URL http://doi.acm.org/10.1145/567806.567807. 100, 183

[38] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html. 143, 146

[39] D. Boden and M. Markowitz. Resistance to human immunodeficiency virus type 1 protease inhibitors. *Antimicrob Agents Chemother*, 42(11):2775–2783, Nov 1998. URL http://www.hubmed.org/display.cgi?uids=9797203. 152

[40] H. G. Boman. Antibacterial peptides: basic facts and emerging concepts. *J Intern Med*, 254(3):197–215, Sep 2003. 58

[41] Brazma, Jonassen, Vilo, and Ukkonen. Pattern discovery in biosequences. In *ICGI: International Colloquium on Grammatical Inference and Applications*, 1998. URL citeseer.nj.nec.com/brazma98pattern.html. 46

[42] S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for protein structure and sequence analysis. *Nucleic Acids Res*, 28(1):254–6, Jan 2000. 131, 133

[43] J. Bresnan, R. M. Kaplan, S. Peters, and A. Zaenen. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635, 1982. Reprinted in W. Savitch et al. (eds) The Formal Complexity of Natural Language, 286–319. Dordrecht: D. Reidel. 32

[44] A. Brik and C. Wong. HIV-1 protease: mechanism and drug discovery. *Org Biomol Chem*, 1(1):5–14, Jan 2003. URL http://www.hubmed.org/display.cgi?uids=12929379. 152

[45] J. Buhler and M. Tompa. Finding motifs using random projections. In *Proceedings of the fifth annual international conference on Computational biology*, pages 69–76. ACM Press, 2001. ISBN 1-58113-353-7. 90, 117

[46] J. Buhler and M. Tompa. Finding motifs using random projections. *J Comput Biol*, 9 (2):225–42, 2002. 47, 117, 125

[47] H. J. Bussemaker, H. Li, and E. D. Siggia. Building a dictionary for genomes: identification of presumptive regulatory sites by statistical analysis. *Proc Natl Acad Sci U S A*, 97(18):10096–100, Aug 2000. 47

[48] Y.-D. Cai, X.-J. Liu, X.-B. Xu, and K.-C. Chou. Support Vector Machines for predicting HIV protease cleavage sites in protein. *J Comput Chem*, 23(2):267–274, Jan 2002. 153

[49] Y. D. Cai, H. Yu, and K. C. Chou. Artificial neural network method for predicting HIV protease cleavage sites in protein. *J Protein Chem*, 17(7):607–15, Oct 1998. 153

[50] G. H. Cassell and J. Mekalanos. Development of antimicrobial agents in the era of new and reemerging infectious diseases and increasing antibiotic resistance. *JAMA*, 285: 601–5, 2001. 57

[51] P. Casteels and P. Tempst. Apidaecin-type peptide antibiotics function through a non-poreforming mechanism involving stereospecificity. *Biochem Biophys Res Commun*, 199 (1):339–345, Feb 1994. 76

[52] Y. Chen, X. Xu, S. Hong, J. Chen, N. Liu, C. B. Underhill, K. Creswell, and L. Zhang. RGD-Tachyplesin inhibits tumor growth. *Cancer Res*, 61(6):2434–2438, Mar 2001. 58

[53] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956. 25, 31

[54] N. Chomsky. *Syntactic Structures*. Mouton and Co., The Hague, 1957. 25

[55] N. Chomsky. *Aspects of the theory of syntax*. MIT Press, Cambridge, Massachusetts, 1965. 25

[56] K. C. Chou. Prediction of human immunodeficiency virus protease cleavage sites in proteins. *Anal Biochem*, 233(1):1–14, Jan 1996. 153

[57] P. Y. Chou and G. D. Fasman. Prediction of the secondary structure of proteins from their amino acid sequence. *Adv Enzymol Relat Areas Mol Biol*, 47:45–148, 1978. 151

[58] G. K. Christophides, E. Zdobnov, C. Barillas-Mury, E. Birney, S. Blandin, C. Blass, P. T. Brey, F. H. Collins, A. Danielli, G. Dimopoulos, C. Hetru, N. T. Hoa, J. A. Hoffmann, S. M. Kanzok, I. Letunic, E. A. Levashina, T. G. Loukeris, G. Lycett, S. Meister, K. Michel, L. F. Moita, H.-M. Muller, M. A. Osta, S. M. Paskewitz, J.-M. Reichhart, A. Rzhetsky, L. Troxler, K. D. Vernick, D. Vlachou, J. Volz, C. von Mering., J. Xu, L. Zheng, P. Bork, and F. C. Kafatos. Immunity–related genes and gene families in Anopheles gambiae. *Science*, 298(5591):159–165, Oct. 2002. 58

[59] P. C. Cirino, K. M. Mayer, and D. Umeno. Generating mutant libraries using error-prone PCR. *Methods Mol Biol*, 231:3–9, 2003. 163

[60] J. C. Clemente, R. E. Moose, R. Hemrajani, L. R. S. Whitford, L. Govindasamy, R. Reutzel, R. McKenna, M. Agbandje-McKenna, M. M. Goodenow, and B. M. Dunn. Comparing the accumulation of active- and nonactive-site mutations in the HIV-1 protease. *Biochemistry*, 43(38):12141–51, Sep 2004. 153

[61] F. S. Collins, M. Morgan, and A. Patrinos. The Human Genome Project: lessons from large-scale biology. *Science*, 300(5617):286–90, Apr 2003. 16

[62] D. J. Crisp and C. J. C. Burges. A geometric interpretation of v-svm classifiers. In *NIPS*, pages 244–250, 1999. 156

[63] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-78019-5. 156

[64] L. V. Danilova, V. A. Lyubetsky, and M. S. Gelfand. An algorithm for identification of regulatory signals in unaligned DNA sequences, its testing and parallel implementation. *In Silico Biol*, 3(1–2):33–47, 2003. 47

[65] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Structure*, volume 5(Suppl. 3), pages 345–352. National Biomedical Reasearch Foundataion, Silver Spring, Md., 1978. 130, 149, 151, 152

[66] S. Dietmann and L. Holm. Identification of homology in protein structure classification. *Nat Struct Biol*, 8(11):953–957, Nov 2001. 111

[67] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error–correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 149

[68] J. Dongarra. Preface: Basic Linear Algebra Subprograms Technical (Blast) Forum Standard I. *The International Journal of High Performance Computing Applications*, 16(1): 1–111, Spring 2002. ISSN 1094-3420. 100, 183

[69] J. Dongarra. Preface: Basic Linear Algebra Subprograms Technical (Blast) Forum Standard II. *The International Journal of High Performance Computing Applications*, 16(2): 115–199, Summer 2002. 100, 183

[70] T. A. Down and T. J. P. Hubbard. NestedMICA: sensitive inference of over-represented motifs in nucleic acid sequence. *Nucleic Acids Res*, 33(5):1445–53, 2005. 53

[71] M. Dsouza, N. Larsen, and R. Overbeek. Searching for patterns in genomic data. *Trends Genet*, 13:497–8, 1997. 128

[72] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998. 99

[73] I. Eidhammer, I. Jonassen, and W. R. Taylor. Structure comparison and structure patterns. *J Comput Biol*, 7(5):685–716, 2000. 108, 111

[74] H. M. Ellerby, W. Arap, L. M. Ellerby, R. Kain, R. Andrusiak, G. D. Rio, S. Krajewski, C. R. Lombardo, R. Rao, E. Ruoslahti, D. E. Bredesen, and R. Pasqualini. Anti-cancer activity of targeted pro-apoptotic peptides. *Nat Med*, 5(9):1032–1038, Sep 1999. 58

[75] R. M. Epand and H. J. Vogel. Diversity of antimicrobial peptides and their mechanisms of action. *Biochim Biophys Acta*, 1462:11–28, 1999. 58, 59

[76] Eric Sayers and David Wheeler. Building Customized Data Pipelines Using the Entrez Programming Utilities. `http://www.ncbi.nlm.nih.gov/books/`, Accessed in Dec 2005. 20

[77] E. Eskin. From profiles to patterns and back again: a branch and bound algorithm for finding near optimal motif profiles. In *RECOMB '04: Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 115–124, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-755-9. 52, 53

[78] E. Eskin and P. A. Pevzner. Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, 18 Suppl 1:354–363, 2002. Evaluation Studies. 47, 90, 123

[79] G. Fasman, editor. *Physical Chemical Data*, volume 1 of *CRC Handbook of Biochemistry and Molecular Biology*. CRC Press, Cleveland, Ohio, 1976. 154

[80] J. L. Fauchère, M. Charton, L. B. Kier, A. Verloop, and V. Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *Int J Pept Protein Res*, 32(4):269–78, Oct 1988. 154

[81] J. Felsenstein. Phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989. 145

[82] A. Floratos. *Pattern Discovery in Biology: Theory and Applications*. PhD thesis, New York University, New York, Jan. 1999. 46, 47, 48

[83] G. B. Fogel, D. G. Weekes, G. Varga, E. R. Dow, H. B. Harlow, J. E. Onyia, and C. Su. Discovery of sequence motifs related to coexpression of genes using evolutionary computation. *Nucleic Acids Res*, 32(13):3826–35, 2004. 47

[84] J. E. Friedl. *Mastering Regular Expressions*. O'Reilly & Associates, Inc., Sebastopol, California, 1997. 35

[85] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. In *4th Annual International Conference on Computational Molecular Biology (RECOMB 2000)*, pages 127–135, Apr 2000. URL `citeseer.ist.psu.edu/friedman99using.html`. 155

[86] M. C. Frith, U. Hansen, J. L. Spouge, and Z. Weng. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res*, 32(1):189–200, 2004. 47

[87] D. J. Galas, M. Eggert, and M. S. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from Escherichia coli. *J Mol Biol*, 186(1):117–28, Nov 1985. 47

[88] R. Ganesh, D. A. Siegele, and T. R. Ioerger. Mopac: motif finding by preprocessing and agglomerative clustering from microarrays. *Pac Symp Biocomput*, pages 41–52, 2003. URL `http://www.hubmed.org/display.cgi?uids=12603016`. 47

[89] T. Ganz. Defensins: antimicrobial peptides of innate immunity. *Nat Rev Immunol*, 3: 710–20, 2003. 58

[90] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness*. W.H. Freeman and Company, New York, 1979. 46, 99

[91] Y. Ge, D. L. MacDonald, K. J. Holroyd, C. Thornsberry, H. Wexler, and M. Zasloff. In vitro antibacterial properties of pexiganan, an analog of magainin. *Antimicrob Agents Chemother*, 43(4):782–788, Apr. 1999. 58, 60

[92] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7:457–472, 1992. 54

[93] Gerstein Laboratory, Yale University. Omes Table. `http://bioinfo.mbb.yale.edu/what-is-it/omes/omes.html`, Dec 2005. 20

[94] A. Giangaspero, L. Sandri, and A. Tossi. Amphipathic alpha helical antimicrobial peptides. *Eur J Biochem*, 268:5589–600, 2001. 58

[95] H. Giladi, D. Goldenberg, S. Koby, and A. B. Oppenheim. Enhanced activity of the bacteriophage lambda PL promoter at low temperature. *Proc Natl Acad Sci U S A*, 92 (6):2184–8, Mar 1995. 163, 165

[96] H. Giladi, S. Koby, G. Prag, M. Engelhorn, J. Geiselmann, and A. B. Oppenheim. Participation of IHF and a distant UP element in the stimulation of the phage lambda PL promoter. *Mol Microbiol*, 30(2):443–51, Oct 1998. 163, 165

[97] H. Giladi, K. Murakami, A. Ishihama, and A. B. Oppenheim. Identification of an UP element within the IHF binding site at the PL1-PL2 tandem promoter of bacteriophage lambda. *J Mol Biol*, 260(4):484–91, Jul 1996. 163, 165

[98] A. Glieder, E. T. Farinas, and F. H. Arnold. Laboratory evolution of a soluble, self-sufficient, highly active alkane hydroxylase. *Nat Biotechnol*, 20(11):1135–9, Nov 2002. 161

[99] A. Goffeau. Genomic-scale analysis goes upstream? *Nat Biotechnol*, 16(10):907–8, Oct 1998. 53

[100] M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers in Chemistry*, 20(1):25–33, 1996. 133

[101] D. GuhaThakurta and G. D. Stormo. Identifying target sites for cooperatively binding factors. *Bioinformatics*, 17(7):608–21, Jul 2001. 53

[102] R. E. Hancock and A. Patrzykat. Clinical development of cationic antimicrobial peptides: from natural to novel antibiotics. *Curr Drug Targets Infect Disord*, 2:79–83, 2002. 60

[103] D. J. Hand and K. Yu. Idiot's bayes – not so stupid after all? *International Statistical Review*, 69(3):385–399, 2001. 155

[104] K. A. Hargreaves and K. Berry. Regex. Free Sotfware Foundation, 675 Mass Ave, Cambridge, MA 02139, Sept. 1992. 128

[105] A. J. Hartemink, D. K. Gifford, T. Jaakkola, and R. A. Young. Bayesian methods for elucidating genetic regulatory networks. *IEEE Intelligent Systems*, 17(2):37–43, 2002. 155

[106] D. Heckerman. A tutorial on learning with bayesian networks, 1995. URL `citeseer.ist.psu.edu/heckerman96tutorial.html`. 155

[107] J. G. Henikoff. Fred Hutchinson Cancer Research Center. Personal communication, October 2005. 138

[108] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Res*, 19:6565–72, 1991. 127

[109] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89(22):10915–10919, Nov 1992. 71, 107, 123, 130, 131, 134, 135, 136, 138, 147, 149, 151, 152

[110] S. Henikoff and J. G. Henikoff. Performance evaluation of amino acid substitution matrices. *Proteins*, 17:49–61, 1993. 130, 131, 134, 136, 138, 149

[111] S. Henikoff and J. G. Henikoff. Protein family classification based on searching a database of blocks. *Genomics*, 19(1):97–107, Jan 1994. URL `http://www.hubmed.org/display.cgi?uids=8188249`. 131

[112] S. Henikoff and J. G. Henikoff. *Amino Acid Substitution Matrices*, volume 54 of *Advances in Protein Chemistry*, pages 73–98. Academic Press, San Diego, 2000. 130, 149

[113] S. Henikoff, J. G. Henikoff, W. J. Alford, and S. Pietrokovski. Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene*, 163(2): GC17–26, Oct 1995. 90, 124

[114] D. Hernandez, R. Gras, and R. Appel. MoDEL: an efficient strategy for ungapped local multiple alignment. *Comput Biol Chem*, 28(2):119–28, Apr 2004. 47

[115] G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7-8):563–577, Jul 1999. 53, 90, 118, 124

[116] D. G. Higgins, A. J. Bleasby, and R. Fuchs. CLUSTAL V: improved software for multiple sequence alignment. *Comput Appl Biosci*, 8:189–91, 1992. 145

[117] K. Hilpert, R. Volkmer-Engert, T. Walter, and R. E. W. Hancock. High-throughput generation of small antibacterial peptides with improved activity. *Nat Biotechnol*, 23(8): 1008–12, Aug 2005. 88

[118] K. Hofmann, P. Bucher, L. Falquet, and A. Bairoch. The PROSITE database, its status in 1999. *Nucleic Acids Res*, 27:215–9, 1999. 36, 60, 99, 127, 130

[119] L. Holm, C. Ouzounis, C. Sander, G. Tuparev, and G. Vriend. A database of protein structure families with common folding motifs. *Protein Sci*, 1(12):1691–1698, 1992. 90

[120] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J Mol Biol*, 233(1):123–138, Oct 1993. 111, 123

[121] L. Holm and C. Sander. Enzyme HIT. *Trends Biochem Sci*, 22(4):116–117, May 1997. Letter. 12, 111, 115

[122] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, Apr 1987. 111

[123] P. Horton. Tsukuba BB: a branch and bound algorithm for local multiple alignment of DNA and protein sequences. *J Comput Biol*, 8(3):283–303, 2001. 47

[124] J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J Mol Biol*, 296:1205–14, 2000. 53

[125] C. G. Hunter and S. Subramaniam. Protein fragment clustering and canonical local shapes. *Proteins*, 50(4):580–588, Apr 2003. Evaluation Studies. 111

[126] D. Hwang, A. G. Rust, S. Ramsey, J. J. Smith, D. M. Leslie, A. D. Weston, P. de Atauri, J. D. Aitchison, L. Hood, A. F. Siegel, and H. Bolouri. A data integration methodology for systems biology. *Proc Natl Acad Sci U S A*, 102(48):17296–301, Nov 2005. 20

[127] T. Ideker and D. Lauffenburger. Building with a scaffold: emerging strategies for high- to low-level cellular modeling. *Trends Biotechnol*, 21(6):255–62, Jun 2003. 20

[128] J. S. Jacobs Anderson and R. Parker. Computational identification of cis-acting elements affecting post-transcriptional control of gene expression in Saccharomyces cerevisiae. *Nucleic Acids Res*, 28(7):1604–17, Apr 2000. 47

[129] K. L. Jensen, M. P. Styczynski, and G. N. Stephanopoulos. All of your blast searches are wrong... sort of. *Bioinformatics*, page submitted, 2006. 131

[130] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, 2005. URL `citeseer.ist.psu.edu/john95estimating.html`. 155

[131] M. Johnson, S. Morris, A. Chen, E. Stavnezer, and J. Leis. Selection of functional mutations in the U5-IR stem and loop regions of the Rous sarcoma virus genome. *BMC Biol*, 2(1):8, May 2004. 162

[132] I. Jonassen, J. F. Collins, and D. G. Higgins. Finding flexible patterns in unaligned protein sequences. *Protein Sci*, 4(8):1587–1595, Aug 1995. 47, 90

[133] I. Jonassen, I. Eidhammer, D. Conklin, and W. R. Taylor. Structure motif discovery and mining the PDB. *Bioinformatics*, 18(2):362–367, Feb 2002. 111

[134] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, New Jersey, 2000. 32, 60

[135] S. Kawashima, H. Ogata, and M. Kanehisa. AAindex: Amino Acid Index Database. *Nucleic Acids Res*, 27(1):368–9, Jan 1999. 151

[136] U. Keich and P. A. Pevzner. Finding motifs in the twilight zone. *Bioinformatics*, 18(10):1374–1381, Oct 2002. Evaluation Studies. 90

[137] U. Keich and P. A. Pevzner. Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, 18(10):1382–1390, Oct 2002. Evaluation Studies. 47

[138] S. M. Kiełbasa, J. O. Korbel, D. Beule, J. Schuchhardt, and H. Herzel. Combining frequency and positional information to predict transcription factor binding sites. *Bioinformatics*, 17(11):1019–26, Nov 2001. 47

[139] D. M. Kim and C. Y. Choi. A semicontinuous prokaryotic coupled transcription/translation system using a dialysis membrane. *Biotechnol Prog*, 12:645–9, 1996. 73

[140] S. Kim, S. S. Kim, Y.-J. B. Kim, Seong-Jin, and B. J. Lee. In vitro activities of native and designed peptide antibiotics against drug sensitive and resistant tumor cell lines. *Peptides*, 24(7):945–953, 2003. 58

[141] D. A. Kimbrell and B. Beutler. The evolution and genetics of innate immunity. *Nat Rev Genet*, 2:256–67, 2001. 57

[142] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983. URL `citeseer.nj.nec.com/kirkpatrick83optimization.html`. 54

[143] P. Koehl and M. Levitt. Structure-based conformational preferences of amino acids. *Proc Natl Acad Sci U S A*, 96(22):12524–9, Oct 1999. 154

[144] R. Kolodny, P. Koehl, and M. Levitt. Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures. *J Mol Biol*, 346(4):1173–88, Mar 2005. 111

[145] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, R. Funke, D. Gage, K. Harris, A. Heaford, J. Howland, L. Kann, J. Lehoczky, R. LeVine, P. McEwan, K. McKernan, J. Meldrim, J. P. Mesirov, C. Miranda, W. Morris, J. Naylor, C. Raymond, M. Rosetti, R. Santos, A. Sheridan, C. Sougnez, N. Stange-Thomann, N. Stojanovic, A. Subramanian, D. Wyman, J. Rogers, J. Sulston, R. Ainscough, S. Beck, D. Bentley, J. Burton, C. Clee, N. Carter, A. Coulson, R. Deadman, P. Deloukas, A. Dunham, I. Dunham, R. Durbin, L. French, D. Grafham, S. Gregory, T. Hubbard, S. Humphray, A. Hunt, M. Jones, C. Lloyd, A. McMurray, L. Matthews, S. Mercer, S. Milne, J. C. Mullikin, A. Mungall, R. Plumb, M. Ross, R. Shownkeen, S. Sims, R. H. Waterston, R. K. Wilson, L. W. Hillier, J. D. McPherson, M. A. Marra, E. R. Mardis, L. A. Fulton, A. T. Chinwalla, K. H. Pepin, W. R. Gish, S. L. Chissoe, M. C. Wendl, K. D. Delehaunty, T. L. Miner, A. Delehaunty, J. B. Kramer, L. L. Cook, R. S. Fulton, D. L. Johnson, P. J. Minx, S. W. Clifton, T. Hawkins, E. Branscomb, P. Predki, P. Richardson, S. Wenning, T. Slezak, N. Doggett, J. F. Cheng, A. Olsen, S. Lucas, C. Elkin, E. Uberbacher, M. Frazier, R. A. Gibbs, D. M. Muzny, S. E. Scherer, J. B. Bouck, E. J. Sodergren, K. C. Worley, C. M. Rives, J. H. Gorrell, M. L. Metzker, S. L. Naylor, R. S. Kucherlapati, D. L. Nelson, G. M. Weinstock, Y. Sakaki, A. Fujiyama, M. Hattori, T. Yada, A. Toyoda, T. Itoh,

C. Kawagoe, H. Watanabe, Y. Totoki, T. Taylor, J. Weissenbach, R. Heilig, W. Saurin, F. Artiguenave, P. Brottier, T. Bruls, E. Pelletier, C. Robert, P. Wincker, D. R. Smith, L. Doucette-Stamm, M. Rubenfield, K. Weinstock, H. M. Lee, J. Dubois, A. Rosenthal, M. Platzer, G. Nyakatura, S. Taudien, A. Rump, H. Yang, J. Yu, J. Wang, G. Huang, J. Gu, L. Hood, L. Rowen, A. Madan, S. Qin, R. W. Davis, N. A. Federspiel, A. P. Abola, M. J. Proctor, R. M. Myers, J. Schmutz, M. Dickson, J. Grimwood, D. R. Cox, M. V. Olson, R. Kaul, C. Raymond, N. Shimizu, K. Kawasaki, S. Minoshima, G. A. Evans, M. Athanasiou, R. Schultz, B. A. Roe, F. Chen, H. Pan, J. Ramser, H. Lehrach, R. Reinhardt, W. R. McCombie, M. de la Bastide, N. Dedhia, H. Blocker, K. Hornischer, G. Nordsiek, R. Agarwala, L. Aravind, J. A. Bailey, A. Bateman, S. Batzoglou, E. Birney, P. Bork, D. G. Brown, C. B. Burge, L. Cerutti, H. C. Chen, D. Church, M. Clamp, R. R. Copley, T. Doerks, S. R. Eddy, E. E. Eichler, T. S. Furey, J. Galagan, J. G. Gilbert, C. Harmon, Y. Hayashizaki, D. Haussler, H. Hermjakob, K. Hokamp, W. Jang, L. S. Johnson, T. A. Jones, S. Kasif, A. Kaspryzk, S. Kennedy, W. J. Kent, P. Kitts, E. V. Koonin, I. Korf, D. Kulp, D. Lancet, T. M. Lowe, A. McLysaght, T. Mikkelsen, J. V. Moran, N. Mulder, V. J. Pollara, C. P. Ponting, G. Schuler, J. Schultz, G. Slater, A. F. Smit, E. Stupka, J. Szustakowski, D. Thierry-Mieg, J. Thierry-Mieg, L. Wagner, J. Wallis, R. Wheeler, A. Williams, Y. I. Wolf, K. H. Wolfe, S. P. Yang, R. F. Yeh, F. Collins, M. S. Guyer, J. Peterson, A. Felsenfeld, K. A. Wetterstrand, A. Patrinos, M. J. Morgan, P. de Jong, J. J. Catanese, K. Osoegawa, H. Shizuya, S. Choi, and Y. J. Chen. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, Feb 2001. 20

[146] N. Landwehr, M. Hall, and E. Frank. *Logistic model trees*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 241–252. Springer–Verlag, 2003. 155

[147] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, Oct 1993. 52, 53, 55, 90, 118, 124

[148] H. C. M. Leung and F. Y. L. Chin. Finding exact optimal motifs in matrix representation by partitioning. *Bioinformatics*, 21 Suppl 2:ii86–ii92, Sep 2005. 53

[149] M. Y. Leung, G. M. Marsh, and T. P. Speed. Over- and underrepresentation of short DNA words in herpesvirus genomes. *J Comput Biol*, 3(3):345–60, 1996. 53

[150] W. Li, L. Jaroszewski, and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–3, Mar 2001. 82

[151] S. Liang, M. P. Samanta, and B. A. Biegel. cWINNOWER algorithm for finding fuzzy dna motifs. *J Bioinform Comput Biol*, 2(1):47–60, Mar 2004. 47

[152] C. D. Lima, K. L. D'Amico, I. Naday, G. Rosenbaum, E. M. Westbrook, and W. A. Hendrickson. MAD analysis of FHIT, a putative human tumor suppressor from the HIT protein family. *Structure*, 5(6):763–774, Jul 1997. 111

[153] D. Liu and W. F. DeGrado. *De novo* design, synthesis, and characterization of antimicrobial beta–peptides. *Journal of the American Chemical Society*, 123(31):7553–7559, 2001. 78

[154] J. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994. 53

[155] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, 2001. 52, 54

[156] Y. Liu, M. P. Vincenti, and H. Yokota. Principal component analysis for predicting transcription-factor binding motifs from array-derived data. *BMC Bioinformatics*, 6: 276, 2005. 53

[157] I. S. Lossos, R. Tibshirani, B. Narasimhan, and R. Levy. The inference of antigen selection on Ig genes. *J Immunol*, 165(9):5122–6, Nov 2000. 162

[158] R. Lutz and H. Bujard. Independent and tight regulation of transcriptional units in Escherichia coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements. *Nucleic Acids Res*, 25(6):1203–10, Mar 1997. 163, 164, 165

[159] K. D. Macisaac, D. B. Gordon, L. Nekludova, D. T. Odom, J. Schreiber, D. K. Gifford, R. A. Young, and E. Fraenkel. A hypothesis-based approach for identifying the binding specificity of regulatory proteins from chromatin immunoprecipitation data. *Bioinformatics*, 22(4):423–9, Feb 2006. 53

[160] T. Madej, J. F. Gibrat, and S. H. Bryant. Threading a database of protein cores. *Proteins*, 23(3):356–369, Nov 1995. 111

[161] D. Maier. The complexity of some problems on subsequences and supersequences. *J. ACM*, 25(2):322–336, 1978. 46

[162] J. V. Maizel and R. P. Lenk. Enhanced graphic matrix analysis of nucleic acid and protein sequences. *Proc Natl Acad Sci U S A*, 78(12):7665–9, Dec 1981. 82

[163] A. Mancheron and I. Rusu. Pattern discovery allowing wild-cards, substitution matrices, and multiple score functions. In *Algorithms in Bioinformatics, Proceedings Lecture notes in Bioinformatics*, pages 124–138. Springer–Verlag, 2003. 93

[164] H. J. Mangalam. tacg–a grep for DNA. *BMC Bioinformatics*, 3:8, 2002. 128

[165] A. Marchler-Bauer, J. B. Anderson, C. DeWeese-Scott, N. D. Fedorova, L. Y. Geer, S. He, D. I. Hurwitz, J. D. Jackson, A. R. Jacobs, C. J. Lanczycki, C. A. Liebert, C. Liu, T. Madej, G. H. Marchler, R. Mazumder, A. N. Nikolskaya, A. R. Panchenko, B. S. Rao, B. A. Shoemaker, V. Simonyan, J. S. Song, P. A. Thiessen, S. Vasudevan, Y. Wang, R. A. Yamashita, J. J. Yin, and S. H. Bryant. CDD: a curated Entrez database of conserved domain alignments. *Nucleic Acids Res*, 31(1):383–387, Feb 2003. 108

[166] M. Markstein, R. Zinzen, P. Markstein, K.-P. Yee, A. Erives, A. Stathopoulos, and M. Levine. A regulatory code for neurogenic gene expression in the Drosophila embryo. *Development*, 131(10):2387–94, May 2004. 47

[167] L. Marsan and M. F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *J Comput Biol*, 7(3–4):345–62, 2000. 47

[168] K. A. Martemyanov, V. A. Shirokov, O. V. Kurnasov, A. T. Gudkov, and A. S. Spirin. Cell-free production of biologically active polypeptides: application to the synthesis of antibacterial peptide cecropin. *Protein Expr Purif*, 21(3):456–461, Apr 2001. 73

[169] G. Mengeritsky and T. F. Smith. Recognition of characteristic patterns in sets of functionally equivalent DNA sequences. *Comput Appl Biosci*, 3(3):223–7, Sep 1987. 47

[170] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, Jun 1953. 52

[171] S. Mitaku, T. Hirokawa, and T. Tsuji. Amphiphilicity index of polar amino acids as an aid in the characterization of amino acid preference at membrane-water interfaces. *Bioinformatics*, 18(4):608–16, Apr 2002. 154

[172] L. Moerman, S. Bosteels, W. Noppe, J. Willems, E. Clynen, L. Schoofs, K. Thevissen, J. Tytgat, J. Van Eldere., J. Van Der. Walt., and F. Verdonck. Antibacterial and antifungal properties of alpha-helical, cationic peptides in the venom of scorpions from southern Africa. *Eur J Biochem*, 269(19):4799–4810, Oct 2002. 57

[173] F. Mueller. A library implementation of POSIX threads under unix. In *Proceedings of the Winter 1993 USENIX Technical Conference and Exhibition*, pages 29–41, San Diego, CA, USA, 1993. 128

[174] V. L. Murthy and G. D. Rose. RNABase: an annotated database of RNA structures. *Nucleic Acids Res*, 31(1):502–504, Jan 2003. 90

[175] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol*, 247(4):536–40, Apr 1995. 133

[176] K. Nakai, A. Kidera, and M. Kanehisa. Cluster analysis of amino acid indices for prediction of protein structure and function. *Protein Eng*, 2(2):93–100, Jul 1988. 151

[177] A. Narayanan, X. Wu, and Z. R. Yang. Mining viral protease data to extract cleavage knowledge. *Bioinformatics*, 18 Suppl 1:S5–13, 2002. 153

[178] G. Navarro. NR–grep: a fast and flexible pattern-matching tool. *Software Practice and Experience*, 31(13):1265–1312, ???? 2001. 128

[179] A. Neuwald and P. Green. Detecting patterns in protein sequences. *Journal of Molecular Biology*, 239:698–712, 1994. 47

[180] H. Ney and S. Ortmanns. Progress in dynamic programming search for LVCSR. *Proceedings of the IEEE*, 88(8):1244–1240, 2000. 142

[181] P. C. Ng and S. Henikoff. Predicting deleterious amino acid substitutions. *Genome Res*, 11(5):863–874, May 2001. URL `http://www.hubmed.org/display.cgi?uids=11337480`. 131

[182] B. H. Normark and S. Normark. Evolution and spread of antibiotic resistance. *J Intern Med*, 252:91–106, 2002. 58

[183] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–217, Sep 2000. URL `http://www.hubmed.org/display.cgi?uids=10964570`. 130

[184] I. of Medicine. *Antimicrobial Resistance: Issues and Options*. National Academy Press, 1998. 58

[185] C. A. Orengo and W. R. Taylor. SSAP: sequential structure alignment program for protein structure comparison. *Methods Enzymol*, 266:617–635, 1996. 111

[186] H. A. Orr. A minimum on the mean number of steps taken in adaptive walks. *J Theor Biol*, 220(2):241–7, Jan 2003. 170

[187] A. R. Ortiz, C. E. M. Strauss, and O. Olmea. MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. *Protein Sci*, 11(11):2606–2621, Nov 2002. Evaluation Studies. 111

[188] J. Palau, P. Argos, and P. Puigdomenech. Protein secondary structure. Studies on the limits of prediction accuracy. *Int J Pept Protein Res*, 19(4):394–401, Apr 1982. 154

[189] L. Parida, I. Rigoutsos, and A. Floratos. MUSCA: an algorithm for constrained alignment of multiple data sequences. In *Proceedings of the Twelfth International Conference on Genome Informatics (GIW)*, Tokyo, 1998. 51

[190] J. Parrish-Novak, S. R. Dillon, A. Nelson, A. Hammond, C. Sprecher, J. A. Gross, J. Johnston, K. Madden, W. Xu, J. West, S. Schrader, S. Burkhead, M. Heipel, C. Brandt, J. L. Kuijper, J. Kramer, D. Conklin, S. R. Presnell, J. Berry, F. Shiota, S. Bort, K. Hambly, S. Mudri, C. Clegg, M. Moore, F. J. Grant, C. Lofton-Day, T. Gilbert, F. Rayond, A. Ching, L. Yao, D. Smith, P. Webster, T. Whitmore, M. Maurer, K. Kaushansky, R. D. Holly, and D. Foster. Interleukin 21 and its receptor are involved in NK cell expansion and regulation of lymphocyte function. *Nature*, 408(6808):57–63, Nov 2000. 36

[191] G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole. Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res*, 32(Web Server issue):W199–203, Jul 2004. 47

[192] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol*, 183:63–98, 1990. 133

[193] W. R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11(3):635–50, Nov 1991. 133, 134

[194] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85:2444–2448, 1988. 127, 130, 142

[195] P. A. Pevzner and S. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings International Conference on Intelligent Systems for Molecular Biology*, pages 269–278. AAAI Press, 2000. 47, 93, 117, 118, 120

[196] W. W. Piegorsch and A. J. Bailer. Statistical approaches for analyzing mutational spectra: some recommendations for categorical data. *Genetics*, 136(1):403–16, Jan 1994. 162

[197] S. Pietrokovski. Searching databases of conserved sequence regions by aligning protein multiple-alignments. *Nucleic Acids Res*, 24(19):3836–3845, Oct 1996. URL `http://www.hubmed.org/display.cgi?uids=8871566`. 131

[198] M. Prabhakaran. The distribution of physical, chemical and conformational properties in signal and nascent peptides. *Biochem J*, 269(3):691–6, Aug 1990. 154

[199] A. Price, S. Ramabhadran, and P. A. Pevzner. Finding subtle motifs by branching from sample strings. *Bioinformatics*, 19 Suppl 2:II149–II155, Oct 2003. 47, 90

[200] G. A. Price, G. E. Crooks, R. E. Green, and S. E. Brenner. Statistical evaluation of pairwise protein sequence comparison with the bayesian bootstrap. *Bioinformatics*, 21 (20):3824–3831, Oct 2005. URL `http://www.hubmed.org/display.cgi?uids=16105900`. 133, 134, 137

[201] K. Putsep, G. Carlsson, H. G. Boman, and M. Andersson. Deficiency of antibacterial peptides in patients with morbus Kostmann: an observation study. *Lancet*, 360:1144–9, 2002. 58

[202] C. Queen, M. Wegman, and L. Korn. Improvements to a program for dna analysis: a procedure to find homologies among many sequences. *Nucleic Acids Research*, 10:449–456, 1982. 47

[203] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 1992. 155

[204] B. Raphael, L.-T. Liu, and G. Varghese. A uniform projection method for motif discovery in dna sequences. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(2):91–94, 2004. ISSN 1545-5963. 53

[205] J. H. Reif. Computing. Successes and challenges. *Science*, 296(5567):478–9, Apr 2002. 15

[206] P. Rice, I. Longden, and A. Bleasby. EMBOSS: the European Molecular Biology Open Software Suite. *Trends Genet*, 16:276–7, 2000. 74, 128

[207] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14:55–67, 1998. 47, 48, 60, 90, 93, 97, 123, 124, 128

[208] I. Rigoutsos, A. Floratos, C. Ouzounis, Y. Gao, and L. Parida. Dictionary building via unsupervised hierarchical motif discovery in the sequence space of natural proteins. *Proteins*, 37:264–77, 1999. 52, 66, 91

[209] E. Rivas and S. R. Eddy. The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–40, Apr 2000. 32

[210] B. Robson and E. Suzuki. Conformational properties of amino acid residues in globular proteins. *J Mol Biol*, 107(3):327–56, Nov 1976. 154

[211] T. Rognvaldsson and L. You. Why neural networks should not be used for HIV-1 protease cleavage site prediction. *Bioinformatics*, 20(11):1702–1709, Jul 2004. 153

[212] D. R. Rokyta, P. Joyce, S. B. Caudle, and H. A. Wichman. An empirical test of the mutational landscape model of adaptation using a single-stranded DNA virus. *Nat Genet*, 37(4):441–4, Apr 2005. 170

[213] J. Rolff and M. T. Siva-Jothy. Invertebrate Ecological Immunology. *Science*, 301(5632): 472–475, 2003. 57

[214] T. M. Rose, E. R. Schultz, J. G. Henikoff, S. Pietrokovski, C. M. McCallum, and S. Henikoff. Consensus-degenerate hybrid oligonucleotide primers for amplification of distantly related sequences. *Nucleic Acids Res*, 26(7):1628–1635, Apr 1998. URL `http://www.hubmed.org/display.cgi?uids=9512532`. 131

[215] M. Sagot, A. Viari, and H. Soldano. Multiple sequence comparison — A peptide matching approach. *Theoretical Computer Science*, 180(1–2):115–137, 1997. 47

[216] C. Salazar, J. Schütze, and O. Ebenhöh. Bioinformatics meets systems biology. *Genome Biol*, 7(1):303, 2006. 20

[217] H. Salgado, S. Gama-Castro, A. Martinez-Antonio, E. Diaz-Peredo, F. Sanchez-Solano, M. Peralta-Gil, D. Garcia-Alonso, V. Jimenez-Jacinto, A. Santos-Zavaleta, C. Bonavides-Martinez, and J. Collado-Vides. RegulonDB (version 4.0): transcriptional regulation, operon organization and growth conditions in Escherichia coli K-12. *Nucleic Acids Res*, 32(Database issue):303–306, Jan 2004. 122

[218] N. H. Salzman, D. Ghosh, K. M. Huttner, Y. Paterson, and C. L. Bevins. Protection against enteric salmonellosis in transgenic mice expressing a human intestinal defensin. *Nature*, 422:522–6, 2003. 58

[219] B. Schittek, R. Hipfel, B. Sauer, J. Bauer, H. Kalbacher, S. Stevanovic, M. Schirle, K. Schroeder, N. Blin, F. Meier, G. Rassner, and C. Garbe. Dermcidin: a novel human antibiotic peptide secreted by sweat glands. *Nat Immunol*, 2:1133–7, 2001. 57

[220] B. Schuster-Böckler, J. Schultz, and S. Rahmann. HMM Logos for visualization of protein families. *BMC Bioinformatics*, 5:7, Jan 2004. 112

[221] M. S. Scott, D. Y. Thomas, and M. T. Hallett. Predicting subcellular localization via protein motif co-occurrence. *Genome Res*, 14(10A):1957–66, Oct 2004. 155

[222] D. B. Searls. The computational linguistics of biological sequences. In L. Hunter, editor, *Artificial Intelligence and Molecular Biology*, pages 47–120. AAAI Press, 1992. 25

[223] D. B. Searls. Linguistic approaches to biological sequences. *Comput Appl Biosci*, 13: 333–44, 1997. 25

[224] D. B. Searls. Reading the book of life. *Bioinformatics*, 17(7):579–580, 2001. 25

[225] D. B. Searls. The language of genes. *Nature*, 420:211–7, 2002. 60

[226] Y. Shai. Mode of action of membrane active antimicrobial peptides. *Biopolymers*, 66: 236–48, 2002. 59, 76, 77

[227] J. Shendure, R. D. Mitra, C. Varma, and G. M. Church. Advanced sequencing technologies: methods and goals. *Nat Rev Genet*, 5(5):335–44, May 2004. 15, 16

[228] S. M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985. 32

[229] R. Siddharthan, E. D. Siggia, and E. van Nimwegen. PhyloGibbs: a gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput Biol*, 1(7):e67, Dec 2005. 53

[230] M. Simmaco, G. Mignogna, and D. Barra. Antimicrobial peptides from amphibian skin: What do they tell us? *Biopolymers*, 47(6):435–450, 1999. 57

[231] S. Sinha. Discriminative motifs. *J Comput Biol*, 10(3-4):599–615, 2003. 47

[232] S. Sinha and M. Tompa. Discovery of novel transcription factor binding sites by statistical overrepresentation. Deparment of Computer Science and Engineering, University of Washington, 2002. 47

[233] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997. 35

[234] J. J. Smith, S. M. Travis, E. P. Greenberg, and M. J. Welsh. Cystic fibrosis airway epithelia fail to kill bacteria because of abnormal airway surface fluid. *Cell*, 85:229–36, 1996. 58

[235] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–7, Mar 1981. 133

[236] C. Solem and P. R. Jensen. Modulation of gene expression made easy. *Appl Environ Microbiol*, 68(5):2397–403, May 2002. 161

[237] R. Staden. Methods for discovering novel motifs in nucleic acid sequences. *Comput Appl Biosci*, 5(4):293–8, Oct 1989. 47

[238] G. D. Stormo and G. W. Hartzell. Identifying protein-binding sites from unaligned DNA fragments. *Proc Natl Acad Sci U S A*, 86(4):1183–7, Feb 1989. 53

[239] M. B. Strom, B. E. Haug, M. L. Skar, W. Stensen, T. Stiberg, and J. S. Svendsen. The pharmacophore of short cationic antibacterial peptides. *J Med Chem*, 46:1567–70, 2003. 60

[240] P. Sumazin, G. Chen, N. Hata, A. D. Smith, T. Zhang, and M. Q. Zhang. DWE: discriminating word enumerator. *Bioinformatics*, 21(1):31–8, Jan 2005. 47

[241] A. L. Swain, M. M. Miller, J. Green, D. H. Rich, J. Schneider, S. B. Kent, and A. Wlodawer. X-ray crystallographic structure of a complex between a synthetic protease of human immunodeficiency virus 1 and a substrate-based hydroxyethylamine inhibitor. *Proc Natl Acad Sci U S A*, 87(22):8805–9, Nov 1990. 12, 152

[242] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990. 142

[243] K. Tharakaraman, L. Mariño-Ramírez, S. Sheetlin, D. Landsman, and J. L. Spouge. Alignments anchored on genomic landmarks can aid in the identification of regulatory elements. *Bioinformatics*, 21 Suppl 1:i440–i448, Jun 2005. 53

[244] The computational biology and functional genomics laboratory at the Dana–Farber Cancer Institute and Harvard School of Public Health. The -omics revolution count. `http://biocomp.dfci.harvard.edu/tgi/omics_count.html`, Dec 2005. 20

[245] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–4680, Nov 1994. 130

[246] E. Tiozzo, G. Rocco, A. Tossi, and D. Romeo. Wide-spectrum antibiotic activity of synthetic, amphipathic peptides. *Biochem Biophys Res Commun*, 249:202–6, 1998. 60, 87

[247] K. Tomii and M. Kanehisa. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *Protein Eng*, 9(1):27–36, Jan 1996. 151

[248] E. Tomita, A. Tanaka, and H. Takahasi. An optimal algorithm for finding all the cliques. *SIG Algorithms*, 12:91–98, 1989. 99

[249] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Regnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol*, 23(1):137–144, Jan 2005. 54, 91

[250] A. Tossi. Antimicrobial sequences database (AMSDb), 2002. http://www.bbcm.univ.trieste.it/ tossi/amsdb.html. 60, 81

[251] A. Tossi, L. Sandri, and A. Giangaspero. Amphipathic, alpha-helical antimicrobial peptides. *Biopolymers*, 55:4–30, 2000. 58, 87

[252] J. van Helden, B. André, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J Mol Biol*, 281(5):827–42, Sep 1998. 47

[253] J. van Helden, A. F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res*, 28(8):1808–18, Apr 2000. 47

[254] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8. 156

[255] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998. ISBN 0-471-03003-1. VAP v 98:1 1.Ex. 156

[256] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. Gabor Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu-Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, V. Di Francesco, P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R. R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Milshina, H. M. Moore, A. K. Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Wang, A. Wang, X. Wang, J. Wang, M. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An, A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. L. Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin,

J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. McCawley, T. McIntosh, I. McMullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y. H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. N. Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn-Deen, K. Wolfe, J. Zaveri, K. Zaveri, J. F. Abril, R. Guigo, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes-Stine, P. Caulk, Y. H. Chiang, M. Coyne, C. Dahlke, A. Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. McDaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, M. Nodell, S. Pan, J. Peck, M. Peterson, W. Rowe, R. Sanders, J. Scott, M. Simpson, T. Smith, A. Sprague, T. Stockwell, R. Turner, E. Venter, M. Wang, M. Wen, D. Wu, M. Wu, A. Xia, A. Zandieh, and X. Zhu. The sequence of the human genome. *Science*, 291(5507):1304–1351, Feb 2001. 20

[257] J. Vizioli, P. Bulet, J. A. Hoffmann, F. C. Kafatos, H.-M. Muller, and G. Dimopoulos. Gambicin: A novel immune responsive antimicrobial peptide from the malaria vector Anopheles gambiae. *PNAS*, 98(22):12630–12635, 2001. 58

[258] G. Vogt, T. Etzold, and P. Argos. An assessment of amino acid exchange matrices in aligning protein sequences: the twilight zone revisited. *J Mol Biol*, 249:816–31, 1995. 130, 149

[259] D. R. Walker, J. P. Bond, R. E. Tarone, C. C. Harris, W. Makalowski, M. S. Boguski, and M. S. Greenblatt. Evolutionary conservation and somatic mutation hotspot maps of p53: correlation with p53 protein structural and functional features. *Oncogene*, 18 (1):211–8, Jan 1999. 162

[260] C. Walsh. Molecular mechanisms that confer antibacterial drug resistance. *Nature*, 406: 775–81, 2000. 58

[261] G. Wang, Y. Li, and X. Li. Correlation of three-dimensional structures with the antibacterial activity of a group of peptides designed based on a nontoxic bacterial membrane anchor. *J Biol Chem*, 280(7):5803–11, Feb 2005. 58, 61

[262] W. Wang and P. A. Kollman. Computational study of protein specificity: the molecular basis of HIV-1 protease drug resistance. *Proc Natl Acad Sci U S A*, 98(26):14937–42, Dec 2001. 152

[263] Z. Wang and G. Wang. APD: the Antimicrobial Peptide Database. *Nucleic Acids Res*, 32(Database issue):D590–2, Jan 2004. 80

[264] M. S. Waterman. Efficient sequence alignment algorithms. *J Theor Biol*, 108:333–7, 1984. 142

[265] R. H. Waterston, K. Lindblad-Toh, E. Birney, J. Rogers, J. F. Abril, P. Agarwal, R. Agarwala, R. Ainscough, M. Alexandersson, P. An, S. E. Antonarakis, J. Attwood, R. Baertsch, J. Bailey, K. Barlow, S. Beck, E. Berry, B. Birren, T. Bloom, P. Bork, M. Botcherby, N. Bray, M. R. Brent, D. G. Brown, S. D. Brown, C. Bult, J. Burton, J. Butler, R. D. Campbell, P. Carninci, S. Cawley, F. Chiaromonte, A. T. Chinwalla, D. M. Church, M. Clamp, C. Clee, F. S. Collins, L. L. Cook, R. R. Copley, A. Coulson, O. Couronne, J. Cuff, V. Curwen, T. Cutts, M. Daly, R. David, J. Davies, K. D. Delehaunty, J. Deri, E. T. Dermitzakis, C. Dewey, N. J. Dickens, M. Diekhans, S. Dodge, I. Dubchak, D. M. Dunn, S. R. Eddy, L. Elnitski, R. D. Emes, P. Eswara, E. Eyras, A. Felsenfeld, G. A. Fewell, P. Flicek, K. Foley, W. N. Frankel, L. A. Fulton, R. S. Fulton, T. S. Furey, D. Gage, R. A. Gibbs, G. Glusman, S. Gnerre, N. Goldman, L. Goodstadt, D. Grafham, T. A. Graves, E. D. Green, S. Gregory, R. Guigó, M. Guyer, R. C. Hardison, D. Haussler, Y. Hayashizaki, L. W. Hillier, A. Hinrichs, W. Hlavina, T. Holzer, F. Hsu, A. Hua, T. Hubbard, A. Hunt, I. Jackson, D. B. Jaffe, L. S. Johnson, M. Jones, T. A. Jones, A. Joy, M. Kamal, E. K. Karlsson, D. Karolchik, A. Kasprzyk, J. Kawai, E. Keibler, C. Kells, W. J. Kent, A. Kirby, D. L. Kolbe, I. Korf, R. S. Kucherlapati, E. J. Kulbokas, D. Kulp, T. Landers, J. P. Leger, S. Leonard, I. Letunic, R. Levine, J. Li, M. Li, C. Lloyd, S. Lucas, B. Ma, D. R. Maglott, E. R. Mardis, L. Matthews, E. Mauceli, J. H. Mayer, M. McCarthy, W. R. McCombie, S. McLaren, K. McLay, J. D. McPherson, J. Meldrim, B. Meredith, J. P. Mesirov, W. Miller, T. L. Miner, E. Mongin, K. T. Montgomery, M. Morgan, R. Mott, J. C. Mullikin, D. M. Muzny, W. E. Nash, J. O. Nelson, M. N. Nhan, R. Nicol, Z. Ning, C. Nusbaum, M. J. O'Connor, Y. Okazaki, K. Oliver, E. Overton-Larty, L. Pachter, G. Parra, K. H. Pepin, J. Peterson, P. Pevzner, R. Plumb, C. S. Pohl, A. Poliakov, T. C. Ponce, C. P. Ponting, S. Potter, M. Quail, A. Reymond, B. A. Roe, K. M. Roskin, E. M. Rubin, A. G. Rust, R. Santos, V. Sapojnikov, B. Schultz, J. Schultz, M. S. Schwartz, S. Schwartz, C. Scott, S. Seaman, S. Searle, T. Sharpe, A. Sheridan, R. Shownkeen, S. Sims, J. B. Singer, G. Slater, A. Smit, D. R. Smith, B. Spencer, A. Stabenau, N. Stange-Thomann, C. Sugnet, M. Suyama, G. Tesler, J. Thompson, D. Torrents, E. Trevaskis, J. Tromp, C. Ucla, A. Ureta-Vidal, J. P. Vinson, A. C. Von Niederhausern, C. M. Wade, M. Wall, R. J. Weber, R. B. Weiss, M. C. Wendl, A. P. West, K. Wetterstrand, R. Wheeler, S. Whelan, J. Wierzbowski, D. Willey, S. Williams, R. K. Wilson, E. Winter, K. C. Worley, D. Wyman, S. Yang, S. P. Yang, E. M. Zdobnov, M. C. Zody, and E. S. Lander. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–562, Dec 2002. URL `http://www.hubmed.org/display.cgi?uids=12466850`. 20

[266] J. E. Wedekind, P. A. Frey, and I. Rayment. The structure of nucleotidylated histidine-166 of galactose-1-phosphate uridylyltransferase provides insight into phosphoryl group transfer. *Biochemistry*, 35(36):11560–11569, Oct 1996. 111

[267] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, L. Y. Geer, W. Helmberg, Y. Kapustin, D. L. Kenton, O. Khovayko, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, K. D.

Pruitt, G. D. Schuler, L. M. Schriml, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, T. O. Suzek, R. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 34(Database issue):D173–80, Jan 2006. 20

[268] C. L. Wilson, A. J. Ouellette, D. P. Satchell, T. Ayabe, Y. S. Lopez-Boado, J. L. Stratman, S. J. Hultgren, L. M. Matrisian, and W. C. Parks. Regulation of intestinal alpha-defensin activation by the metalloproteinase matrilysin in innate host defense. *Science*, 286:113–7, 1999. 58

[269] I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. 154, 155

[270] F. Wolferstetter, K. French, G. Herrmann, and T. Werner. Identification of functional. *Computer Applications in the Biosciences*, 12(1):71–80, 1996. 47

[271] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe, NM, 1995. URL `citeseer.ist.psu.edu/wolpert95no.html`. 157

[272] J. C. Wootton and S. Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Computers in Chemistry*, 17:149–163, 1993. 133

[273] C. T. Workman and G. D. Stormo. Ann-spec: a method for discovering transcription factor binding sites with improved specificity. *Pac Symp Biocomput*, pages 467–478, 2000. URL `http://www.hubmed.org/display.cgi?uids=10902194`. 53

[274] M. Wu and R. E. Hancock. Interaction of the cyclic antimicrobial cationic peptide bactenecin with the outer and cytoplasmic membrane. *J Biol Chem*, 274(1):29–35, Jan 1999. 85

[275] S. Wu and U. Manber. Agrep — A fast approximate pattern-matching tool. In *Usenix Winter 1992 Technical Conference*, pages 153–162, San Francisco, Jan. 1992. 128

[276] L. You, D. Garwicz, and T. Rögnvaldsson. Comprehensive bioinformatic analysis of the specificity of human immunodeficiency virus type 1 protease. *J Virol*, 79(19):12477–86, Oct 2005. 153

[277] M. J. Zaki. Scalable algorithms for association mining. *Knowledge and Data Engineering*, 12(2):372–390, 2000. URL `citeseer.ist.psu.edu/zaki00scalable.html`. 93

[278] M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *In Proceedings of 3 rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, 1998. URL `citeseer.ist.psu.edu/zaki98theoretical.html`. 93

[279] M. Zasloff. Antimicrobial peptides in health and disease. *New England Journal of Medicine*, 347(15):1199–1200, Oct. 2002. 58, 60

[280] M. Zasloff. Antimicrobial peptides of multicellular organisms. *Nature*, 415:389–95, 2002. 58, 59, 60, 76

[281] L. Zhang, W. Yu, T. He, J. Yu, R. E. Caffrey, E. A. Dalmasso, S. Fu, T. Pham, J. Mei, J. J. Ho, W. Zhang, P. Lopez, and D. D. Ho. Contribution of human alpha-defensin 1, 2, and 3 to the anti-HIV-1 activity of CD8 antiviral factor. *Science*, 298:995–1000, 2002. 58

[282] W. Zhong, P. Zeng, P. Ma, J. S. Liu, and Y. Zhu. RSIR: regularized sliced inverse regression for motif discovery. *Bioinformatics*, 21(22):4169–75, Nov 2005. 53