UNIVERSITY of CALIFORNIA
SANTA CRUZ

PROTEIN-CODING GENE STRUCTURE PREDICTION USING GENERALIZED HIDDEN MARKOV MODELS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

David C. Kulp

March 2003

The Dissertation of David C. Kulp is approved:

Contents

List of Figures

List of Tables

Abstract

Protein-coding Gene Structure Prediction Using Generalized Hidden Markov Models

by

David C. Kulp

This paper describes a computer method for predicting the exon-intron structures and protein-coding regions of genes in genomic DNA along with its application to several model organisms. Expanding on earlier work applying linguistics and state machines to DNA analysis, the problem is introduced here using a novel generalized hidden Markov model that allows arbitrary length symbols per model state. This model provides a simple representation of complex grammatical structure and reduces some of the parameterization and training burden of standard hidden Markov models. A key characteristic of the method is the abstraction of score-generating feature sensors from the gene-structure model topology. A computer program called "Genie" embodies the method described here. Employing mostly standard metrics for feature scoring, the basic gene-prediction method is shown to work better than other known methods, identifying as much as 40% of exact coding sequences correctly. An expanded method, which uses constraints on the set of possible outputs, allows for the incorporation of messenger RNA or protein sequence homology to boost gene prediction sensitivity and specificity by approximately 10%. The results of whole genome studies on several complete genome sequences are presented. Engineering details of the software design are discussed including flexible run-time configurations and methods to reduce the running time from cubic to linear in the size of the input sequence.

In memory of my grandmother, Ethel Zimmerman.

Acknowledgements

I extend my thanks to family and friends who provided moral support and collaborators whose collegial support made this work possible. Specifically, my deepest gratitude is extended to my wife, Laura, whose patience is unequaled, and to my mother and father, Ann and Paul, who have remained supportive through the many years of detours and diversions. Among friends, I thank Cyrus Harmon for his enthusiastic support. Of note, Martin Reese was a valued collaborator for many years. Martin's neural network signal sensors were used in some incarnations of the Genie system described here; he also produced curated DNA annotations that were critical for evaluation of the Genie system and he was the catalyst for some interesting experiments and projects. Ewan Birney's uncommon goodwill has resulted in publication success that might otherwise be impossible. Jim Kent wrote "pslayout", which was key to cDNA constraints, and he worked with me to re-annotate C. elegans as part of his "intronerator" project. Ray Wheeler wrote "AltMerge", which, like "pslayout", was essential to the success of the constraint system. Hari Tammana helped me test the software performance. Alan Williams wrote two versions of batch management software for running whole genome annotations. Ray and Alan were collaborators in the human genome annotation and Ray and I worked together on the mouse genome annotation. Robin Matlib produced the curated set of Chlamydomonas sequences. Peter Brokstein provided 5'/3' clone pairs used in the Drosophila annotation.

The following people contributed to early related work not described here:

Melissa Cline provided me with sample Matlab code for feed-forward neural networks used in the content sensor in the first iteration; David Le attempted early studies using cDNA homology with Genie; and Kevin Karplus generated early alignment models of splice sites. Naomi Harris performed early gene finder evaluations, developed a meta-gene-finder that included Genie, developed a visualization system that included Genie annotations of Drosophila, and maintained a flyGenie web server.

Lastly, I am grateful to my advisor, David Haussler, for his mentorship, insight, inspiration, and friendship over many long years.

Chapter 1

Introduction

The task of separating the wheat from the chaff is a significant problem in DNA sequence analysis. Faced with billions of nucleotides, much effort has been given to locating the functional elements along the chromosomes. Most efforts are directed towards the identification of genes. A gene is classically defined as a region of a chromosome that is genetically linked with a phenotype. More specifically, a gene is typically a discrete sequence of nucleotides composed of a subsequence that is physically transcribed by a polymerase and proximal sequence influential in that transcription; that is, a gene is a set of regulatory elements and a primary transcript. Of particular interest are those primary transcripts that are translated by the ribosome into amino acid chains: the protein-coding genes. Many RNA transcripts in eukaryotes are spliced into smaller sequences through the excision of introns by the spliceosome, leaving a set of concatenated exons to be passed to the ribosome. The specific problem addressed here is the prediction of both the exon-intron structure and the translated peptide sequence of protein-coding genes. I define this limited problem as "gene prediction" and the exon-intron structure more generally as the "gene structure."

In Chapter 2, a background of molecular biology and gene prediction methodology is presented. In this section I introduce standard terminology, refer to the significant and influential work in gene prediction methods, and identify efforts related to my own. Important discriminating features of genes are explained and the basic concepts behind the likelihood models used for gene finding are presented.

Chapter 3 is concerned with a principled derivation of the generalized hidden Markov model from its linguistic roots. I show that all necessary restrictions on the structure of protein-coding genes can be described in terms of a regular grammar and that the corresponding probabilistic finite state automaton can be equivalently mapped to a new generative graph that emits multi-symbol observations along its arcs. The algorithms necessary for path scoring for this so-called generalized hidden Markov model are briefly presented and the precise model used by the Genie software is shown.

The focus of Chapter 4 is on the software engineering aspects related to the configurability, flexibility, expressibility, and efficiency of the system. Technical details of the implementation are revealed including the software architecture, optimizations to manage running time and memory constraints, the format for run-time specification, and the practical problem of making predictions in long DNA contigs.

Chapter 5 includes descriptions of three different mechanisms for constraining the search space and scoring of parses given external evidence. The first involves the use of protein similarity in a coding-exon likelihood model; the second is a software method for restricting the search space by traversing only subgraphs of the induced graph of possible parses; the third is an artificial scoring scheme to force the inclusion of certain paths in the optimal parse while allowing statistical prediction elsewhere. For organisms with large expressed-sequence databases, such as human and mouse, the problem of gene prediction is greatly simplified by incorporating the alignments from these databases as fixed constraints.

Chapter 6 provides a disclosure of various configurations used in different experiments and genome annotations. Results are then presented demonstrating the efficacy of the methods described in the previous chapters. In addition, I present evidence that the number of training samples necessary for effective gene prediction is surprisingly low and I show that the posterior probabilities derived from the hidden Markov model serve as a good score function.

Finally, the general results of several whole genome annotation efforts are presented in Chapter 7. I give particular attention to the recent generation of a set of protein-coding genes for the mouse genome and conclude with an assessment of the number of genes in mouse based on a novel method using EST mate pairs.

Chapter 2

Background

2.1 Basic molecular biology

DNA (deoxyribonucleic acid) is the symbolic code of life. In each cell the DNA is divided into chromosomes, and each chromosome is a helical strand of nucleotides—adenine (A), guanine (G), cytosine (C), and thymine (T)—the so-called bases. Combined, there are up to billions of bases per cell, collectively containing sufficient information for the creation and functions of life. The bases are arranged into two complementary strands such that A-T and G-C are always paired.

Much effort has been given to locating the functional elements along these genomic sequences and most efforts are directed towards the identification of genes. A gene is classically defined as a region of a chromosome that is genetically linked with a phenotype. In molecular biology, a gene is typically defined as a discrete sequence of nucleotides composed of a subsequence that is transcribed by a polymerase into

RNA (ribonucleic acid) and proximal sequence influential in that transcription. Thus, a gene is a set of regulatory elements and a primary transcript. Of particular interest are those primary transcripts that are translated by the ribosome into amino acid chains: the protein-coding genes. In this paper, the term gene is typically used loosely to refer to a protein-coding primary transcript.

Since one strand is the complement of the other, in DNA analysis only one strand is stored in the databases. Which strand is represented is generally arbitrary and unimportant, but in this paper the represented sequence is called the forward strand and the implicit complement is the reverse. A DNA sequence is always represented, by convention, in the direction of DNA replication. The left end of the sequence is also termed upstream or 5' and the right end is downstream or 3'. Transcription occurs on single-stranded DNA, either the forward or reverse complement strand. Although transcription has been observed to occur at the same physical genomic position on both strands, this work assumes that there are no overlapping transcripts. Automatic analysis methods must evaluate both the explicitly represented sequence and the implicit reverse complement simultaneously.

Transcription is performed by a DNA polymerase molecule that binds to a DNA molecule. Conserved nucleotide patterns in the region near transcription initiation are called promoters, and the attachment of the DNA polymerase is shepherded by a set of promoter-binding molecules. Transcription, like replication, occurs in a downstream direction along the sequence. This copying of the DNA into RNA often terminates after transcribing a nucleotide pattern called the polyadenylation signal.

Once the RNA has been copied, the last transcribed bases are capped with a series of usually 10 or more adenine bases creating the poly-A tail.

Many transcribed messages (pre-mRNA) in eukaryotes are spliced into smaller sequences called processed or mature mRNA through the excision of introns by the spliceosome, leaving a set of concatenated exons to be passed to the ribosome. The locations in the DNA where the introns are removed are the splice sites, individually called the 5' and 3' splice sites. At least 99% of 5' splice sites begin with "GT" and 3' splice sites end with "AG", called the consensus dinucleotides.

Within the processed mRNA sequence, a subsequence called the coding sequence (CDS) is translated into a peptide chain by the ribosome. The coding sequence is a set of nucleotide triplets called codons and each codon is predictably translated into one of the 20 amino acids. The coding sequence usually begins with the codon "ATG" (sometimes "GTG" in some lower organisms) and terminates with one of three stop codons, "TAA", "TGA", or "TAG". Usually the start codon is the first "ATG" in the processed mRNA (Kozak's Rule[1]). The offset positions in a coding sequence modulo three indicates the frame or phase. Codons, by definition, always begin in frame zero. If one of the three stop codon nucleotide triplet combinations is observed in frame zero, this is called an in-frame stop codon. A sequence of DNA with no in-frame stop codons is called an open reading frame (ORF).

Figure 2.1 graphically depicts this central dogma.

Terminology: Gene Features

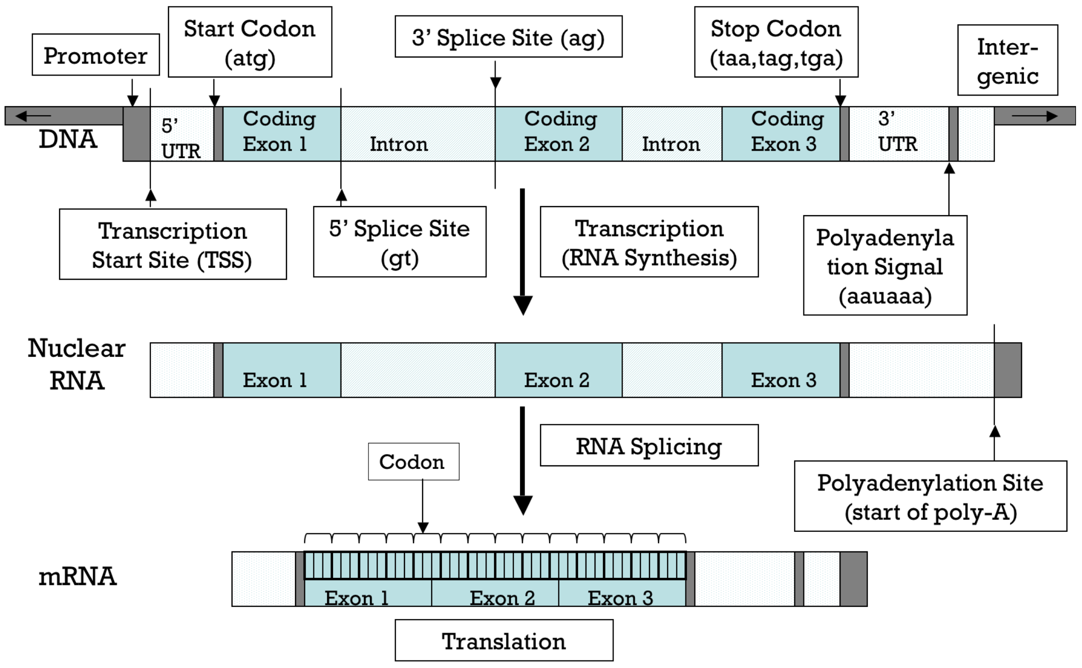


Figure 2.1: Central dogma of molecular biology in eukaryotes. A primary transcript region starting at the promoter is copied into pre-mRNA. The transcript is then spliced to produce the mature cytoplasmic message. The message is translated into peptides. Note that codons may span splice boundaries. Although not shown in this diagram, splicing is possible in the untranslated regions.

2.2 Variability

The processes of transcription and splicing, that is, the creation of mature mRNA is called expression. Unfortunately, the precise details of expression are not deterministic for any gene. The location of transcription initiation can vary due to multiple promoter sites. Transcription termination is quite variable, and each alternative termination site can vary on the order of tens of bases. Splicing can vary as well. Different or new splice sites can be used by the spliceosome. Thus, alternative transcription and alternative splicing provide a mechanism for expressing many different species of the same gene. Indeed, alternative splicing can, in theory, lead to the generation of different mRNAs that is exponential in the number of splice sites. But, while this expression variability appears to be increasingly common in higher organisms such as in human, where estimates of alternative splicing frequency range as high as 60%, in most cases the number of alternative forms is typically only a few. Nevertheless, alternative mRNAs for the same gene can lead to different CDSs, complicating the problem of gene prediction and the assessment of the results.

2.3 Gene features

Given the molecular biology processes defined in the previous section, there are important statistical properties of the corresponding features in a DNA sequence. Before the advent of systematic gene-structure-prediction methods, researchers were concerned with measuring the properties of individual features and devising discriminants for each. Following Staden[2], the feature types can be divided into two categories: signals and contents.

2.3.1 Signals

Signal features are fixed patterns in DNA that describe transition points for transcription, splicing, or translation. Signals loosely correspond to binding sites or special functional patterns recognized by the polymerase, spliceosome, or ribosome. The promoter and polyadenylation sites are signal features for transcription, 5' and 3' splice sites are signal features for splicing, and translation start and stop sites are signal features for translation. Each of these signals are somewhat conserved across different genes and different species. A signal is often a combination of highly conserved nucleotides flanked by less conserved, weaker signals.

For example, the splice sites have consensus dinucleotides, but are flanked by regions with less specific nucleotide patterns. Profiles are often used to describe nucleotide patterns of fixed length[3]. A profile is a simple two-dimensional matrix $P(1 \leq i \leq m, 1 \leq j \leq 4)$ such that $P(i, j)$ is the probability of nucleotide j at position i in a motif of length m. Frequencies can be used to generate these probabilities, priors can be introduced when data is sparse, or more sophisticated contexts can be represented such as dinucleotide frequencies (order-2 Markov profile) at each position resulting in a $M \times 16$ matrix, etc. These profiles have the convenient property of simply generating a likelihood probability for any test sequence represented as a vector $S(1 \leq i \leq m, 1 \leq j \leq 4)$, where $S_{i,j} = 1$ for the nucleotide j found at position i and

0 elsewhere, by $\prod_{i=1\ldots m}\prod_{j=1\ldots 4} P_{i,j}^{S_{i,j}}$. Stormo presented thermodynamic, likelihood, and information theoretic justifications for their use[4]. Figure 2.2 shows examples of simple frequency profiles for start codon and splice sites in *C. elegans*.

In contrast to splicing and translation signals, transcription signals are weak, small, and variably positioned. The polyadenylation site is usually "AATAAA", but there are many variants. Moreover, the motif is small, is not located predictably near other contextual features, and may not even be present. The promoter site is an elaborate complex of weak binding sites and variably ordered.

Many methods have been advanced for detecting signal features in DNA and this paper will not attempt to fully summarize. Gelfand offers a fine review[6]. In my work, I have considered profiles including elaborations proposed by Burge[7], and the neural network methods of Brunak & colleagues[8] and Reese[9, 10] for splice site and promoter prediction. Another important approach are the discriminant analysis methods of Solovyev[11] and Zhang[12], which are not addressed in this work because of their incompatibility with likelihood models.

2.3.2 Contents

Content features are compositional patterns across a stretch of DNA. Content features refer to those extents of DNA that are transcribed, spliced, or translated, i.e., protein-coding regions, untranslated exonic regions, introns, and untranscribed DNA. The intron, untranslated exon, and untranscribed DNA regions are are often treated as a common non-coding class in contrast to the protein-coding regions.

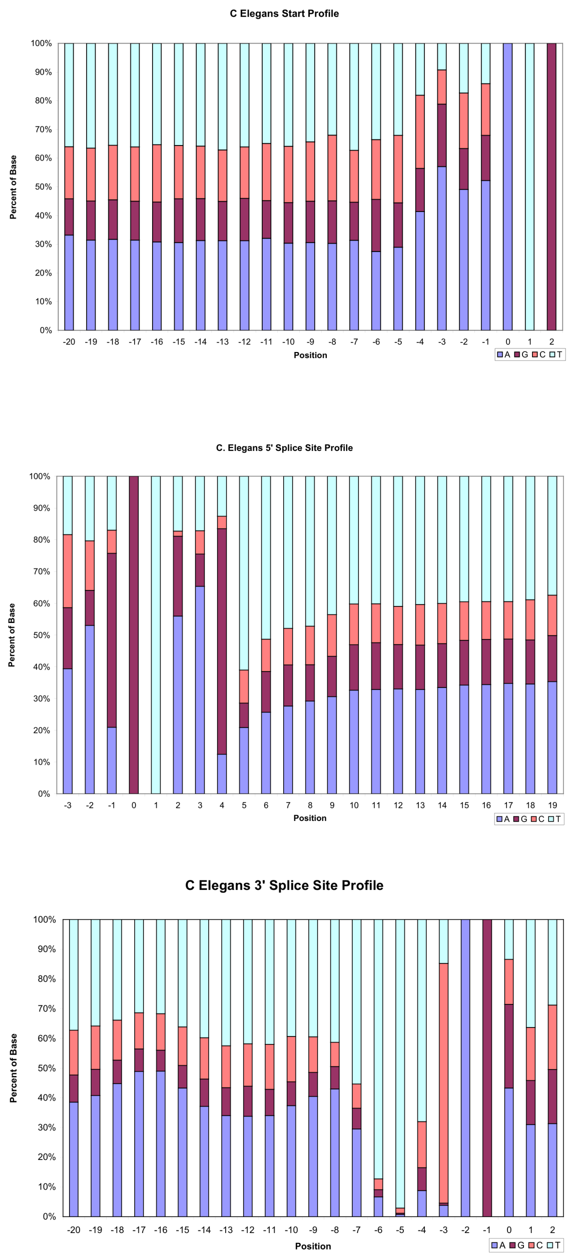
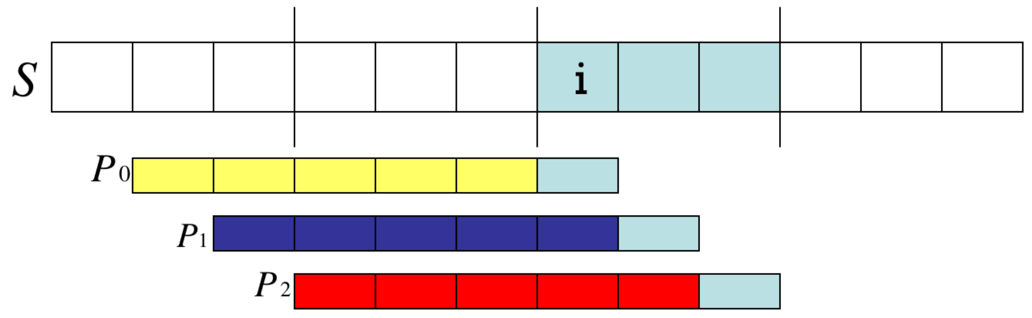


Figure 2.2: *C. elegans* start codon and splice site profiles. Windows of 20 bases upstream of the start codon (position 0), 20 bases downstream of the 5' exon junction ("GT" is positions 0 and 1), and 20 bases upstream of the 3' exon junction ("AG" is positions -1 and -2) were selected from curated gene sequences[5].

Most early research in gene finding focused on coding measures for the discrimination of protein-coding from non-coding regions. All of the coding measures are based on the observations of periodicity (due to the 3-base non-overlapping codons), non-uniformity of amino acids (amino acid preference), non-uniformity of codon selection for a given amino acid (codon preference), and local nucleotide bias. The combination of amino acid and codon preference was defined by Staden as codon usage[3], and this measure lies at the heart of most successful coding measures. Fickett and Tung provide an assessment of many of the proposed coding measures and they found that phased hexamer counts, first published by Claverie, et. al.[13], is the most effective measure[14]. An equivalent model of hexamer counts is a set of three phased order-N Markov chains [15] where $N = 5$, as shown in Figure 2.3, and this is the method employed here. Figure 2.4 shows the distribution of negative log likelihood scores using the three phased 5th-order Markov chains and intronic 5th-order Markov chain for coding regions of length 99. Extending to 6th order Markov chains when sufficient training data is available offers no significant improvement, as shown in Figure 2.4.

Guigo and Fickett[16], showed that all content models are highly correlated with G+C bias, i.e., the local (on the order of 100s or 1000s of bases) frequency of G and C nucleotides versus A and T. Thus, all measures for coding and non-coding benefit from incorporating G+C as an additional parameter or by training separate models for different ranges of G+C.

The lengths of exons and introns are not random (see, e.g. Smith, 1988), but have distinct distributions. For example, both gene density and intron length are



$$P(\text{Codon}_i) = \prod_{f=0}^{2} P_f(S[i+f] \mid S[i+f-5 \ldots i+f-1])$$

Figure 2.3: Three phased Markov chains are used to detect codon usage and dependencies among neighboring bases. A 5th-order Markov chain is similar to hexamer counts.

correlated with G+C (Figure 2.5). The length distribution of contents is an important additional parameter to incorporate into models. For example, internal coding exons of most organisms have a typical length of about 150 to 200 bases. By contrast, intron distributions are broad and vary among organisms. In Figure 2.6, a majority of *Drosophila* and *C. elegans* introns are less than 100 bases and quite regularly about 60-70 nucleotides. Human introns are rarely less than 100 bases and show a long exponential distribution to 10^6 bases.

One simple and effective coding measure is the identification of ORFs of minimal length. For genomes with little or no splicing, or when annotating mature mRNA, the problem of identifying protein-coding regions is a relatively simple problem of finding large open reading frames. ORF detection can be an effective protein-coding region identification method even for organisms with frequent splicing. The likelihood of a 150-base ORF (the length of an average exon) is only 0.11 in random DNA and even less likely in out-of-phase coding regions. Thus, the typical coding exon is

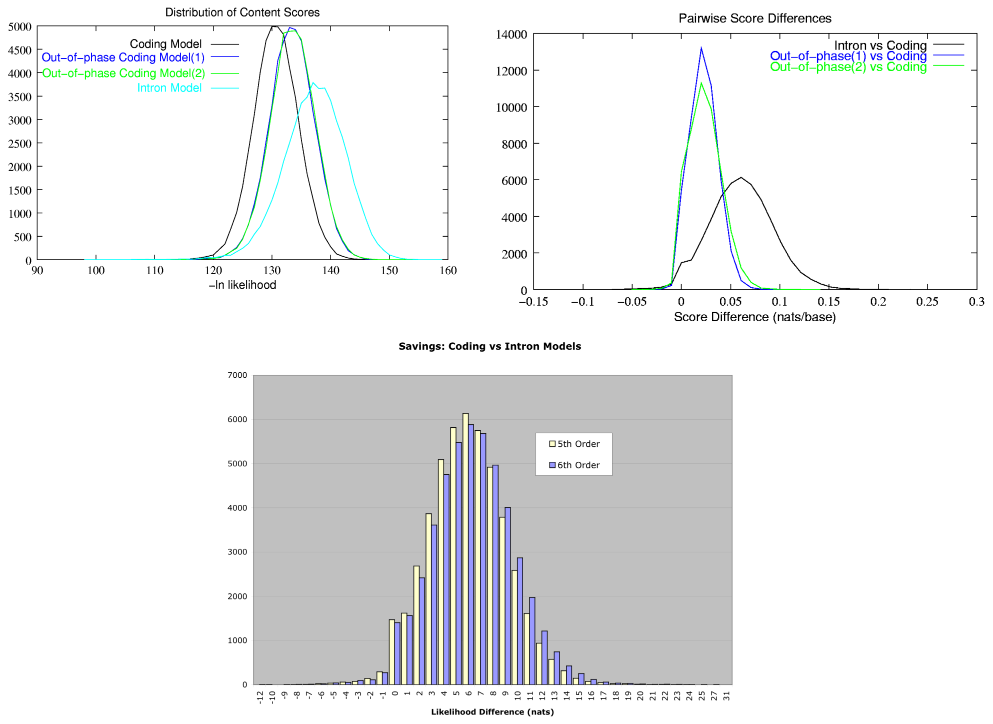


Figure 2.4: Score distributions for Markov chain models of coding and intronic DNA. Three phased Markov chains were trained on 53,183,460 coding bases and one Markov chain was trained on 16,149,264 intronic bases from the well-annotated protein-coding exons of *C. elegans*. For each of 48,124 exons, a 99-nucleotide in-frame coding region was scored using the coding model, two out-of-phase coding models, and the intron model. The top left graph shows the distribution of $-\ln$ likelihoods for 5th-order Markov chains. The top right graph is the distribution of savings per base of the 5th-order coding model versus the intron and out-of-phase 5th-order models for each coding exon. Pairwise comparisons with likelihoods from out-of-phase models were only made if there were no in-frame stop codons in the alternative frame. Note, that although the score distributions in the top left figure are distinct, there is significant overlap, but the top right graph shows that there is good pairwise separation, i.e. most differences between model scores are greater than zero. The bottom graph shows a comparison of the savings per base for 5th- and 6th-order chains. There is a very small, improvement with higher order models.

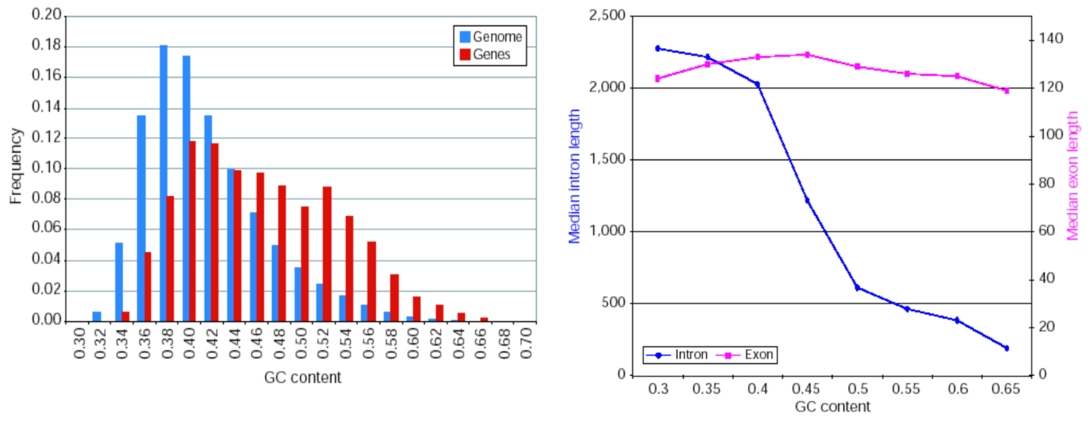


Figure 2.5: G+C dependency on gene density and feature length. The first graph shows frequency of G+C for 135,867 non-overlapping windows from human DNA of length 20,000 and 9,315 windows of length at least 20,000 that fully included well-aligned full-length human mRNAs in finished human DNA contigs. The second graph shows the distribution of exon and intron lengths with respect to G+C. G+C was calculated according to a window at least 10,000 bases wide that fully included the intron or exon.

statistically significant simply due to the presence of an ORF of that length, regardless of other nucleotide patterns. The problem of gene finding is thus an “end game”, to a large degree, in which most of the coding bases are readily identified and the difficulties lie in the prediction of splice sites, the prediction of small coding exons, and, importantly, the incorporation of exons into a consistent arrangement of protein-coding exons. In the ideal, a computer method for predicting genes would be one that merely excelled at signal predictions, mimicking the biological processes. In practice, discrimination is often not possible simply due to the density of putative signals. Thus, the consistent incorporation of predicted features into a unified gene structure is the important “meta-model” of successful gene finding systems as described in the next chapter.

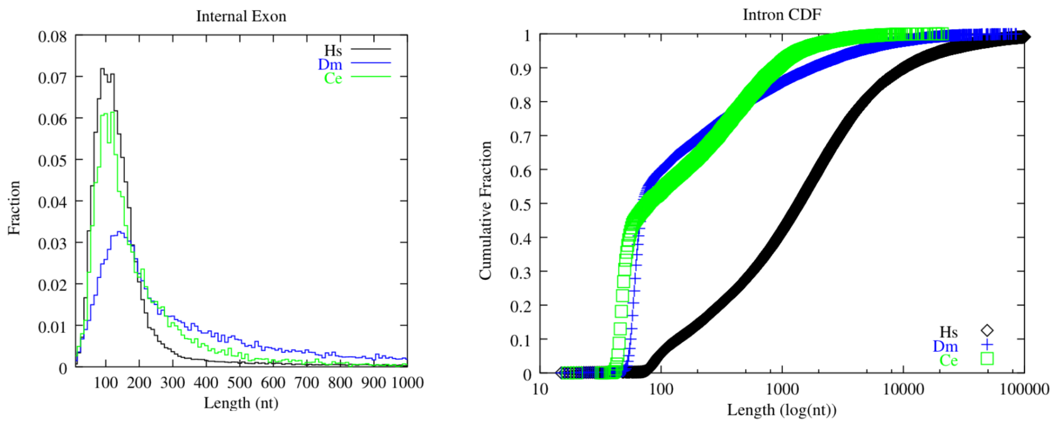


Figure 2.6: Length distributions of exons and introns. The left graph shows a histogram distribution of the length of internal exons (those coding exons that do not contain a start or stop codon) and the right graph shows a cumulative distribution of the lengths of introns plotted in log scale. Both graphs show lengths for human (Hs), *Drosophila melanogaster* (Dm), and *C. elegans* (Ce). The data was generated by aligning (using Blat[17]) 106,158 exons from 10,428 full-length mRNAs for Hs from Genbank to the Dec 2001 assembly, 47,542 exons from 9,820 full-length mRNAs for Dm from the "DGC v1 & 2" from BDGP to the release 3 assembly, and 24,137 exons from 3,627 complete CDS sequences for Ce from WormBase to the final assembly. The longest alignment of greater than 90% of the mRNA length was selected. Gaps less than 15 nucleotides in the genomic or mRNA were merged. Exon distributions tend to be similar among species. The majority of introns are about 70 bases for Dm and Ce; Hs does not have a peak length and tends to be much longer. All have approximately an exponential distribution for intron length.

Chapter 3

Gene Models

3.1 Gene structure models

In the early 1990s, several new methods were published describing complex systems that used different metrics to generate an integrated gene structure prediction [18, 19, 20, 21]. Whereas previous programs predicted exons, splice sites, or other individual features, these new systems combined information to predict a gene structure—that is, a series of connected coding exons. Earlier work used rules to filter through possible combinations of features in a way that mimicked that manual annotation process of the day. Snyder and Stormo[22] showed how the optimal combination of features could be obtained using dynamic programming.

3.1.1 Linguistic gene models

Dong and Searls[23, 24] were the first major proponents for describing gene structure in linguistic terms. Searls showed how gene structure could be modeled in terms of formal grammar productions. The concept is appealing because it provides an abstract representation for the biological processes and a means for computation. Gene structure can be described at the top-level as transcription, splicing, and translation. Using standard BNF syntax, the structure of genes in DNA can be expressed in terms of a context-free grammar. For example, a grammar might contain

```
        DNA  →  Untranscribed Transcribed Untranscribed
Transcribed  →  5′UTR Coding 3′UTR
     Coding  →  Exon Intron Exon
     Intron  →  Intron Exon Intron
        ...
```

Unlike Searls, who required a context-free grammar in his models, it is possible to describe gene structure in terms of a regular grammar, which offers significant computational advantages. In my derivation, to simplify mapping from regular grammars to FSAs, I will only use a modified right-regular production such that any production of the form $Z \rightarrow (a|b)Y$ is written as the two productions $Z \rightarrow XY$ and $X \rightarrow (a|b)$. This style will map conveniently to hidden Markov model state transition and arc distributions. Here is the regular grammar derivation:

```
           DNA  →  IntergenicBase DNA
           DNA  →  IntergenicBase ForwardTranscribed
           DNA  →  IntergenicBase ReverseTranscribed
           DNA  →  null
IntergenicBase  →  A | G | C | T
```

The reverse strand is handled analogously to the forward strand, but not explicitly presented here.

$$\texttt{ForwardTranscribed} \rightarrow \texttt{5}'\texttt{UTR}$$

The 5' UTR region can be interrupted by one or more introns and the CDS follows the 5' UTR.

$$\begin{aligned}
\texttt{5}'\texttt{UTR} &\rightarrow \texttt{ExonBase 5}'\texttt{UTR} \\
&\rightarrow \texttt{ExonBase Translated} \\
&\rightarrow \texttt{ExonBase Intron} \\
\texttt{Intron} &\rightarrow \texttt{GT Intron}' \\
\texttt{Intron}' &\rightarrow \texttt{IntronicBase Intron}' \\
&\rightarrow \texttt{AG 5}'\texttt{UTR} \\
\texttt{ExonBase} &\rightarrow \texttt{A | C | G | T}
\end{aligned}$$

The translated region is similarly handled, but to ensure frame consistency the coding regions must maintain phase across introns. Let the naming convention, $\texttt{Intron}_i$ mean an intron that is interrupting a coding region in frame i, and $\texttt{Base}_j$ refer to the jth base of a partial codon.

$$
\begin{array}{rcl}
\texttt{Translated} & \rightarrow & \texttt{ATG CDS} \\
\texttt{CDS} & \rightarrow & \texttt{Codon CDS} \\
 & \rightarrow & \texttt{Codon Intron}_0 \\
 & \rightarrow & \texttt{Codon Base}_0\ \texttt{Intron}_1 \\
 & \rightarrow & \texttt{Codon Base}_0\ \texttt{Base}_1\ \texttt{Intron}_2 \\
 & \rightarrow & \texttt{Stop 3}'\texttt{UTR} \\
\texttt{Stop} & \rightarrow & \texttt{TAG | TGA | TAA} \\
 & & \\
\texttt{Intron}_0 & \rightarrow & \texttt{GT Intron}'_0 \\
\texttt{Intron}'_0 & \rightarrow & \texttt{IntronicBase Intron}'_0 \\
 & \rightarrow & \texttt{AG CDS} \\
 & & \\
\texttt{Intron}_1 & \rightarrow & \texttt{GT Intron}'_1 \\
\texttt{Intron}'_1 & \rightarrow & \texttt{IntronicBase Intron}'_1 \\
 & \rightarrow & \texttt{AG Base}_1\ \texttt{Base}_2\ \texttt{CDS} \\
 & & \\
\texttt{Intron}_2 & \rightarrow & \texttt{GT Intron}'_2 \\
\texttt{Intron}'_2 & \rightarrow & \texttt{IntronicBase Intron}'_2 \\
 & \rightarrow & \texttt{AG Base}_2\ \texttt{CDS} \\
 & & \\
\texttt{IntronicBase} & \rightarrow & \texttt{A | C | G | T} \\
\texttt{Base}_0 & \rightarrow & \texttt{A | C | G | T} \\
\texttt{Base}_1 & \rightarrow & \texttt{A | C | G | T} \\
\texttt{Base}_2 & \rightarrow & \texttt{A | C | G | T} \\
\texttt{Codon} & \rightarrow & \texttt{AAA | AAC | AAG | AAT | ACA ...}
\end{array}
$$

Last, the 3' UTR is like the 5' UTR:

$$
\begin{array}{rcl}
\texttt{3}'\texttt{UTR} & \rightarrow & \texttt{ExonBase 3}'\texttt{UTR} \\
 & \rightarrow & \texttt{ExonBase DNA} \\
 & \rightarrow & \texttt{ExonBase Intron} \\
\texttt{Intron} & \rightarrow & \texttt{GT Intron}' \\
\texttt{Intron}' & \rightarrow & \texttt{IntronicBase Intron}' \\
 & \rightarrow & \texttt{AG 3}'\texttt{UTR} \\
\texttt{ExonBase} & \rightarrow & \texttt{A | C | G | T}
\end{array}
$$

One final detail is that the grammar must not allow in-frame stop codons to span across introns. This can be handled with additional states. Replace the productions derived from the "Translated" non-terminal as follows:

$$\begin{array}{rcl}
\texttt{Translated} & \rightarrow & \texttt{ATG CDS} \\
\texttt{Stop} & \rightarrow & \texttt{TAG | TGA | TAA} \\
\texttt{CDS} & \rightarrow & \texttt{Codon CDS} \\
 & \rightarrow & \texttt{Codon Intron}_0 \\
 & \rightarrow & \texttt{Codon T Intron}_1\texttt{-T} \\
 & \rightarrow & \texttt{Codon Not-T Intron}_1 \\
 & \rightarrow & \texttt{Codon TA Intron}_2\texttt{-TA} \\
 & \rightarrow & \texttt{Codon TG Intron}_2\texttt{-TG} \\
 & \rightarrow & \texttt{Codon Not-T[AG] Intron}_2 \\
 & \rightarrow & \texttt{Stop UTR3'}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Not-T} & \rightarrow & \texttt{A | C | G} \\
\texttt{Not-T[AG]} & \rightarrow & \texttt{AA | AC | AG | AT | CA | CC | CG | CT} \\
 & \rightarrow & \texttt{GA | GC | GG | GT | TC | TT}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Intron}_0 & \rightarrow & \texttt{GT Intron}'_0 \\
\texttt{Intron}'_0 & \rightarrow & \texttt{IntronicBase Intron}'_0 \\
 & \rightarrow & \texttt{AG CDS}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Intron}_1 & \rightarrow & \texttt{GT Intron}'_1 \\
\texttt{Intron}'_1 & \rightarrow & \texttt{IntronicBase Intron}'_1 \\
 & \rightarrow & \texttt{AG Base}_1\texttt{ Base}_2\texttt{ CDS}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Intron}_1\texttt{-T} & \rightarrow & \texttt{GT Intron}'_1\texttt{-T} \\
\texttt{Intron}'_1\texttt{-T} & \rightarrow & \texttt{IntronicBase Intron}'_1\texttt{-T} \\
 & \rightarrow & \texttt{AG Not-[AG|AA|GA] CDS}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Intron}_2 & \rightarrow & \texttt{GT Intron}'_2 \\
\texttt{Intron}'_2 & \rightarrow & \texttt{IntronicBase Intron}'_2 \\
 & \rightarrow & \texttt{AG Base}_2\texttt{ CDS}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Intron}_2\texttt{-TA} & \rightarrow & \texttt{GT Intron}'_2\texttt{-TA} \\
\texttt{Intron}'_2\texttt{-TA} & \rightarrow & \texttt{IntronicBase Intron}'_2\texttt{-TA} \\
 & \rightarrow & \texttt{AG Not-[AG] CDS}
\end{array}$$

$$\begin{array}{rcl}
\texttt{Intron}_2\texttt{-TG} & \rightarrow & \texttt{GT Intron}'_2\texttt{-TG} \\
\texttt{Intron}'_2\texttt{-TG} & \rightarrow & \texttt{IntronicBase Intron}'_2\texttt{-TG} \\
 & \rightarrow & \texttt{AG Not-A CDS}
\end{array}$$

```
Not-[AG|AA|GA]  →  AC | AT | CA | CC | CG | CT
                →  GC | GG | GT | TA | TC | TG | TT
         Not-A  →  C | G | T

  IntronicBase  →  A | C | G | T
         Base0  →  A | C | G | T
         Base1  →  A | C | G | T
         Base2  →  A | C | G | T
         Codon  →  AAA | AAC | AAG | AAT | ACA ...
```

This grammar is nearly sufficient to recognize only biologically legitimate gene structures. Only sequences containing zero or more transcribed genes with no in-frame stop codons, a CDS length being a multiple of three, and splicing occurring only at "GT" and "AG" junctions are recognized. The complete grammar is also shown graphically as an FSA in Figure 3.1. The grammar does not quite describe biological reality. It is possible, although rare, for genes to overlap, but this phenomenon is not captured in this grammar. It is also possible for the start and stop codons to span introns, but is specifically prohibited here. Fewer than 1% of splice sites do not contain consensus dinucleotides. In addition, the model does not include minimum or maximum length restrictions for loop features. Minimum lengths can obviously be added to the grammar by increasing the number of states. Maximum length restrictions cannot be reasonably represented in this framework without an enormous expansion of states. The problem of length restrictions will be addressed generally in terms of length distributions in Section 3.2.

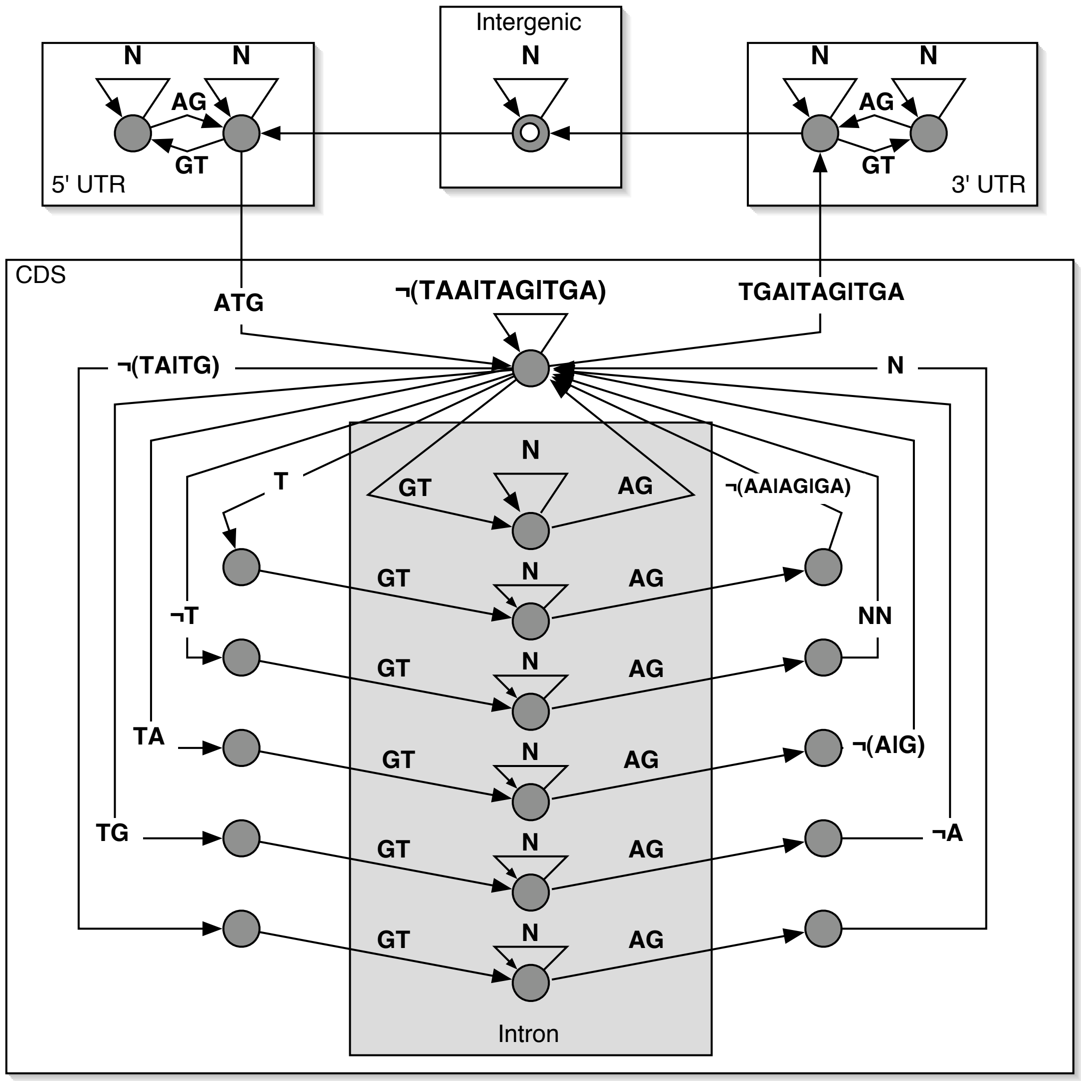


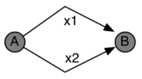
Figure 3.1: Finite state automaton that recognizes gene structures. This FSA recognizes all protein-coding gene structures such that there are no in-frame stop codons, a CDS length of a multiple of three, "GT"/"AG" splicing, and no intron-spanning stop codons. (Intron-spanning stop codons are, in fact, legal, but very rare, and so are explicitly excluded.)

3.1.2 Hidden Markov models

The FSA described above is a language recognizer for genes in DNA, and the computational complexity of such a recognizer is only $O(NM)$ where N is the length of the sequence and M is the number of states. However, the gene finding problem is not only to determine whether a given query sequence can be accepted by our language—in fact, all sequences are accepted by the grammar as completely intergenic—but which is the parse of the data that represents reality. This problem lends itself to a probabilistic grammar framework in which a likelihood estimate can be attributed to each possible parse by assigning probabilities to each production in the grammar. Formally, this is equivalent to a hidden Markov model, as follows:

Given a normalized, right-regular grammar with productions of the form

$$\begin{array}{lcl} \mathtt{A} & \rightarrow & x_1\ \mathtt{B} \\ \mathtt{A} & \rightarrow & x_2\ \mathtt{B} \end{array}$$

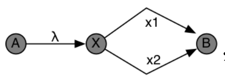
and a corresponding FSA of 

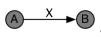
then a set of probabilities can be derived from the graph where $P(A)$ is the probability of applying one of the productions for the non-terminal A, $P(x, B|A)$ is the conditional probability of emitting the terminal symbol x and the non-terminal B given the application of A, etc. The probability of generating symbol x using this production is

$$\begin{aligned} P(x, AB) &= P(A)P(x, B|A) \\ &= P(A)P(x|A)P(B|x, A) \\ &= P(A)P(x|A)P(B|A) \end{aligned} \tag{3.1}$$

All such productions can be written equivalently, as in the DNA gene grammar above, as

```
A  →  X B
X  →  x1
X  →  x2
```

creating a new state machine 

, where λ emits no symbol. Applying the same logic, the joint probability is $P(x, AXB) = P(A)P(x|X)P(B|A)$. It is convenient to collapse all sets of terminal productions and label the arc of the FSA by the *state* X: 

. I introduce a function $\mathsf{arc}(A, B)$ that returns the terminal production state X. We can then rewrite the joint probability as

$$P(x, AXB) = P(A)P(x|\mathsf{arc}(A, B))P(B|A)$$

In short, this new FSA is equivalent to any normal FSA, but a shorthand method is employed such that terminal productions are all on the arcs. This then leads to the following definitions:

x	$=$	A nucleotide in the set $\{A, G, C, T\}$.
S	$=$	The sequence of length N, where $S[i]$ is the ith symbol in S.
Q	$=$	The set $\{j, k, \ldots\}$ of M non-terminal productions according to the convention described in the derivation, above. Equivalently, the set of states in the shorthand FSA.
$P(x\|\mathsf{arc}(j,k))$	$=$	The probability distribution over terminal productions. Equivalently, the probability of emitting symbol x during the transition from state j to k.
$P(k\|j)$	$=$	The probability distribution over each non-terminal production. Equivalently, the probability of transitioning into state k conditioned on being in state j.
Φ	$=$	A parse. A sequence of $N+1$ states selected from Q, symbol $S[i]$ is emitted on the arc between states Φ_{i-1} and Φ_i. $\Phi_0 = 0$, a special start state.

Given these definitions, the joint probability of a sequence and some parse, Φ, is

$$P(S, \Phi) = \prod_{i=1\ldots n} P(S[i]|\mathsf{arc}(\Phi_{i-1}, \Phi_i))P(\Phi_i|\Phi_{i-1}) \tag{3.2}$$

While an FSA can report whether a sequence is recognized in a language, this probabilistic FSA can report the probability that a sequence is generated by the language, that is,

$$P(S) = \sum_{\Phi} P(S, \Phi) \tag{3.3}$$

Moreover, an individual parse that results in a high likelihood of S can be interpreted as a better or more desirable parse. Thus, the objective is to identify Φ that maximizes the likelihood of S, i.e.

$$\arg\max_{\Phi} P(S, \Phi) \tag{3.4}$$

The best Φ can be deduced easily through an inductive procedure that is obvious from Equation 3.2. The so-called Viterbi dynamic programming method provides an efficient solution[25]. Equivalently, a depth-first search can be applied on a graph $G = (V, E)$ where the vertices, V, are the cells in the dynamic programming array, i.e. state/position pairs, and there is an edge in E between each pair of vertices with a non-zero transition probability (i.e. where $P(i|j) > 0$). Such a sparse matrix implementation is useful when sequences and state counts are large, not all states are possible at all positions, and memory is a significant limitation. The running time of either algorithm is proportional to the length of the input sequence and the number of states ($O(NM)$).

In pseudocode, let $S[i]$ be a letter in the input sequence where $1 \leq i \leq N$, and let $V_{i,j}$ be the vertex corresponding to the state j, $1 \leq j \leq M$, at position $S[i]$ in the sequence. score($V_{i,j}$) is the memoized score at that node, and parent($V_{i,j}$) is the memoized parent node representing the optimal path. Both score() and parent() are initially undefined. The following recursive function populates the data structure V with scores and optimal parent paths.

```
Function $F(i, j)$
    If score($V_{i,j}$) is defined Return score($V_{i,j}$)
    If $i = 1$
    Then
        Set score($V_{i,j}$) $= P(S[i]|j)P(j|0)$
    Else
    For each $k$, where $P(j|k) > 0$
```

```
        If $F(i-1,k) \cdot P(S[i]|\text{arc}(k,j))P(j|k) > \text{score}(V_{i,j})$
        Then
            Set $\text{score}(V_{i,j}) = F(i-1,k) \cdot P(S[i]|\text{arc}(k,j))P(j|k)$
            Set $\text{parent}(V_{i,j}) = (i-1,k)$
    Return $\text{score}(V_{i,j})$
```

$F(N,M)$ returns $\arg\max_\Phi P(S,\Phi)$ and, as a side effect, the populated data structure V now contains the parse, Φ.

Determining $P(S)$ requires a simple modification of $F(i,j)$ to sum (Equation 3.3), instead of maximize (Equation 3.4), over possible parses:

```
Function $G(i,j)$
    If $\text{score}(V_{i,j})$ is defined Return $\text{score}(V_{i,j})$
    If $i = 1$
    Then
        Set $\text{score}(V_{i,j}) = P(S[i]|j)P(j|0)$
    Else
    For each state $k$, where $P(j|k) > 0$
        Set $\text{score}(V_{i,j}) = \text{score}(V_{i,j}) + G(i-1,k) \cdot P(S[i]|\text{arc}(k,j))P(j|k)$
    Return $\text{score}(V_{i,j})$
```

It is often useful to determine the posterior probability $P(\Phi_{i-1} = k, \Phi_i = j|S)$. This value provides the answer to such questions as the probability that the nucleotide "C" at position 22 is an intronic base, for example. The solution is obtained by summing only over all parses that contain the specified arc at the specified position. By creating a reverse summation function, G', and a second set of vertices, V', this value is easily determined:

```
Function $G'(i,j)$
    If $\text{score}(V'_{i,j})$ is defined Return $\text{score}(V'_{i,j})$
    If $i = N$
    Then
```

```
        Set score($V'_{i,j}$) $= 0$
    Else
    For each $k$, where $P(k|j) > 0$
        Set score($V'_{i,j}$) $=$ score($V'_{i,j}$) $+ G'(i+1,k) \times P(S[i+1]|\mathsf{arc}(j,k))P(k|j)$
    Return score($V'_{i,j}$)
```

Then the posterior can be calculated from functions G and G' as:

$$\begin{aligned} P(\Phi_{i-1} = k, \Phi_i = j|S) &= \frac{P(S, \Phi_{i-1}=k, \Phi_i=j)}{P(S)} \\ &= \frac{P(S[1...i-1], \Phi_{i-1}=k)P(j|k)P(S[j...N], \Phi_i=j)}{P(S)} \qquad (3.5) \\ &= \frac{G(i,k)P(S[i]|\mathsf{arc}(k,j))P(j|k)G'(i,j)}{P(S)} \end{aligned}$$

This derivation is a variant of the forward/backward algorithm. Note, that if Z and Z' are fully populated by calling $G(N,M)$ and $G'(1,1)$, then all subsequent calls for any values of i and j are precomputed.

To summarize, it is possible to formally describe the processes of transcription, splicing, and translation in terms of a finite state automaton representing productions in a regular grammar. A generalization of the FSA with probabilities assigned to transitions between nodes expands the power of the model from language recognition to parse prediction. A modified FSA with all terminal symbol productions placed on the arcs is equivalent to a probabilistic FSA, but naturally translates to an HMM framework with probability distributions over state transitions and observed symbols. A recursive graph traversal, which is equivalent to the dynamic programming used in the standard Viterbi and forward/backward methods, can be used for determining the HMM solutions. Thus, the HMM framework for gene finding provides a method to integrate likelihood models for different gene features into a single system, provides grammatical constraints to ensure the integrated parse is legitimate, and provides a probabilistic interpretation of the DNA input sequence.

Krogh, Mian, and Haussler[26] first proposed the use of HMMs for non-splicing gene finding in *E. coli*, although not in a linguistic-motivated way as I describe here. Their work also took advantage of automated learning methods for standard single-symbol emission HMMs, which I do not pursue. My contribution involves improvements to the HMM framework to increase the expressibility of the model described in the next section.

3.2 Generalized HMMs

The primary weakness of the gene finding HMM framework from the previous chapter is the demand that each state emits a fixed number of symbols—in this case, one symbol. A regular grammar cannot restrict the number or distribution of productions used in a loop construct. For example, note that the CDS productions "`CDS` $\rightarrow$ `Codon CDS`" with a probability p and "`CDS` $\rightarrow$ `Codon`" with a probability $p - 1$ allow for the generation of an arbitrary number of consecutive codons. If the production probability is p, then the probability of CDS segments of length l is geometric in $p^{l-1}(1 - p)$. But as shown in Figure 2.6, the length distribution of exons is not geometric. It would be convenient if length distributions could be specified for an arbitrary segment of symbols.

In addition, the basic gene finding HMM framework lacks a distinction between the likelihood models for features and the gene-structure model. In a classic HMM, the distributions of symbols in a given state are learned from training data as part of the Expectation Maximization learning procedure, but the incorporation of likelihood models such as the three phased 5th-order Markov chain would require a proliferation of states and parameter "tying". Other likelihood models are simply not amenable to incorporation into the standard HMM framework. For example, I used neural networks for measuring coding potential in [27]. Burge describes a set of profiles selected using a decision tree to characterize the 5' splice site, which would require a "lookahead" and modified training methods using a standard HMM[7]. Although Krogh has demonstrated that careful model construction and parameter tying can produce a high performance gene finding HMM[28], it is certainly convenient to separate the implementation of the likelihood models from the grammatical constraints of the state machine.

Thus, there are two goals: to provide length restrictions and to abstract the likelihood models from the HMM framework. To accomplish this, I expand on the explicit-state-duration hidden Markov models described by Rabiner[25]. Each state is now allowed to generate an arbitrary number of symbols and a length distribution is added to all states in the model. This differs from Rabiner in that it is applied to *all* states in a model rather than as a special case. To allow multiple symbols, a subtle addition to notation is used. Previously we use the square brackets to denote a single symbol in S. Now, in addition, let the set $X = S_1, S_2, \ldots, S_r$ be any set of non-overlapping segments of S such that the ordered concatenation of the segments is the original sequence S. Thus, each segment is a subsequence of S where $S_p = S[i_p \ldots i_{p+1} - 1]$. A new type of parse, Φ', is the set of states $\{\Phi'_1, \ldots, \Phi'_r\}$ where the segment S_p is emitted between states Φ'_{p-1} and Φ'_p. Finally, an additional length term per segment is incorporated in the joint probability as follows:

$$P(S, \Phi', X) = \prod_{p=1 \ldots r} P(S_p | \mathsf{arc}(\Phi'_{p-1}, \Phi'_p)) P(\Phi'_p | \Phi'_{p-1}) P(\mathsf{len}(S_p) | \mathsf{arc}(\Phi'_{p-1}, \Phi'_p))$$

The additional utility in this formulation is that $P(S_p | \mathsf{arc}(\Phi'_{p-1}, \Phi'_p))$ can be drawn from any desired likelihood model. The combination of the state duration HMM and the abstraction of likelihood models I define as a generalized HMM (GHMM). Since the sequence S can be composed of any set of non-overlapping segments, then the optimal parse is now found by maximizing over all possible parses and all possible segmentations of S, that is,

$$\arg\max_{\Phi', X} P(S, \Phi', X)$$

As currently described, the run-time complexity of the DFS is now $O(N^2 M)$ (N is the length of S and M is the number of states), which introduces a serious computational challenge. In Chapter 4, I will explain how judicious attention to the graph topology ensures that the running time, in practice, is only $O(N)$.

3.2.1 The Genie GHMM

A software program called Genie was written to experiment with the concepts of a generalized hidden Markov model. Although there are some organism-specific variations described in later chapters, the GHMM broadly used for many different organisms is essentially the same.

The Genie model is directly derived from the finite state automaton in Figure 3.1. The FSA is modified to incorporate multi-symbol likelihood models for coding and non-coding exons. Fixed-length likelihood models replace the 2-base splice sites. The length of the coding regions in the initial and final coding exon differs from internal coding exons because the exon is interrupted by a start or stop codon. In addition, the length of coding regions in non-spliced mRNA tends to be significantly longer than spliced coding regions. Therefore, separate likelihood models for initial, internal, final, and single coding exons are introduced. The detailed model is shown in Figure 3.2. New arcs are introduced to accommodate the different types of exons. This model allows for the simultaneous analysis of forward and reverse strand, and produces an optimal labeling of 0, 1, or more genes on either strand.

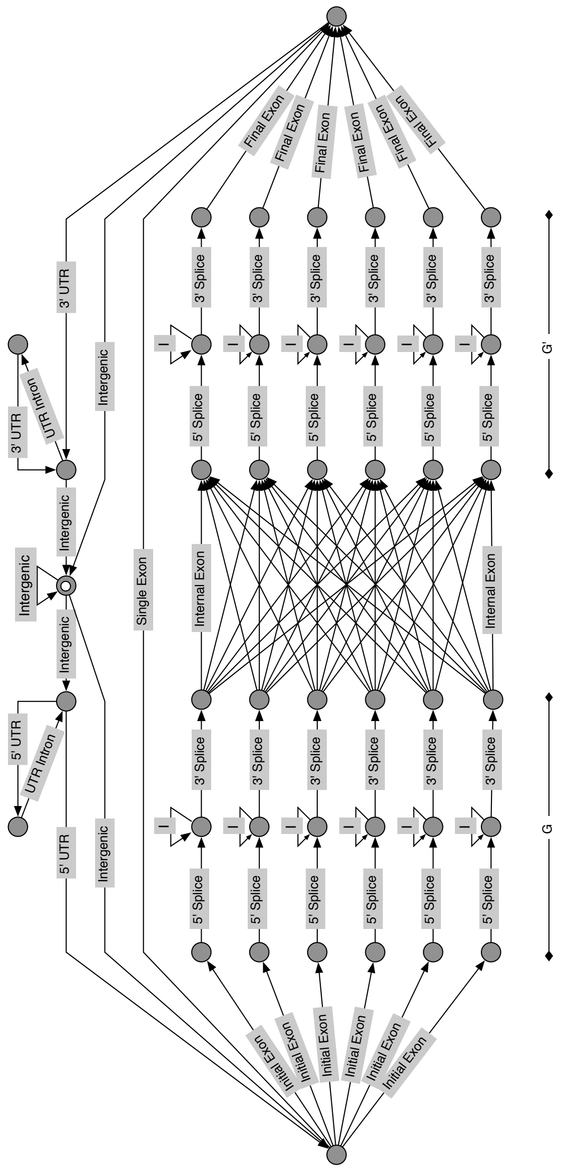


Figure 3.2: Generalized hidden Markov model used by the Genie gene finding system. The grey boxed labels along each arc represent multi-symbol likelihood models and length distributions over all possible sequences. For clarity, the subgraphs identified on the left and right by the horizontal bars labeled G and G' are duplicated in the display, but in the implementation there is only one set of these nodes and arcs. The six parallel paths in this subgraph maintain state necessary to ensure both frame consistency and avoid intron-spanning stop codons (as shown in the grammar and derived FSA in Chapter 3.1.1). Partial genes are accommodated by adding additional exon coding arcs from the intergenic node directly to the 5' splice and translation stop states and from the 3' splice site and translation start states directly back to the intergenic node. The reverse strand is not shown, but is the exact mirror image except for the shared intergenic state. To enforce a single gene prediction, separate begin and end intergenic states can be substituted for the single intergenic source and sink.

Chapter 4

Implementation

4.1 Data structures and initialization

Several implementation details of the gene finding system allow for a flexible and efficient solution. A motivation of the GHMM framework is the abstraction of likelihood models of gene regions from the HMM topology. Such an abstraction provides flexibility in the choice of likelihood models that would be difficult to implement in a classic HMM framework. In addition, this abstraction lends itself to a modular framework for software development in which the model topology and the likelihood models associated with states can all be specified at run-time. In the Genie program, a set of C++ classes define the different parts of the system. The model topology, that is, the grammatical constraints represented by connections of arcs between states, is embodied in a "GeneModel." The set of all possible parses of a specific sequence given the gene model, also called the induced graph, is a "GeneGraph" object. Thus, there are two graphs: the GHMM syntactic graph (Figure 3.2) and the induced graph for a specific sequence (Figure 4.1). A state in the GeneModel is called a "TransitionType"—for example, the state between the initial exon arc and the 5' splice site arc—and a "Transition" is a lightweight object corresponding to a node in the induced graph. A node is a position/state pair. In short, the GeneModel contains the details of the interconnections of TransitionType states (e.g., 5' splice site state follows start translation state), while a GeneGraph contains a collection of Transition nodes (e.g., a possible position of a start translation node at position 539 and the 5' splice site nodes that connect to it downstream at positions 985, 1044, etc.)

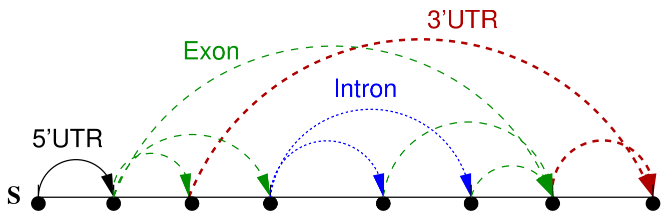


Figure 4.1: The induced gene graph. A directed acyclic graph representing all possible parses given a GHMM and an input sequence. The sequence, S, is the horizontal axis. Each node represents a state/position pair. The different colored arcs represent different likelihood models between states. This example graph is different than one that would be derived from the GHMM in Figure 3.2 and is meant only to give an intuitive grounding for the concept of the induced graph. The directed acyclic graph for a specific sequence is computed by scanning the input sequence to identify putative state/position nodes and connecting nodes according to the GHMM topology.

The GeneModel also contains the specifications for the types of likelihood models used along each arc leaving each TransitionType and the transition probabilities to follow each arc. For example, both a single exon arc and an initial exon arc leave the start translation state and different transition probabilities are assigned to the two arcs. Objects called "MetaContent" contain the code specific for supplying an appropriate likelihood score for each type of arc.

The graph and class terminology is graphically shown in Figure 4.2 and summarized as follows:

- The *GHMM* is a *GeneModel*
- The *induced graph* is a *GeneGraph*
- *States* in the GHMM graph, never called nodes, are *TransitionTypes*
- *Nodes* in the induced graph, never called states, are *Transitions*
- The *likelihood models* for arcs on the induced graph are *MetaContents*

As a pre-processing step, code is executed to identify the sites of putative Transition nodes in a GeneGraph. For example, a start translation node can only occur at a position in a sequence that begins with an "ATG". Specific objects called "Signals" provide the implementation specific to each state. This is an important detail of the software design because it allows for the creation of sparse GeneGraphs. By contrast, the typical dynamic programming method for HMM analysis requires the allocation of memory to accommodate all possible states at all possible positions—usually a $N \times M$ matrix. Many nodes are not possible at any given position and the graph implementation is essentially an analog for a sparse matrix in a dynamic programming implementation.

When the program starts, the details of the interconnections of TransitionTypes, the types of MetaContents, and the types of Signals are loaded into a Gen-

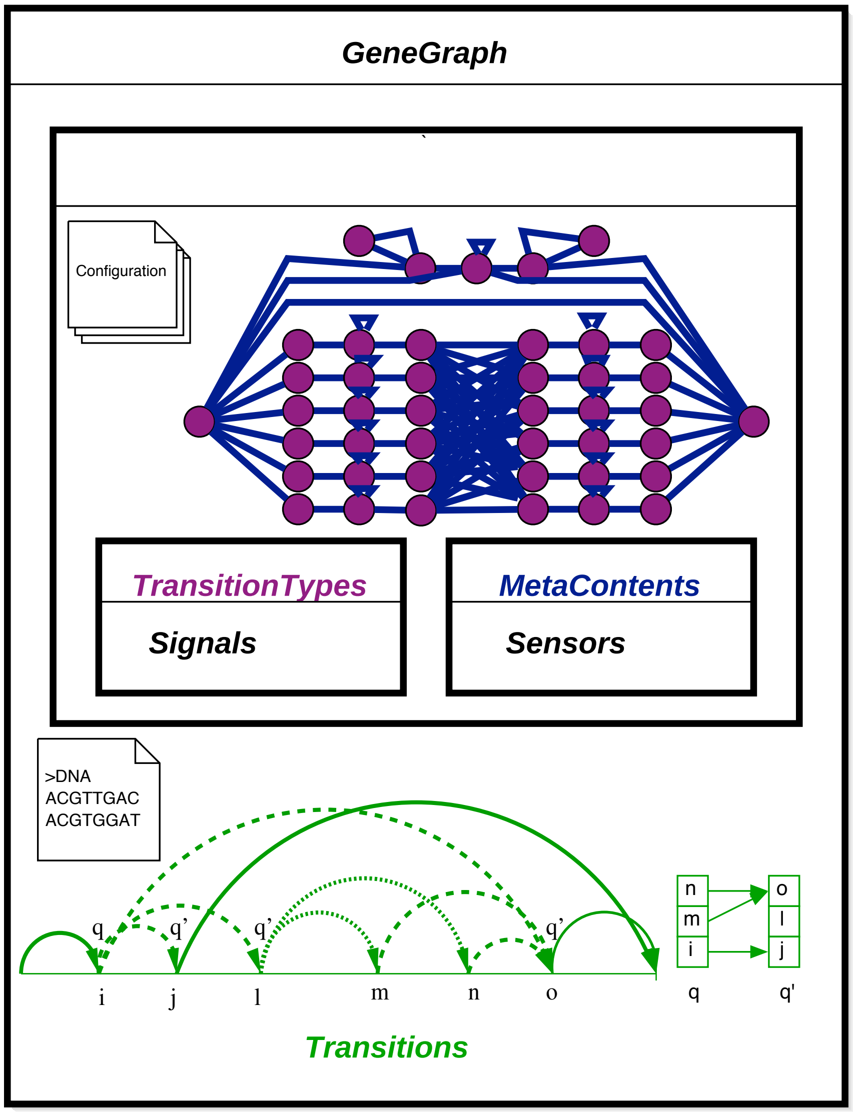


Figure 4.2: Software object architecture. The primary data structures used in the implementation of Genie. A GeneGraph is instantiated per input sequence. It contains a GeneModel describing the topology and a collection of TransitionTypes and MetaContents for creating and scoring the GeneGraph, respectively. Signals identify allowable positions for a TransitionType and one or more Sensors implement the likelihood models for scoring the arcs by the MetaContents. The induced graph is represented by a set of arrays of Transitions (bottom right) and each Transition contains a set of pointers to offsets in the arrays representing nearest connected nodes.

eModel from a set of configuration files. When a DNA sequence is loaded, a GeneGraph is initialized containing a GeneModel. The induced graph for an input sequence is determined as follows:

For each position i in the sequence from N down to 1, for each possible TransitionType q, the Signal object is asked whether the state q is allowed at position i. If the position is allowable, then i is pushed onto a stack, T_q, of Transition nodes of type q. The node is the pair (i, q).

Next, the connectivity of node (i, q) with all other nodes is deduced. Since the GeneGraph is a directed acyclic graph, then (i, q) can only be adjacent to nodes (j, q') where $j > i$ and where q' follows q in the GeneModel. The nearest possible adjacent node for each state q' is simply the top member of the each stack $T_{q'}$. Thus, for each node (i, q), a pointer is stored to the current top of the stack for all $T_{q'}$, where q' is adjacent to q in the GeneModel.

Connectivity from node (i, q) to more distant nodes can be determined by simply tracing through the stack, so only one arc is explicitly stored for each possible adjacent state q'.

The creation of the GeneGraph requires only $O(kNM)$ in time and space to build the graph, where k is the maximum number of adjacent TransitionType states—typically on the order of 1 to 6. This process is repeated in the reverse direction to generate the upstream links to accommodate the backward part of the forward/backward summation.

4.2 Graph traversal

The GeneGraph fully describes the induced graph specific to the sequence of interest. The recursive depth-first algorithm is then applied starting with the Transition object corresponding to the first position of the sequence at the initial start state of the model and searches for the Transition object at the last position of the sequence and in the final model state. Each call to the DFS simply takes an identifier of a current Transition and the goal Transition. In each call of the DFS routine, all outgoing arcs are scored for the current transition, if not yet visited, and then each Transition is recursively called. For the Viterbi, the best score and the adjacent node corresponding to the best path is stored, requiring $O(NM)$ storage for scores and paths. For the forward algorithm, the sum of scores is stored and no path information is saved.

This approach presents some running time challenges. In the worst case, all states at all positions have arcs to all states at all downstream positions. The additional power of a multi-symbol state brings an additional multiplicative N term to the running time yielding calls to score functions on the order of $O(N^2M^2)$. Even worse, the time to score any putative region between any two points can be, at worst, on the order of the size of the region to score. Thus, running time could be as bad as $O(N^3M^2)$.

In practice, this is not the case. First, the GeneModel is not particularly dense. The number of types of adjacent states is usually only 1 or 2 and can be at most 6 for the GHMM in Figure 3.2. Further, the Signal pre-processing stage allows only legal nodes in the GeneGraph, which is a small fraction of all possible state/position pairs in a fully connected directed acyclic graph. Thus, immediately a N term and a M term are practically eliminated. Four additional efficiencies are employed to ensure manageable running time: short-circuiting based on in-frame stop codons, length restrictions, cumulative scoring, and modifications to the model topology.

4.2.1 In-frame stop codon short-circuiting

The values that are promulgated through the DFS and memoized at each Transition are real valued probabilities between 0 and 1 that are internally represented as logarithms. A special math class, "Preal", is used for all numerical operations and the standard arithmetic operators are overridden so that user knowledge of the internal representation of the numeric values is not required. The Preal class contains a special constant to indicate whether the variable has a valid value. All scores stored in the memoization data structure are initialized as invalid, allowing the DFS to proceed without requiring a separate "processed" flag. TransitionTypes hold Preals to describe the transition probabilities between states and MetaContent functions return Preal values corresponding to scores of arbitrary regions. The Preal class has the benefits of fast multiplication and avoidance of overflow or underflow when using very large or small numbers, a common problem in probabilistic model calculations.

In an obvious implementation of likelihood model scoring, any sequence that does not meet basic criteria returns a likelihood value of zero. However, it is convenient to use the special invalid Preal value instead of zero in certain circumstances such as in the presence of an in-frame stop codon in a putative coding region. A special rule is added to the DFS graph traversal routine such that any likelihood model that returns an invalid value will cause the function to short-circuit, not scoring any other *longer* regions of the same type from the same node. Since stop codons occur randomly in non-coding DNA, most arcs of sufficient length in the GeneGraph between putative 3' and 5' splice site nodes contain stop codons. Once the first arc containing a stop codon is identified, then all longer arcs are skipped. Thus, the biological requirement of an open reading frame keeps the scoring of variable-length coding exon features under control.

The same short-circuit technique is used to enforce Kozak's Rule in the 5' UTR likelihood model as described in Section 6.6.

4.2.2 Length restrictions

All multi-symbol likelihood models have a corresponding length distribution histogram read from an external parameter file. No attempt is made to score regions greater than the maximum length value stored in the histogram. For the variable-length UTR likelihood model (where stop codons are not relevant), the maximum exon length from the histogram caps the number of possible scored arcs, reducing the overall running time for such sensors to be on the order of the maximum allowed length instead of the sequence length. Typical input sequences are on the order of 100,000s of nucleotides and length restrictions are typically 10,000–20,000 nucleotides.

4.2.3 Cumulative scoring

With a DFS implementation, each likelihood model function is called in an unpredictable order inhibiting scoring optimizations that could be achieved by a predictable call order. But most likelihood models include Sensors that score every possible base in the region allowing for a simple and well-known tiling optimization by storing cumulative scores. For example, since a codon is theoretically possible at every position in the sequence, a preprocessing step is employed such that the likelihood of every possible codon is computed and the cumulative codon scores are stored in an array C where $C_i = C_{i-3}\mathsf{codonScore}(i)$. Thus, the codon model score between any positions i and j (where $(j - i)\%3 = 0$) is simply computed as $C_j - C_i$. The one complication is that stop codons disrupt the computation, since their score is zero. The solution is to reset the cumulative scores after each stop codon and create an additional array that reports the nearest upstream in-frame stop codon at each position i. These pre-processed cumulative scores along with the stop codon information is generated in $O(N)$ in time and space and reduces the time to compute a score for a region to $O(1)$.

4.2.4 Model topology

A careful review of the graph in Figure 3.2 shows that the intron and intergenic states are loops, yet the GHMM approach allows for multi-symbol states. Observe, however, in Figure 2.6 that the length distribution of introns is roughly geometric. Thus, an appropriate transition probability for a single symbol state can adequately approximate the intron length distribution. Similarly, intergenic regions have no real length distribution, and so a large self-loop probability very close to 1 is sufficient. Since the large number of possible introns and intergenic regions dominates the search space, the significant advantage to this choice of model is that the time to score intron and intergenic arcs is reduced to constant time. By contrast, for a multi-symbol intron likelihood model, the arcs are essentially unbounded in length, and the number of candidate 5' and 3' splice node pairs is enormous. The running time improvement by creating single-symbol arcs for introns and intergenic states is offset by an increase in space to maintain additional state: six intron nodes (for frame and stop codon state management) and an intergenic node, composed of score and path variables, must be stored for every position in the sequence. While the space and time upper-bounds do not change, the practical result is a significant speed improvement at the expense of additional RAM.

These four optimizations reduce the running time by reducing the number of nodes, reducing the number of arcs to score, and reducing the time to score an arc. If the number of arcs to score is relatively small, the number of adjacent states is constant, and the time to score arcs is constant, then the overall running time should be about $O(NM)$. In Figure 4.3, the running time is, indeed, linear in the size of the input, requiring about 1 second per 10,000 input bases during the DFS.

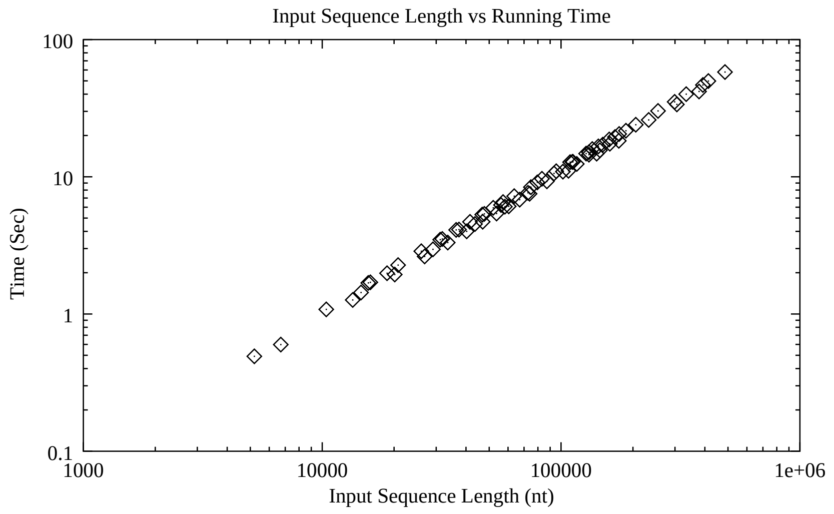


Figure 4.3: Practical running time of depth-first search. The running time of only the depth-first search was measured. Input sequence pre-processing was not included. To generate a broad selection of lengths, five sequences of lengths approximately 5, 7, 15, 16, and 19 thousand bases were chosen and each DNA sequence was concatenated with itself 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, and 25 times. Tests were performed on a 1Ghz Pentium running the GNU/Linux OS with 1GB RAM.

4.3 MetaContent: content scoring

In the software implementation, the class providing abstraction of the content likelihood model is called "MetaContent". Each likelihood model can be implemented in an arbitrary way as long as it can return a score for any requested region. Typically a likelihood model integrates evidence contributed from multiple sources and estimates a likelihood of a subsequence from the combined information. To accomplish this, the MetaContent class uses a simple dynamic program to combine the independent scores of multiple sub-components called Sensors. For example, it is convenient to use a single coding exon MetaContent function to score a candidate region, but its implementation might use a combination of a codon model, a start codon model, a repeat model, a protein homology model, and a cDNA constraint model. Figure 4.4 shows a simple example. Each one of these separate Sensors, like the rest of the system parameters, are specified at run-time.

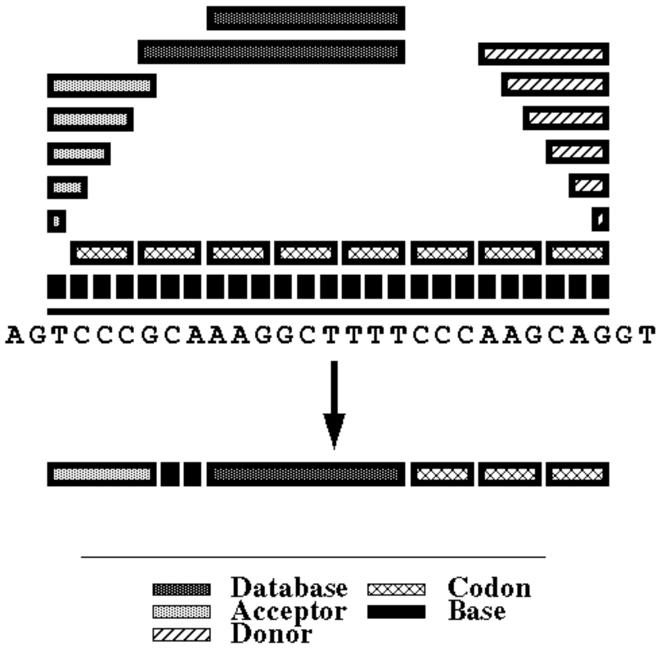


Figure 4.4: Example of content sensor scoring using dynamic programming. The best scoring set of tiled values is chosen as the likelihood for a region.

The MetaContent class provides a means of combining multiple sources of information—but only a subset of the data is relevant for any sub-sequence in question. For example, a protein homology Sensor may provide scores for a region with Blastx similarity, but have no data for other regions. In general, each individual Sensor provides incomplete scoring information for an entire region, and the MetaContent sensor must efficiently obtain the relevant pieces for the region of interest. To accomplish this, a modification of a red/back interval tree is used [29]. In this implementation, both minimum and maximum values are stored at each node in the tree to allow iteration over all overlapping regions in $O(\ln(N)+k)$, where k is the number of overlapping regions in the query interval.

4.4 Flexible configuration

The configuration parameters, including the GeneModel topology and transition parameters, Signal types, MetaContent types and their Sensors, are all stored in a common configuration file format. The format is a simple name/value dictionary format that allows for nested values. The C pre-processor is used at program initiation to filter the configuration allowing for simple selection of alternative configurations. For example, a user can choose a single-gene versus a multiple-gene configuration or a forward versus both strand analysis by specifying run-time options that correspond to pre-processor macros to modify the model topology. A simple excerpt from the configuration file shows how the GHMM model topology is specified at run-time:

```
/********** START ('ATG') **********/
        Start_Node : {
            Signal : { \Start }
            Nodes  : { \Donor_Node \End_Node \PartialR_Node }
            Arcs   : { \Initial_Exon \Single_Exon \Initial_Exon }
            Prob   : [ ONE_LESS_SP_PP SINGLE_P PARTIAL_P ]
            FrameMatch : 0
        }

/********** DONOR ( -3 from 'GT') **********/
/* FAN OUT TO 6 DIFFERENT STATES ACROSS INTRON TO ACCOUNT FOR INTRON
   SPANNING STOP CODONS AND STATE.  */

        Donor_Node : {
            Signal : { \Donor }
            Nodes  : { \IntronL_Node_0
                        \IntronL_Node_1_x \IntronL_Node_1_T
```

```
                  \IntronL_Node_2_x \IntronL_Node_2_TA
                  \IntronL_Node_2_TG }
    Arcs   : { \Donor_Site \Donor_Site \Donor_Site
                  \Donor_Site \Donor_Site \Donor_Site }
    Prob   : [ 0.167 0.167 0.167 0.167 0.167 0.167 ]
}
```

In this example, the major grouping is a TransitionType (state) showing two such states: Start_Node and Donor_Node. The Signal object used in the pre-processing step is named, and the graph connectivity is described. For example, the Start_Node connects to the Donor_Node, the arc between the two nodes is the MetaContent named Initial_Exon, and the probability of traversing that arc is ONE_LESS_SP_PP (a pre-processor macro defined elsewhere).

A convenient feature of the object-oriented class design and the nested configuration is that implementations of sensors need not worry about the reverse strand—a consistent source of programming errors in DNA sequence analysis. Instead, a simple Sensor called RevComp provides a reverse complemented sequence to a Sensor that it contains and performs the coordinate transformations. Similarly a special Signal called RevSig can contain any other Signal. For example, one type of Sensor used for start and stop codons is the TripletMatch subclass. It matches one or more 3-symbol sequences and returns a fixed probability. In this case, the MetaContent is a 3-symbol fixed-length state that only contains the stop codon Sensor. While I emphasize that this example is overly complex for just Stop codon scoring, it succinctly demonstrates the flexibility of the run-time configuration for more elaborate modeling requirements.

Here is an excerpt from the standard configuration file:

```
Stop : {
        Class : TripletMatch
        Triplets : [ "TAA" "TAG" "TGA" ]
        Probs : [ 0.57 0.18 0.25 ]
}
RStop : {
        Class : RevComp
        Contains : {
                \Stop
        }
}
RStopContent : {
        Components : {
                \RStop
        }
        MinLength : 3
        MaxLength : 3
        Frame : None
}
```

In this case, reading the configuration from bottom to top, the MetaContent "RStopContent" contains one Sensor component "RStop". That component is implemented by the C++ class "RevComp" (loaded via dynamic library linking), which in turn contains the Sensor "Stop". The "Stop" class is implemented by the C++ class "TripletMatch" (again, dynamically loaded by name). The parameters associated with each class are completely dynamic. In this case, the Sensor expects two parameters named "Triplets" and "Probs", which are automatically provided to it via the bootstrapping loader from this configuration.

In addition to flexible run-time configuration and dynamic loading, perhaps the greatest flexibility is provided through the mechanism for triggering external program actions. Any Sensor can specify in its configuration that a system call be performed. Special environment variables are provided to the program that specify the location of the input sequence and any other files provided to the Genie program at run-time. As a result, rapid prototyping of stand-alone Sensors are made possible. Often sensors are just Perl scripts that might perform a simple data transformation such as converting a RepeatMasker or Blastx result file into likelihood model scores. To make the process even simpler, external scripts can output results in the standard GFF format and a Sensor subclass called "GFFSensor" can process this format. Thus, adding additional sensors is a relatively simple process of writing an external script that takes as input a DNA sequence and outputs a GFF file. The rest of the integration is handled by the Genie software system.

4.5 Weakness of graph traversal

One weakness of the DFS graph traversal is that the recursive algorithm is limited by the maximum stack size provided by the operating system. Typically the operating system's stack size is about 65000 and the stack will be exceeded during a recursive traversal along a series of single-base arcs, i.e. a long intergenic or intronic region. To work around this limitation, the DFS routine is repeatedly called at intervals of, arbitrarily, 10000 bases, e.g. from bases $N - 10000 \ldots N$, $N - 20000 \ldots N$, $\ldots$, $1 \ldots N$. Because of the model topology, it is impossible to exceed the stack size as windows grow larger because of memoized results stored in previous calls for smaller windows. However, the final Viterbi path can no longer be reported recursively. Instead, the tail recursion is replaced by a loop.

4.6 Annotating large sequences

When this research began in 1995, only a few annotated DNA contigs existed in the database and most sequences were less than 50,000 nucleotides long. Unannotated BAC sequences on the order of 150,000 bases were just appearing. Today, the genome projects have released gapless contigs on the order of tens of megabases. The Genie program requires about 1000 bytes per base of input sequence, depending on complexity of model, size of floating point numbers, and the algorithm (Viterbi or forward/backward). Sequences of up to 200,000 bases can be comfortably annotated using memory capacities typical of current workstations, but larger sequences require a windowed approach. For large sequences, the input is first fragmented into smaller overlapping pieces (110,000 bases with 10,000 bases of overlap is typical). The Genie software processes each piece and a post-processing program combines the results. For conflicting predictions in the overlapping regions, the software first determines a gene region based on the conflicting, overlapping predictions, extending the region by 1000 bases or halfway to the next adjacent prediction, whichever is shorter. The new DNA region is extracted and a new set of predictions are generated by recalling the Genie software for the conflict region. Generally, this method is satisfactory for large sequence annotation efforts. The only serious risk is that an intron will completely include the overlapping region. If this occurs, then it is likely that the software will produce two disjoint gene predictions in adjacent fragments and no conflict will be identified.

Chapter 5

Constrained Systems

5.1 Protein matches

The system so far described is based purely on simple frequency-derived statistics of gene features. Homology to other known protein-coding regions is another method for finding genes in DNA[30]. In this work, protein alignments are treated simply as an additional likelihood sensor for coding-exon arcs.

Database homology raises the problem of assigning a “fair score” for a match relative to other scoring Sensors such as the codon Sensor. My solution is to generate a likelihood score from a theoretical encoding bit-cost of the DNA subsequence match given the protein database match. Let the estimated encoding cost of the homologous DNA be the sum of the encoding cost to represent the offset into the database and the translation cost.

The offset is described as the encoding cost for uniquely specifying where the match in the database is located. If the number of starting positions for matches in the database is D and assuming that all positions are uniformly likely, then the cost of encoding the offset of the subject in the database is $-\log(1/D) = \log(D)$. The translation cost of the target is determined using a substitution matrix to translate from the subject to the target.

This method was prototyped using both the BLOCKS database[31] and Blastx queries against protein databases. The BLOCKS database is a collection of over 2000 highly conserved protein motifs without insertions or deletions. I derived a profile for each motif using a nine-component Dirichlet mixture[32]. The translation cost of a target nucleotide sequence given a BLOCKS homology is the combined cost of encoding the target protein product using the BLOCKS motif profile and the cost of translating the target protein product from an amino acid sequence into a nucleotide sequence. The profile cost is simply the product of the probability of each residue in each column. The protein-to-nucleotide cost is computed using the three phased fifth order Markov chains, but the probability of a codon is normalized to sum to one over those codons that translate to the same amino acid. For Blastx matches, the translation cost is similar to that used for the BLOCKS database, but the BLOSUM-62 substitution matrix is used instead of the motif profile. A Blastx threshold of $E = 1$ was used in the searches, allowing very weak matches in favorable contexts to contribute to coding exon prediction. Results are shown in Section 6.7.1.

5.2 Perfect alignments

Perfect or near-perfect alignments can provide the true "answer" to the gene finding problem. If two Blastx homology matches are found from a single protein sequence and the two peptide subsequences are adjacent, then one can conclude that the "match pair" implies an insertion of protein-coding or non-coding nucleotides. If the insertion is of non-coding nucleotides, then a pair of splice sites can be inferred from a pair of homology matches. To reduce the chance that a match pair is the result of an insertion of coding nucleotides, I only consider those match pairs with suitable splice site patterns at the match boundaries. This constraint was achieved in the Genie system by forcing all intron region scores to zero that did not meet the match pair constraint[33]. Similar logic can be applied to mRNA alignments from full-length cDNAs or single read ESTs. If the mRNA aligns perfectly and flanks a putative pair of splice sites, then the intron is obviously revealed.

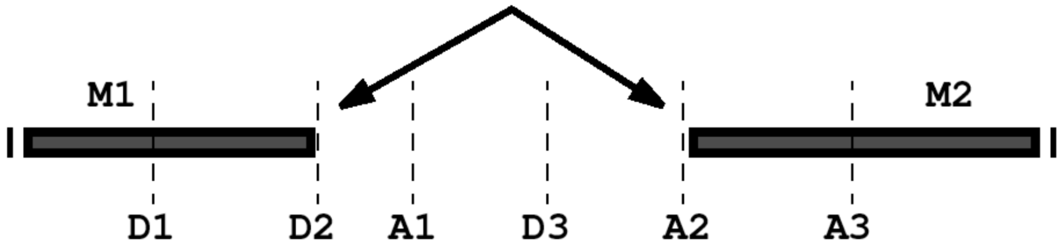


Figure 5.1: Match pair constraints. A match pair, M_1 and M_2, imply a pair of splice sites, D_2 and A_2. The DFS is constrained such that D_2 is used in the final parse if and only if A_2 is used.

The simplicity of this constraint system is due to the multi-symbol intron states; any intron that is flanked by one but not both match pair splice sites was not scored. However, with the current model topology (Figure 3.2), intron states emit only a single symbol to allow for efficient running time (Section 4.2.4). As a result, it is not possible to consider jointly the flanking splice sites for a candidate intron. A more general method is needed for constraining parses using perfect alignment information. Two approaches were implemented: an interactive graph control method and a sensor scoring method.

5.2.1 Interactive graph control

One means of constraining the system is to modify the graph traversal such that only those parts of the induced graph with unknown labeling are considered. For example, if a cDNA alignment irrefutably shows a forward strand intron located in positions $i \dots j$, then an obvious solution is to partition the analysis into two segments flanking the constraint. For the right side, a DFS is performed from position j in the 3' splice site state to the final node and similarly a second DFS is performed from position i in the 5' splice site state in reverse to the start node. The Viterbi results and constraint information are concatenated for a final sequence annotation.

This approach has the additional benefit that the likelihood of any region can be easily returned to the user via the forward/backward algorithm. In the example above, it is possible to both determine the best parse including an intron from $i \dots j$ and to report the likelihood of the intron. One can imagine a graphical interactive program in which a user could experiment with different exon structures and receive information from the gene finding system regarding the likelihood of the current configuration.

A simple scripting language was implemented for the Genie software to achieve interactive control of the GeneGraph. The script language allows a user to specify the sequence, build the GeneGraph, perform the Viterbi, forward, and backward on the entire sequence and report the results for fragments of the graph. For example:

```
# initialize
M=MODEL ModelName
DATA M SEQUENCE
>DNA
ctgcgtactaagacccgtgtgcagcagcggcggcggcggtagaggcggcggcggcggcgg
...
.
G=GRAPH M
BUILD G 0 4862

# perform viterbi and forward in both directions
VITERBI G Begin_Node 0 0 Donor_Node 2045 0
SUM G Begin_Node 0 0 Final_Node 4862 0

VITERBI G Final_Node 4862 0 Begin_Node 0 0
SUM G Final_Node 4862 0 Begin_Node 0 0

# get best parse that includes intron from i..j
REPORT G Donor_Node i 0 Begin_Node 0 0
REPORT G Acceptor_Node j 0 Final_Node 4862 0

# try shifting the donor site to k
REPORT G Donor_Node k 0 Begin_Node 0 0
QUIT
```

While such a system provides maximum flexibility, no automated front-end driver for the system has been developed. There is only limited utility in practical applications because it becomes very difficult to provide multiple constraints manually in different segments of the query sequence and then merge the results. Separate induced gene graphs must be generated for each segment and manual bookkeeping becomes too difficult. The user needs to recognize that sometimes multiple parses must be considered; for example, if the parse begins or ends in an intron state, the user must consider all six possible intron states. Tying this scripting system to an automated graphical genome annotation tool would be an interesting extension to this work, but it would be a complicated challenge because the front-end requires specific knowledge of the underlying GeneModel.

5.2.2 Sensor score constraints

A simple-minded alternative to the graph control method is to provide the program with artificial false scores for regions of interest. The key feature to exploit is the convention in the Genie system to prohibit a prediction of any region that includes an invalid score generated by any of the Sensors in the corresponding MetaContent likelihood model.

A program was written to convert constraints specified in terms of commonly understood feature names into sensor scores acceptable by the Genie system.

For example, to force an intron between position $100\ldots200$, a user simply creates a GFF file of the form:

```
# GFF
MYSEQ   99    100   splice5   0    +    .
MYSEQ   100   200   intron    0    +    .
MYSEQ   200   201   splice3   0    +    .
```

These constraint lines are translated into a set of Sensor scores. For example, the Sensor scores for the intron base model will be set to one for the region from $i\ldots j$, and the codon model scores will be set to invalid for the same region, inhibiting any prediction of coding region in the intron. Similarly, the system will make the intron score at $i-1$ and $j+1$ invalid to ensure the designated splice sites are chosen.

It is relatively straightforward to generate false sensor scores automatically from a set of user-specified constraints. More than one non-overlapping constraint can be specified and the constraints can be scattered throughout the sequence. The end result is a gene structure prediction that includes the desired constraints and augments the annotations using the standard gene finding methods in the flanking regions.

The weakness of this approach is as obvious as it is simple. It is not possible to obtain scores for the constrained regions, if desired, and it is not possible to create more subtle weighting systems based on less than perfect alignments. Despite these weaknesses, this approach proves to be quite useful in whole genome annotation. Typically a user is not interested in the academic question of the efficacy of gene finding systems, but rather wishes that all “known” genes, i.e. those with cDNA evidence, be reported exactly as it is known and the remainder of the genome be annotated as best as possible.

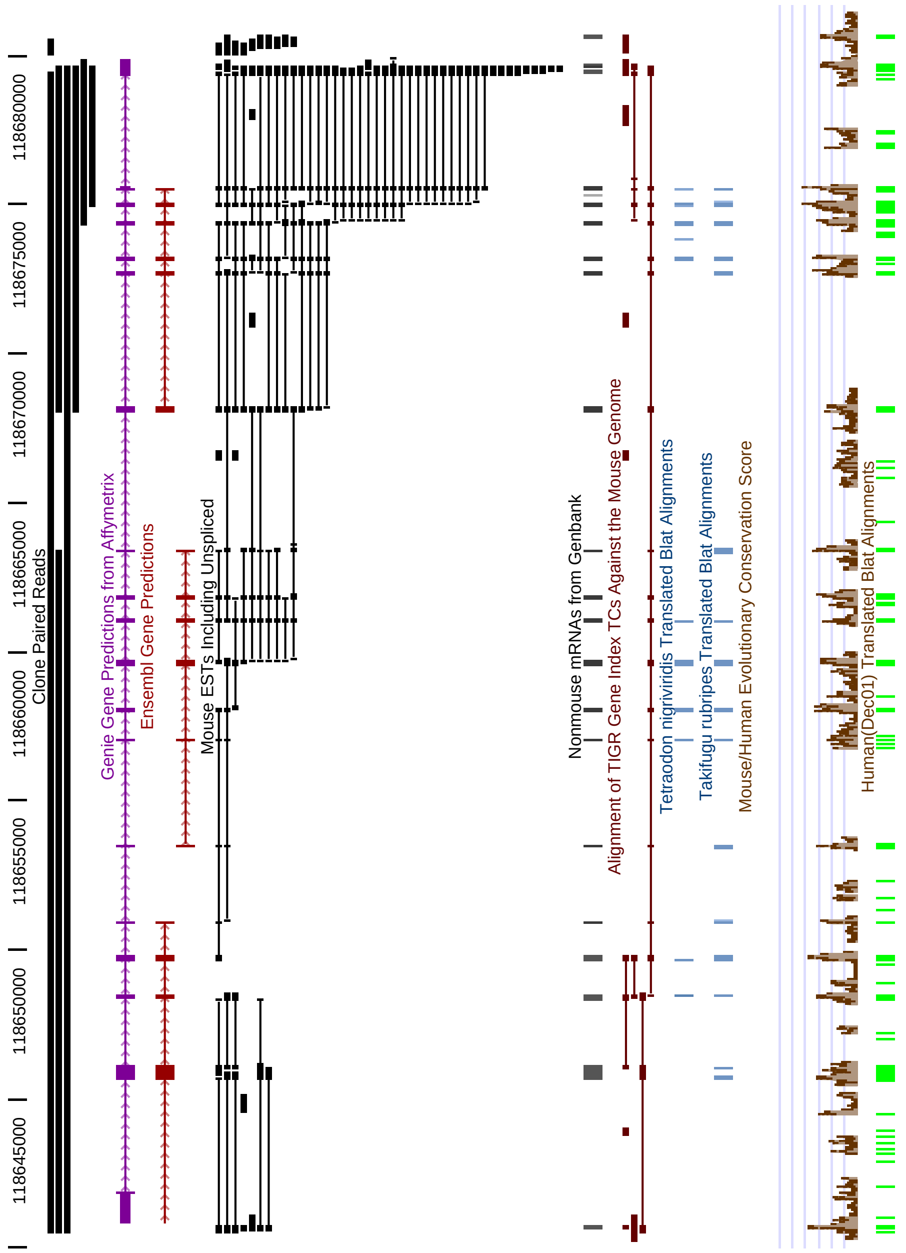

118645000
118650000
118655000
118660000
118665000
118670000
118675000
118680000
Clone Paired Reads
Genie Gene Predictions from Affymetrix
Ensembl Gene Predictions
Mouse ESTs Including Unspliced
Nonmouse mRNAs from Genbank
Alignment of TIGR Gene Index TCs Against the Mouse Genome
Tetraodon nigriviridis Translated Blat Alignments
Takifugu rubripes Translated Blat Alignments
Mouse/Human Evolutionary Conservation Score
Human(Dec01) Translated Blat Alignments

Figure 5.2: Example of perfect alignment mRNA constraints. This region is taken from chromosome 7 of the mouse genome at `http://genome.ucsc.edu/`. Note, that the majority of the gene structure is determined by spliced EST alignments. The top-most set of bars represent clone bounds, i.e. extents along the genome delineating a region between the aligned 5' and 3' read pairs from a common clone. The gene prediction (second track, in purple), uses the sensor score constraint system to ensure that a single gene is predicted along the clone bounds and that the exon-intron structures implied by the EST alignments are respected.

Chapter 6

Methods and Results

6.1 Sensors, signals, length distributions, and transition probabilities

Each MetaContent in the Genie system scores an arc between two states and a MetaContent contains one or more Sensors for scoring a region. Signals pre-process the input sequence to identify potential state/position nodes in the GeneGraph (see Chapter 4). Each Sensor is trained as a separate likelihood model, using frequency counts to estimate probabilities. Parameters for Sensors from older versions using neural networks were estimated using training sets in which each nucleotide position was encoded as 4 binary inputs. Some Sensors are shared among multiple MetaContents, analogous to parameter tying used in other HMM applications. Neural net Signals were trained on example sites with nearby decoy sites used as negative training examples.

The Sensors and Signals in Table 6.1 and 6.2 are used in the Genie system.

Different Sensors and Signals are used in different implementations of the system and for different organisms. Subsequent results will refer to these version numbers to indicate the configuration used. Version 3 and 5 include all components from version 2. Version 4 includes all components from version 3.

Component	Description	Arc	Version
Non-coding Base	A single Nth order Markov chain.	Intergenic, 3'UTR, 5'UTR	2,3,4,5
Codon	Three-phased Nth order Markov chain.	Initial, Internal, Final, and Single Coding Exon	2,3,4,5
Intron Base	A single Nth order Markov chain.	Intron	2,3,4,5
Start Codon	A fixed likelihood of 1 when matching ATG.	Initial, Single Coding Exon	2,3,4,5
Donor Tract	An order-1 Markov profile covering the last 3 coding bases and the first 5 intronic bases	5' Splice Site	2,3,4,5
Pyrimidine Tract	An order-1 Markov profile covering the last 20 intronic bases and the first 3 coding bases	3' Splice Site	2,3,4,5
Branch Tract	An order-2 Markov profile with average frequencies from neighboring bases at + and -1, as in Burge[7]	3' Splice Site	2,3,4,5
Stop Codon	A fixed likelihood based on frequencies of 3 stop codons	Single, Final Coding Exon	2,3,4,5
Constraints	Externally provided likelihoods (1 or invalid)	All Sensors	2,3,4,5
Start Profile	An order-0 Markov profile covering the last 7 bases of the 5' UTR and covering the start codon	5' UTR	5

Table 6.1: Genie sensors

Component	Description	Arc	Version
Stop Profile	An order-0 Markov profile covering the stop codon and subsequent 7 bases into the 3' UTR	3' UTR	5
Promoter Tract	Promoter Site neural network converted to likelihood using Bayes rule.	5' UTR	4
No-Start-Codon	Scores a special invalid for any ATG. Enforces Kozak's Rule([1]).	5' UTR	4
Non-coding Base	300nt windowed frequency	Intergenic, 3'UTR, 5'UTR	1
Codon	Two-layer neural network with 17 hidden units. Inputs of G+C (300 nt window) and previous codon.	Initial, Internal, Final, and Single Coding Exon	1
Intron	300nt windowed frequency	Intron	1
Donor Tract	Donor site neural network converted to likelihood using Bayes rule. (see [27])	5' Splice Site	1
Acceptor Tract	Acceptor site neural network converted to likelihood using Bayes rule. (see [27])	3' Splice Site	1
Codon	Coding likelihood from protein homology	Initial, Internal, Final, and Single Coding Exon	1

Table 6.1: Genie sensors

Signal	Description	Version
Any base	All positions	2,3,4,5
Donor Site	GT	2,3,4,5
Acceptor Site	AG	2,3,4,5
Start Site	All ATG	2,3,4,5
Start Site	Beginning of ORF $> 20nt$	2,3,4,5

Table 6.2: Genie signals

Signal	Description	Version
Stop Site	TAG,TGA,TAA	1,2,3,4,5
Donor Site	Two-layer neural network with 50 hidden units. Inputs of $-7 \ldots +8$ bases around site.	1
Acceptor Site	Two-layer neural network with 40 hidden units. Inputs of $-21 \ldots +20$ bases around site.	1
Start Site	Neural network. Inputs of $-10 \ldots -1$ bases before site.	1,2
Promoter Site	Time-delay neural network. (See [10])	4
Polyadenylation Site	3' aligned end of 3' EST or mRNA	3,4
Promoter Site	5' aligned end of 5' EST or mRNA	3,4
Donor Site	Site of EST or mRNA spliced alignment	3,4
Acceptor Site	Site of EST or mRNA spliced alignment	3,4

Table 6.2: Genie signals

Smoothed histograms are used to represent the length distributions for exons. Length frequencies are tallied from annotated exons and smoothed by averaging over a sliding window of 150 nt for the single coding exon and 100 nt for other coding exon. Most state transitions are measured from the training data according to the frequencies of features. For example, transition probabilities from an intron into the internal versus the final exon state are based on the counts of internal exons in the training data; intron-intron transition probabilities are estimated according to the average intron size.

While the model accommodates 5' and 3' UTR, it is important to note that these sections of the model are essentially inactive during the standard *ab initio* prediction. That is, unless there is additional evidence provided for transcription outside of the translated area or evidence of a promoter or polyadenylation site, then no UTR will be predicted.

6.2 Performance metrics

Performance metrics are those used by Burset and Guigo[34] with the addition of exact CDS structure identification (sometimes erroneously referred to as an exact gene prediction metric). Sensitivity or Recall (Sn) refers to the fraction of correctly predicted coding bases out of the total set of annotated coding bases. Specificity or Precision (Sp) refers to the fraction of correctly predicted coding bases out of the total set of predicted coding bases. $1 - Sp$ is the false positive rate. Exact exon measures refer to the prediction of a coding exon with the correct flanking splice sites or start and stop codons. Thus, Sn and Sp are the fractions of correctly predicted exons out of the set of annotated and predicted exons, respectively. A "wrong" exon is identified as an exon that does not overlap any annotated exon. The "wrong exon" metric is the fraction of wrong exons out of the total set of predicted exons. A "missing" exon is an annotated exon that does not overlap any predicted exon. The "missing exon" metric is the fraction of missing exons out of the total set of wrong exons. Exon Sn is inversely correlated with missing exons, and exon Sp is inversely correlated with

	Method:	Genie
Per Base	Sn	0.91
	Sp	0.98
Exact Exon	Sn	0.78
	Sp	0.84
	Missing	0.12
	Wrong	0.05
Exact CDS	Sn	0.40
	Sp	0.39
	Missing	0.07
	Wrong	0.06

Table 6.3: *Ab initio* performance for *Chlamydomonas*. (Genie version 2.)

wrong exons. Finally, a set of exact CDS metrics are used in the same manner as the exact exon metrics. CDS refers to the entire coding region from start to stop codon, inclusive.

6.3 Basic *ab initio* results

To assess the simple gene finder performance, a set of 60 curated protein-coding genes for *Chlamydomonas reinhardtii* in 59 DNA sequences were selected from Genbank DNA records. All sequences were pairwise aligned using Blastn and those sequences with greater than 50% similarity were eliminated. Only sequences with consistent gene structure accepted by the Genie grammar were allowed in a manner similar to other reference data sets previously generated (see, e.g.,[27]). An additional set of coding regions from 51 unrelated mRNAs from Genbank were used for training only. Using a hold-one-out cross validation on the 59 sequences resulted in performance metrics shown in Table 6.3.

A 2.9Mb region of the *Drosophila* genome containing the *Adh* locus was manually curated using unpublished cDNA sequences. Reese, et. al. collected gene predictions from five different *ab initio* gene finding systems and performed an independent assessment[35]. The assessors published a training set of 416 DNA sequences with annotated protein-coding exon structures. An additional set of coding regions from Drosophila mRNAs augmented the training set. For the published assessment, two data sets were used, called "std1" and "std3", where "std1" represented only full-length mRNAs and "std3" was a larger set based, circularly, on gene finder results and other automated, predictive evidence. "std1" was used to measure *Sn*-related metrics and "std3" for *Sp*-related metrics. For more discussion, see Reese [35].

Results are show in Figure 6.4. Detailed discussion of the successes and failures of the Genie method are documented by Reese [36]. In this test case, it is noteworthy that 40% of CDS were predicted correctly because no better performance is known to have been published for higher eukaryotes.

6.4 Learning ability

The likelihood models used in this gene finder depend on a data set of sufficient size to properly estimate true probabilities from frequencies. Each training sample—a DNA sequence with a single gene—typically provides about 1000 examples for each of the three phased coding Markov chains, several thousand intronic base examples, and 100s to 10s of thousands of intergenic base examples. With 1024 pa-

	Method:	Fgenes$_1$	Fgenes$_2$	Fgenes$_3$	GeneID$_1$	GeneID$_2$
Per Base	Sn(1)	0.89	0.49	0.93	0.48	0.86
	Sp(3)	0.77	0.86	0.60	0.84	0.83
Exact Exon	Sn(1)	0.65	0.44	0.75	0.27	0.58
	Sp(3)	0.49	0.68	0.24	0.29	0.34
	Missing(1)	0.11	0.46	0.06	0.54	0.21
	Wrong(3)	0.32	0.17	0.53	0.48	0.47
Exact CDS	Sn(1)	0.30	0.09	0.37	0.02	0.26
	Sp(3)	0.27	0.18	0.10	0.05	0.10
	Missing(1)	0.09	0.35	0.09	0.44	0.14
	Wrong(3)	0.24	0.25	0.52	0.22	0.31

	Method:	Genie	HMMGene	Grail
Per Base	Sn(1)	0.96	0.97	0.81
	Sp(3)	0.92	0.91	0.86
Exact Exon	Sn(1)	0.70	0.68	0.42
	Sp(3)	0.57	0.53	0.41
	Missing(1)	0.08	0.05	0.24
	Wrong(3)	0.17	0.20	0.29
Exact CDS	Sn(1)	0.40	0.35	0.14
	Sp(3)	0.29	0.30	0.12
	Missing(1)	0.05	0.07	0.16
	Wrong(3)	0.11	0.15	0.24

Table 6.4: *Ab initio* performance for the *Adh* locus in *Drosophila*. "1" refers to the test set "std1" and "3" refers to "std3". The division into two tables is arbitrary to fit onto the page. (Genie version 2.)

rameters per 5th-order Markov chain, one would expect that several sequences would be required to begin to estimate the Markov chain parameters accurately. By contrast, only a few splice site examples are observed per training sequence, so those likelihood models, although having fewer parameters, are expected to be more difficult to estimate.

To understand the learning rate of the system, the 59 sequences from the *Chlamydomonas* data set (Section 6.3) were used to create training sets of varying sizes. For each test sequence, a training set was created by randomly sampling from the available training samples, testing the performance, and then repeating the sampling, growing the training set by one sample sequence. Thus, for each training set size there were 59 separate tests, one for each test sequence.

Figure 6.1 shows the results of the tests in terms of base level *Sn* and *Sp*. As the number of training samples increases, the performance rises steeply and plateaus. If the splice site recognition played a significant role, then one might expect to see continued improvement at the exon level, but this is not the case. The result suggests that the majority of signal necessary for gene finding is found in the coding potential and syntactic restrictions of a valid parse.

6.5 Feature scoring

The forward/backward algorithm reports the likelihood for all labels as the sum of the likelihoods of all parses of the sequence that contain that label. If the

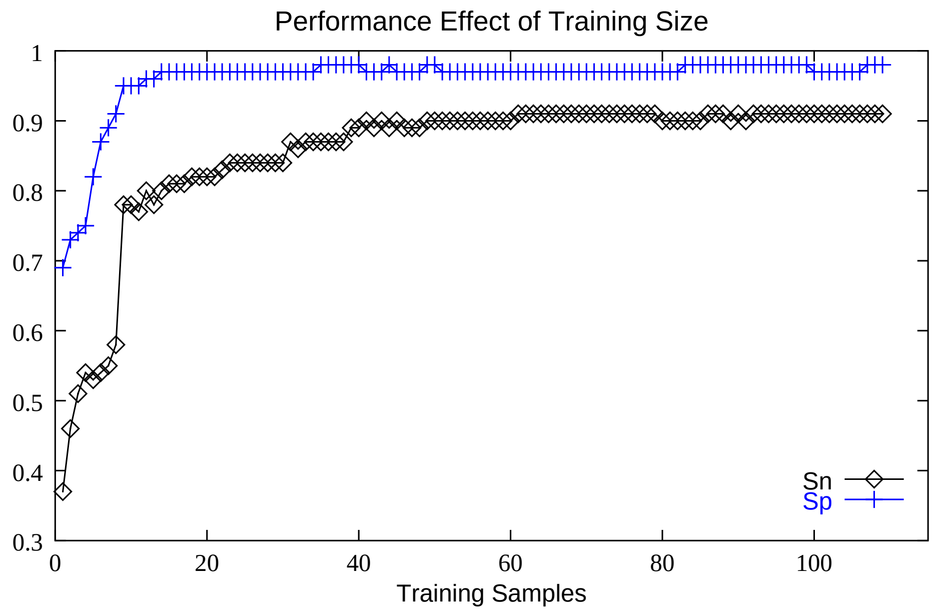


Figure 6.1: Effect of training size. Each point represents the cumulative base-level performance statistic for 59 separate sequences with the given number of training examples. (Genie version 2)

GHMM is truly a good model then the forward/backward likelihood scores reflect the true probabilities of each feature. To study this question, the coding exons predicted for all sequences in a hold-one-out cross-validated test using the Chlamydomonas data were ranked by score and compared to the known annotations. If the HMM feature scores are correlated with the true likelihoods, then a ranking of those values with respect to performance metrics should be monotonic and much better than chance. Figure 6.2 shows that the forward/backward likelihood is indeed a legitimate metric of prediction quality.

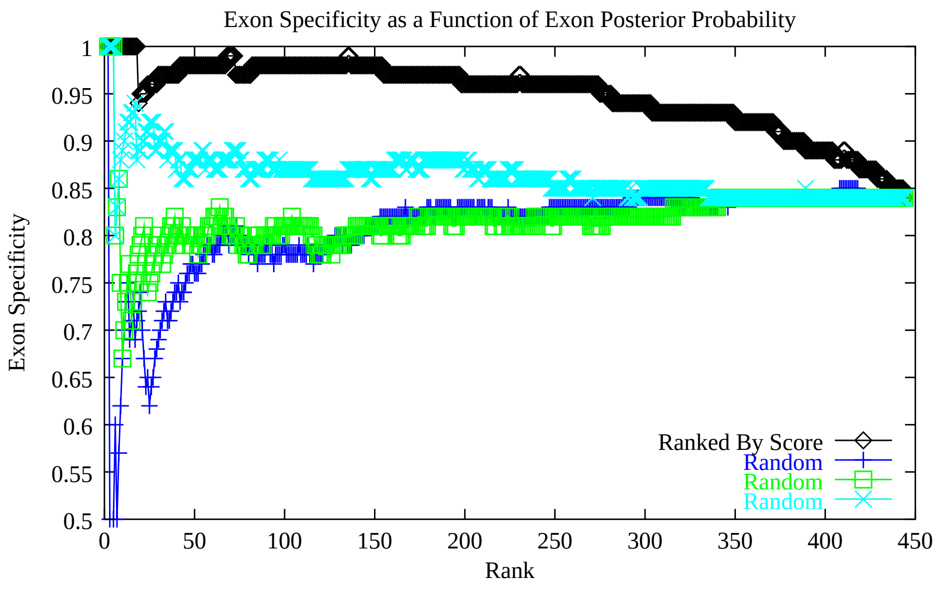


Figure 6.2: Performance of posterior probability score function. The left-most rank represents the one best scoring exon of 443 from 59 Chlamydomonas DNA sequences, the second rank is the two highest scoring, etc., and the rightmost rank represents all predicted coding exons (ties are arbitrarily ordered). Each point is the exon specificity among the set at that rank. The ranking by score is monotonically decreasing and very different from random.

6.6 Model extensions

Section 4.4 describes mechanisms for simple modifications of the gene models. One example of this is the incorporation of a promoter prediction method. Reese[10] describes a method for predicting transcription start sites using time-delay neural networks. Using this method independently to scan an input sequence results in a high false positive rate of about 1:1000 as shown in [37]. The false positive rate can be regulated by requiring the transcription start site to be proximal to a protein-coding gene. To control the distance between a putative transcription start site and the translation start site, Kozak's Rule can be employed. This rule states that the translation start site is the first "ATG" codon following the transcription start site. Such a rule can be easily implemented by modifying the likelihood model for the 5' UTR such that any putative region containing an "ATG" is assigned a probability of zero. Using the short-circuit method described in Section 4.2.1, this is simply implemented in the Genie system by creating a Sensor that returns an invalid value for "ATG". This Sensor is added to the 5' UTR MetaContent model to enforce Kozak's Rule. The short-circuit mechanism automatically ensures that no more distal promoter sites be considered if a closer site fails the rule. To test this configuration, Reese's neural network was added as a Signal for the transition between intergenic and 5' UTR and the Kozak's Rule Sensor was added to the 5' UTR MetaContent model (version 4). In [35], this model was tested against the same *Drosophila Adh* region as above and we found that the false positive rate was reduced to 1:14710, while maintaining a

	Method:	*ab initio*	w/ protein	GeneID+	GeneParser3
Per Base	Sn	0.87	0.95	0.91	0.86
	Sp	0.88	0.91	0.90	0.91
Exact Exon	Sn	0.69	0.77	0.73	0.56
	Sp	0.70	0.74	0.70	0.58
	Missing	0.10	0.04	0.07	0.14
	Wrong	0.15	0.13	0.13	0.09

Table 6.5: Prediction results on 570 protein-coding genes. The NCBI non-redundant protein database (nr) and BLOCKS protein database were used to generate likelihood model scores for homologous coding regions. For fair comparison, only sequences of length less than 8000 nucleotides were considered due to limitations of other gene finders. (Genie version 1)

sensitivity comparable to the best competing promoter-prediction methods. Thus, the context and grammatical constraints improved the accuracy of a component of the model over its independent performance. Moreover, this was achieved through some relatively simple modifications to the run-time configuration.

6.7 Constraint results

6.7.1 Protein homology

Burset and Guigo[34] published a set of vertebrate protein-coding genes. DNA segments with protein homology are scored as described in Section 5.1 and incorporated into the content sensors as described in Section 4.3. Table 6.5 shows the results of *ab initio* Genie, the addition of protein homology matches, and comparisons with two other gene finders that also integrated protein homology.

6.7.2 Fixed sensor score constraints

A set of RefSeq mouse mRNAs was aligned to the draft mouse genome using the BLAT alignment tool[17] (greater than 90% of the query sequence was required to align with at least 90% identity). The CDS regions of the mRNAs were required to fully align, splicing was required to conform to consensus splice sites (including the most common degenerate dinucleotides), and splicing must be strand consistent (i.e. all splice sites must indicate that the mRNA is sense oriented). No gaps were allowed in the CDS alignment that would have caused frame shifts and no sequence errors were allowed that would have caused in-frame stop codons. The resulting exon structure and coding regions were extracted from the mRNA alignments and the DNA sequence including the full-length mRNA plus 1000 flanking bases was extracted into a set of 4414 sequence files. The minimum DNA sequence length was 2339 bases, the maximum was 1.3 million bases, and the median was 14897.

A random 9/10ths of the set of sequences was set aside for training and the remaining was used for testing. To test the performance of a constrained gene finding system, all available ESTs were used, but all mRNAs were excluded since mRNAs were used to create the reference annotations. For each test sequence, all mouse ESTs were aligned and merged into partial transcripts based on common overlapping regions using the AltMerge program[38]. The longest AltMerge transcript of an overlapping set of alternatively spliced transcripts was selected and the largest open reading frame in each AltMerge transcript was annotated as the putative CDS. In addition to the

AltMerge transcript constraints, primary transcript regions were inferred from 5'/3' ESTs pairs sequenced from the same clone. An *ab initio* annotation was performed using the Genie software. (All parameters were derived from the training sets.) A second constrained prediction was performed using the AltMerge transcripts and clone bounds as constraints to Genie. The AltMerge transcript constrains Genie by creating scores of invalid or 1 for sensors corresponding to splice sites, introns, and exonic regions. Similarly, the clone bounds generate invalid scores for the intergenic sensors, effectively forcing a prediction of a primary transcript in that region. Thus, four gene predictions are compared:

1. Reference CDS from high quality mRNA alignment
2. Longest ORF within AltMerge transcripts
3. *ab initio* Genie
4. AltMerge and clone-bound constrained Genie

Sample results are shown in Figure 6.3 and summary statistics in Table 6.6. Due to the large number of mouse ESTs, the ORF method alone is sufficient to identify a significant portion of the coding regions. Using the Genie constraint system improves on the ORF method and sometimes bridges partial transcripts.

It is not surprising that constraints from perfect alignments and clone bounds improves gene prediction performance over *ab initio* predictions. It is, however, surprising, that the ORF method is, in some ways, better than the constrained predictions.

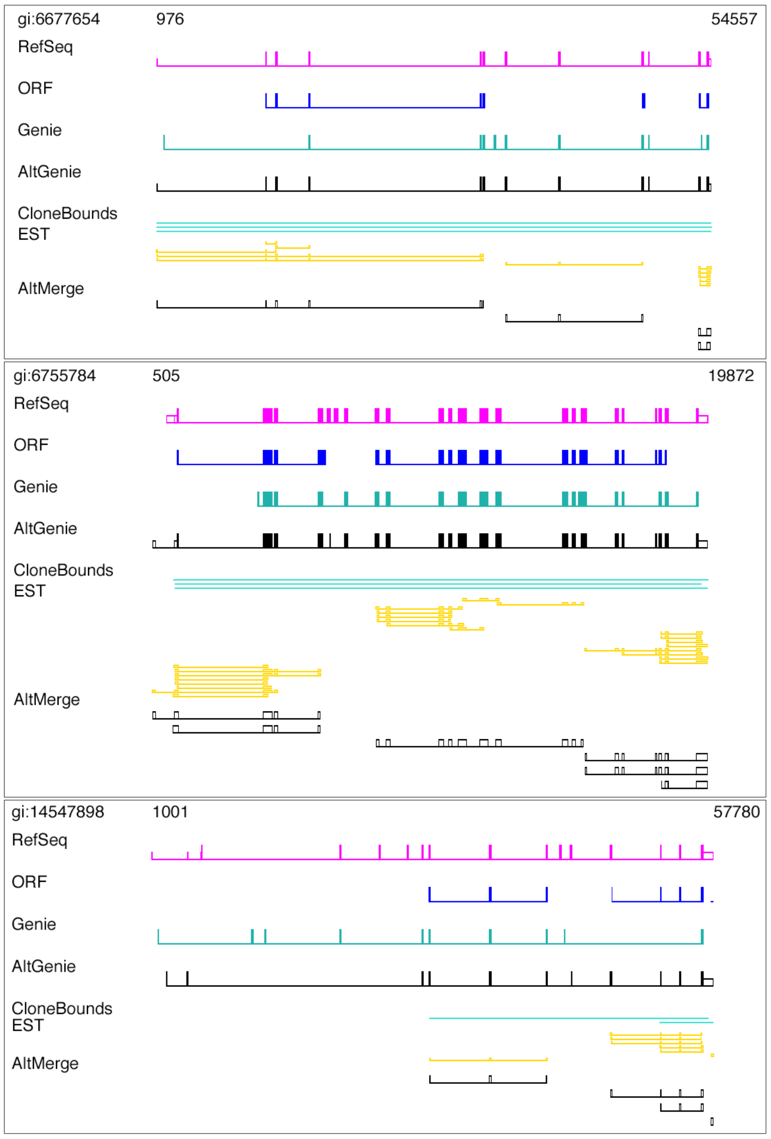


Figure 6.3: Constraint-based prediction results. The reference sequence is a full-length RefSeq mRNA. *Ab initio* predictions are labeled Genie. A set of non-redundant transcripts inferred from the EST alignments are labeled AltMerge. Clone bounds represent extents between 5' and 3' reads from the same clone. The ORF track is the largest open reading frame from the AltMerge transcripts. The track labeled AltGenie is derived from *ab initio* plus fixed constraints from clone bounds and AltMerge.

	Method:	*ab initio*	constrained	ORF
Per Base	Sn	0.86	0.77	0.77
	Sp	0.77	0.91	0.89
Exact Exon	Sn	0.66	0.71	0.70
	Sp	0.63	0.80	0.82
	Missing	0.20	0.21	0.19
	Wrong	0.24	0.11	0.05
Exact CDS	Sn	0.14	0.36	0.39
	Sp	0.14	0.37	0.31
	Missing	0.03	0.13	0.10
	Wrong	0.03	0.10	0.14

Table 6.6: Constrained gene prediction performance. (Genie version 3)

Almost 40% of exact protein-coding CDS structures can be determined using nothing more complicated than an ORF finder on aligned ESTs. Also, when looking at these individual loci, the vast majority have one or more clone bounds describing some or all of the primary transcript region. Since a large number of true genes are covered by clone bounds, one can surmise that many unknown gene regions can be identified by the clone bounds, too.

It is possible that for novel mRNAs, the constrained Genie system would not perform as well because the full-length sequences used for testing tend to be those genes that are heavily expressed. Thus, there are likely to be many ESTs for those genes that are known.

It is interesting that the *ab initio* gene prediction method and the EST assembly method produce such similar results when using completely different information. This suggests that *ab initio* gene predictors can play an important role in annotation when carefully monitored for false positive rates such as in this experiment.

Chapter 7

Whole Genome Gene Finding

7.1 Whole genome experiments

In this chapter I briefly discuss the application of the Genie system to whole genome analysis. In most cases, it is difficult to definitively assess performance, and in some cases, I simply note the availability of the results for completeness. For the fruit fly, human, and mouse whole genome annotations, gene prediction sets were generated by the Genie system but the detailed functional analyses and assessments were performed by collaborators as referenced in each section. Thus, this chapter serves to summarize findings using Genie in whole genome analysis and to describe any implementation details not published elsewhere.

7.1.1 *C. elegans*

A set of genes for the *Caenorhabditis elegans* genome was generated. These genes were predicted using constraints from Jim Kent's Intronerator system for predicting introns from cDNA alignments. The 16,423 gene predictions are available in the on-line *C. elegans* genome browser described by Kent[39] where they can be visually compared with the the manual curations from the *C. elegans* Sequencing Consortium[5]. (Genie version 5.)

7.1.2 *Chlamydomonas*

The complete genome sequence of the green algae, *Chlamydomonas reinhardtii*, is currently being sequenced. In collaboration with Gary Stormo's lab at Washington University St Louis, I trained a version of the software called GreenGenie and cross-validated performance metrics as shown in Section 6.3. A recent independent assessment of the GreenGenie predictions was compared to other *ab initio* predictions and cDNA alignments[40, 41], and based on that assessment, the total number of genes in green algae is estimated to be between 12,000 and 16,400. As a result of that study, the software is now actively being used for further contig annotation. A web server is available for single contig predictions and the software is available for other *Chlamydomonas* researchers upon request.

	Method:	Genie	GeneMark	Genescan
Exact Exon	Sn	0.75	0.54	0.75
	Sp	0.84	0.58	0.78
TAP-predicted Exact Exon	Sn	0.70	0.56	0.64
	Sp	0.81	0.67	0.53

Table 7.1: Assessment of Genie on the *Chlamydomonas* genome as reported in[41]. TAP-predicted exons refer to exons deduced by the TAP software[40] from EST alignments. (Genie version 2.)

7.1.3 *Drosophila*

To annotate the complete assembly of the fruit fly genome, a cadre of bioinformaticists pored through thousands of gene predictions from Genie and Genscan along with cDNA alignments and protein homology. Most of the final gene predictions are based on—if not derived exactly from—predictions from the Genie software. 13,189 gene predictions were produced using the Genie system. About 10,000 genes with database homology were reviewed manually. Based on the Genie predictions among that set, the curators chose to include approximately 3000 additional unreviewed Genie predictions yielding a final complement of 13,601 genes. A detailed assessment of the gene annotations was performed by Adams, et. al.[42]. All of these gene structures were deposited in Genbank and are available from the Flybase web site. A revised version of the Genie software is planned to be hosted by the Berkeley Drosophila Genome Project to assist in continued curation.

Drosophila was the first whole genome annotation effort in which I used clone bounds. Peter Brokstein from the Berkeley Drosophila Genome Project supplied a collection of 5'/3' mate paired ESTs. Martin Reese provided a curated set of annotated

	Both	Genie only	Genscan only	No Prediction	Total
EST + protein match	4,708	223	229	57	5,217
EST match only	3,192	261	212	51	3,716
Protein match only	1,462	82	112	54	1,710
No matches	2,551	338	69	0	2,958
Total	11,913	904	622	162	13,601

Table 7.2: Drosophila genome gene predictions[42]. Gene prediction programs were used in combination with searches of protein and EST databases to create a putative set of protein-coding genes. The first column are those selected genes with overlapping predictions by both Genie and Genscan. For Genscan only predictions with no homology support, only 69 out of more than 4,000 predictions were included, whereas all Genie only predictions were included. (Genie version 3.)

DNA sequences from Genbank from which the frequency counts for gene features were derived. cDNA constraints were implemented crudely as Blastn hits in the same manner as the protein homology models using all available public cDNA data along with unreleased (at that time) cDNAs from the BDGP. Precise splice site constraints could not be obtained. Since multiple exons in the 3' UTR are uncommon in Drosophila, the GHMM model topology was modified to disallow introns in the 3' UTR. Of note, the length of the majority of introns in Drosophila are peaked at about 70 nucleotides, making the gene predictions much more compact and reliable than for higher organisms.

7.1.4 ***Homo sapiens***

The effort to annotate the draft assembly of the human genome was far more challenging than generating the gene set for the *Drosophila* genome. A consortium effort generated gene predictions and the authors devised an automated system for selecting a gene set. The task was particularly onerous because the exon-intron structure predictions generated on the genomic contigs were expected to be used by protein bioinformaticists to assess the functional content of the protein-coding genes. It is well known that gene prediction in human is an imposing challenge even given high quality small contigs[34] and worse in large random genomic contigs[43]. False positive rates of *ab initio* prediction programs in random genomic contigs (versus the gene-rich contigs from which the programs are trained) tend to be quite high[44]. To control the false positive rate in a whole genome analysis and attempt to optimize for correct exon bounds, colleagues Ray Wheeler, Alan Williams, Cyrus Harmon, and I decided to only include gene predictions with some evidence of splicing from aligned cDNAs. In human genes, the 3' UTR is considerably longer than the rest of the exons and the majority of ESTs are derived from the 3' end of polyadenylated mRNAs. Thus, some of the Genie predictions were undoubtedly derived from completely untranslated regions.

Unlike the relatively clean set of Drosophila cDNAs, the EST data for human tended to be of lower quality as a result of genomic contaminants, immature mRNAs, and non-genic transcription (for example from transposon promoters), and often did not include protein-coding regions. Thus, the human genome annotation was subject to lower quality genomic and EST sequences. Sequencing errors result in situations where a legal parse cannot be generated from the EST/mRNA constraints because of insertions, deletions, and in-frame stop codons. Given sequencing errors, the constrained Genie system may predict a truncated or erroneous, out-of-frame CDS in order to accommodate the constraints. To ensure that downstream protein annotation efforts were not negatively impacted by sequencing errors and erroneous gene predictions, our final gene set replaced Genie predictions with annotated CDSs from known RefSeqs and full-length mRNAs when the experimental and predicted methods overlapped.

Nevertheless, some of the Genie predictions were novel protein-coding regions that had no similarity to known proteins, even though the gene structure may have been nearly correct due to EST alignments. As a result, the consortium chose to discard any predictions with no overlap to the Ensembl prediction set, which was largely influenced by protein homology. On the other hand, because we used clone bounds for gene region detection, our method tended to generate longer gene predictions where the Ensembl method produced highly fragmented results. In such conflicting cases, the longer Genie predictions were chosen as the representative gene structure. In summary, of the 10083 Genie predicted genes and 6315 full-length gene annotations submitted by myself and colleagues, only 4057 of the predicted genes were used in the published integrated gene index.

Several methods were employed to evaluate the sensitivity, specificity, and fragmentation of the human gene index. The gene set was compared to a small set of newly sequenced cDNAs generated after the genome annotation effort to assess sensitivity and fragmentation, compared to a large set of mouse cDNAs to assess sensitivity and an estimate of potential novel genes with no known protein, and compared to manual curation of the human chromosome 22. All of these analyses suggested that the total number of protein-coding genes was roughly in agreement with other inde-

	Knowns	Ensembl-Genie	Ensembl	Total
Number	14,882	4,057	12,839	31,778
Average length (amino acids)	469	443	187	352
Matches to non-human proteins	85%	74%	63%	75%
Matches to RIKEN mouse cDNA set	78%	74%	57%	69%
Matches to RIKEN mouse cDNA set but not to non-human proteins	36%	47%	31%	34%

Table 7.3: Human genome gene predictions. "Knowns" are a combination of SwissProt and peptide translations from RefSeq and TrEMBL. Note that the average peptide length of the Ensembl-Genie set is more than twice the length of the Ensembl-only predictions. This is partly due to the clone bounds and EST data serving to extend the gene predictions beyond the peptide homologies identified by the Ensembl method.

pendent estimates, that sensitivity (coverage of the total protein-coding gene set) was 70-85%, about 10-20% of predicted genes were false or pseudogenes, and that the set was significantly fragmented. Unfortunately, it is nearly impossible to relate those results to the Genie subset in any meaningful way except to suggest that the fragmentation of the total gene set would have been greater without it, as shown by the average peptide length in Figure 7.3. The detailed data generated by co-authors at that time was not made available and is since lost. Moreover, due to the magnitude of the analyses, no direct consideration of splice site prediction and exon-intron structure was given—rather analysis was largely based on protein similarity. For more details, see [45].

7.1.5 *Mus musculus*

More recently, a near-complete whole genome shotgun assembly of the mouse genome was published along with gene annotations using the Ensembl and Genie systems. Ray Wheeler and I worked together to generate the Genie results. To annotate the entire genome, the chromosome contigs were partitioned into smaller pieces with an average size of 7MB. Partitioning occurred at long repetitive regions when possible. For each segment, putative gene bounds were identified from clone bounds inferred from paired EST reads, *ab initio* mammalian Genie trained from human samples, and all best-in-genome cDNA alignment (at least 90% identity along 90% of the cDNA required). Maximal overlapping gene regions are inferred from these three methods. The *ab initio* gene finder serves to connect adjacent regions that may not have connecting cDNA evidence. Regions containing only an *ab initio* prediction or of length less than 500 bases are discarded. Each remaining region is processed independently per strand by creating a set of Genie constraints composed of a set of non-overlapping AltMerge transcripts and clone bounds for the region. One or more genes are predicted from the region and those with a coding region less than 100 bases are discarded.

While the approach was very similar to that employed for the human genome, it would seem that the results were more favorably received by our colleagues—probably for a number of different reasons. First, anecdotally, it seems that the EST data as a whole tended to be of higher quality in terms of rate of contamination and low quality reads. Second, only the highest quality cDNA alignments were included in our analysis (requiring 90% identity along 90% of the cDNA). Third, the EST coverage for many mouse genes appears to be denser and more complete than for human. Although the total number of mouse and human ESTs (about 2,400,000) and the 3'/5' ratio (about 1.4:1) was about the same, visual inspection of known gene structures along with EST alignments typically show more favorable EST coverage and quality per gene than for human. In particular, although a detailed analysis is lacking, it would appear that alternative messages, particularly alternative polyadenylation and partially processed mRNA, are more prevalent in human than in mouse. The mouse cDNA evidence tends to be more consistent among the overlapping aligned cDNAs. Fourth, my treatment of clone bounds was much more rigorous, excluding many false bounds as described below. Fifth, a detailed comparison of gene predictors at the exon-level was performed by the authors in which seven gene-prediction systems were compared: 23,026 genes from Ensembl[46], 37,793 genes from Fgenes++, 46,158 genes from the NCBI pipeline using the GenomeScan[47] program, 46,646 genes from an alternative Ensembl system that used only EST data, 48,451 genes from SGP[48], 48,548 genes from Twinscan[49], and 14,006 genes from SLAM[50], and 18,548 gene predictions using the Genie system. A 7-way Venn diagram of exon prediction overlap was generated along with a lengthy subjective evaluation. The authors decided to take as the primary set of gene predictions the union of the Ensembl and Genie predictions because those methods were most confirmed by other methods, while the other methods each tended to add additional predictions that were unique only to that method—suggesting mostly false positives. The fact that the Ensembl gene predictions were dominated by protein homology and the Genie predictions were dominated by EST evidence, yet the two methods agreed substantially provided additional confidence of the specificity of both methods. Last, predictions from other methods were used to extend gene predictions from Ensembl or Genie if more than 80% of the length of the transcript from the alternative prediction was shared by Ensembl or Genie. The total number of mouse protein-coding genes was extrapolated from the final set of gene predictions in a manner similar to the human gene count estimates in [45] yielding an estimate of $27,000 - 30,500$. For full details, see [51].

Because of the high quality of the cDNA and genomic data, a large fraction of exon-intron structures are fully described by the cDNA alignments. Using the AltMerge program[38] to generate putative transcripts merged from ESTs was sufficient to delineate gene structure, and the Genie software served mostly to "stitch" fragmented transcripts together and to predict the open reading frame in the AltMerge constraints.

An example of the complexities, pitfalls, and advantages of the Genie system is shown for an unsequenced gene in Figure 7.1.

7.2 Clone bounds and gene counts

How significant are the clone bounds in gene finding annotations? Considering the mouse genome, there are 197,289 clones with multiple EST reads (as many as 4). Of these, 207,239 3'/5' pairs are from the same clone, and of these, both ESTs align to the genome with marginal specificity for 188,197 pairs based on the alignment methods

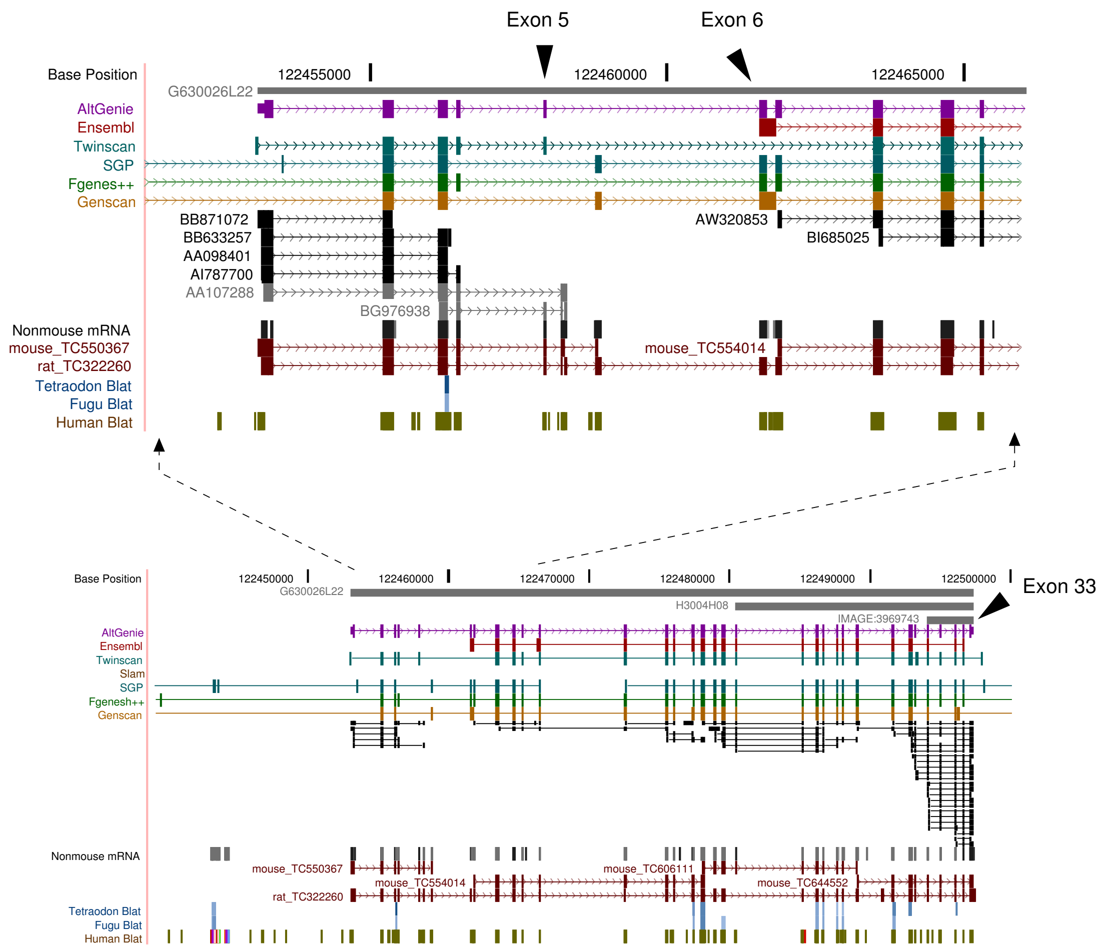


Figure 7.1: An example mouse gene. From top to bottom are shown clone bounds from EST mate pairs, 7 automated gene predictions, alignment of mouse ESTs, non-mouse mRNAs, assembled cDNA transcripts from TIGR, and sequence similarity with the tetraodon, fugu, and human genomes. Although no full-length cDNAs are present, virtually the entire gene structure is delineated by EST data except for a single small gap. A few conflicts exist among the aligned ESTs suggesting minor alternative splicing variations, truncated messages, or low quality tails that are incorrectly aligned. The one gap in EST evidence includes the Genie predictions for exon 5 and 6, predicted solely from statistical sequence information (shown in the top, magnified panel). Note, that the grey ESTs (AA107288 and BG976938) were *not* used as constraints because the alignments were not of sufficiently high quality, although in this case one of these two ESTs lends support to exon 5 as does homology with rat and human. Both *ab initio* and comparative genome gene finders (Fgenesh++ and SGP) and rat cDNA alignments lend credibility to exon 6. The final exon 33, while clearly justified by the cDNA evidence, contains only 4 codons, so it is understandably missed by the other gene finders, which do not use cDNA evidence. There are three pairs of 3'/5' reads that align to this gene region (top grey track) and the 5' read from clone G630026L22 is probably aligned to the beginning of the primary transcript since there are no other upstream ESTs in the vicinity. It is unlikely that the *ab initio* and comparative genome gene finders (SGP, Fgenesh++, GeneID, and Genscan) are correct in their upstream and downstream extensions. This example and others showing overextension argue that the statistical gene finders tend to overextend, and that the merge rule used in generating the mouse gene set was probably unwise. There are four separate TIGR "TC" transcripts for mouse for this one gene, suggesting that EST clustering is inferior to EST-genomic alignment when deducing transcripts.

described above. About 8.4% (15,864) of these aligned pairs match to different contigs, and 7.6% (14,371) aligned to different chromosomes (suggesting that partitioning at 7MB was not a serious problem, but that the alignment criteria could be stricter). For the remaining 91.6% (172,333) that aligned well to the same contig, 132,151 (76.7%) aligned with at least 90% identity.

I define a pair of reads as agreeing on genomic strand when the 5' EST aligns in the forward orientation and the 3' EST aligns in the reverse, or vice-versa. A pair agrees in order if the 5' EST is upstream of the 3' EST along the inferred genomic strand. 117,078 pairs (67.9% of the well aligned pairs) agree in strand and order, but only 72,882 (42.3%) have strong evidence of orientation based on splicing or polyadenylation.

Many of the pairs of reads overlap with other pairs from different clones. Since a majority of pairs had weak orientation evidence, I created sets of gene bounds in an iterative process. Starting with the pair with the strongest orientation evidence, new pairs were merged with overlapping pairs. A new gene bound was not created unless the orientation score was above a threshold and no gene bound already existed on the opposite strand. Without this rule, clone reversal would cause the prediction of gene regions in the same position on complementary strands. This merging process created $23,116$ gene regions with a minimum, maximum, mean, and median primary transcript length of 168, 5.1×10^6, 26120, and 5699, respectively—lengths comparable to primary transcripts for known full-length mRNAs.

This analysis suggests that clone pairs aligned to the mouse genome with high specificity provide significant support for gene regions for a large number of distinct loci—indeed on the order of the total number of predicted genes. Visual inspection shows that these clone bounds are consistently in agreement with known genes. This suggests that such bounds are important anchors for the prediction of more detailed gene structure and can help reduce the false positive rate of *ab initio* gene prediction by reducing the input sequences to those with gene bounds defined by clone read pairs.

Because EST sequencing is limited in its tissue diversity, a lower bound on the total number of genes can be extrapolated by considering the fraction of clone bounds that overlap the 7979 RefSeq mRNA that align to the mouse genome with high specificity. 6608 (82.8%) of these mRNAs overlap with clone bounds. Some of the clone bounds may be contaminants, and others may represent non-protein-coding genes. Nevertheless, this suggests that there are at least $27,900$ $(23,116/.828)$ genes in the mouse genome. This number is within the range of $27,000 - 30,000$ genes that we estimated in [51] based on a very different analysis of sensitivity of the integrated gene set with respect to known gene collections.

Chapter 8

Closing Remarks

8.1 Discussion

The GHMM framework described here still lacks a few important features that I have not addressed. As mentioned in the previous chapter, overlapping genes cannot be generally accommodated, although this is not a serious weakness. In practice, a pre-processing step can identify putative overlapping genes and then analyze the sequences independently, if necessary. In addition to overlapping genes, some rare gene structures also cannot be represented. It is obviously possible for true start and stop codons to span introns, although from experience it is much rarer than decoy stop codons. This model specifically inhibits intron-spanning stop codons and cannot detect intron-spanning start codons. Also, while it is estimated that less than 1% of splice sites do not include the consensus dinucleotides, this model requires consensus sites unless cDNA evidence implies otherwise. It is possible to relax the model to allow splicing at additional positions, but preliminary experiments suggest that doing so decreases overall accuracy and significantly increases running time.

Cardinality constraints in the number of exons, or, equivalently, in the total size of the predicted message, cannot be modeled well. I described in [27] how the exon-intron-exon loop can be "unspooled" so that the distribution of the first k exons can be modeled separately, although this was never implemented due to the complexity of such models.

An important weakness is in the hidden Markov model itself. A generative HMM requires that the parameters over the transitions and observations either be learned through an EM method or that the scoring methods provided through other means conform to likelihood distributions to ensure a balanced system. But there are many methods for discriminating features that simply produce a score that is optimally thresholded to obtain the desired balance of sensitivity and specificity. Sometimes these scores can be treated as posteriors and converted into likelihoods given some assumptions about the training procedure and using Bayes rule, but my attempts, for example, to convert neural network outputs into likelihoods were not satisfactory. Probably the most effective internal exon prediction schemes are based on discriminant analysis[11, 12], yet these methods are not amenable to incorporation into likelihood models.

The cDNA constraint system that I describe should be generalized to allow more subtle weighting schemes. Such a system could include consideration for the number or strength of homologous sequence alignments or incorporate other types of data such as readouts from micro-arrays. In the extreme, the problems of orthologous sequence alignment between genomes and gene prediction can be unified into a single modeling problem, as has been done with the HMMs developed by Pachter, et. al.[50], in their comparative genome gene finder and aligner. I believe that this is the natural extension of HMM-based gene finding in future research.

As shown in Section 6.7.2, it is remarkable that the number of ESTs in the public databases today are sufficient to describe as much as 40% of gene structures and protein-coding regions, despite the high noise rate in these expressed sequences. Thus, it would seem that for some genomes, a sophisticated gene prediction system is less important than a careful management of cDNA sequences. Nevertheless, there will continue to be a need for gene prediction methods that provide a complement to the cDNA data when quality or quantity is lacking, and that role is played well by Genie.

One topic of future research that is particularly interesting is the relationship of gene features to expression level. With several completed genomes, large collections of ESTs, and whole genome micro-array expression assays, it would be interesting to consider the correlation of expression to gene features in the same way that G+C content has been found to correlate with coding metrics and intron length.

8.2 Conclusion

In this paper I showed the efficacy of a gene-prediction system based on a state-duration hidden Markov model (GHMM) allowing multiple symbols per state and imposing length distributions over the observations. The grammatical structure of genes in DNA, including frame consistency and intron-spanning stop codons, can be modeled using such an HMM and this leads to a consistent, complete gene prediction. In addition, such a model lends itself to a modular implementation that provides for the integration of independent feature sensors into a unified system. While the theoretical running time is cubic in the length of the input sequence, I describe an implementation that, in practice, is linear.

To accommodate external information, I propose different methods for either constraining the search space or modifying the likelihood model scoring system. I show that this has the practical value of improving gene prediction by using cDNA or protein alignments to genomic sequence to delineate exonic and protein-coding regions, respectively.

Finally, I present results of the application of the Genie software system to the annotation of several model organisms and describe how those results led to predictions of the total number of protein-coding genes in those organisms. In addition, I argue that the use of clone bounds is an effective means of identifying gene loci, estimating gene counts, and minimizing false positive rates from *ab initio* gene predictions.

Bibliography

[1] M. Kozak. Structural features in eukaryotic mRNAs that modulate the initiation of translation. *J Biol Chem*, 266(30):19867–70, 1991.

[2] R. Staden. Finding protein coding regions in genomic sequences. *Methods Enzymol*, 183:163–80, 1990.

[3] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res*, 12(1 Pt 2):505–19, 1984.

[4] G. D. Stormo. Consensus patterns in DNA. *Methods Enzymol*, 183:211–21, 1990.

[5] The C. elegans Sequencing Consortium. Genome sequence of the nematode C. elegans: a platform for investigating biology. *Science*, 282(5396):2012–8, 1998.

[6] M. S. Gelfand. Prediction of function in DNA sequence analysis. *J Comput Biol*, 2(1):87–115, 1995.

[7] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *J Mol Biol*, 268(1):78–94, 1997.

[8] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the dna sequence. *J Mol Biol*, 220(1):49–65, 1991.

[9] M. G. Reese, F. H. Eeckman, D. Kulp, and D. Haussler. Improved splice site detection in Genie. *J Comput Biol*, 4(3):311–23, 1997.

[10] M. G. Reese. Application of a time-delay neural network to promoter annotation in the Drosophila melanogaster genome. *Comput Chem*, 26(1):51–6, 2001.

[11] V. V. Solovyev, A. A. Salamov, and C. B. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Res*, 22(24):5156–63, 1994.

[12] M. Q. Zhang. Identification of protein coding regions in the human genome by quadratic discriminant analysis. *Proc Natl Acad Sci U S A*, 94(2):565–8, 1997.

[13] J. M. Claverie, I. Sauvaget, and L. Bougueleret. Computer generation and statistical analysis of a data bank of protein sequences translated from GenBank. *Biochimie*, 67(5):437–43, 1985.

[14] J. W. Fickett and C. S. Tung. Assessment of protein coding measures. *Nucleic Acids Res*, 20(24):6441–50, 1992.

[15] M. Borodovsky and J. McIninch. Recognition of genes in DNA sequence with ambiguities. *Biosystems*, 30(1-3):161–71, 1993.

[16] R. Guigo and J. W. Fickett. Distinctive sequence features in protein coding genic non-coding, and intergenic human DNA. *J Mol Biol*, 253(1):51–60, 1995.

[17] W. J. Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12(4):656–64, 2002.

[18] C. A. Fields and C. A. Soderlund. gm: a practical tool for automating DNA sequence analysis. *Comput Appl Biosci*, 6(3):263–70, 1990.

[19] R. Guigo, S. Knudsen, N. Drake, and T. Smith. Prediction of gene structure. *J Mol Biol*, 226(1):141–57, 1992.

[20] G. B. Hutchinson and M. R. Hayden. The prediction of exons through an analysis of spliceable open reading frames. *Nucleic Acids Res*, 20(13):3453–62, 1992.

[21] E. C. Uberbacher and R. J. Mural. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proc Natl Acad Sci U S A*, 88(24):11261–5, 1991.

[22] E. E. Snyder and G. D. Stormo. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucleic Acids Res*, 21(3):607–13, 1993.

[23] D. B. Searls. The linguistics of DNA. *American Scientist*, 80:579–591, 1992.

[24] S. Dong and D. B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23(3):540–51, 1994.

[25] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77(2):257–286, 1989.

[26] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology. applications to protein modeling. *J Mol Biol*, 235(5):1501–31, 1994.

[27] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. *Proc Int Conf Intell Syst Mol Biol*, 4:134–42, 1996.

[28] A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. *Proc Int Conf Intell Syst Mol Biol*, 5:179–86, 1997.

[29] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Interval trees. In *Introduction to Algorithms*, pages 290–294. MIT Press, 1990.

[30] W. R. Gish and D. J. States. Identification of protein coding regions by database similarity search. *Nature Genetics*, 3, 1993.

[31] S. Henikoff and J. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Res*, 19(23):6565–6572, 1991.

[32] K. Sjolander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Comput Appl Biosci*, 12(4):327–45, 1996.

[33] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. Integrating database homology in a probabilistic gene structure model. *Pac Symp Biocomput*, pages 232–44, 1997.

[34] M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–67, 1996.

[35] M. G. Reese, G. Hartzell, N. L. Harris, U. Ohler, J. F. Abril, and S. E. Lewis. Genome annotation assessment in Drosophila melanogaster. *Genome Res*, 10(4):483–501, 2000.

[36] M. G. Reese, D. Kulp, H. Tammana, and D. Haussler. Genie–gene finding in Drosophila melanogaster. *Genome Res*, 10(4):529–38, 2000.

[37] J. W. Fickett and A. G. Hatzigeorgiou. Eukaryotic promoter recognition. *Genome Res*, 7(9):861–78, 1997.

[38] R. Wheeler. A method of consolidating and combining EST and mRNA alignments to a genome to enumerate supported splice variants. In R. Guigo and D. Gusfield, editors, *Algorithms in Bioinformatics : Second International Workshop, WABI*, Rome, 2002.

[39] W. J. Kent and A. M. Zahler. The intronerator: exploring introns and alternative splicing in Caenorhabditis elegans. *Nucleic Acids Res*, 28(1):91–3, 2000.

[40] Z. Kan, E. C. Rouchka, W. R. Gish, and D. J. States. Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. *Genome Res*, 11(5):889–900, 2001.

[41] J. Li, S. Lin, H. Jia, H. Wu, B. A. Roe, D. Kulp, G. D. Stormo, and S. K. Dutcher. Analysis of Chlamydomonas reinhardtii genome structure using large-scale sequencing of regions on linkage groups iii and vi. *submitted*, 2003.

[42] M. D. Adams, S. E. Celniker, R. A. Holt, C. A. Evans, J. D. Gocayne, P. G. Amanatides, S. E. Scherer, P. W. Li, R. A. Hoskins, R. F. Galle, R. A. George, S. E. Lewis, S. Richards, M. Ashburner, S. N. Henderson, G. G. Sutton, J. R. Wortman, M. D. Yandell, Q. Zhang, L. X. Chen, R. C. Brandon, Y. H. Rogers, R. G. Blazej, M. Champe, B. D. Pfeiffer, K. H. Wan, C. Doyle, E. G. Baxter, G. Helt, C. R. Nelson, G. L. Gabor, J. F. Abril, A. Agbayani, H. J. An, C. Andrews-Pfannkoch, D. Baldwin, R. M. Ballew, A. Basu, J. Baxendale, L. Bayraktaroglu, E. M. Beasley, K. Y. Beeson, P. V. Benos, B. P. Berman, D. Bhandari, S. Bolshakov, D. Borkova, M. R. Botchan, J. Bouck, P. Brokstein, P. Brottier, K. C. Burtis, D. A. Busam, H. Butler, E. Cadieu, A. Center, I. Chandra, J. M. Cherry, S. Cawley, C. Dahlke, L. B. Davenport, P. Davies, B. de Pablos, A. Delcher, Z. Deng, A. D. Mays, I. Dew, S. M. Dietz, K. Dodson, L. E. Doup, M. Downes, S. Dugan-Rocha, B. C. Dunkov, P. Dunn, K. J. Durbin, C. C. Evangelista, C. Ferraz, S. Ferriera, W. Fleischmann, C. Fosler, A. E. Gabrielian, N. S. Garg, W. M. Gelbart, K. Glasser, A. Glodek, F. Gong, J. H. Gorrell, Z. Gu, P. Guan, M. Harris, N. L. Harris, D. Harvey, T. J. Heiman, J. R. Hernandez, J. Houck, D. Hostin, K. A. Houston, T. J. Howland, M. H. Wei, C. Ibegwam, et al. The genome sequence of Drosophila melanogaster. *Science*, 287(5461):2185–95, 2000.

[43] R. Guigo, P. Agarwal, J. F. Abril, M. Burset, and J. W. Fickett. An assessment of gene prediction accuracy in large DNA sequences. *Genome Res*, 10(10):1631–42, 2000.

[44] I. Dunham, N. Shimizu, B. A. Roe, S. Chissoe, A. R. Hunt, J. E. Collins, R. Bruskiewich, D. M. Beare, M. Clamp, L. J. Smink, R. Ainscough, J. P. Almeida, A. Babbage, C. Bagguley, J. Bailey, K. Barlow, K. N. Bates, O. Beasley, C. P. Bird, S. Blakey, A. M. Bridgeman, D. Buck, J. Burgess, W. D. Burrill, K. P. O'Brien, and et al. The DNA sequence of human chromosome 22. *Nature*, 402(6761):489–95, 1999.

[45] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, R. Funke, D. Gage, K. Harris, A. Heaford, J. Howland, L. Kann, J. Lehoczky, R. LeVine, P. McEwan, K. McKernan, J. Meldrim, J. P. Mesirov, C. Miranda, W. Morris, J. Naylor, C. Raymond, M. Rosetti, R. Santos, A. Sheridan, C. Sougnez, N. Stange-Thomann, N. Stojanovic, A. Subramanian, D. Wyman, J. Rogers, J. Sulston, R. Ainscough,

S. Beck, D. Bentley, J. Burton, C. Clee, N. Carter, A. Coulson, R. Deadman, P. Deloukas, A. Dunham, I. Dunham, R. Durbin, L. French, D. Grafham, S. Gregory, T. Hubbard, S. Humphray, A. Hunt, M. Jones, C. Lloyd, A. McMurray, L. Matthews, S. Mercer, S. Milne, J. C. Mullikin, A. Mungall, R. Plumb, M. Ross, R. Shownkeen, S. Sims, R. H. Waterston, R. K. Wilson, L. W. Hillier, J. D. McPherson, M. A. Marra, E. R. Mardis, L. A. Fulton, A. T. Chinwalla, K. H. Pepin, W. R. Gish, S. L. Chissoe, M. C. Wendl, K. D. Delehaunty, T. L. Miner, A. Delehaunty, J. B. Kramer, L. L. Cook, R. S. Fulton, D. L. Johnson, P. J. Minx, S. W. Clifton, T. Hawkins, E. Branscomb, P. Predki, P. Richardson, S. Wenning, T. Slezak, N. Doggett, J. F. Cheng, A. Olsen, S. Lucas, C. Elkin, E. Uberbacher, M. Frazier, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

[46] T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. Clamp. The Ensembl genome database project. *Nucleic Acids Res*, 30(1):38–41, 2002.

[47] R. F. Yeh, L. P. Lim, and C. B. Burge. Computational inference of homologous gene structures in the human genome. *Genome Res*, 11(5):803–16, 2001.

[48] G. Parra, P. Agarwal, J. F. Abril, T. Wiehe, J. W. Fickett, and R. Guigo. Comparative gene prediction in human and mouse. *Genome Res*, 13(1):108–17, 2003.

[49] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17 Suppl 1:S140–8, 2001.

[50] L. Pachter, M. Alexandersson, and S. Cawley. Applications of generalized pair hidden Markov models to alignment and gene finding problems. *J Comput Biol*, 9(2):389–99, 2002.

[51] R. H. Waterston, K. Lindblad-Toh, E. Birney, J. Rogers, J. F. Abril, P. Agarwal, R. Agarwala, R. Ainscough, M. Alexandersson, P. An, S. E. Antonarakis, J. Attwood, R. Baertsch, J. Bailey, K. Barlow, S. Beck, E. Berry, B. Birren, T. Bloom, P. Bork, M. Botcherby, N. Bray, M. R. Brent, D. G. Brown, S. D. Brown, C. Bult, J. Burton, J. Butler, R. D. Campbell, P. Carninci, S. Cawley, F. Chiaromonte, A. T. Chinwalla, D. M. Church, M. Clamp, C. Clee, F. S. Collins, L. L. Cook, R. R. Copley, A. Coulson, O. Couronne, J. Cuff, V. Curwen, T. Cutts, M. Daly, R. David, J. Davies, K. D. Delehaunty, J. Deri, E. T. Dermitzakis, C. Dewey, N. J. Dickens, M. Diekhans, S. Dodge, I. Dubchak, D. M. Dunn, S. R. Eddy, L. Elnitski, R. D. Emes, P. Eswara, E. Eyras, A. Felsenfeld, G. A. Fewell,

P. Flicek, K. Foley, W. N. Frankel, L. A. Fulton, R. S. Fulton, T. S. Furey, D. Gage, R. A. Gibbs, G. Glusman, S. Gnerre, N. Goldman, L. Goodstadt, D. Grafham, T. A. Graves, E. D. Green, S. Gregory, R. Guigo, M. Guyer, R. C. Hardison, D. Haussler, Y. Hayashizaki, L. W. Hillier, A. Hinrichs, W. Hlavina, T. Holzer, F. Hsu, A. Hua, T. Hubbard, A. Hunt, I. Jackson, D. B. Jaffe, L. S. Johnson, M. Jones, T. A. Jones, A. Joy, M. Kamal, E. K. Karlsson, et al. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–62, 2002.