Development of Gene-Finding Algorithms for Fungal Genomes- Dealing with Small Datasets and Leveraging Comparative Genomics

by

Allan Lazarovici

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[June 2003]

May 23, 2003

Author.....
.....
Department of Electrical Engineering and Computer Science
May 23, 2003

Certified by.....
.....
Christopher Burge
Assistant Professor
Thesis Supervisor

Accepted by.....
.....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Development of Gene-Finding Algorithms for Fungal Genomes- Dealing with Small Datasets and Leveraging Comparative Genomics

by

Allan Lazarovici

Submitted to the Department of Electrical Engineering and Computer Science on May 23, 2003 in Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Computer Science and Engineering

Abstract

A computer program called FUNSCAN was developed which identifies protein coding regions in fungal genomes. Gene structural and compositional properties are modeled using a Hidden Markov Model. Separate training and testing sets for FUNSCAN were obtained by aligning cDNAs from an organism to their genomic loci, generating a 'gold standard' set of annotated genes. The performance of FUNSCAN is competitive with other computer programs design to identify protein coding regions in fungal genomes. A technique called 'Training Set Augmentation' is described which can be used to train FUNSCAN when only a small training set of genes is available. Techniques that combine alignment algorithms with FUNSCAN to identify novel genes are also discussed and explored.

Thesis Supervisor: Christopher Burge
Title: Assistant Professor

Acknowledgements

I would like to thank my advisor, Christopher Burge, for giving me the opportunity to work in his lab and introducing me to the project of developing a gene-finder for fungal genomes. His input and encouragement was invaluable.

Working in the Burge lab was very enjoyable because I was surrounded by a talented group of PhD students and postdocs who were always available to answer questions and provide feedback on my work. Gene Yeo answered many of my questions and showed me how to use several bioinformatics tools in the lab. Dirk Holste showed me how to use GENOA, the gene annotation script which enabled me to obtain my training and testing sets. Uwe Ohler, Brad Friedman, Ben Lewis, Mehdi Yahyanejad, Zefeng Wang and David Wine all provided me with feedback on my work.

Contents

Chapter 1

INTRODUCTION

For centuries, biology has been a field in which its practitioners have dealt mostly with specimens and Petri dishes. However, in recent years, the development of technology for efficient, automated DNA sequencing has led to the accumulation of large databases of DNA and protein sequences. As a result, there has been a sudden ascendancy of computing in biology over the past five years and many of the challenges in biology have become challenges in computing. Bioinformatics, a branch of computing concerned with the acquisition, storage and analysis of biological data, has become a linchpin in the completion of several major biology projects such as the Human Genome Project (HGP).

Section 1.1 examines the reasons why biologists are excited about major genome sequencing projects such as the HGP and Section 1.2 explains why some scientists are beginning to shift their gaze towards fungal genomes. Section 1.3 provides a brief review of basic molecular biology and Section 1.4 describes some of the work that has been done in the area of gene-finding. Section 1.5 describes the main objectives of this thesis.

1.1 The Purpose of Sequencing Genomes

The HGP has been compared to the Apollo space program by the National Human Genome Research Institute and a leading science writer declared the completion of the HGP to be "the greatest intellectual moment in human history, bar none."

Given that scientists have been enraptured by the HGP, a person not familiar with biology and the HGP might be tempted to ask several questions. For example, what is a genome? What do scientists mean when they say that they've 'sequenced' a genome? What was the purpose of sequencing the human genome and what are the potential benefits?

A genome is the entire collection of chromosomes in each cell of an organism. A chromosome is a very long chain of nitrogenous bases. When dealing with nucleotide sequences in biology, the beginning of a sequence is known as the 5' end the end of a sequence is known as the 3'end. There are four different kinds of nitrogenous bases: adenine, cytosine, guanine and thymine and they are generally referred to by their initial letters, A, C, G and T. The two main regions of a chromosome are the coding regions and the intergenic regions. Coding regions are also called genes and they code for the production of proteins. Proteins are made of smaller molecules, called amino acids, which are strung together in chains that are normally several hundred amino acids long. Just as there are only four different nucleotides, there are only 20 different amino acids.

It is these proteins that play an important role in molecular biology; almost every molecule in a eukaryotic organism is either a protein or the direct result of a protein's activity. Coding regions occupy a small portion of a higher eukaryotic genome; it is estimated that coding regions constitute only 3% of the human genome. However, for *Schizosaccharomyces pombe*, a fission yeast, it is estimated that the protein-coding regions occupy 60.2% of the *S. pombe* genome.

The task of scientists working on sequencing the genome of an organism is to obtain a complete genomic sequence and to identify a complete set of genes. An important secondary goal of sequencing the genome of an organism is to gain an understanding of when, where and how a gene is turned on. This process is known as gene expression. Scientists can then compare how a gene is expressed under normal conditions and in an

altered state, such as in disease. Ultimately, one would also like to know the functions of all the proteins encoded by the genome.

1.2 Fungal Genome Initiative

Now that the HGP is complete, genome centers are prioritizing new organisms for genome sequencing. For example, the Whitehead Institute Center for Genome Research has recently defined a Fungal Genome Initiative, which is an effort to provide an impetus for research on the fungal kingdom by prioritizing a set of fungi for genome sequencing.

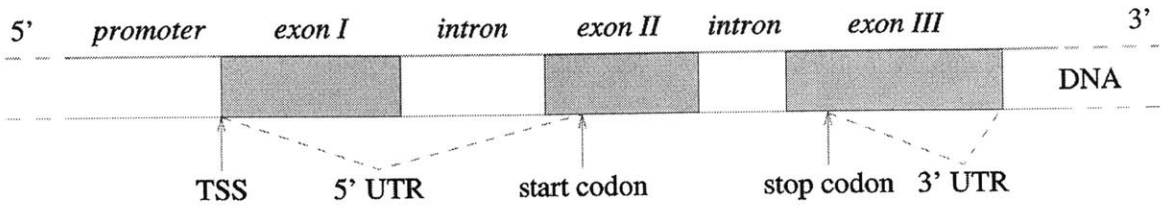
Researchers are interested in sequencing the genomes of various fungi because in addition to presenting threats to human health, they can serve as useful tools for biomedical research. Fungi also have numerous agricultural and industrial uses.

1.3 Gene Structure

Before discussing how coding regions are identified, it is important to review the structure of genes as this will give one a better understanding of the challenges involved in finding genes in uncharacterized DNA. The coding regions of a chromosome are not contiguous since coding regions for different genes are separated by intergenic regions. Furthermore, even the coding region for a single gene is often not continuous since eukaryotic genes are typically they are interrupted by noncoding regions called introns.

One feature of DNA structure that must be taken into consideration when building a gene-finder is that DNA is not one long strand of nucleotides, but rather two strands. However, the second strand is merely the reverse complement of the first strand. The reverse complement strand is obtained by replacing every A with a T, every T with an A, every C with a G and every G with a C. If a DNA strand is 5' ACCGTCGAGA 3', then the reverse complement is 5' TGGCAGCTCT 3'. It can therefore be said that all information about a genomic sequence can be obtained by examining just one strand. However, both strands actually code for the production of proteins. Although there are coding regions on both strands, these coding regions rarely overlap. As a result, when

building a gene-finder, one must ensure that gene predictions for both strands do not overlap.



This picture shows gene structure in eukaryotes. Note that the start codon does not appear in the first exon. As a result, the first exon is never translated.

The process by which DNA codes for the production of proteins and ensure the inheritance of information is explained by the Central Dogma of Molecular Biology. This dogma is represented by four major processes:

1) Replication: The DNA replicates itself in a process involving many protein enzymes.

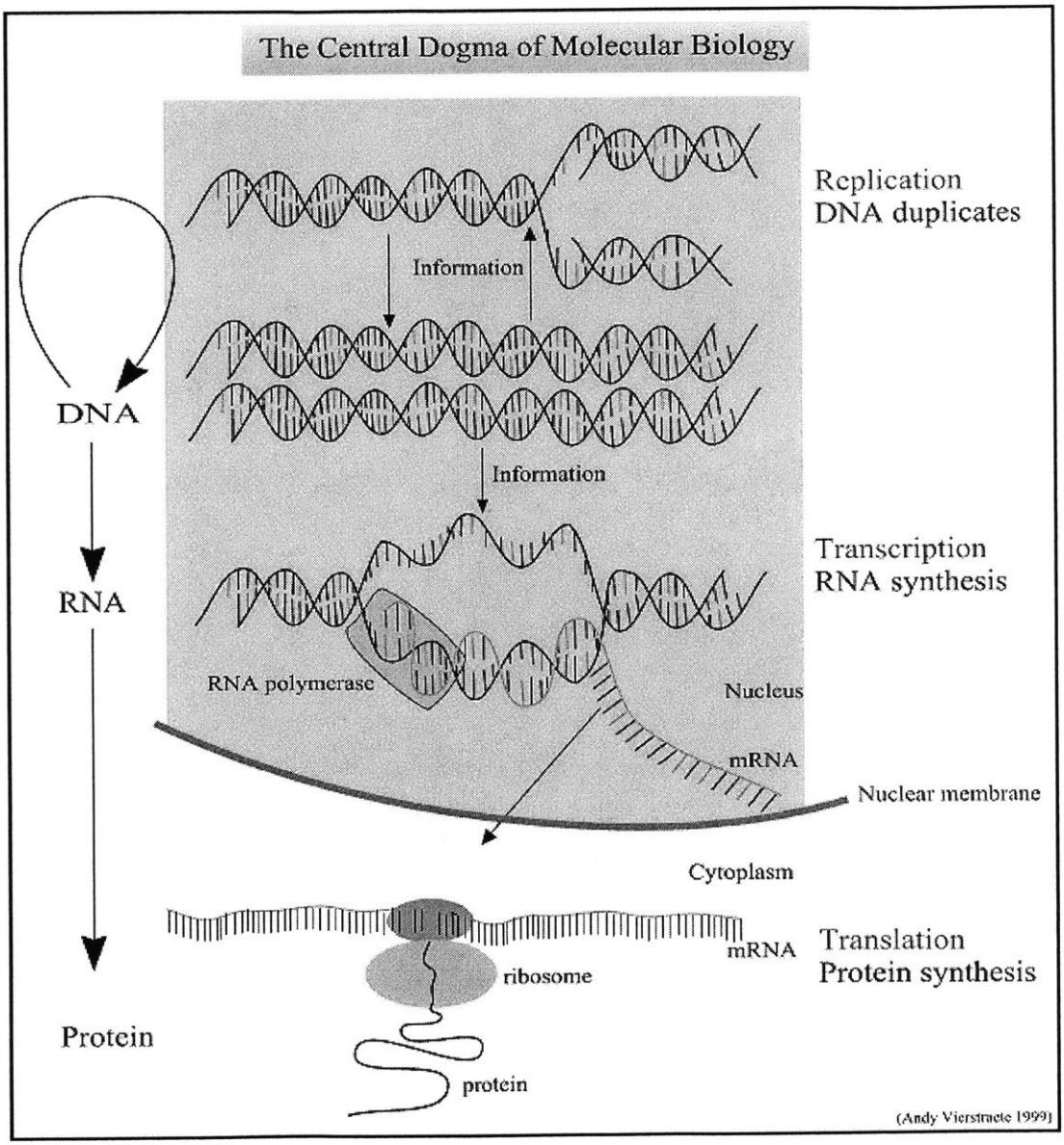
2) Transcription: DNA codes for the production of RNA. RNA is very similary to DNA since the RNA is composed of four nitrogenous bases: guanine (G), adenine (A), cytosine (C) and uracil (U).

3) Processing: Introns are removed from the RNA by a process called splicing and exons are assembled to form messenger RNA (mRNA)

4) Translation: mRNA carries coded information to the ribosomes, where the information is 'read' and used for protein synthesis.

mRNA is a long chain of A, C, G and U nucleotides. However, when thinking about mRNA, it is better to think of mRNA as being a chain of three-nucleotide-long 'words' flanked on either side by untranslated regions (UTRs) known as the 5' and 3' UTRs.

The words are located in the middle of the mRNA and generally occupy most of the mRNA. These three nucleotide words are known as codons. Recall that proteins are long chains of molecules called amino acids. Given that there are four possible nucleotides and given that codons are three nucleotides long, there are $4^3 = 64$ different possible codons. Since cells only use 20 different types of amino acids to construct proteins, it is sufficient for codons to be 3 nucleotides long.



This figure shows the the Central Dogma of Molecular Biology and is avaiable online at http://allserv.rug.ac.be/~avierstr/principles/centraldogma.html

When mRNA is read by the ribosome, it first reads the 5' UTR. The ribosomes only start to direct the construction of amino acid chains once it encounters a special codon called the 'start' codon and stops directing this construction when a 'stop' codon is encountered. The first codon of an mRNA specifies which amino acid will be the first on the chain. Similarly, the second codon specifies which amino acid will be the second on the chain, and so on. In almost all cases, the first codon (known as the 'start' codon) is AUG and the last codon (known as the 'stop' codon) is either UAA, UAG or UGA. Given that mRNA can be thought of as being a chain of three-nucleotide-long words, the length of the protein-coding region of mRNA is always a multiple of 3. This has ramifications in the design of computer programs that predict gene structure in a given DNA sequence input.

The Central Dogma of Molecular Biology may at first appear to some to be an unwieldy process. After all, why would the cell use RNA as an intermediary between DNA and the proteins that it encodes? One advantage of using RNA is that DNA can remain protected in the nucleus and away from the cytoplasm where the ribosomes are located. Another advantage is that it is easier to regulate the expression of a gene when there are more opportunities to control gene expression, e.g. at the level of synthesis of the initial RNA transcript, at the level of RNA splicing, at the level of translation of the processed mRNA, or at the level of the stability of the mRNA.

1.4 Gene-Finding

Once scientists have obtained the genomic sequence of an organism, they must next identify which regions of the sequence code for the production of proteins. The two main methods used to identify genes are (1) sequencing expressed sequence tags (ESTs) and (2) computational approaches.

ESTs are short strands of DNA sequence (typically 500 nucleotides long). They are generated by sequencing either one or both ends of the mRNA of an expressed gene. If scientists have the genome sequence of an organism, they can use these 'tags' to identify genes in chromosomal DNA. Isolating mRNA is essential if one wishes to generate

ESTs. However, mRNA is very unstable outside of a cell. Scientists therefore use enzymes to convert mRNA to cDNA. cDNA is a much more stable compound representing the same sequence information as the mRNA it was derived from. 5'ESTs are usually obtained from the regions of cDNAs that code for a protein while 3'ESTs are usually obtained from regions that fall in the 3'UTR. 5'ESTs therefore tend to be conserved to a far greater degree across species than 3'ESTs.

Although the EST approach has proven to be very fruitful, it is not a panacea. One limitation of the EST approach is that highly expressed genes are represented thousands of times in EST databases while genes expressed at low levels may be absent. Another limitation is that ESTs represent only bits and pieces of mRNAs, not complete sequences, and even when many are known for a given gene, it is common that they cover only parts of the whole mRNA.

There are two main types of computational methods for finding genes: similarity-based approaches and ab initio gene recognition. With homology approaches, exons are identified by comparing conceptually translated regions of a genomic sequence to databases of known protein sequences. Among homology approaches, BLASTX is one of the more popular algorithms. In addition to genomic:protein comparison methods, there are also genomic:genomic comparison methods. One example of a genomic:genomic comparison method is the GLASS/ROSETTA approach. This method of gene-recognition simultaneously analyzes homologous loci from human and mouse genomic sequences and identifies coding exons. GLASS is an alignment algorithm that provides global alignments of human and mouse genomic regions and ROSETTA is a program that identifies coding exons in both species (Batzoglou et al 2000).

Programs that perform ab initio recognition only use the information contained in the input sequence itself when predicting gene structure. Homology approaches, by contrast, also use databases and cross-species comparisons. Most ab initio programs available to the public were developed for gene-finding in vertebrates (Chen and Zhang 1997). Many gene-finding programs use Hidden Markov Models (HMMs), a stochastic signal model, for predicting gene structures. Examples of HMM-based gene-finders for vertebrates include GENSCAN (Burge 1997), HMMgene (Krogh 1997) and Genie (Kulp et al. 1996). Other vertebrate gene-finders include FGENEH (Soloyev, 1995), which uses

dymanic programming and pattern recognition algorithms, and Morgan (Salzberg et al. 1998), which uses a combination of decision trees, dynamic programming and Markov chains.

Several comparative analyses of the performance of gene-finding programs have been performed in recent years. In 1996, Burset and Guigo (Burset and Guigo 1996) compared the performance of GeneID, SORFIND, GeneParser2 and GeneParser3, GRAIL 2, GenLang, FGENEH and Xpound. In 2001, Rogic, Mackworth and Ouellette (Rogic et al, 2001) performed a comparative analysis of several new programs that were developed since the Burset/Guigo analysis was published. These programs were: FGENES, GeneMark.hmm, Genie, GENSCAN, HMMgene, Morgan and MZEF.

While there are a wide variety of gene-finders for mammalian sequences, there haven't been nearly as many gene-finders developed for fungal genomes. Pombe (Chen and Zhang 1997) is a publicly available program that was designed to identify protein coding regions in *Schizocassharomyces pombe* (*S. pombe*) genomic sequence. This program uses linear discriminant analysis and dynamic programming. Another program designed for fungal genomes is Fgenesh, which is an HMM-based program with the algorithm similar to GENSCAN and Genie. Fgenesh is not available publicly.

One might think that scientists annotating genomes may only be interested in only using the best available gene-finders. However, scientists annotating genomes often make use of several different gene-finders. When the *Neurospora crassa* genome was sequenced and annotated, FGENESH, FGENESH+ and Genewise were all used (Galagan et al. 2003).

1.5 Goals of this thesis

While there are several HMM-based gene-finders available for mammalian sequences, there doesn't exist a publicly available HMM-based gene-finder for fungal genomes. The first goal of this thesis was to develop FUNSCAN, a new HMM-based gene-finder for fungal genomes using some of the ideas from HMM-based gene-finders such as GENSCAN and to apply FUNSCAN to all available fungal genomes. Once FUNSCAN

performs at a level comparable or better than pombe, I will like to make it publicly available. In addition to the predictive goal of high predictive accuracy, the following properties for FUNSCAN were also considered desirable:

1) FUNSCAN should be trainable with a minimum of effort. Since one objective of developing FUNSCAN was to apply it to several different genomes, it was considered desirable to automate as much of the training process as possible.

2) FUNSCAN should perform at a speed that is comparable to most other gene finding programs. For example, FUNSCAN should be able to process sequences of 100 kilobases (10^5 bases) in less than one minute.

The second goal of this thesis was to explore new ways in which small datasets and comparative genomics could be leveraged. Since fungal genes are much less studied than mammalian genes (with the exception of the model organism *Saccharomyces cerevisiae*), there are often only a small number of high-quality genes available for training fungal gene-finders. Furthermore, since several new fungal genomes will be made publicly available in the next few years, it is worth exploring how tools such as BLAST and FUNSCAN can be combined to develop new algorithms for similarity-searches.

1.6 Outline of this work

This work is structured as follows:

Chapter 2 provides a concise introduction to HMMs. After discussing HMM-based algorithms and providing an example of an HMM, I describe how HMMs can be applied to gene-finding and I provide a brief overview of the architecture of GENSCAN, an HMM-based genefinder

Chapter 3 describes the architecture of FUNSCAN and explains why this design was chosen. The training and testing of FUNSCAN is also described

Chapter 4 is devoted to exploring how FUNSCAN can be trained with small datasets and how it can be used in comparative genomics.

Chapter 5 provides a summary of this work.

Chapter 2

BACKGROUND IN HMMs AND BIOINFORMATICS

Signals can be classified broadly into two classes: deterministic and stochastic signals. A deterministic signal is a signal in which each value of the signal is fixed and can be determined by a mathematical equation, rule or table. Deterministic models exploit a known property of the signal (i.e., signal is a sine wave), thus making signal specification very straightforward. All that is required is to obtain accurate estimates of the parameters of the signal model (i.e. frequency, amplitude, phase).

When dealing with stochastic signals, one does not try to characterize the precise behavior of the signal. Instead, one tries to characterize the statistical properties of the signal. Examples of statistical models include Gaussian processes, Poisson processes, Markov processes and Hidden Markov processes.

The main difference between deterministic and stochastic signals is that while the values of deterministic signals can be calculated from past values with complete confidence, stochastic signals have uncertainty about their future behavior.

Chapter 2 discusses Hidden Markov process and examines how they can be used to build a gene-finder. Section 2.1 provides a brief review and an example of a Markov process. This discussion provides a suitable background for section 2.2, in which HMMs are explained and an example is discussed. Section 2.3 discusses how HMMs can be used to identify coding regions in genomic sequences and provides an overview of GENSCAN, an HMM-based gene-finder is provided. The discussion of FUNSCAN in Chapter 3 builds upon and extends many of the ideas discussed in this chapter.

2.1 Discrete Markov Processes

Consider a system which is described as being in one of a set of N distinct states, S_1, S_2, ... S_N. If the probability of transition from state S_i at time t to state S_j at time $t + 1$ depends only on S_i (i.e. the state at time t) and not on previous history of the process (i.e. the states at times t -1, t -2, etc..), then the process is said to be a first-order Markov chain. It can be described in the following manner:

$$P[q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \ldots] = P[q_{t+1} = S_j | q_t = S_i]$$

Similarly, if the probability of transition from state *Si* at time n to state *Sj* at time t + 1 depends only on *Si* (the state at time t) and on *Sk* (the state at time t-1), the process is said to be a second order Markov chain. More generally, an nth-order Markov chain can be described in the following manner:

$$P[q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \ldots, q_0 = S_z]$$
$$= P[q_{t+1} = S_j | q_t = S_i, \ldots, q_{t-n+1} = S_x]$$

An example of a first-order Markov model is a simple 3-state model of the weather. In this model, the weather can be described as being in the following states:

State 1: rain
State 2: cloudy
State 3: sunny

Since this is a first-order Markov model, the weather on day $t+1$ depends only on the weather on day t. For a first-order Markov model, the state transition probabilities are defined in the following manner:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$$

All state transition probabilities have a value which is greater than or equal to zero. A matrix of state transition probabilities for the Markov model of the weather is the following

$$a_{ij} = \begin{vmatrix} 0.5 & 0.3 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{vmatrix}$$

From this matrix of state transition probabilities, one can make some observations about the behavior of our weather system. For example, if the weather is in state x on day t, then on day $t + 1$, the state that the weather is most likely to be in is state x.

2.2 Hidden Markov Models

Markov models are considered to be too restrictive to be applicable to many problems of interest because each state corresponds to an observable event. A stochastic process where the observation is a probabilistic function of a hidden state is called a Hidden Markov Model. The coin toss model described in the following paragraph is an example of a Hidden Markov Model (HMM). This example is also discussed in a paper written by Lawrence Rabiner entitled "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition" (Rabiner, 1989), which is an excellent reference for HMMs.

Assume that you are in a room with a curtain behind which you cannot see what is happening. On the other side of the curtain, another person is performing a series of coin flips and is telling you the result of each coin flip. For each coin flip, this person is using a gold coin, a silver coin or a bronze coin. However, this person will not tell you which coin is used for each coin flip, i.e. a series of *hidden* coin tossing experiments is being performed.

The *observation sequence* of this HMM is the series of heads and tails results from the experiment and the hidden *state sequence* of this HMM is the sequence in which the coins were used. Here is an example of a typical observation and state sequence

$$\mathbf{O} = O_1\ O_2\ O_3\ \ldots O_T$$
$$= H\ H\ T\ T\ T\ H\ T\ T\ H\ \ldots\ H$$

$$\mathbf{S} = S_1\ S_2\ S_3\ \ldots\ S_T$$
$$= G\ G\ B\ S\ S\ G\ S\ G\ \ldots\ B$$

Here are what typical state transition probability distributions and observation probability distributions would be:

State Transition Probability

	To Gold	To Silver	To Bronze
From Gold	.8	.1	.1
From Silver	.5	.3	.2
From Bronze	.6	.3	.1

Observation Probability

	COIN		
	Gold	Silver	Bronze
P(heads)	.8	.5	.2
P(tails)	.2	.5	.8

More generally, an HMM is characterized by the following (Rabiner 1989):

1) *N, the number of states.* In the coin-tossing experiment, the states correspond to the coins that were used for each flip. The set of individual states is denoted as S = $\{S_1, S_2, \ldots, S_N\}$.

2) *M, the number of distinct observation symbols per state.* In the coin-tossing experiment, the observation symbols for all three states are Heads and Tails. The individual symbols are denoted as V = $\{v_1, v_2, \ldots, v_M\}$.

3) *The state transition probability distribution $A(a_{ij})$, which is the probability oftransition from A_i -> A_j.* In the coin-tossing experiment, the state transition probabilities describe which of the three coins the experimenter will use next. The probabilities are a function of which coin the experimenter has just used.

4) *Observation symbol probability distribution in each state.* In the coin-tossing experiment, this corresponds to the probability of getting either heads or tails for each of the biased coins. The probability of observing the symbol k in state j is $b_j(k)$, where

$$b_j(k) = P[v_k \text{ at } t | q_t = S_j], \text{ for } \quad 1 \leq i \leq N$$
$$1 \leq k \leq M$$

5) *Initial State Distribution.* This is the probability of being in each possible state at the start of the experiment. This is denoted in the following manner:

$$\pi_i = P[q_1 = S_i], 1 \leq i \leq N$$

In the coin-tossing experiment, a few properties can be inferred about the coins that are being used and the coin preference of the person performing the experiment. For example, the silver coin is unbiased while the gold and bronze coins are biased. In addition, the person performing the experiment behind the curtain has an obvious preference for using the gold coin in the experiments. As a result, if the person announces several heads in a row, then it is reasonable to hypothesize that the gold coin was used for those flips.

Given the definition of HMMs, there are two basic problems that must be solved in order for the model to be useful in gene-finding applications:

1) Given an observation sequence $\mathbf{O} = O_1, O_2, O_3, \ldots, O_N$ and an HMM model, how do you efficiently computer P(**O**|model)?

2) Given an observation sequence $\mathbf{O} = O_1, O_2, O_3, \ldots, O_N$ and a model, how do you choose the state sequence $\mathbf{S} = S_1, S_2, S_3, \ldots, S_N$ that maximizes P(**O**, **S**)? In this case, **S** is the state sequence that best "explains" the observations.

The solution to the first problem is the forward algorithm (Rabiner 1989). The key idea behind the forward algorithm is that one must only computer $P(O, S_t = i|\text{model})$ for each state I recursively. P(O|model) will then be the sum of the probabilities in each of the possible states at time N. Consider the forward variable:

$$\alpha_t(i) = P(O_1\ O_2\ \ldots\ O_t,\ q_t = S_i|\text{model})$$

This is the probability of the partial observation sequence *O1 O2 ... O_t* and also having the HMM be in S_j at time *t*, given our model. Given the above definition of the forward variable $\alpha_t(i)$, it can be seen that if one can calculate $\alpha_T(i)$ for each possible state *i*, then the first problem can be solved by simply taking the sum of $\alpha_T(i)$ for each possible state *i*. $\alpha_T(i)$ for each *i* is calculated in the following manner:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad 1 \le t \le T-1, \quad 1 \le j \le N$$

3. Termination:

$$P(O|\text{model}) = \sum_{i=1}^{N} \alpha_T(i).$$

The solution to the second problem is the Viterbi algorithm, which is very similar to the Forward algorithm. The Viterbi algorithm finds the single best state sequence Q = {*q1 q2 ... qT*} for the given observation sequence O = {*O1 O2 .. O_T*}. Just as an $\alpha_t(i)$ variable was defined for the first problem, a $\delta_t(i)$ variable is defined for the Viterbi algorithm, where $\delta_t(i)$ is the probability of the observing the observation sequence when the underlying state sequence is the best path, but ends in q_i.

$$\delta_t(i) = \max \quad P[q_1\ q_2\ \ldots\ q_t = i,\ O_1\ O_2\ \ldots\ O_t|\ \text{model}]$$

$\delta_t(j)$ can be calculated recursively in the following manner:

$$\delta_{t+1}(j) = [\max \delta_t(i) a_{ij}] \cdot b_j(O_{t+1})$$

Just as for the forward algorithm, where there was an $\alpha_T(i)$ for each state i, there will be an $\delta_T(i)$ for each state i in the Viterbi algorithm. However, for the Viterbi algorithm, one must select the $\delta_T(i)$ that has the maximum value. In the Forward algorithm, we merely take the sum of all $\alpha_T(i)$ values.

Once the maximum $\delta_T(i)$ is established, it is known that the last state of **S** is S_i. To retrieve the rest of S, we need to keep track of the state that maximizes $\delta_{t+1}(j)$. Here is the complete procedure for finding the best state sequence:

1. Initialization:

$$\delta_1(j) = \pi_i b_i(O_1), \qquad 1 \le i \le N$$

2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \, b_j(O_t), \qquad 2 \le t \le T, \quad 1 \le j \le N$$

$$\psi_t(j) = \operatorname{argmax}_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \qquad 2 \le t \le T, \quad 1 \le j \le N$$

3. Termination:

$$q_T = \operatorname{argmax}_{1 \le i \le N} [\delta_T(i)]$$

4. Path (state sequence) backtracking:

$$q_t = \psi_{t+1}(j)(q_{t+1}) \qquad t = T-1, T-2, \ldots, 1.$$

2.3 Application of HMMs to gene-finding

As can be seen from the discussion of HMMs in the previous section, HMMs have the ability to model *grammar*. Suppose one wanted to modify the coin-tossing experiment from the previous chapter to include the following rule: when the person behind the

curtain uses a silver coin, the person cannot use a gold coin for the next coin toss. This rule could easily be incorporated into the HMM by setting $a_{silver, gold}$ equal to 0. If the states of the coin-toss experiment were considered as being 'words' in a language, one rule of the grammar of this language would be that one cannot have a sentence of the form ....silver-gold.... .

The problem of analyzing genomic sequences and predicting gene structures also has a grammatical structure. If one considers 'intergenic', 'exons' and 'introns' as being words in a sentence, then the sentences of gene-structure must be of the following form exon-intron-exon-intron...exon-intron-exon. These sentences must always begin with an exon and exons cannot be adjacent to one another; they must be separated by exactly one intron.

2.3.1 Signals in gene-finding

In addition to gene structures possessing a grammatical structure, genes also possess several notable signals which must be taken into consideration when building a gene-finder. With very rare exceptions, eukaryotic genes possess each of the following properties:

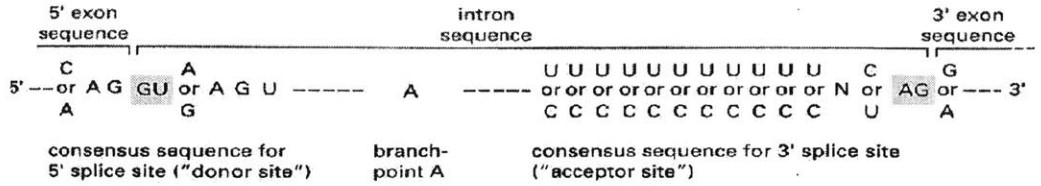
1) Start codon: The first three bases of the first exon, are ATG

2) Stop codon: The last three bases of the last exon of a gene are one of the three stop codons TAA, TAG or TGA.

2) Donor and acceptor sites: The first two bases of an intron are GT (the donor or 5' site) and the last two bases of an intron are AG (the acceptor or 3' site).

Consensus sequence of splice donor and acceptor sites

These sites are used by most pre-mRNA molecules.



This picture shows the donor and accpetor site of an intron. In addition, a base from the branch signal is shown. Picture is available online at (http://amiga1.med.miami.edu/Medical/Werner/Images/Lecture6/Consensus%20splice%20sites.JPG)

3) Branch signal: Although branch signals are conserved to varying degrees in different organisms, they are well-conserved in fungi. The branch signal is a seven base-pair long sequence that is located 10-30 nucleotides away from the end of an intron. The branch signal in fungi and methods for identifying it will be discussed in greater detail in Chapter 3.

2.3.2 Overview of GENSCAN

GENSCAN is a semi-Markov HMM-based gene-finder. The difference between a Hidden Semi Markov Model (HSMM) and an HMM is how state duration is modeled. In an HMM, the amount of time that is spent in a particular state follows a geometrical distribution. However, with HSMM models, the time spent in a particular state is not necessarily geometric; it can be modeled by other distributions. It can be stated that an HMM is a HSMM where state duration is modeled with a geometric distribution. GENSCAN was designed to predict genes in vertebrate organisms. It is desirable to use

an HSMM when building a gene-finder for vertebrates because there is a great deal of variability in the length of vertebrate introns. For fungi, the length of introns does not vary a great deal. In vertebrates, however, intron length can vary from 100 to 50000 nucleotides.

The Gene model of the input DNA sequence of GENSCAN has 27 states and simultaneously predicts genes on both strands. As we will see, FUNSCAN first processes the forward strand and then processes the reverse strand. GENSCAN models the intergenic region and also the promoter and 5'UTR, which are both located before the first exon. Since some genes do not contain introns (i.e. they are 'single-exon' genes), GENSCAN models single-exon genes and intron-containing genes in two different manners. The FUNSCAN gene-model, on the other hand, models one generalized gene type (which may or may not have introns).

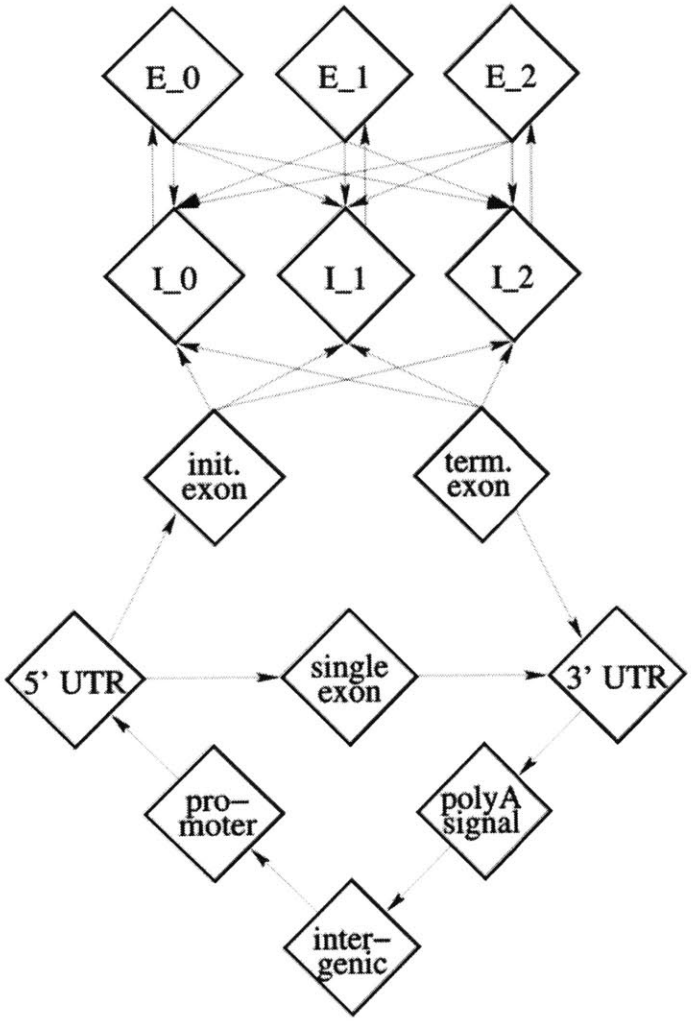
Although the gene-structure for intronless genes is straightforward, the gene-structure for genes with introns is more elaborate. The E_0+, E_1+, E_2+, I_0+, I_1+ *and* I_2+ states are used to ensure that the length of the coding region of the mRNA is a multiple of 3.

Gene signals such as the start codons, stop codons, donator sites and acceptor sites were modeled implicitly in GENSCAN as part of the associated exon models in order to keep the number of states manageable (Burge 1997).

The purpose of examining the GENSCAN model is to demonstrate that there are several design decisions to be made when building an HMM-based gene-finder, a fact that will become readily apparent when the FUNSCAN gene model is discussed in the next chapter. The following table summarizes the features of GENSCAN and FUNSCAN that were discussed in this chapter:

FEATURE

	State duration	**Overlapping genes**	**State design**
GENSCAN	HSMM	Genes on both strands predicted simultaneously	Gene signals modeled implicitly in states
FUNSCAN	Classical HMM	Strands are processed separately	Gene signals modeled explicitly using states



This is the gene model for GENSCAN. 13 coding regions states and the intergenic state is shown. The 13 states not shown in this diagram are identical to the 13 coding region states shown above.

Although HMMs are effective for solving the gene-finding problem, they are not sufficient for all problems in bioinformatics. With RNA folding, there are correlations between bases far apart from each other in the linear sequence. For RNA folding, Stochastic Context-Free Grammars (SCFGs), which generalize HMMs, have proven to be effective.

Chapter 3

IMPLEMENTATION AND TESTING OF FUNSCAN

This chapter covers the implementation and testing of the FUNSCAN program. Section 3.1 describes the GenBank format for annotating genes and how training and test data sets were obtained. Section 3.2 describes the model architecture of FUNSCAN and Section 3.3 goes into greater detail into how both the observation symbol probability distribution and the state transition probability distribution were obtained. FUNSCAN performance and testing is discussed in section 3.4.

3.1 Construction of Datasets

Publicly available DNA sequences are archived at GenBank, the National Institute of Health (NIH) genetic sequence database that is available online. Sequences are submitted to GenBank from scientists around the world; many journals even require submission to a database at time of publication. A certain errors are present in sequences that are submitted to GenBank and in the annotation that accompanies the sequence. For example, sequences may not have the proper donor and acceptor sites (GT and AG respectively); in one case, a sequence had an intron that was only 1 nucleotide long.

In addition to this, some of the genes structures submitted to GenBank are computationally predicted, as opposed to experimentally verified ones. It is for this

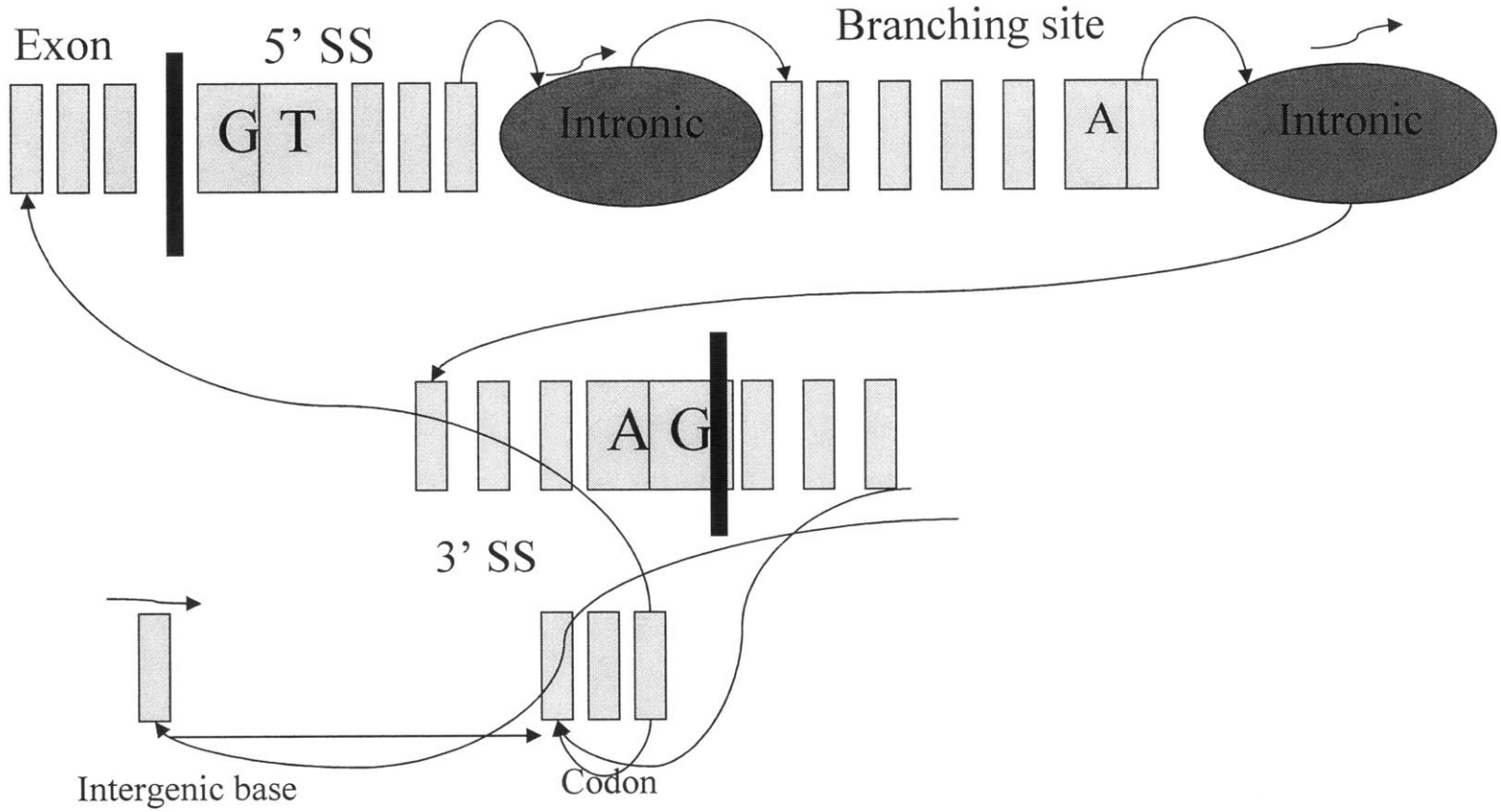
reason that a gene annotation script called GENOA was developed in the Burge lab (Lee P. Lim, Ru-Fang Yeh, Dirk Holste and Chris Burge, unpublished). GENOA takes available cDNAs for an organism and aligns then to their genomic loci, generating a 'gold standard' set of annotated genes.

GenBank has a standard format, known as 'GenBank format' for submissions of DNA sequences with associated annotation of gene-structures. An example of a DNA sequence in Genbank format is included in Appendix A. For the purposes of training and testing FUNSCAN, the most important data element is the 'CDS' field, which indicates the region of the DNA sequence that corresponds with the sequence of amino acids in a protein (CDS includes start and stop codons).

3.2 Gene Model of FUNSCAN

The figure discussed in this section is a conceptual model of FUNSCAN. The only difference between the conceptual model of FUNSCAN and the actual model is that the conceptual model does not explain how FUNSCAN ensures that the lengths of predicted coding regions will always be a multiple of 3. Still, the actual model is much more effective at highlighting the key features of FUNSCAN.

Starting from the intergenic state, a gene is first defined by a special start codon state. Notice that the promoter region and the 5' UTR is not modeled. After the start state, the next structure defined is the codon state. After the codon state, the next state can either be the stop codon state or the intergenic state. After the intergenic state, the next state is the codon state. For all genes, the last state that is entered is the stop codon state. The next state after the stop codon state is the intergenic state.



This is the conceptual model of FUNSCAN. The intronic state is represented by a circle, but all other states are represented by rectangles. Note that the arrows above the Intergenic and Intronic states mean that it is possible to be in the Intergenic state at both times t and t+1. In this model, when two rectangles are adjacent to each other and no arrow is present, the state transition probability between them is equal to 1.

Each state in FUNSCAN emits exactly one base. The following states are fixed, meaning that they will always emit the same base(s):

Start State	Base Emitted
1	A
2	T
3	G

Stop State	Base Emitted
1	T
2	A or G
3	A or G (function of previous 2 emissions)

Intron State Position	Base Emitted
1	G
2	T
L - 1	A
L	G

One of the states in the branch site is also fixed, but this will be discussed in the next section.

The method by which FUNSCAN ensures that the lengths of predicted coding regions are a multiple of 3 is very similar to the method that is used by HMMgene (Krough, 1998).

3.2.1 Branch Site Located in Introns

Of all the gene elements included in the FUNSCAN gene model, the branch site is the only one that was not discussed in the Chapter 1. The third stage of the Central Dogma of Molecular Biology, discussed in section 1.3, is where introns are removed from RNA. This process is known as RNA splicing and a seven-nucleotide-long structure known as the branch site is involved in this process. The branch site is located approximately 10 to 40 nucleotides away from the acceptor site. The distance from the branch site to the acceptor site is not uniform in each organism; it varies for each intron in an organism and the average distance varies between organisms. Although the branch site is present in all introns, it is conserved to varying degrees in different organisms. In *Saccharomyces cerevisiae* (budding yeast), the sequence is almost invariably TACTAAC. While the sequence is not conserved as strongly in other organisms, the sixth base is always 'A' and the other six bases are also well-conserved in most fungi.

RNA splicing is an area in which active research is being conducted. This is because many genes can be spliced in more than one way. Such genes are said to be 'alternatively spliced'. Since some genes are alternatively spliced, knowledge of a genome sequence does not imply a comprehensive knowledge of its transcriptome- the set of gene transcripts (mRNAs) that the genome encodes (Burge 2001). Just as biochemists in the 1960s identified the rules governing the translation of mRNAs into amino acid chains, today's scientists are attempting to identify the rules that govern the ways in which an RNA transcript is spliced. The rules governing RNA splicing are likely to be more complicated than those of translation, and they will not be the same for all organisms.

3.3 Observation and Transition Probabilities

Recall from the discussion on Hidden Markov Models in Section 2.2 that Hidden Markov Models are characterized by the following five features:

- *N, the number of states*
- *M, the number of distinct observation symbols per state*
- *The state transition probability distribution $A(a_{ij})$, which is the probability of transition from A_t -> A_{t+1}.*
- *Observation symbol probability distribution in each state (emission probabilities)*
- *Initial State Distribution*

Information about the number of states and the number of distinct observation symbols per state can be obtained from the figures. However, only limited information about the state transition probabilities and observation symbol probability distribution can be obtained from the FUNSCAN gene model in Section 3.2.

3.3.1 Calculating the Observation Symbol Probability Distributions

With the exception of the fixed states that were discussed in the previous chapter, every state can emit any one of the four possible nucleotides (A, C, G, T). How does one determine what the probability distributions should be for these states?

Before answering this question, let us first examine how one might determine observation symbol probability distributions for the coin toss experiment introduced in section 2.2, assuming that a person has access to the gold, silver and bronze coins that are used in the experiment. One would flip each coin many times and keep track of the result of each flip. Using the maximum likelihood estimator, the observation symbol probability distribution for the gold coin is simply:

P(heads|gold coin) = # of heads / # of flips P(tails|gold coin) = # of tails / # of flips

The same procedure is repeated for the silver and bronze coins. One might then choose to use the maximum likelihood estimator for calculating the observation symbol probability distributions. However, such a design does not take into account important facts about coding regions. For example, stop codons (TAA, TAG, TGA) never appear in coding regions, so the conditional probability of an A or G at the third site of a codon following TA at the first two positions should be zeros and other biases exist at the level of DNA triplets. It is for this reason that the observation symbol probability distributions of FUNSCAN are functions of what the previous two bases were.

For the intergenic state, the intronic state and for codon state 1, codon state 2 and codon state 3, the observation symbol probability distributions are stored in 16x4 matrices. Appendix B shows the format that is used to submit observation symbol probability distributions to FUNSCAN. Each row of the matrix corresponds to the previous two bases and each column corresponds to the observation symbol probability distribution of the four nucleotides. For example, $b_{intergenic}$(A|previous two bases were AA) corresponds to the $Intergenic_{1,1}$ matrix entry, $b_{intergenic}$(T|previous two bases were TT) corresponds to the $Intergenic_{16,4}$ matrix entry and $b_{intergenic}$(C|previous two bases were GC) corresponds to the $Intergenic_{11,2}$ matrix entry. Because stop codons do not occur in coding regions, one can also deduce that $b_{codon\_state\_3}$(A|previous two bases were TA), $b_{codon\_state\_3}$(A|previous two bases were TG) and $b_{codon\_state\_3}$(G|previous two bases were TA) will always equal 0. Since the intergenic, intronic and the three codon states are dependent on the previous two bases, the emission probabilities of FUNSCAN are

second-order HMM. Second-order probabilities are still estimated using the maximum likelihood estimator discussed in chapter 2; they are a conditional version of maximum likelihood.

Although the maximum likelihood estimator is not suitable for the intergenic, intronic and coding states, it is used by FUNSCAN for the donor and acceptor sites. The maximum likelihood estimator is also used for determining the observation symbol probability distributions for each of the seven bases for the branching site.

To summarize, here is a table that lists how the observation symbol probability distributions are obtained:

STATE	METHOD OBTAINED
Start	Maximum likelihood
Stop	Maximum likelihood
Intergenic	Second order
3 codon states	Second order
Intronic	Second order

3.3.2 Gibbs Sampling for the Branching Site

The only states missing from the above table are the branch site states. These states are actually obtained by using the maximum likelihood method. However, *identifying* the branch sites in introns is not trivial; a procedure called Gibbs Sampling is used (Lawrence et al. 1993). Since Gibbs sampling is a stochastic process, it is not guaranteed to produce the same results each time it is performed. In order for the branch site to be identified by the Gibbs sampler, only the last 40 bases of each intron must be submitted to the Gibbs sampling algorithm. If entire introns are submitted, the Gibbs sampling algorithm will not be able to identify the intron regions.

The Gibbs sampler takes as input several sequences and the length of the motif (in the case of FUNSCAN, the motif is the branch site). The output is the motif and statistics about the motif, such as the frequency that the motif appears in the input sequence and maximum likelihood estimators for each motif position. In addition, the Gibbs sampler

will indicate where the motif is located in each sequence. This is very useful for calculating state transition probabilities, as we will see in the next section.

3.3.3 Calculating State Transition Probability Distributions.

The methods used for calculating the state transition probabilities are very intuitive. However, the parameters obtained by these methods did not always prove to be optimal. The quality of parameters was verified by calculating the state transition probabilities according to the formulas described below and then using them when testing on the training set. The reason why the gene-finder was tested on the training set is because one doesn't want to test on a testing set until a final gene-finder is ready. Automated testing has been set up for FUNSCAN. Whenever a series of GenBank files is provided as input, the output is a series of test statistics. These test statistics will be discussed in greater detail later in this chapter.

These values were then modified slightly by hand in order to ensure that they were indeed the optimal values. Below is a brief description of how several transition probabilities were calculated. In addition, comments are included that describe how successful each method proved to be:

> ***Intergenic to start codon*:** In a training set, the number of intergenic bases and the number of genes present are computed. The transition probabilily for $\mathbf{A_{intergenic,start}}$ is calculated using the following formula: #genes/#intergenic bases. The transition probability for $\mathbf{A_{intergenic,intergenic}}$ is equal to $\mathbf{1 - A_{intergenic,start}}$. These formulas proved to be fairly effective; the number of genes predicted was about right. This result is not very surprising, given that a gene is a very strong signal and that it would require a very poor parameter formula in order to miss a complete gene. This hypothesis was tested by testing a gene-finder on the training set and modifying the order of this parameter by a factor of 1000. Since the gene-finder still performed well, one can conclude that there can be a lot of variability in the value of this parameter.

Codon transition formulas: When the model is in the codon state, the model can either remain in the codon state or change to the stop codon state or one of the intergenic states. When the training set was processed, the number of exons and the total number of exonic bases was computed. The following formulas were then used:

$\mathbf{A_{codon\_state\_3,stop\_codon\_1}}$ = #genes / # bases in exons with stop codon

$\mathbf{A_{codon\_state\_3,intronic}}$ = (#total exons - #exons with stop codon)/ (# bases in all exons except exons with stop codons)

$\mathbf{A_{codon\_state\_3,codon\_state1}} = 1 - (A_{codon\_state\_3,intronic} + A_{codon\_state\_3,stop\_codon\_1})$

Given that exons are hundreds of bases long, the values of the first two parameters are very small and the value of the third parameter is very close to 1. The success of these formulas varied greatly. On some organisms, these formulas worked very well. However, for other organisms, the value for $A_{codon\_state\_3,intergenic}$ was from optimal. After testing on the training set, it sometimes turned out that the optimal value for this parameter differed by a factor of 10 from the actual optimal value. This might be because exon length is not actually distributed geometrically. Recall that for HMMs, all state durations are geometrically distributed.

Intron transition probabilities: There are two parameters that must be calculated here since there is an intron state before the branch site and an intron site after the branch site. The follow are the equations that are used in calculating these parameters. Explanations are included where necessary

(i) $\mathbf{A_{intronic\_before\_branch,branch\_site}}$ = num_introns / (num_intronic_bases – [num_genes(20 + 7 + avg_dist_branch_acceptor)])

(ii) $\mathbf{A_{intronic\_before\_branch,intronic\_before\_branch}} = 1 - A_{intronic\_before\_branch,branch\_site}$

Explanation: For introns, FUNSCAN models the first 10 bases, the last 10 bases and the seven branch site bases using position-specific probabilities. This is the purpose of having 20+7 in the equation. Since the branch site is located 10-30 bp upstream of the acceptor site, this amount must also be taken into consideration when computing a parameter. The denominator of the first term is the total number of intronic bases excluding the first ten from each intron, the

branch site and the bases downstream from the branch site of each intron. The second equation follows from the first equation.

(iii) $A_{intronic\_after\_branch,acceptor\_site}$ = num_introns / (num_intronic_bases – [num_genes(7 + all_genes_upstream_of_branch_site + 10)])

(iv) $A_{intronic\_after\_branch,intronic\_after\_branch} = 1 - A_{intronic\_after\_branch,acceptor\_site}$

Explanation: The logic behind the equations for these parameters is similar to the logic used in obtaining parameters for the two parameters above. Since introns are on average over 100 bases long, one would expect the value of (ii) to be higher than (iv). This is always the case, since (iii) is greater than (i).

Although these parameters are not always optimal, they need to be adjusted by relatively small factors. When training a model, it is recommended to modify these parameters simultaneously with codon transition probabilities in order to find an optimal solution.

The state transition probabilities are very useful for trouble-shooting. Whenever training FUNSCAN, one should always first test on the training set in order to identify problems that can be corrected by adjusting the state transition probabilities. A test set should always be kept separate from the training data. The following table discusses common problems with training FUNSCAN and how they can be corrected by modifying state transition probabilities. All of these situations have arisen when training FUNSCAN for different organisms

PROBLEM	SOLUTION
Too many or too few genes are being predicted	Modify $A_{intergenic,start}$ accordingly
Too many or too few introns are being predicted	Modify $A_{codon\_state\_3,intronic}$ accordingly
FUNSCAN finds intronless genes well, but is not effective at finding genes with introns	Modify $A_{codon\_state\_3,intronic}$ accordingly

3.4 Evaluating the performance of FUNSCAN

When building a gene-finder set, how does one select test statistics that will provide useful information for the person attempting to improve the performance of the gene-finder? There are two main types of test statistics: accuracy by nucleotide or accuracy by correct gene structure. An example of accuracy by nucleotide would be statistics such as percentage of exonic nucleotides predicted. Examples of accuracy by correct gene structure are statistics such as percentage of exon structure and intron structures accurately predicted (ie gene-finder predicts correct boundaries).

For the evaluation of FUNSCAN performance, several statistics on accuracy by correct gene structure were chosen. Statistics that examine the percentage of correct gene structures were chosen because they give describe more precisely the performance of the gene-finder. A good performance for accuracy by correct gene structure predictions implies that the gene-finder will also perform well when evaluated with accuracy by nucleotide statistics. However, the converse is not necessarily true.

The following statistics were selected in order to evaluate the performance of FUNSCAN:

Fraction of Correct Start/Stop Exons: A start/stop exon is an intronless gene.

Fraction of Correct Start Exons: A start exon is the first exon in genes that contain introns

Fraction of Correct Stop Exons: A stop exon is the last exon in genes that contain introns

Fraction of Correct Middle Exons: A middle exon are exons that are not Start exons or Stop exons in genes that contain at least two introns.

Fraction of introns correct (intron accuracy): This statistic measures what percentage of introns in a dataset are identified correctly. Since gene density is quite high in fungal genomes, one must 'trim' each gene in the test set. This means leaving 500 nucleotides flanking each side of the CDS region. If more bases are left, then it is possible that FUNSCAN may be discovering new genes, even though they have not been annotated.

When testing FUNSCAN, special attention will be paid to the intron accuracy statistic. This is because genes that contain introns are the most difficult to identify, and successful prediction of introns can assist homology methods in confirming gene predictions.

3.4.1 FUNSCAN Performance for *S. pombe*

Schizosaccharomyces pombe (*S. pombe*) is a fission yeast that was the sixth eukaryotic genome to be sequenced (Wood et al. 2002). Of all the fungi of interest, GENOA (the gene annotation script used in the Burge lab) obtained the most sequences for *S. pombe*. GENOA also obtained a large number of genes for *Saccharomyces cerevisiae (S. cerevisiae)*, a budding yeast that was the first eukaryotic organism to be sequenced (Wood et al. 2002). However, *S. cerevisiae* is not a very interesting organism for the purposes of gene-finding because very few of its genes have introns. As a result, it is very easy to build a gene-finder for *S. cerevisiae*.

GENOA obtained 234 genes for *S. pombe*. From these genes, 132 genes were used as a training set and 102 genes were used as a test set. Below are the test statistics for FUNSCAN when it was tested on the 102 genes. In this table, the test statistic is the exact prediction of exon/intron boundaries; both boundaries of a structure must be correctly identified.

	Start/Stop Exon	**Start Exon**	**Stop Exon**	**Middle Exon**	**Intron**
# correct	34	36	42	37	80*
# incorrect	16	16	10	13	22
Sensitivity	68%	69%	81%	74%	78.4%

* A total of 115 intron structures were predicted by FUNSCAN

Results obtained from FUNSCAN are competitive with results obtained from the Pombe gene-finder. For FUNSCAN, the sensitivity and specificity of intron

identification were 78.4% and 69.5% while for Pombe, these numbers were 67.8% and 77% respectively (Chen and Zhang 1998). Although FUNSCAN identifies exon/intron structures more accurately, FUNSCAN is also making more predictions, thus leading to lower specificity.

Here is the performance of FUNSCAN when only 60 *S. pombe* genes are used for training. Not surprisingly, FUNSCAN does not perform as well when only 60 genes are used for training. However, as will be seen in the next chapter, there are some methods that can be used to improve the performance of FUNSCAN when only small datasets are available. The sensitivity was 70.5% and the specificity was 66%.

	Start/Stop Exon	**Start Exon**	**Stop Exon**	**Middle Exon**	**Intron**
# correct	29	34	42	30	72*
# incorrect	21	18	10	20	30
Sensitivity	58%	65%	81%	60%	70.5%

*109 introns were predicted

3.4.2. FUNSCAN Performance for *N. crassa*

Neurospora crassa (N. crassa) is another fungal genome that was recently sequenced (Galagan et al. 2003). The *Neurospora* genome is over three times larger than the *S. pombe* genome. Here are the performance statistics for FUNSCAN on *Neurospora crassa (N. crassa).* For this experiment, only 65 genes were available, so cross-validations were used for testing. The number of introns predicted was 151 . For this experiment, the sensitivity was 57.9% and the specificity was 77/151 = 51%.

	Start/Stop Exon	Start Exon	Stop Exon	Middle Exon	Intron
# correct	1	24	35	37	77
# incorrect	4	22	11	50	56
Accuracy	20%	52%	76%	42.5%	57.9%

The performance of FUNSCAN in this experiment is inferior to the performance when only 60 genes were used in a training set for *S. pombe*. This might be explained by the fact that there are more introns in *N. Crassa* genes than in *S. pombe* genes. For the N. crassa experiment, the start exons, stop exon and middle exon provide more interesting insight into the performance of FUNSCAN. Notice that FUNSCAN was much better at identifying start and stop exons than at identifying middle exons. This is not very surprising since start and stop exons have stong signals (the start and stop codons)

3.4.3 Phylogenetic generality of FUNSCAN

When trained and tested on *S. pombe*, *N. crassa* and *A. nidulans,* FUNSCAN can identify at least 70% of introns correctly. An interesting question to pose is how does FUNSCAN perform when it is trained on *S. pombe*, for example, and then tested on either *N. crassa* or *A. nidulans*? Some literature on GENSCAN suggests that it is possible for a gene-finder to possess a high degree of phylogenetic generality. Although GENSCAN was trained on vertebrate sequences, GENSCAN performed well when tested on a set of 202 complete *Drosophila* (fruit fly) genes and on a set of 41 complete maize genes (Burge 1997). However, there are also indications that GENSCAN performs poorly when tested with *C. elegans* (round worm) genes.

To examine more closely the phylogenetic generality of FUNSCAN, the following experiment was conducted:

1. Three different version of FUNSCAN were trained on training sets for *S. pombe*, *N. crassa* and *A. nidulans*.

2. Each version of FUNSCAN from step 1 was tested against the two organisms for which it was not trained for. The intron accuracy statistic was recorded in each case.

3. Each version of FUNSCAN was then tested on how well it identified genes of the organism on which it was trained:

- Since there is a dataset of 234 *S. pombe* genes, the last 102 genes were set aside as a test set
- The last 50 genes of the N. crassa dataset were set aside as a test set
- Since the dataset of A. nidulans is very small, cross-validation tests were used to determine how well an *A. nidulans* trained FUNSCAN gene-finder could identify genes in *A. nidulans*.

4. 50 genes were selected from each of the following: *S. pombe* training set, *N. crassa* training set and *A. nidulans* training set. These 150 genes were then used to train FUNSCAN. This new version of FUNSCAN was then tested with the same test sets from (3) for *S. pombe* and *N. crassa*. Given the small dataset for *A. nidulans*, it was not possible to set aside a test set of 50 A. nidulans genes. As a result, the 50 *A. nidulans* genes that were used in the 150 gene training set were also used as a test set for the FUNSCAN gene-finder that was trained on genes from three different organisms. Although using training genes as a test set is not ideal, the 100 genes from *S. pombe* and *N. crassa* should help mitigate this effect and the results of this experiment should still be useful. Here are two tables that summarize the results from this experiment.

	TESTED ON		
TRAINED ON	**S. pombe**	**N. crassa**	**A. nidulans**
S. pombe	78%	31%	41%
N. crassa	4.5%	76%	31%
A. nidulans	13%	58%	69%
50 genes each from all three organisms	41%	61%	71%

Explanation of table: FUNSCAN was trained on the organism specified at the beginning of each row and tested on the organism specified at the top of each column. The test statistic is intron accuracy.

When examining the first table, the first thing that becomes immediately apparent is that it is very difficult to identify *S. pombe* genes unless the gene-finder has been trained on *S. pombe*. However, a gene-finder trained on *S. pombe* can identify genes from other fungal genomes reasonably well. This might indicate that certain characteristics are unique to the *S. pombe* genome, or are at least not present to the same degree in the genomes of *N. crassa* or *A. nidulans*. While not 'strong' enough to prevent an *S. pombe*-trained gene-finder from identifying genes from other fungal genomes, these characteristics seem to prevent *S. pombe* genes from being identified by gene-finders trained on different organisms.

Given this observation, one would expect that if a gene-finder were trained on genes from all three organisms, it would be most difficult for this gene-finder to identify *S. pombe* genes. And this is exactly what happened when step 4 of the experiment described above was performed. The second table shows that the performance of such a gene-finder on *S. pombe* is noticeably poorer than for genes from *N. crassa* and *A. nidulans*.

Another interesting observation that can be made from the above tables is that the gene-finder that was trained on genes from three different genomes was more successful at identifying *A. nidulans* genes than the gene-finder that was only trained on *A. nidulans* genes. The most likely explanation is that the multi-organism gene finder benefited from having a significantly larger training set than was available for the A. nidulans-specific gene finder, emphasizing the importance of having a sufficiently large training set for reliable parameter estimation.

Chapter 4

DEALING WITH SMALL DATASETS AND LEVERAGING COMPARATIVE GENOMICS

One might think that training gene-finders becomes routine once a gene-finder has been built and the training and testing procedures have become automated. However, the performance of gene-finders can be improved when they are trained in more creative ways, especially when training data are united. Section 4.1 some modifications that can be made to the process of training gene-finders when only a small number of genes are available. Section 4.2 examines how genes that would not normally be detected by sequence alignment techniques can be identified using a gene-finder

4.1 Dealing with small datasets

As was the case with *A. nidulans* in the previous chapter, there may not be enough high-quality genes available to train a gene-finder using the traditional method of analyzing a set of known genes and then computing certain statistics.

If one has a small dataset and cannot obtain additional known genes, then an acceptable substitute may be computationally predicted genes. These computationally predicted genes can be obtained from a gene-finder that was trained on the original set of

known genes. A procedure called Training Set Augmentation (TSA) is outlined below which incorporates computationally predicted genes into the training of gene-finders:

1. Take initial dataset and train a gene-finder in the normal manner
2. Take a chromosome or other large sequence containing uncharacterized genes of the organism for which the gene-finder is being trained. Submit this chromosome to the gene-finder in step (1).
3. Train a new gene-finder on the genes that were predicted in step 2.

When examining this procedure, one might think that since the initial training set is small, the computationally predicted genes may be inaccurate and that therefore performance will decline when those genes are used for training. However, it could be that a gene finder trained on a very small set may not be 'flexible' enough to recognize as many genes as a gene-finder trained with a larger dataset, even one which contains some erroneous annotations. The TSA procedure was tested in order to distinguish between these possibilities.

When attempting to evaluate the performance of TSA, it makes sense to perform this evaluation on an organism where a large training set is available and then to compare the performances before and after TSA. If TSA produces a gene-finder that is inferior to the original gene-finder, then it can be safely concluded that TSA does not work. In the following table, the sensitivity and specificity of a normally trained FUNSCAN gene-finder for *S. pombe* is compared with a gene-finder that was trained using TSA. A set of 132 genes were present in the initial training set; the first row contains the same data as the table in Section 3.4.1.

	Start/Stop Exon	**Start Exon**	**Stop Exon**	**Middle Exon**	**Intron**	**Intron Sensitivity**	**Intron Specificity**
FUNSCAN trained normally	34/50 = 68%	36/52 = 69%	42/52 = 81%	37/50 = 74%	80/102 115 predictions	80/102 = 78%	80/115 = 69.5%
FUNSCAN trained with TSA	33/50 = 66%	41/52 = 79%	42/52 = 81%	41/50 = 82%	86/102 123 predictions	86/102 = 84.4%	86/123= 69.9%

The initial indications about the performance of TSA look promising. While it might seem that TSA has the potential to improve the performance of gene-finders, one might also wish to pose the following questions:

1. Can this phenomenon be demonstrated in other organisms?
2. Would the 'delta' (improvement in intron sensitivity) be greater with a smaller training set?

To investigate the first question, an experiment was designed where 50 *A. nidulans* genes were used for the training-set and 34 genes were used for the test set. Here are the result that were obtained.

	Sensitivity	**Specificty**
FUNSCAN trained normally	51% (40/78)	70% (40/57)
FUNSCAN trained with TSA	62% (48/78)	65% (48/74)

This experiment does not provide overwhelming evidence that this phenomenon can be demonstrated in other organisms since although the sensitivity increases, the specificity decreases. Still, this TSA-trained gene finder is at least as good as the standard one. To answer the second question, the first experiment was repeated, but only 50 genes were used in the training set. Here are the results of that experiment

Initial Dataset size (genes)	**Normal sensitivity**	**Normal specificity**	**Sensitivity after TSA**	**Specificity after TSA**
Size = 120	78%	70%	84%	70%
Size = 50	64%	69%	79%	69%

The delta for the case when N = 120 is 6% (84.4% - 78.4%) and the delta for the case when N = 50 is 15.4% (79.4% - 64%). This experiment suggests that the delta might be bigger when the test set is smaller, although there must be a limit to how small the initial dataset size can be before the initial gene finder becomes hopelessly inaccurate. This

experiment would have to be performed on several different organisms with several different training set sizes to fully characterize this phenomenon.

In the bioinformatics literature, there are very few discussions of iterative learning methods. One paper discusses how an *ab initio* interative Markov modeling procedure was used to automatically perform the partition of microbial genomic sequences into three subsets: coding, coding on opposite strand, and intergenic segments (Audic and Claverie 1998).

4.2 Leveraging Comparative Genomics

Comparative genomics tools can be used to identify genes that are similar across different organisms. One example of a comparative genomics tool is BLAST. The BLAST package provides programs that take as input (1) a query sequence (sequence can be either a nucleotide sequence or amino acid sequence) and (2) a database (this can be a database of nucleotide sequences or proteins). The output of this program is a list of candidate nucleotide sequences or proteins that match the query sequence. These candidate sequences are scored according to how closely they match the query sequence.

So, how does one use such alignment tools to search for genes in an organism that has not been annotated? The simplest strategy to follow would be the following:

1. Blast the entire genome of this organism against an organism that has been annotated and is phylogenetically close to the organism to that we are trying to annotate.
2. The output of step 1 will be pairs of sequences of length 50 to 500 bp that are very similar to each other. In each pair, one sequence will come from the annotated sequence and the other sequence will come from the organism that we are trying to annotate.

3. If the sequence from the annotated organism is a coding region, then there is a reasonable probability that the sequence from the other organism is also a coding region.

A more interesting question is the following: How does one search for novel genes (genes that have not been annotated) in an annotated genome using only gene-predictions from an un-annotated genome? For example, *S. pombe* has been annotated but *A. nidulans* has not yet been annotated. However, a draft of the unannotated sequence of *A. nidulans* has recently been released. It is not reasonable to expect that the gene annotation that genome centers perform is 100% accurate. It is possible that genome centers will miss some genes and predict other genes that are not correct. It would be interesting to use FUNSCAN predictions for *A. nidulans* to attempt to identify novel genes in *S. pombe*. The following strategy was used in order to perform this task:

For both *A. nidulans* and *S. pombe*, do the following:

1. Submit the entire genomic sequence of the organism to FUNSCAN
2. The output of FUNSCAN will be nucleotide sequences of predicted genes. Convert these nucleotide sequences into their translated amino acid sequences and BLAST these amino sequences against a database of all known proteins
3. Discard proteins that score well. Since these proteins scored well, they can be easily identified with BLAST methods similar to the one described above. After step 3, we will have a set of amino-acid sequences of predicted genes for *S. pombe* and *A. nidulans*.
4. When these three steps have been performed for both organisms, BLAST all *S. pombe* amino acid sequences against all *A. nidulans* sequences. If an amino-acid sequence from *S. pombe* aligns well with an *A. nidulans* amino acid sequence, then it is possible that a novel gene has been discovered.

For the purposes of this experiment, a poor score for step 3 was considered to be when there was less than 50% alignment between the protein sequences. Afterwards, when the *S. Pombe* and *A. nidulans* amino acid sequences were aligned, a good score was considered to be when 75% alignment.

Unfortunately, this method did not identify any candidates for novel genes in *S. pombe*. It is possible that if two amino acid sequences align well after step 3, then their corresponding nucleotide sequences could be identified using the first method described in this section. Although novel genes were not identified using this method, many genes predicted from both organisms obtained high scores when aligned with sequences from protein databases. This suggests that in addition to test sets, aligning predicted gene sequences against known protein sequences is a strategy that can be used to evaluate the quality of gene-finders. Such an alignment test method can be used to confirm many of the predicted genes. If a gene-finder does not perform well when tested using the alignment method, then it wil surely not perform well when using the more traditional test statistics of nucleotide accuracy and gene-structure accuracy. An alignment test method can be used to gain insight into the types of genes that a gene-finder is able to accurately predict. Is alignment successful when the genes are long or short? Is alignment successful when the gene is located in a GC-rich region of the genome? Although such statistics can also be obtained using test-sets, it might be worthwhile to also compute such statistics using alignment methods since the number of predicted genes is much larger than the number of genes available in any test set.

Chapter 5

CONCLUSION

In Section 1.5, the two main objectives of this thesis were discussed: (1) designing a gene-finder whose performance on fungal genomes is competitive with other existing gene-finders and (2) the exploration of new methods that can be used to train gene-finder when only small datasets are available and to leverage comparative genomics. Have these goals been achieved? Below is a brief discussion:

1. *Design of a gene-finder competitive with existing gene-finder for fungal genomes*: This goal has been accomplished. As can be seen from the discussion of the results for *S. pombe* in Section 3.4.1, the performance of FUNSCAN is competitive with the performance of Pombe. Furthermore, as was seen in section 4.1, the performance of FUNSCAN is improved when Training Set Augmentation was performed on the 120 gene training set for *S. pombe*. The training procedure for FUNSCAN is highly automated and only a cursory inspection of the state transition probabilities is required to ensure that they are optimal (Section 3.3.3). Also, FUNSCAN processes 100 kilobase sequences in well under a minute

2. *Developing new ways of dealing with small datasets and leveraging comparative genomics*: Only the first part of this goal was accomplished. As was seen in the discussion in Section 4.1, Training Set Augmentation does indeed improve the

performance of FUNSCAN. The effect of Training Set Augmentation is particularly noticeable when only small datasets are available for training. As was seen for *S. pombe* when only 50 genes are available for training, sensitivity increased from 64% to 80% and specificity was unchanged when Training Set Augmentation was used in the training process. Training Set Augmentation was also shown to work for *A. nidulans*. Unfortunately, a method designed to leverage comparative genomics was not successful at identifying novel genes. Still, it was surprising to see how many predicted genes align well with genes in a database of known proteins.

Although the topics discussed in this thesis dealt specifically with improving the performance of gene-finders, some aspects of this work may be of interest to bioinformaticians working on other tasks. Although Training Set Augmentation was used only to improve the accuracy of identifying introns, it might also be possible to use it to identify other gene structures such as promoter regions of genes. Since Markov processes have also been used to identify promoters, the challenges involved in identifying introns and promoters are somewhat similar. However, identifying promoters is a little more difficult because all genes do not possess promoters (Ohler 2001). It might also be possible to use iterative learning methods such as TSA to solve problems bioinformatics besides those dealing with genomic DNA sequence.

Although the fungal genome sequencing projects have not made nearly as many headlines as the sequencing of the human genome, scientists working on sequencing fungal genomes have little to envy from their counterparts working with human genomic DNA sequences. As discussed in Section 1.2, fungi have numerous industrial and agricultural uses. In addition, since the size of fungal genomes is usually a manageable size (10 million to 40 million bases), fungal genomes can be sequenced much more rapidly. This gives researchers the opportunity to develop new methods for identifying gene-structures. For example, a recent paper from the Whitehead Institute (Kellis et al 2003) describes how gene structures were identified aligning the genomic DNA of four different fungi. An interesting project would be to compare the strengths and weaknesses of these alignment methods with traditional gene-finders and to explore how they can be

used together to identify new genes in fungal genomes. Using alignment methods is not always practical when dealing with genomic sequences that are billions of base pairs long, but they can be readily applied to fungal genomes. As a result, fungal gene-finders are not the only solution available to scientists attempting to identify gene structures. Instead, the *ab initio* gene-finder is one weapon in an arsenal that researchers can draw upon. If used skillfully, the total value of this gene-finding arsenal might prove to be greater than the sum of its individual parts. Fungi, given the manageable size of their genomic DNA sequences, are ideal organisms for researchers to use in order to explore how traditional techniques such as HMM-based gene-finders and alignment methods can be used as a platform for more innovative techniques.

Appendix A

Below is a sample GENBANK file. This is the format of the training and testing sets for FUNSCAN. The main field of interest for FUNSCAN is the CDS field since this field indicates the coding regions. For example, one of the CDS fields is 687..3158, meaning that a start codon starts at 687 and a stop codon finishes at 3158. Another CDS field is complement(3300..4037). This means that there is a gene located on the complement strand with the start codon that starts at base 4037 and the stop codon that finishes at base 3300. As a result, bases 4037, 4036 and 4035 are t, a and c respectively. This corresponds to a start codon a, t and g on the complement strand. Also, bases 3302, 3301 and 3300 are a, t and t. This means that the stop codon on the complement strand is taa.

```
LOCUS       SCU49845     5028 bp    DNA             PLN       21-JUN-1999
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, partial cds, and Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      baker's yeast.
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Hemiascomycetes; Saccharomycetales;
            Saccharomycetaceae; Saccharomyces.
REFERENCE   1  (bases 1 to 5028)
  AUTHORS   Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
  TITLE     Cloning and sequence of REV7, a gene whose function is required for
            DNA damage-induced mutagenesis in Saccharomyces cerevisiae
  JOURNAL   Yeast 10 (11), 1503-1509 (1994)
  MEDLINE   95176709
REFERENCE   2  (bases 1 to 5028)
  AUTHORS   Roemer,T., Madden,K., Chang,J. and Snyder,M.
  TITLE     Selection of axial growth sites in yeast requires Axl2p, a novel
            plasma membrane glycoprotein
  JOURNAL   Genes Dev. 10 (7), 777-793 (1996)
  MEDLINE   96194260
REFERENCE   3  (bases 1 to 5028)
  AUTHORS   Roemer,T.
  TITLE     Direct Submission
  JOURNAL   Submitted (22-FEB-1996) Terry Roemer, Biology, Yale University, New
            Haven, CT, USA
FEATURES             Location/Qualifiers
     source          1..5028
                     /organism="Saccharomyces cerevisiae"
                     /db_xref="taxon:4932"
                     /chromosome="IX"
                     /map="9"
     CDS             <1..206
                     /codon_start=3
                     /product="TCP1-beta"
                     /protein_id="AAA98665.1"
                     /db_xref="GI:1293614"
                     /translation="SSIYNGISTSGLDLNNGTIADMRQLGIVESYKLKRAVVSSASEA
                     AEVLLRVDNIIRARPRTANRQHM"
     gene            687..3158
```

```
                     /gene="AXL2"
     CDS             687..3158
                     /gene="AXL2"
                     /note="plasma membrane glycoprotein"
                     /codon_start=1
                     /function="required for axial budding pattern of S.
                     cerevisiae"
                     /product="Ax12p"
                     /protein_id="AAA98666.1"
                     /db_xref="GI:1293615"
                     /translation="MTQLQISLLLTATISLLHLVVATPYEAYPIGKQYPPVARVNESF
                     TFQISNDTYKSSVDKTAQITYNCFDLPSWLSFDSSSRTFSGEPSSDLLSDANTTLYFN
                     VILEGTDSADSTSLNNTYQFVVTNRPSISLSSDFNLLALLKNYGYTNGKNALKLDPNE
                     VFNVTFDRSMFTNEESIVSYYGRSQLYNAPLPNWLFFDSGELKFTGTAPVINSAIAPE
                     TSYSFVIIATDIEGFSAVEVEFELVIGAHQLTTSIQNSLIINVTDTGNVSYDLPLNYV
                     YLDDDPISSDKLGSINLLDAPDWVALDNATISGSVPDELLGKNSNPANFSVSIYDTYG
                     DVIYFNFEVVSTTDLFAISSLPNINATRGEWFSYYFLPSQFTDYVNTNVSLEFTNSSQ
                     DHDWVKFQSSNLTLAGEVPKNFDKLSLGLKANQGSQSQELYFNIIGMDSKITHSNHSA
                     NATSTRSSHHSTSTSSYTSSTYTAKISSTSAAATSSAPAALPAANKTSSHNKKAVAIA
                     CGVAIPLGVILVALICFLIFWRRRRENPDDENLPHAISGPDLNNPANKPNQENATPLN
                     NPFDDDASSYDDTSIARRLAALNTLKLDNHSATESDISSVDEKRDSLSGMNTYNDQFQ
                     SQSKEELLAKPPVQPPESPFFDPQNRSSSVYMDSEPAVNKSWRYTGNLSPVSDIVRDS
                     YGSQKTVDTEKLFDLEAPEKEKRTSRDVTMSSLDPWNSNISPSPVRKSVTPSPYNVTK
                     HRNRHLQNIQDSQSGKNGITPTTMSTSSSDDFVPVKDGENFCWVHSMEPDRRPSKKRL
                     VDFSNKSNVNVGQVKDIHGRIPEML"
     gene            complement(3300..4037)
                     /gene="REV7"
     CDS             complement(3300..4037)
                     /gene="REV7"
                     /codon_start=1
                     /product="Rev7p"
                     /protein_id="AAA98667.1"
                     /db_xref="GI:1293616"
                     /translation="MNRWVEKWLRVYLKCYINLILFYRNVYPPQSFDYTTYQSFNLPQ
                     FVPINRHPALIDYIEELILDVLSKLTHVYRFSICIINKKNDLCIEKYVLDFSELQHVD
                     KDDQIITETEVFDEFRSSLNSLIMHLEKLPKVNDDTITFEAVINAIELELGHKLDRNR
                     RVDSLEEKAEIERDSNWVKCQEDENLPDNNGFQPPKIKLTSLVGSDVGPLIIHQFSEK
                     LISGDDKILNGVYSQYEEGESIFGSLF"
BASE COUNT      1510 a   1074 c    835 g   1609 t
ORIGIN
        1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac ggaaccattg
       61 ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct
      121 ctgcatctga agccgctgaa gttctactaa gggtggataa catcatccgt gcaagaccaa
      181 gaaccgccaa tagacaacat atgtaacata tttaggatat acctcgaaaa taataaaccg
      241 ccacactgtc attattataa ttagaaacag aacgcaaaaa ttatccacta tataattcaa
      301 agacgcgaaa aaaaaagaac aacgcgtcat agaacttttg gcaattcgcg tcacaaataa
      361 attttggcaa cttatgtttc ctcttcgagc agtactcgag ccctgtctca agaatgtaat
      421 aatacccatc gtaggtatgg ttaaagatag catctccaca acctcaaagc tccttgccga
      481 gagtcgccct cctttgtcga gtaattttca cttttcatat gagaacttat tttcttattc
      541 tttactctca catcctgtag tgattgacac tgcaacagcc accatcacta gaagaacaga
      601 acaattactt aatagaaaaa ttatatcttc ctcgaaacga tttcctgctt ccaacatcta
      661 cgtatatcaa gaagcattca cttaccatga cacagcttca gatttcatta ttgctgacag
      721 ctactatatc actactccat ctagtagtgg ccacgcccta tgaggcatat cctatcggaa
      781 aacaataccc cccagtggca agagtcaatg aatcgtttac atttcaaatt tccaatgata
      841 cctataaatc gtctgtagac aagacagctc aaataacata caattgcttc gacttaccga
      901 gctggctttc gtttgactct agttctagaa cgttctcagg tgaaccttct tctgacttac
      961 tatctgatgc gaacaccacg ttgtatttca atgtaatact cgagggtacg gactctgccg
     1021 acagcacgtc tttgaacaat acataccaat ttgttgttac aaaccgtcca tccatctcgc
     1081 tatcgtcaga tttcaatcta ttggcgttgt taaaaaacta tggttatact aacggcaaaa
     1141 acgctctgaa actagatcct aatgaagtct tcaacgtgac ttttgaccgt tcaatgttca
     1201 ctaacgaaga atccattgtg tcgtattacg gacgttctca gttgtataat gcgccgttac
     1261 ccaattggct gttcttcgat tctggcgagt tgaagtttac tgggacggca ccggtgataa
     1321 actcggcgat tgctccagaa acaagctaca gttttgtcat catcgctaca gacattgaag
     1381 gattttctgc cgttgaggta gaattcgaat tagtcatcgg ggctcaccag ttaactacct
     1441 ctattcaaaa tagtttgata atcaacgtta ctgacacagg taacgtttca tatgacttac
     1501 ctctaaacta tgtttatctc gatgacgatc ctatttcttc tgataaattg ggttctataa
     1561 acttattgga tgctccagac tgggtggcat tagataatgc taccatttcc gggtctgtcc
     1621 cagatgaatt actcggtaag aactccaatc ctgccaattt ttctgtgtcc atttatgata
     1681 cttatggtga tgtgatttat ttcaacttcg aagttgtctc cacaacggat ttgtttgcca
     1741 ttagttctct tcccaatatt aacgctacaa ggggtgaatg gttctcctac tatttttgc
     1801 cttctcagtt tacagactac gtgaatacaa acgtttcatt agagtttact aattcaagcc
```

```
1861 aagaccatga ctgggtgaaa ttccaatcat ctaatttaac attagctgga gaagtgccca
1921 agaatttcga caagctttca ttaggtttga aagcgaacca aggttcacaa tctcaagagc
1981 tatattttaa catcattggc atggattcaa agataactca ctcaaaccac agtgcgaatg
2041 caacgtccac aagaagttct caccactcca cctcaacaag ttcttacaca tcttctactt
2101 acactgcaaa aatttcttct acctccgctg ctgctacttc ttctgctcca gcagcgctgc
2161 cagcagccaa taaaacttca tctcacaata aaaaagcagt agcaattgcg tgcggtgttg
2221 ctatcccatt aggcgttatc ctagtagctc tcatttgctt cctaatattc tggagacgca
2281 gaagggaaaa tccagacgat gaaaacttac cgcatgctat tagtggacct gatttgaata
2341 atcctgcaaa taaaccaaat caagaaaacg ctacaccttt gaacaacccc tttgatgatg
2401 atgcttcctc gtacgatgat acttcaatag caagaagatt ggctgctttg aacactttga
2461 aattggataa ccactctgcc actgaatctg atatttccag cgtggatgaa aagagagatt
2521 ctctatcagg tatgaataca tacaatgatc agttccaatc ccaaagtaaa gaagaattat
2581 tagcaaaacc cccagtacag cctccagaga gcccgttctt tgacccacag aataggtctt
2641 cttctgtgta tatggatagt gaaccagcag taaataaatc ctggcgatat actggcaacc
2701 tgtcaccagt ctctgatatt gtcagagaca gttacggatc acaaaaaact gttgatacag
2761 aaaaactttt cgatttagaa gcaccagaga aggaaaaacg tacgtcaagg gatgtcacta
2821 tgtcttcact ggacccttgg aacagcaata ttagcccttc tcccgtaaga aaatcagtaa
2881 caccatcacc atataacgta acgaagcatc gtaaccgcca cttacaaaat attcaagact
2941 ctcaaagcgg taaaaacgga atcactccca caacaatgtc aacttcatct tctgacgatt
3001 ttgttccggt taaagatggt gaaaattttt gctgggtcca tagcatggaa ccagacagaa
3061 gaccaagtaa gaaaaggtta gtagattttt caaataagag taatgtcaat gttggtcaag
3121 ttaaggacat tcacggacgc atcccagaaa tgctgtgatt atacgcaacg atattttgct
3181 taattttatt ttcctgtttt atttttttatt agtggtttac agatacccta tattttattt
3241 agtttttata cttagagaca tttaatttta attccattct tcaaatttca tttttgcact
3301 taaaacaaag atccaaaaat gctctcgccc tcttcatatt gagaatacac tccattcaaa
3361 attttgtcgt caccgctgat taatttttca ctaaactgat gaataatcaa aggccccacg
3421 tcagaaccga ctaaagaagt gagttttatt ttaggaggtt gaaaaccatt attgtctggt
3481 aaattttcat cttcttgaca tttaacccag tttgaatccc tttcaatttc tgctttttcc
3541 tccaaactat cgaccctcct gtttctgtcc aacttatgtc ctagttccaa ttcgatcgca
3601 ttaataactg cttcaaatgt tattgtgtca tcgttgactt taggtaattt ctccaaatgc
3661 ataatcaaac tatttaagga agatcggaat tcgtcgaaca cttcagtttc cgtaatgatc
3721 tgatcgtctt tatccacatg ttgtaattca ctaaaatcta aaacgtattt ttcaatgcat
3781 aaatcgttct ttttattaat aatgcagatg gaaaatctgt aaacgtgcgt taatttagaa
3841 agaacatcca gtataagttc ttctatatag tcaattaaag caggatgcct attaatggga
3901 acgaactgcg gcaagttgaa tgactggtaa gtagtgtagt cgaatgactg aggtgggtat
3961 acatttctat aaaataaaat caaattaatg tagcatttta agtataccct cagccacttc
4021 tctacccatc tattcataaa gctgacgcaa cgattactat ttttttttttc ttcttggatc
4081 tcagtcgtcg caaaaacgta taccttcttt ttccgacctt ttttttagct ttctggaaaa
4141 gtttatatta gttaaacagg gtctagtctt agtgtgaaag ctagtggttt cgattgactg
4201 atattaagaa agtggaaatt aaattagtag tgtagacgta tatgcatatg tatttctcgc
4261 ctgtttatgt ttctacgtac ttttgattta tagcaagggg aaaagaaata catactattt
4321 tttggtaaag gtgaaagcat aatgtaaaag ctagaataaa atggacgaaa taaagagagg
4381 cttagttcat ctttttttcca aaaagcaccc aatgataata actaaaatga aaaggatttg
4441 ccatctgtca gcaacatcag ttgtgtgagc aataataaaa tcatcacctc cgttgccttt
4501 agcgcgtttg tcgtttgtat cttccgtaat tttagtctta tcaatgggaa tcataaattt
4561 tccaatgaat tagcaatttc gtccaattct ttttgagctt cttcatattt gctttggaat
4621 tcttcgcact tcttttccca ttcatctctt tcttcttcca aagcaacgat ccttctaccc
4681 atttgctcag agttcaaatc ggcctctttc agtttatcca ttgcttcctt cagtttggct
4741 tcactgtctt ctagctgttg ttctagatcc tggtttttct tggtgtagtt ctcattatta
4801 gatctcaagt tattggagtc ttcagccaat tgctttgtat cagacaattg actctctaac
4861 ttctccactt cactgtcgag ttgctcgttt ttagcggaca aagatttaat ctcgttttct
4921 ttttcagtgt tagattgctc taattctttg agctgttctc tcagctcctc atatttttct
4981 tgccatgact cagattctaa ttttaagcta ttcaatttct ctttgatc
//
```

Appendix B

Below is an example of a file that is provided as input to FUNSCAN. This file contains the observation symbol probability distributions for the following states (listed in order): Intergenic state, codon state 3, codon state 1 and codon state 2 and intronic state. The first column of each 'matrix' is a combination of two letters such as 'aa'. With the Intergenic Probabilities, for the row that begins with aa, each number indicates the probability of observing each base *given* that the previous two bases were aa.

Recall that codon state n is the nth base in a codon. For the second matrix, the title is "we are given one and two, predict three". This means that we know what the first two bases of the codon are. Notice that for this matrix, the probability of obtaining a 't' or an 'a' when the first two bases of a codon are ta is 0.000 and that the probability of obtaining an 'a' when the first two bases of a codon are tg is also 0.000. This is not surprising, since there can be no stop codons in coding regions. For the third matrix, the title is "we are given two and three, predict one". This means that we know what the second and third bases of the previous codon were, and that the probabilities listed below are for the first base of a codon, given that the previous codon finished with the bases indicated in the first column.

/**********BEGINNING OF SAMPLE INPUT FILE

Intergenic Probabilities

	a	c	g	t
aa	0.274	0.222	0.243	0.261
ac	0.269	0.257	0.194	0.280
ag	0.267	0.265	0.238	0.230
at	0.232	0.254	0.239	0.275
ca	0.264	0.199	0.264	0.273
cc	0.274	0.242	0.201	0.283
cg	0.259	0.251	0.240	0.249
ct	0.178	0.282	0.253	0.287
ga	0.272	0.220	0.240	0.268
gc	0.258	0.237	0.205	0.301
gg	0.257	0.271	0.221	0.251
gt	0.226	0.264	0.218	0.292

ta	0.225	0.242	0.210	0.322
tc	0.244	0.253	0.199	0.304
tg	0.271	0.251	0.241	0.237
tt	0.161	0.287	0.247	0.305

We are given one and two, predict three

	a	c	g	t
aa	0.167	0.293	0.385	0.154
ac	0.219	0.356	0.194	0.231
ag	0.166	0.440	0.158	0.236
at	0.081	0.381	0.286	0.252
ca	0.229	0.215	0.395	0.161
cc	0.187	0.282	0.241	0.290
cg	0.203	0.350	0.213	0.235
ct	0.107	0.350	0.258	0.286
ga	0.215	0.268	0.310	0.206
gc	0.176	0.330	0.192	0.302
gg	0.189	0.362	0.138	0.311
gt	0.081	0.408	0.212	0.299
ta	0.000	0.636	0.000	0.364
tc	0.184	0.299	0.246	0.271
tg	0.000	0.326	0.478	0.197
tt	0.083	0.441	0.245	0.231

We are given two and three, predict one

	a	c	g	t
aa	0.249	0.207	0.348	0.196
ac	0.266	0.248	0.298	0.188
ag	0.296	0.243	0.307	0.154
at	0.219	0.224	0.421	0.136
ca	0.301	0.191	0.299	0.210
cc	0.324	0.183	0.313	0.179
cg	0.256	0.262	0.293	0.188
ct	0.179	0.267	0.387	0.166
ga	0.292	0.207	0.310	0.191
gc	0.284	0.212	0.303	0.201
gg	0.300	0.252	0.279	0.169
gt	0.195	0.236	0.405	0.163
ta	0.240	0.257	0.292	0.212
tc	0.289	0.223	0.291	0.197
tg	0.281	0.247	0.319	0.153
tt	0.166	0.238	0.457	0.140

We are given three and one, predict two

	a	c	g	t
aa	0.348	0.239	0.171	0.243
ac	0.297	0.243	0.191	0.268
ag	0.381	0.245	0.197	0.177
at	0.202	0.309	0.157	0.333
ca	0.385	0.222	0.131	0.263
cc	0.267	0.236	0.203	0.295
cg	0.379	0.218	0.224	0.179
ct	0.175	0.318	0.191	0.316
ga	0.375	0.229	0.145	0.250
gc	0.273	0.232	0.198	0.296
gg	0.388	0.251	0.167	0.194
gt	0.199	0.313	0.124	0.364
ta	0.246	0.270	0.127	0.357
tc	0.228	0.251	0.182	0.339
tg	0.322	0.249	0.216	0.214
tt	0.165	0.334	0.142	0.359

Intronic Probabilities

	a	c	g	t
aa	0.208	0.254	0.237	0.301
ac	0.270	0.215	0.160	0.356
ag	0.173	0.221	0.405	0.201
at	0.253	0.235	0.242	0.270
ca	0.233	0.205	0.288	0.274
cc	0.268	0.270	0.165	0.297
cg	0.257	0.280	0.183	0.280
ct	0.261	0.216	0.242	0.280
ga	0.227	0.262	0.219	0.292
gc	0.249	0.217	0.142	0.391
gg	0.170	0.194	0.142	0.494
gt	0.318	0.193	0.236	0.253
ta	0.294	0.210	0.198	0.299
tc	0.249	0.237	0.178	0.337
tg	0.301	0.291	0.180	0.228
tt	0.158	0.278	0.220	0.345

**********END OF SAMPLE INPUT FILE **********/

References

Audic, S. and Claverie, J. Self-identification of protein-coding regions in microbial genomes. In *Proc. Natl. Acad. Sci.*, pages 10026-10031, August 1998.

Batzolglou, S., Pachter, L., Mesirov, J., Berger, B. and Lander, E. Human and Mouse Gene Structure: Comparative Analysis and Application to Exon Prediction. In *Genome Research*, pages 950-958, May 2000.

Birren, B., Fink, G. and Lander, E. Fungal Genome Initiative. White paper developed at the Whitehead Institute, February 2002.

Burge, C. Identification of genes in human genomic DNA. PhD Thesis submitted at Stanford University, September 1997.

Burge, C. Chipping away at the transcriptome. In *Nature Genetics*, pages 232-234, March 2001.

Burset, M. and Guigo, R. Evaluation of gene structure prediction programs. In *Genomics*, pages 353-367, June 1996.

Chen, T. and Zhang, M. Pombe: a gene-finding and exon-intron structure prediction system for fission yeast. In *Yeast*, pages 701-710, June 1998.

Galagan, J. et al. The genome sequence of the filamentous fungus Neurospora crassa. In *Nature*, pages 859-868, April 2003.

Kellis, M., Patterson, N., Endrizzi, M., Birren, B. and Lander, E. Sequencing and comparison of yeast species to identify genes and regulatory elements. In *Nature*, pages 241-254, May 2003.

Krough, A. An Introduction to Hidden Markov Model for Biological Sequences. In *Computational Methods in Molecular Biology*, pages 45-63, 1998.

Krogh, A. Two methods for improving performance of an HMM and their application for gene finding. In *Proc Intl Conf Intell Syst Mol Bio*, pages 179-186, 1997.

Kulp, D., Haussler, K., Reese, M. and Eeckman, F. A generalized hidden Markov model for the recognition of human genes in DNA. In *Proc Intl Conf Intell Syst Mol Bio*, pages 134-142, 1996.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F. and Wootton, J. C. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. In *Science*, pages 208-14, June 1993.

Lim, L. and Burge, C. A computational analysis of sequence features involved in recognition of short introns. In *Proc. Natl. Acad. Sci.*, pages 11193-11198, September 2001.

Ohler, Uwe. Computational Promoter Recognition in Eukaryotic Genomic DNA. PhD thesis submitted to the University of Nurenberg, 2001. Available online at http://genes.mit.edu/documents/ohler_diss.pdf.

Rabiner, Lawrence. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, pages 257-286, February 1989.

Rogic, S., Mackworth, A. and Ouellette, F. Evaluation of Gene-Finding Programs on Mammalian Sequences. In Genome Research, pages 817-832, February 2001.

Salzberg, S., Delcher, A., Fasman, K. and Henderson, J. A decision tree system for finding genes in DNA. In *Journal of Computational Biology*, pages 667-680, Winter 1998.

Soloyev, V., Salamov, A. and Lawrence, C. Identification of human gene structure using linear discriminant functions and dynamic programming. In *Proc Int Conf Intell Syst Mol Bio*, pages 367-375, 1995.

Various. A science primer menu (National Center for Biotechnology Information homepage). URL: http://www.ncbi.nlm.nih.gov/About/primer/index.html.

Wood, V. et al. The genome sequence of *Schizosaccharomyces pombe*. In *Nature,* pages 871-880, February 2002.