Homologous Gene Finding with a Hidden Markov Model

by

Xuefeng Cui

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The homology search problem and the gene finding problem are two fundamental problems in bioinformatics. The homology search problem is to find the homologous regions of two biological sequences; the gene finding problem is to find all the genes in both strands of a genomic sequence. Recently, gene finding research has demonstrated that homology search results can be used to improve the accuracy of gene finding. By combining the two problems, we define a new problem called the homologous gene finding problem. The homologous gene finding problem is to find homologous genes of a query gene in a target genomic sequence.

Consequently, we present a new homologous gene finding algorithm in this thesis. We borrow the idea of gene mapping and alignment algorithms, and apply existing seed-based homology search algorithms and hidden Markov model-based (HMM-based) gene finding algorithms to solve the homologous gene finding problem. After we find high-scoring segment pairs (HSPs) between the query gene and the target genomic sequence, we locate target regions that we believe contain a gene homologous to the query gene. Then, we extend existing HMM-based gene finding algorithms to find homologous gene candidates. To improve the accuracy of homologous gene finding, we train a HMM to be biased toward the query gene. We also introduce a new coding sequence (CDS) length penalty as a measure of how the CDS lengths of the query gene and its homologous gene vary to further improve the accuracy. We use the new CDS length penalty together with our enhanced Viterbi algorithm and our flexible finish condition to improve the speed of homologous gene fining without harming the accuracy. Finally, we use protein alignment to pick and rank the best homologous gene candidates.

In this thesis, we also describe several experiments to evaluate and support our homologous gene finding algorithm.

Acknowledgments

I would like to thank both of my supervisors, Ming Li and Tomás Vinar, as well as Prof Dennis Shasha from New York University for their guidance and support. I would like to thank Brona Brejová, Dongbo Bu and Shuai Cheng Li for their discussions. I would also like to thank my readers, Daniel G. Brown and Brendan J. McConkey, for reading and commenting this thesis. Finally, I would like to thank my parents and Bing Wang for supporting and encouraging me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, we introduce a new algorithm to find homologous genes from different species. In Chapter 1, we first introduce the biological background related to our research. Then, we define the homologous gene finding problem, and review several related problems. In Chapter 2, we introduce our homologous gene finding algorithm. In Chapter 3, we demonstrate the usefulness of our algorithm by presenting its implementation, and evaluating its performance on homologous gene data sets. Finally, we discuss our homologous gene finding algorithm and future work in Chapter 4.

1.1 Genome, Gene and Protein

In this section, we introduce the complete biological background related to the homologous gene finding problem. In particular, we introduce gene structures, and the process of producing proteins from genes.

Deoxyribonucleic acid (DNA) consists of two strands of nucleotides, joined together as a single chain to form a double helical structure. Each DNA nucleotide consists of three components: a sugar molecule called deoxyribose, a phosphate group, and a nitrogen-containing compound called base. There are five carbons in the sugar molecule, and they are numbered $1'$, $2'$, $3'$, $4'$ and $5'$. The hydroxyl groups on the $5'$ and $3'$ carbons link to the phosphate groups to form the backbone of DNA. The $5'$ end of a DNA strand is called the upstream, and the $3'$ end of a DNA strand is called the downstream. There are four types of bases for

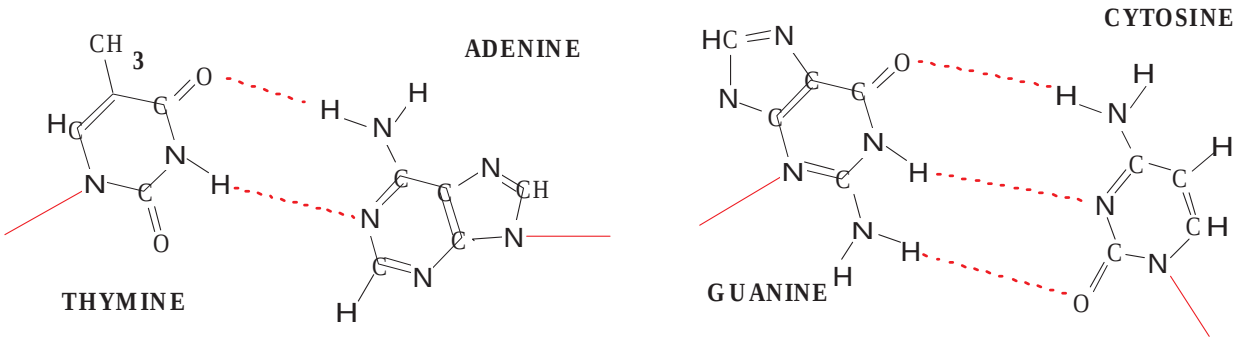


Figure 1.1: DNA complementary base pairing

DNA: adenine (A), cytosine (C), guanine (G) and thymine (T). The two strands are joined together by complementary base pairing: adenine always pairs with thymine, and guanine always pairs with cytosine, as shown in Figure 1.1.

The process of DNA copying is called replication. During replication, the two strands of the double helix structure first separate to individual strands. Then, each strand is paired with floating complementary bases within the cell to form a new double helical structure. Rarely, nucleotides are inserted, deleted (indel) or mutated during replication.

Ribonucleic acid (RNA) consists of a single nucleotide chain. Each RNA nucleotide consists of three components: a sugar molecule called ribose, a phosphate group and a nitrogen-containing compound called base. Again, there are four types of bases for RNA: adenine (A), cytosine (C), guanine (G) and uracil (U).

The genome of an organism consists of the complete DNA sequences of the organism. Genes are regions of the genome that encode the information to produce proteins. Human genes have between 100 and 1,000,000 nucleotides, and their coding regions make up less than 2 percent of the complete genome [15].

In eukaryotes, genes may contain coding regions and non-coding regions. The coding regions are called exons, and exons are separated by non-coding regions called introns. The boundaries of exons and introns are called splice sites. The splice site located after an exon and before an intron is called a donor site. Otherwise, it is called an acceptor site. In prokaryotes, there are no introns.

Generally, there are signals nearby splice sites in the genome. We used the program WebLogo [37] to draw Figures 1.2 and 1.3. These two figures show the signals observed

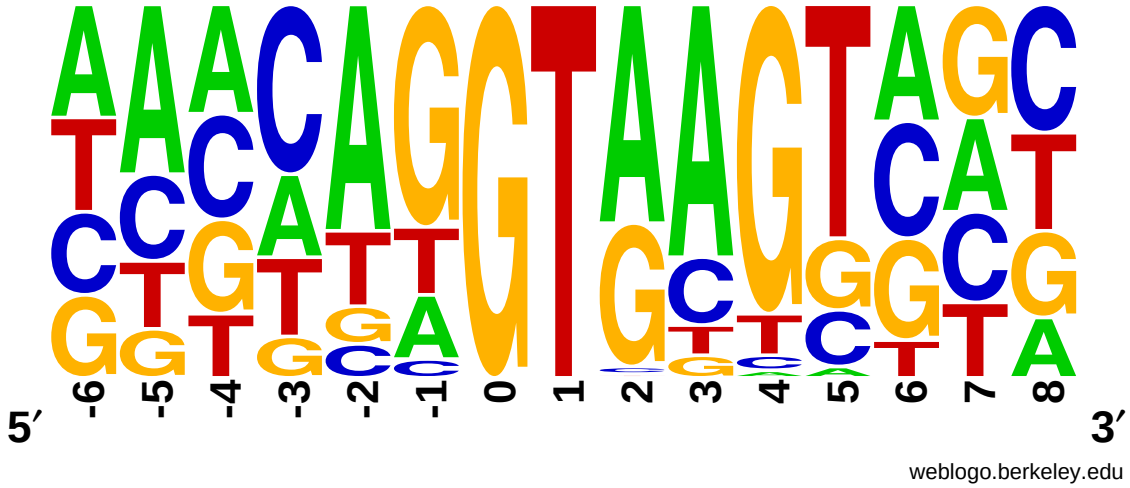


Figure 1.2: Sequence logo of the donor site signal

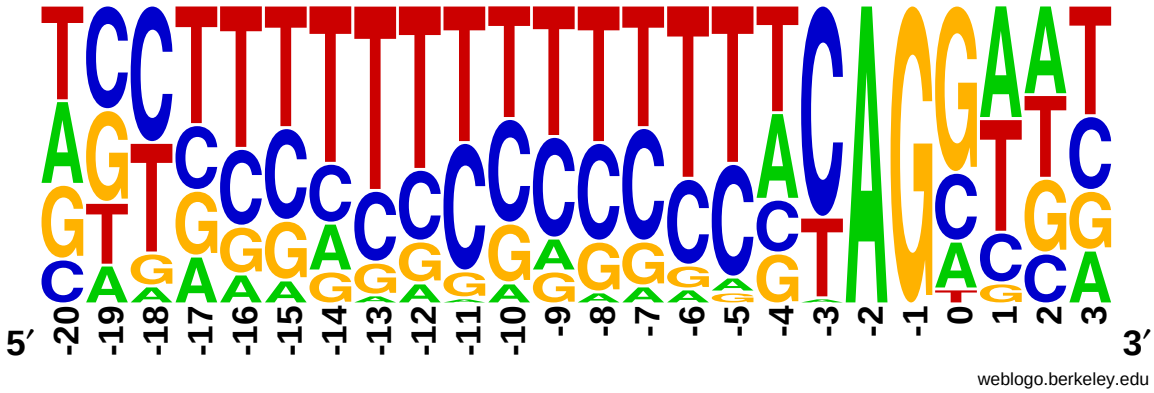


Figure 1.3: Sequence logo of the acceptor site signal

from 100 donor sites and 100 acceptor sites of human genes. In these figures, the splice sites are the boundaries between position -1 and position 0. Each column represents a position around the splice site in the nucleotide sequence, and each letter represents a nucleic acid at that position. The probability of a particular nucleic acid being observed at a position is proportional by the height of that letter. The nucleic acid with the highest probability of being observed at a particular position is shown on the top of that column. From Figures 1.2 and 1.3, we can see that the strongest signal is that the nucleotides "GT" always appear right after donor sites, and the nucleotides "AG" always appear right before acceptor sites. These signals are very useful to find accurate gene structures.

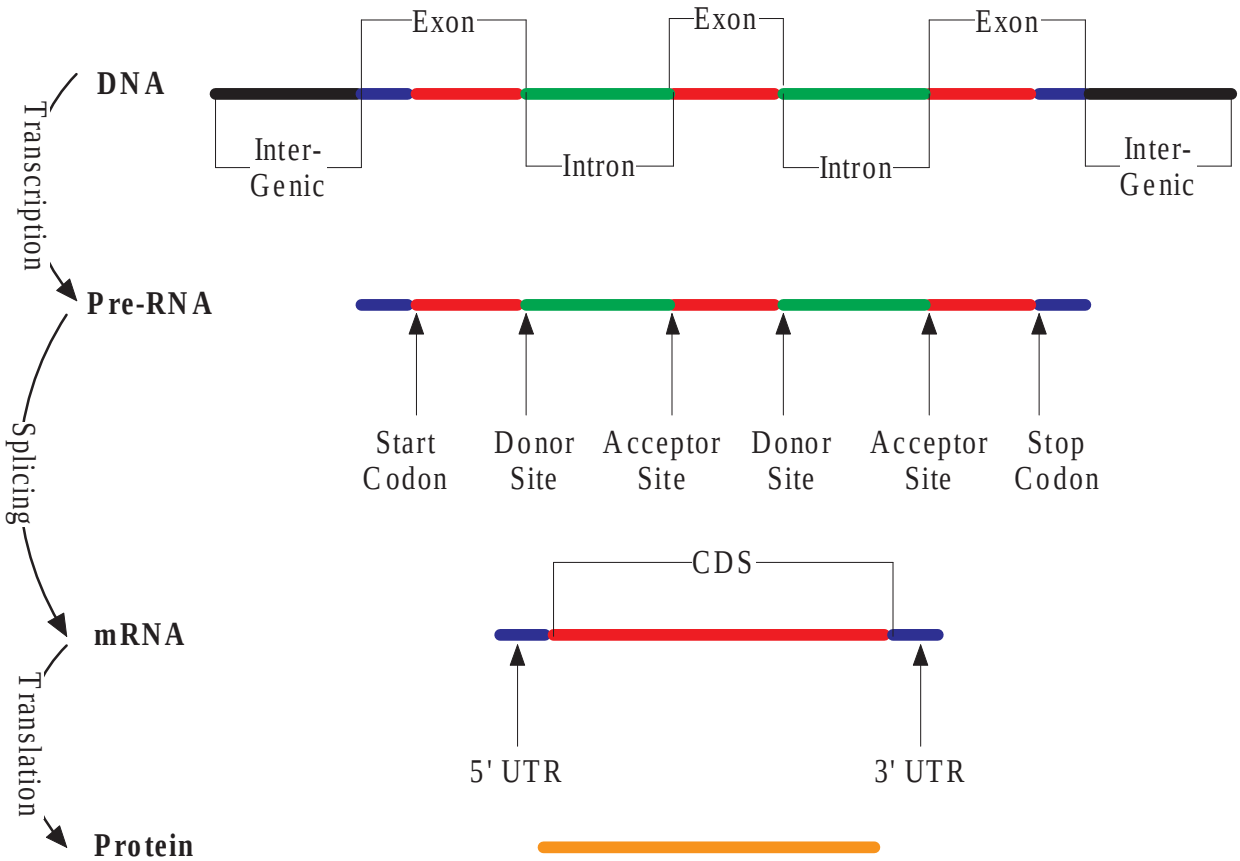


Figure 1.4: Protein synthesis

The process of producing proteins from genes is called protein synthesis, and it can be divided into two steps: transcription and translation. Figure 1.4 shows the complete process of protein synthesis.

During the transcription, the information encoded in the DNA is transcripted to a messenger RNA (mRNA). In prokaryote, this step simply copies a fragment of DNA to mRNA. However, in eukaryotes, the gene is first copied to a preliminary mRNA (pre-mRNA), and then introns are spliced out. The remaining exons are connected in the original order to form a mature mRNA.

During the translation, an mRNA is translated to a protein, and Figure 1.5 shows an example of how an mRNA sequence is translated into a protein sequence. At the $5'$ and $3'$ ends of an mRNA sequence, there are untranslated regions (UTR) that are not translated to the protein. Other than UTRs, the coding sequence (CDS) is translated to the protein, and

		2nd base			
		T	C	A	G
1st base	T	TTT Phe/F TTC Phe/F TTA Leu/L TTG Leu/L	TCT Ser/S TCC Ser/S TCA Ser/S TCG Ser/S	TAT Tyr/Y TAC Tyr/Y TAA **Stop** TAG **Stop**	TGT Cys/C TGC Cys/C TGA **Stop** TGG Trp/W
	C	CTT Leu/L CTC Leu/L CTA Leu/L CTG Leu/L	CCT Pro/P CCC Pro/P CCA Pro/P CCG Pro/P	CAT His/H CAC His/H CAA Gln/Q CAG Gln/Q	CGT Arg/R CGC Arg/R CGA Arg/R CGG Arg/R
	A	ATT Ile/I ATC Ile/I ATA Ile/I ATG Met/M, **Start**	ACT Thr/T ACC Thr/T ACA Thr/T ACG Thr/T	AAT Asn/N AAC Asn/N AAA Lys/K AAG Lys/K	AGT Ser/S AGC Ser/S AGA Arg/R AGG Arg/R
	G	GTT Val/V GTC Val/V GTA Val/V GTG Val/V	GCT Ala/A GCC Ala/A GCA Ala/A GCG Ala/A	GAT Asp/D GAC Asp/D GAA Glu/E GAG Glu/E	GGT Gly/G GGC Gly/G GGA Gly/G GGG Gly/G

Table 1.1: Genetic Code

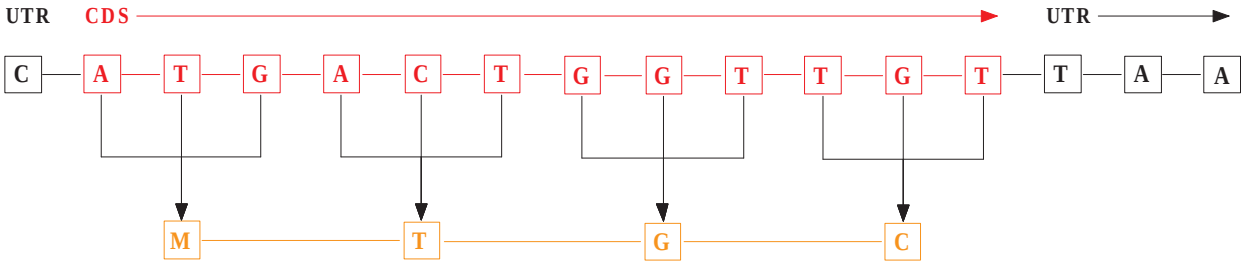


Figure 1.5: Simple translation example

a CDS is made up of nucleotide triplets called codons. Each codon has 3 frames numbered from 1 to 3, and each codon encodes an amino acid using the Genetic Code, as shown in Table 1.1. There are 64 possible codons, but there are only 20 amino acids. Thus, different codons may be translated into the same amino acid. If the codon frame is shifted by one or two, the CDS may be translated to a completely different protein with different functions. Thus, most codon frame-shifts are harmful. During the translation, codons starting with start codon "ATG" are translated into amino acids until one of the stop codons "TAG", "TGA" or "TAA" is reached. Start codons are translated into the protein, but stop codons only serve to abort the translation, and are not translated.

1.2 Homologous Gene Finding

Before we define homologous genes and the homologous gene finding problem, we have to introduce similarity between two sequences. Generally, we use the scores of pairwise sequence alignments to measure the similarity between two sequences. An alignment between two sequences describes how one sequence can be edited to the other one through insertions, deletions (indels) and point mutations. We give positive scores to matches, and negative scores to indels and mutations. The score of an alignment is the total score of the matches, indels and mutations in the alignment, and the optimal alignment is the one with maximum score. There are mainly two types of alignments: global and local. A global alignment describes the global relationships between two whole sequences, while a local alignments describes the local relationships between subregions of two sequences.

Two genes are homologous if they share a common evolutionary ancestry. There are two types of homology: orthology or paralogy. If this homology is a result of a speciation event (one species diverges into two species), we call the two genes orthologous. If this homology is a result of a duplication event, we call the two genes paralogous. Unfortunately, it is hard to define homology using only sequences, without knowing complete evolutionary histories. In bioinformatics, we often presume that two genes are homologous if they have highly similar sequences.

Finding homologous genes is essential for biologists. Since homologous genes are likely to have similar functions, finding homologous genes can help biologists to annotate gene functions. In this thesis, we explore the homologous gene finding problem:

Definition 1 ***Homologous Gene Finding Problem:*** *We are given a query gene, more precisely the query genomic sequence and the annotation of all CDSs of the query gene. We are also given a DNA sequence of the target genomic sequence. We presume that two genes are homologous, if and only if the protein sequences encoded by the query and target genes are highly similar (alignment score above some threshold). We define a target gene as a gene in the target genomic sequence that is homologous to the query gene. The task of homologous gene finding is to find all target genes in the target genomic sequence.*

To measure the accuracy of the homologous gene finding algorithms, we use statistical

measures of sensitivity and specificity, which have been widely used to measure the accuracy of gene finding algorithms [10, 14, 16]. Sensitivity and specificity are defined as follows:

Definition 2 ***Sensitivity and Specificity:*** *Let TP be the number of true positives (true predicted as true), FP be the number of false positives (false predicted as true), and FN be the number of false negatives (true predicted as false). Then, sensitivity, SN, is computed as $SN = \frac{TP}{TP+FN}$, and specificity, SP, is computed as $SP = \frac{TP}{TP+FP}$.*

On the nucleotide level, sensitivity is the percentage of correctly predicted coding nucleotides over true coding nucleotides; specificity is the percentage of correctly predicted coding nucleotides over predicted coding nucleotides. On the exon or the gene level, sensitivity is the percentage of predicted exons or genes over true exons or genes; specificity is the percentage of correctly predicted exons or genes over predicted exons or genes, respectively. An exon is predicted correctly, if and only if both the start and end of the exon are predicted correctly. A gene is predicted correctly, if and only if all exons of the gene are predicted correctly.

1.3 Related Work

Homologous gene finding includes several areas of sequence analysis: the homology search problem, the gene mapping and alignment problem, and the gene finding problem. These problems have been well studied in bioinformatics, and there are many existing algorithms available to solve them. In this section, we briefly review each of these related problems, and discuss their differences to the homologous gene finding problem.

1.3.1 Homology Search

The homology search problem is to predict homologous regions between two or more sequences. In fact, the homology search problem can be simplified as finding local alignments with high alignment scores. If two regions have a high similarity, the probability that they are unrelated is low. Thus, we presume that they are homologous, and we call each pair of highly similar sequences a High-scored Sequence Pair (HSP).

The homology search problem is traditionally solved by finding the optimal HSPs using the Smith-Waterman algorithm [33], which is a dynamic programming algorithm that finds the best local alignments. The main advantage of the Smith-Waterman algorithm is its high sensitivity because all similar regions will be uncovered. The main disadvantage of the Smith-Waterman algorithm is its runtime and memory consumption. If we would like to find homologous regions between two sequences of sizes m and n, the Smith-Waterman algorithm requires $\theta(mn)$ time and $\theta(\min(m, n))$ space. This is often not practical because m and n can be as large as 10^{10}. Therefore, the Smith-Waterman algorithm is only used when high sensitivity is critical and query sequences are short.

The first fast homology search software package FASTA [21] was introduced by Lipman and Pearson in 1985. The idea of FASTA is to firstly find short exact matches of a given length between query sequences, and then run a Smith-Waterman-like algorithm on regions containing high density cluster of such matches. In the worst case, the algorithms may take time $\theta(mn)$. In practice, the number of such matches may not be too many, and FASTA is efficient.

Basic Local Alignment Search Tool (BLAST) [2] is one of the most popular sequence analysis tools available in the public domain. It was introduced by Altschul and others in 1990. Later, a variety of implementations, and extensions of the BLAST family [2, 3, 41, 40, 25] have been implemented. BLAST uses contiguous matches as seeds and extends each seed in both directions to form an HSP. Similar to FASTA, BLAST has a tradeoff between the sensitivity and the speed. Users can specify the seed length to balance the sensitivity and the speed.

While both FASTA and BLAST trade the sensitivity for the speed, PatternHunter [22, 20, 18] achieves higher sensitivity without slowing down the runtime much. Instead of using contiguous matches as seeds, PatternHunter uses discontiguous matches, called spaced seeds. Similar to BLAST, PatternHunter extends each seed in both directions to form an optimal HSP. Spaced seeds increase the sensitivity of homology search significantly without slowing down the runtime too much. Moreover, we can train a seed for a particular task, and use several spaced seeds to increase the sensitivity [20].

Existing homology search tools can not be used to solve the homologous gene finding

problem directly for several reasons. First, predicted homologous region boundaries found by sequence alignment software are most likely incorrect splice sites, because homology search tools always try to extend exons a few nucleotides to achieve better alignment scores. Second, the homology search tools may not be able to find micro-exons (short exons with length of up to 25 nucleotides) [36], inserted exons, deleted exons or highly mutated exons. Finally, the homology search tools cannot distinguish true exons from duplicated partial exons, and the same thing with genes. In this thesis, the above mentioned issues are considered when designing the homologous gene finding algorithm.

1.3.2 Gene Mapping and Alignment

Given a spliced mRNA sequence of a gene and a genomic sequence containing exactly that gene, the gene mapping and alignment problem is to map and align the mRNA sequence to exons of the gene in the genomic sequence. Some popular gene mapping and alignment tools include sim4 [12], Spidey [38], BLAT [17] and GMAP [39]. The algorithms of these gene mapping and alignment tools can be divided into following steps:

1. **Locate a target region:** An alignment algorithm or a hash table is used firstly to find all HSPs [12, 38] or k-mer hits (k contiguous matches) [17, 39] between the mRNA sequence and the genomic sequence. If a region in the genomic sequence contains a high density of HSPs or k-mer hits, it is very likely that the gene or a close homolog is located in that region. Thus, a single region is chosen to find approximate exons in the next step.

2. **Find approximate exons:** In this step, approximate exons (may not have correct splice sites) are found using HSPs or clustering k-mer hits from the previous step. This simple approach usually works well because the mRNA sequence is nearly identical (better than 95% identical [17]) to exons in the genomic sequence.

3. **Adjust splice sites:** Splice site signals are used to find correct splice sites with the constrant that adjacent exons have to follow each other precisely in the mRNA sequence. If there is a gap between two adjacent approximate exons in the mRNA sequence, splice sites are adjusted by either greedy or alignment algorithms. Generally,

greedy algorithms [12, 38] try to extend exons toward each other until they overlap, and then adjust splice sites in the overlapped region. On the other hand, alignment algorithms use either seed based algorithms [17] or dynamic programming algorithms [39] to fill the gap by finding short missed exons between adjacent approximate exons in the genomic sequence.

The gene mapping and alignment tools work well as long as the mRNA and the genomic sequence are from the same species. Although these tools support cross species mode, the accuracy drops significantly even for close species. This is mainly because the assumption that the mRNA sequence is nearly identical to the exons in the genomic sequence no longer holds. It has been reported that if sequence identity is lower than 90% [17], the gene mapping and alignment tools have poor accuracy. Obviously, a lot of homologous gene pairs have sequence identities lower than 90%. Moreover, the gene mapping and alignment tools do not look for inserted exons. Therefore, gene mapping and alignment tools are not suitable to solve the homologous gene finding problem.

1.3.3 Gene Finding

Given one or more genomic sequences, the gene finding problem is to find all genes in the genomic sequences. During the past ten years of study of the gene finding problem, many algorithms have been introduced. Among the most successful algorithms are the ones based on hidden Markov models (HMM). In this section, we focus on HMM-based gene finding algorithms, and briefly introduce the basic idea of them. Finally, we discuss differences between the gene finding problem and the homologous gene finding problem.

1.3.3.1 Hidden Markov Models

A hidden Markov model (HMM) is a generative probabilistic model for modeling sequences, such as DNA sequences. An HMM is a combination of states and transitions. Each state has its own initial probabilities, transition probabilities, and emission probabilities. An initial probability of a state is the probability that the state is used as the initial state. An transition probability of a state is the probability that the current state of the HMM transits from the

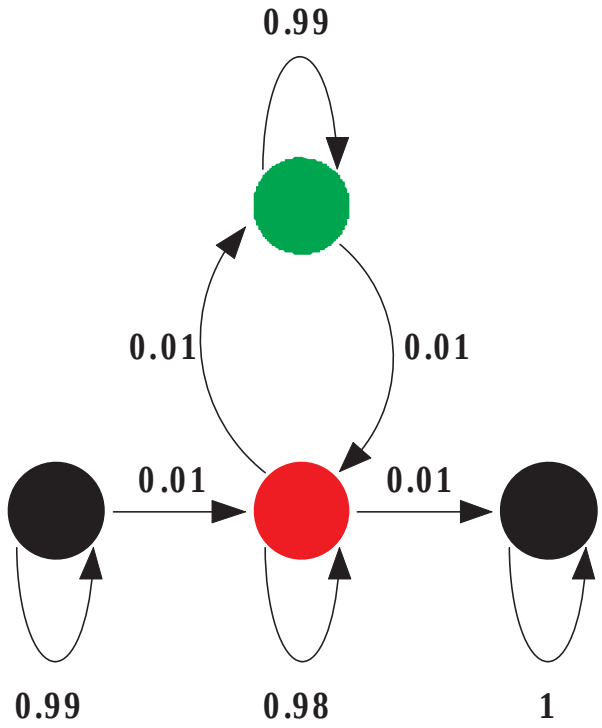
state to itself or another state. An emission probability of a state is the probability that a symbol is emitted by the state.

A sequence is emitted by emitting symbols one by one from a sequence of states, called state path. Initially, the current state of the HMM starts from an initial state. Every time after the current state emits a symbol, the current state transits to the next state. The current state keeps emitting a symbol and transiting to the next state until a final state is reached. Then, the sequence of the current states forms a state path, and a sequence is emitted by the state path. The probability that the sequence is emitted by the state path is computed as the product of the emission probabilities and the transition probabilities in the state path.

Hidden Markov models have been widely used in the gene finding problem. We use states to represent structural properties of genes, such as exons, introns or intergenic regions. Each state may emit a nucleotide from $\{A, C, G, T\}$. We use transitions to represent structural property changes. Thus, each state path emitting a genomic sequence represents a gene structures of the genomic sequence.

Each state has a table of the initial probabilities, the transition probabilities, and the emission probabilities. We define $P_i(s)$ as the initial probability that state s is used as the initial state, and set $P_i(s) = 0$ if state s cannot be used as the initial state. We define $P_t(s, s')$ as the transition probability that the current state transits from state s to state s' directly, and set $P_t(s, s') = 0$ if the current state cannot transit from state s to s' directly. We define $P_e(n, s)$ as the emission probability that nucleotide n is emitted by state s, and set $P_e(n, s) = 0$ if nucleotide n cannot be emitted by state s.

Figure 1.6 shows a simple HMM modeling a DNA sequence with a single gene. The circles represent states, and the arrows represent transitions. The red state represents exons, the green state represents introns, and the black states represent intergenic regions. In order to distinguish the two intergenic states, we simply call the black state on the left side the initial state, and the black state on the right side the final state. The the emission probabilities of the exon state are shown in the red table; the the emission probabilities of the intron state are shown in the green table; and the the emission probabilities of intergenic states are shown in the black table. In fact, each intergenic state should have its own emission



Nucleotide	Probability
A	0.23
C	0.27
G	0.27
T	0.23

Nucleotide	Probability
A	0.26
C	0.22
G	0.24
T	0.28

Nucleotide	Probability
A	0.25
C	0.25
G	0.25
T	0.25

Figure 1.6: Simple HMM for a single gene

probability table, but only one table is shown here. Finally, the probability on each arrow represents the transition probability.

Figure 1.7 shows an example of a simple state path through the HMM shown in Figure 1.6. The state path represents a gene structure for the short genomic sequence "CATGTAA". It basically shows that "ATG" is a gene with a single exon, and the rest of the genomic sequence is not translated.

Let $G = G_{1...L}$ be a genomic sequence of length L, and $H = H_{1...L}$ be a state path of length L. Given genomic sequence G and state path H, we use the following formula to compute joint probability $P_{HMM}(G, H)$ that G is emitted by H.

$$P_{HMM}(G, H) = P_i(H_1) \cdot \prod_{i \in [1...L-1]} P_t(H_i, H_{i+1}) \cdot \prod_{i \in [1...L]} P_e(G_i, H_i). \tag{1.1}$$

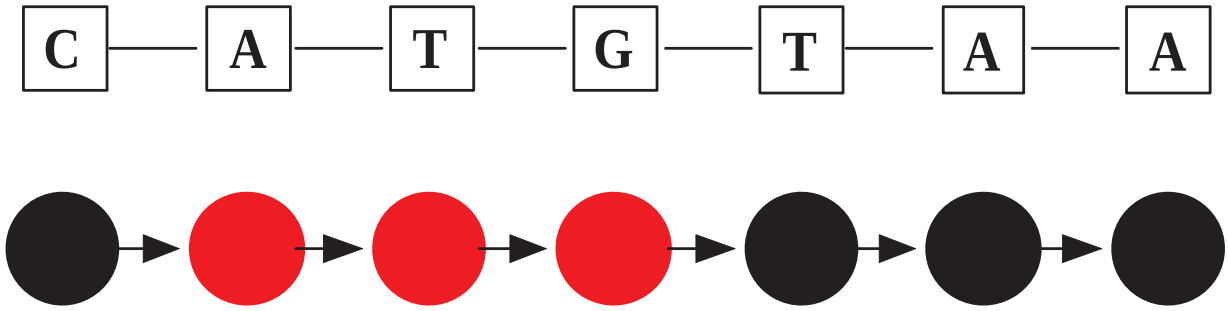


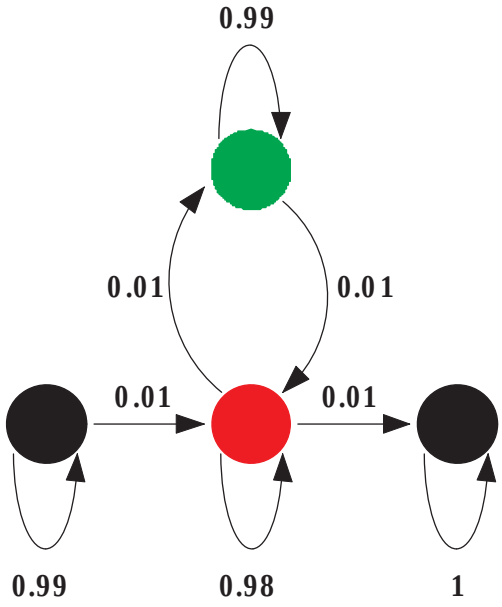
Figure 1.7: Simple state path for the DNA sequence "CATGTAA".

Given a genomic sequence G, we want to find the optimal state path H that maximizes the posterior probability $P(H|G) = P_{HMM}(G, H)/P(G)$. Since $P(G)$ is a constant for a given sequence G, maximizing $P(H|G)$ is equivalent to maximizing $P_{HMM}(G, H)$. Thus, the idea of HMM-based gene finding is to find the state path with the maximum $P_{HMM}(G, H)$, and such a state path represents the best candidate gene structure [30].

In this thesis, we focus on finding a unique gene in the genomic sequence. Thus, we use only the HMMs modeling a single gene, and we simply assume that there is a unique initial state and a unique final state in the HMM (like the one shown in Figure 1.6). Algorithms for the HMMs modeling multiple genes with multiple initial and final states can be derived similarly.

1.3.3.2 Higher-Order HMMs

The HMM shown in Figure 1.6 assumes that the emission probabilities are independent from previously emitted nucleotides. This assumption is obviously false. Exons are made up of codons of triples of nucleotides, and nucleotides within each codon are strongly dependent. There are also dependencies between neighboring codons. Therefore, conditional probabilities are used to emit the emission probabilities of higher-order HMMs. If the emission probabilities of state s depend on the previous N_s nucleotides, where $N_s \geq 0$, we say that state s has order N_s. Different states of the same higher order HMM many have different orders. Let G be the genomic sequence. We define $P_e(G_i, s|G_{i-N_s \ldots i-1})$ to be the emission probability that nucleotide G_i is emitted by state s knowing that the previous N_s emitted



Previous Nucleotide	Current Nucleotide	Probability
A	A	0.24
	C	0.25
	G	0.30
	T	0.21
C	A	0.25
	C	0.30
	G	0.19
	T	0.26
G	A	0.23
	C	0.29
	G	0.29
	T	0.19
T	A	0.17
	C	0.27
	G	0.33
	T	0.23

Figure 1.8: Higher-order HMM for a single gene, whose exon state has order 1.

nucleotides are $G_{i-N_s \ldots i-1}$. Using higher-order HMMs, we compute $P_{HMM}(G, H)$ as follows:

$$P_{HMM}(G, H) = P_i(H_1) \cdot \prod_{i \in [1 \ldots L-1]} P_t(H_i, H_{i+1}) \cdot \prod_{i \in [1 \ldots L]} P_e(G_i, H_i | G_{i-N_{H_i} \ldots i-1}), \tag{1.2}$$

where we deal with $i < N_{H_i}$ sensibly.

Figure 1.8 shows an example of a higher-order HMM. The major differences between it and the simple HMM in Figure 1.6 are the emission probability tables. In order to make things simple, only the emission probability table assigned to the exon state is shown, and the exon state has order 1.

Using a higher-order HMM improves the sensitivity and specificity of gene finding. However, using a very high order increases the number of the emission probabilities of the state. This may cause overfitting due to the limited training resources. Thus, we should choose the order of HMM states carefully based on our training resources.

1.3.3.3 HMM Training

After we build an HMM topology and choose the initial probabilities based on our understanding of gene structures, we need to estimate the emission and the transition probabilities. This process is called HMM training.

The simplest HMM training method is the maximum likelihood estimation method to compute the frequency of each transition and emission in a training data set. This method works only if we have enough genes with complete reference sequences and CDS annotations, and there is a one-to-one correspondence between an annotation and a state path. Let $T(x, y)$ be the number of transitions from state x to state y observed in the training data set, and $E(n, s)$ be the number of emissions of nucleotide n at state s observed in training data set. Let S be the set of all states in the HMM. We set HMM probabilities as follows:

$$P_t(x, y) = \frac{T(x, y)}{\sum_{i \in S} T(x, i)}, \tag{1.3}$$

$$P_e(n, s) = \frac{E(n, s)}{\sum_{j \in \{A,C,G,T\}} E(j, s)}. \tag{1.4}$$

Ideally, we would have enough training resource, and the trained HMM probabilities would be very close to the true values. However, this is sometimes not the case, and overfitting occurs. Overfitting causes good performance on the training data, but worse performance on unseen data. In order to avoid overfitting, we use a pseudocount $C > 0$, when computing the transition and emission probabilities:

$$P_t(x, y) = \frac{T(x, y) + C}{\sum_{i \in S} (T(x, i) + C)}, \tag{1.5}$$

$$P_e(n, s) = \frac{E(n, s) + C}{\sum_{j \in \{A,C,G,T\}} (E(j, s) + C)}. \tag{1.6}$$

1.3.3.4 Viterbi Algorithm

In the previous sections, we described how to build and train an HMM. Here, we review the Viterbi algorithm that finds the optimal state path that emits a genomic sequence with the highest probability. The algorithm uses dynamic programming: we remember the optimal state path ending at each state emitting each prefix of the genomic sequence.

Let G be the genomic sequence, and L be the length of G. Let S be the set of all states in the HMM, and K be the number of states in S. Let S_1 be the initial state, and S_K be the final state. Let $P_{DP}(n, s)$ be the highest probability of a state path that emits the prefix $G_{1...n}$, and emits G_n in state s. Then, it is easy to compute $P_{DP}(n, s)$ as follows:

$$P_{DP}(n,s) = \begin{cases} 1, \text{ if } n = 1 \text{ and } s = S_1, \\ 0, \text{ if } n = 1 \text{ and } s \neq S_1, \\ P_e(G_n, s) \cdot \max_{i \in S}(P_{DP}(n-1, i) \cdot P_t(i, s)), \text{ if } n > 1. \end{cases} \tag{1.7}$$

The order of computation of the Viterbi algorithm is from $n = 1$ to $n = L$. The optimal state path is constructed by tracing back pointers from $P_{DP}(G_L, S_K)$ to $P_{DP}(G_1, S_1)$. Since we need to compute $\theta(LK)$ probabilities, and we require $\theta(K)$ time to compute each probability, the runtime complexity of the Viterbi algorithm is $\theta(LK^2)$.

1.3.3.5 HMM-based Gene Finding Algorithms

We can divide HMM-based gene finding algorithms into three categories: single-genome gene finding algorithms, dual-genome gene finding algorithms, and multi-genome gene finding algorithms.

Single-genome gene finding algorithms require only a genomic sequence and training data as input, and then find genes using an approach similar to the one introduced in this Chapter. Two popular algorithms are GENSCAN [9] and AUGUSTUS [34].

Dual-genome gene finding algorithms are based on the assumption that protein coding regions should be better conserved than non-coding regions. In addition to the training data, they requires two genomic sequences of pairwise alignments between the two genomic sequences as input. DOUBLESCAN [23] and SLAM [1, 28] use pair-HMMs, which emit pairwise alignments instead of sequences, to find and align genes in both genomes at the

same time. TWINSCAN [19, 11] and SGP2 [29] model pre-processed pairwise alignment information into HMM probabilities to find genes in one or both genomes at the same time.

Multi-genome gene finding algorithms are the newest algorithms in the gene finding problem. EXONIPHY [31] and N-SCAN (TWINSCAN 3.0) [13] use pre-processed multi-genome alignments to improve the accuracy of gene finding. They build evolutionary information into an HMM as a phylogenic hidden Markov model (phylo-HMMs). ExonHunter [6] uses different kinds of sequences as evidence to improve the accuracy of gene finding. This evidence includes ESTs, cDNAs and proteins from other species.

Existing gene finding tools are not suitable to solve the homologous gene finding problem because they will find all genes instead of only the homologous genes in the genomic sequence. Filtering the homologous genes after finding all genes is computationally intensive, and thus not a good idea. Moreover, gene finding tools do not use the query gene information as much as our homologous gene finding algorithm, and the query gene information helps to improve the accuracy of homologous gene finding, as demonstrated in Sections 3.5 to 3.10.

1.3.4 Current Homologous Gene Finding Tools

GeneWise [5] and Projector [24] are two pair-HMM-based homologous gene finding tools. In this section, we briefly summary both of them.

GeneWise [5] takes a protein sequence and a genomic sequence as input. It uses the protein sequence to guide homologous gene finding in the genomic sequence. GeneWise combines a protein-DNA alignment pair-HMM and a single-genome gene finding HMM. If the alignment pair-HMM contains m_s states and m_t transitions, and the gene finding HMM contains n_s states and n_t transitions, the combined model contains $m_s n_s$ states and $m_t n_t$ transitions. Thus, if the protein-DNA alignment pair-HMM and the single-genome gene finding HMM are complicated, the combined HMM is very complicated, and very likely causes overfitting.

Projector [24] accepts a target genomic sequence, an informant genomic sequence, and CDS annotations of an informant gene in the informant genomic sequence as input. Similar to pair-HMM-based gene finding tools, Projector uses a pair-HMM to find homologous genes in the target genomic sequence, and align the two genomic sequences at the same time.

Different from pair-HMM-based gene finding tools, Projector restricts transitions to satisfy the CDS annotations of the informant gene. Since Projector requires a training data set of homologous genes with complete reference sequences and CDS annotations between the target and informant species, it might be difficult to choose appropriate training data sets.

Chapter 2

Finding Homologous Genes

In this chapter, we introduce a new algorithm to solve the homologous gene finding problem. For a given query gene and a target genomic sequence, the goal is to find all genes (also called target genes) that are homologous to the query gene in the target genomic sequence. We borrow the idea of gene mapping and finding algorithms, and use a homology search algorithm and a gene finding algorithm in our homologous search. We divide our homologous gene finding algorithm into four steps.

1. **Finding HSPs:** In this step, we find HSPs between the query gene and the target genomic sequence using a seed-based homology search algorithm. We also filter out the HSPs with low alignment scores to locate accurate target regions in the next step.

2. **Locating Target Regions:** Using the HSPs found in the previous step, we locate regions of the target genomic sequence that we believe contain a target gene. These regions are called target regions. By locating the target regions before finding gene candidates, both the accuracy and speed of homologous gene finding are improved.

3. **Finding Gene Candidates:** For each target region, we predict several gene candidates that are homologous to the query gene. We use an HMM-based gene finding algorithm that is modified for this task. Using HMM-based gene finding, we can recover micro-exons that cannot be recovered by homology search algorithms.

4. **Assessment of Gene Candidates:** Finally, we rank the candidates from the previous step by aligning them to the query gene using global protein alignment.

We explain the details of each step in the following sections.

2.1 Finding HSPs

The first step of our homologous gene finding algorithm is to find the HSPs between the query gene and the target genomic sequence. We adopt a seed-based homology search algorithm, as described in Section 1.3.1. The difference between our HSP finding and general homology search is that we are interested in only the HSPs located in the coding regions of the target genomic sequence. Therefore, we need to choose the homology search algorithm that performs well in the coding regions.

Homologous genes do not necessarily contain highly similar CDSs. This is because multiple codons can be translated to the same amino acid, as indicated in Section 1.1. Thus, translated proteins may be very similar, even if they originate from very different CDSs. This raises a problem that homologous regions may not contain contiguous matches, which are used as seeds by some homology search algorithms, such as BLAST family [2, 3, 41, 40, 25]. There are two approaches described below to solve the problem:

1. Spaced seed models, designed especially for coding sequences, and multiple spaced seed models can be employed to improve the sensitivity. This approach was studied in work on optimal spaced seeds [7] and PatternHunter [20].

2. The protein sequence, encoded by the query gene, and the target genomic sequence can be aligned. The problem of finding the HSPs between a protein sequence and a nucleotide sequence was studied in tBLASTx [25, 40] and tPatternHunter [18].

Comparing the above two approaches, the multiple spaced seed approach performs faster with an acceptable good sensitivity, while the protein-nucleotide alignment approach performs slower with a better sensitivity. In our work, we choose the protein-nucleotide alignment approach mainly for two reasons. First, finding the HSPs with a high sensitivity is critical for locating the accurate target regions in the next step. Second, the bottleneck of the homologous gene finding is the HMM-based gene finding step, and the running time used for the homology search is not a limiting factor. The protein-nucleotide alignment approach is examined in Section 3.3.

Even though we choose to use the protein-nucleotide alignment approach, the multiple spaced seed approach can be adopted easily when speed is critical. In order to avoid harmful codon frame-shifts, as described in Section 1.1, we do not allow gaps in the HSPs. Disallowing gaps can also speed up the homology search algorithm.

We observed regions of the target genomic sequence that contains many HSPs with low alignment scores. These HSPs are likely emitted by a combination of duplications, mutations and indels during evolution. For example, a complete gene or a part of a gene was first duplicated, and then a lot of mutations, and indels happened in the duplicated region. As a result, we find many HSPs with low alignment scores in the region, but there is no functional gene in the region. We are not interested in these regions, and these regions are not desirable in the next step. In order to locate accurate target regions, we filter out the HSPs with alignment scores lower than a threshold.

We choose the threshold carefully so that most of the noise is filtered out, and most of the useful information is retained. Let TP_{in} be the number of true positives retained, and TP_{out} be the number of true positives filtered out after filtering. Let FP_{in} be the number of false positives retained, and FP_{out} be the number of false positives filtered out after filtering. Here, an HSP is a true positive if it hits a homologous region, and an HSP is false positive if it does not hit a homologous region. We pick a threshold to maximize the following value:

$$\frac{FP_{out}}{FP_{in}+FP_{out}} + \frac{TP_{in}}{TP_{in}+TP_{out}}. \tag{2.1}$$

In this equation, $\frac{FP_{out}}{FP_{in}+FP_{out}}$ is the percentage of noise filtered out, and $\frac{TP_{in}}{TP_{in}+TP_{out}}$ is the percentage of useful information retained. We maximize the sum of the percentage of noise filtered out and the percentage of useful information retained.

There are two advantages of filtering HSPs in this way. First, while most of the noise is filtered out, and most of the useful information is retained, we likely increases the weight of useful information, as well as the accuracy to locate target regions. Second, by keeping most of the useful information, we keep the option to use this information in the gene finding step. However, we currently do not use this information in the gene finding step, and we will use it in the future. The effects of filtering the HSPs are discussed in Section 3.4.

In summary, we translate the query gene to a protein sequence, and find high scoring

HSPs without gaps between the protein sequence and the target genomic sequence. These HSPs are used in the next step to locate the target regions.

2.2 Locating Target Regions

A target region is a region of the target genomic sequence that is likely to contain a target gene. Ideally, we should find several HSPs in the true target region that contains a target gene, because there should be homologous regions between the query gene and the target gene. Thus, we locate target regions by finding the HSP clusters with high total alignment scores. This approach works well as long as we find enough HSPs in the true target region.

To locate target regions, we cluster the HSPs using the nearest neighbor clustering algorithm. Two HSPs belong to the same cluster, if and only if they are both located on the same strand of the target genomic sequence, and their distance is smaller than a given threshold. To choose a reasonable threshold, we download annotations of known genes from UCSC Genome Bioinformatics Site [35], and study the lengths of the 2051 known genes on human chromosome 6. We set the threshold as $650,000$, which is approximately half of the length of the longest known gene (with gene ID NM_013988 [35]) on human chromosome 6. Then, we compute the score of each HSP cluster as the sum of the alignment scores of individual HSPs within the cluster. If a HSP cluster scores highly, it is likely that there is a target gene in that region. Thus, we pick several highest scoring clusters to locate the target regions. We study several variants of this method in Section 3.5.

We can only use HSP clusters to estimate the approximate location of the target region. It is difficult to estimate where the target region starts and ends, especially if the $5'$ or $3'$ ends of the query gene and a target gene are not homologous. This is possible if the function of these proteins is determined by the middle of the query gene and that of a target gene, and the other regions are not conserved. The most obvious solution is to define a big enough constant for the target region size. However, finding a short gene in a large region is obviously slower than in a region of approximately correct size. Such a search also adds too much noise that causes poor gene finding results. Therefore, we describe the target region by a nucleotide near the middle of the target region, and the flexible ranges of the target region on the $5'$ and $3'$ ends. In particular, we describe each target region by three terms:

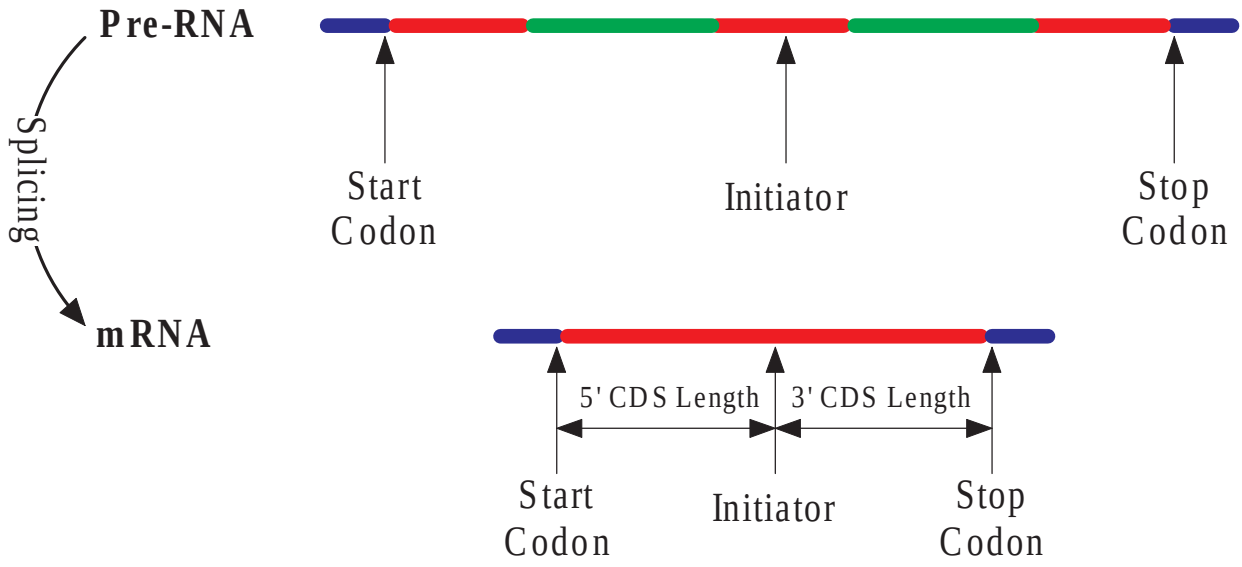


Figure 2.1: Initiator and its associated 5' and 3' CDS lengths

Initiator: Initiator I_0 is a nucleotide in the target genomic sequence which is likely to be located within an exon. The initiator is chosen as the middle nucleotide of the highest scoring HSP in the target region, as shown in Figure 2.1. We can also predict in which codon frame the initiator lies. The frame should match the codon frame of the nucleotide, to which the initiator is aligned, in the query gene. We will use this information later to determine where to start HMM dynamic programming (the base case).

$5'$ **CDS Length:** The $5'$ CDS length $\lambda_{5'}$, associated with I_0, is the expected number of coding nucleotides on the $5'$ side of I_0. We compute $\lambda_{5'}$ as the number of coding nucleotides on the $5'$ side of I_0 in the query gene, as shown in Figure 2.1. We will use it later to determine when to stop our enhanced Viterbi algorithm in the $5'$ direction.

$3'$ **CDS Length:** The $3'$ CDS length $\lambda_{3'}$, associated with I_0, is the expected number of coding nucleotides on the $3'$ side of I_0. We compute $\lambda_{3'}$ as the number of coding nucleotides on the $3'$ side of I_0 in the query gene, as shown in Figure 2.1. Similarly, we will use this information to determine when to stop our enhanced Viterbi algorithm in the $3'$ direction.

The idea behind the use of I_0, $\lambda_{5'}$ and $\lambda_{3'}$ to describe a target genomic sequence is based on the observation that the CDS lengths generally do not change much, as demonstrated in Section 3.9. We also observed that the gene lengths can change significantly during the evolution of homologous genes, as demonstrated in Section 3.9.

There is no guarantee that the highest scoring HSP always overlaps an exon of the target gene, and thus we might choose an incorrect initiator from the highest scoring HSP. To avoid this problem, we choose several initiators from several highest scoring HSPs for each target region. We study several variants of this method in Section 3.6.

In summary, we first locate target regions by clustering the HSPs, and then choose several initiators with their associated $5'$ and $3'$ CDS lengths for each target region. Locating the target regions before gene finding can improve both the accuracy and speed of homologous gene finding. Both the accuracy and the speed are improved because those noise regions, which we are not interested in, are filtered out before the gene finding step.

2.3 Finding Homologous Gene Candidates

In this step, we find a candidate gene structure for each initiator and its associated $5'$ and $3'$ CDS lengths. We later evaluate these gene candidates and select several target genes. To find candidate gene structures, we introduce a new HMM-based gene finding algorithm, and enhance previous HMM-based gene finding algorithms as follows:

Biased Training: We train an HMM to be biased toward the query gene to improve the accuracy of the homologous gene finding.

Enhanced Viterbi Algorithm: We modify the Viterbi algorithm to be able to find gene structures starting from the initiator, and progressing in both the $5'$ and $3'$ directions.

CDS Length Penalty: We introduce the new CDS length penalty as a measure of how the CDS lengths of the query gene and its homologous gene vary. We also combine the HMM state path probability and the CDS length penalty to define a new gene scoring system to further improve the accuracy of the homologous gene finding.

Stopping Condition: Using the new CDS length penalty and the new scoring system, we stop the enhanced Viterbi algorithm when a stopping condition is reached. The new stopping condition improves the speed of homologous gene finding without harming the accuracy.

In Section 2.3.1, we explain the HMM used to predict gene candidates. From Section 2.3.2 to Section 2.3.6, we explain the details of above mentioned enhancements.

2.3.1 Hidden Markov Model for Homologous Gene Finding

We use the HMM in Figure 2.2 to find candidate gene structures. In the figure, circles represent HMM states; triangles represent signal submodels; and arrows represent transitions. State S is the silent initial state, and state F is the silent final state. The red states represent exons, and the green states represent introns. The exon and intron states are of the 4^{th} order. The left triangle represents the start signal submodel, and the right triangle represents the stop signal submodel. The orange triangles represent the donor signal submodels, and the pink triangles represent the acceptor signal submodels. The HMM in Figure 2.2 is a single gene submodel that does not have any states representing either UTR or intergenic regions. The overall performance of this HMM for homologous gene finding is studied in Section 3.8.

The start and stop signal submodels are depicted in Figure 2.3 by notations similar to those in Figure 2.2. In addition, the black states represent the regions before the start codon or after the stop codon. The states, which show a single symbol inside, can emit only the nucleotide symbolized by that symbol. Compared to the states in Figure 2.2, there is less data to train the emission probabilities of the states in the start and stop signal submodels. This occurs because the number of nucleotides that can be used to train a signal state is approximately the same as the number of exons, whereas the number of nucleotides that can be used to train an exon or an intron state is approximately the total length of the exons or introns, respectively. For this reason, we use states of the 2^{nd} order for the start and stop signal submodels to avoid overfitting.

The donor and acceptor signal submodels are shown in Figure 2.4 by notations similar to those in Figure 2.3. Similarly as in the case of the start and stop signal submodels, we

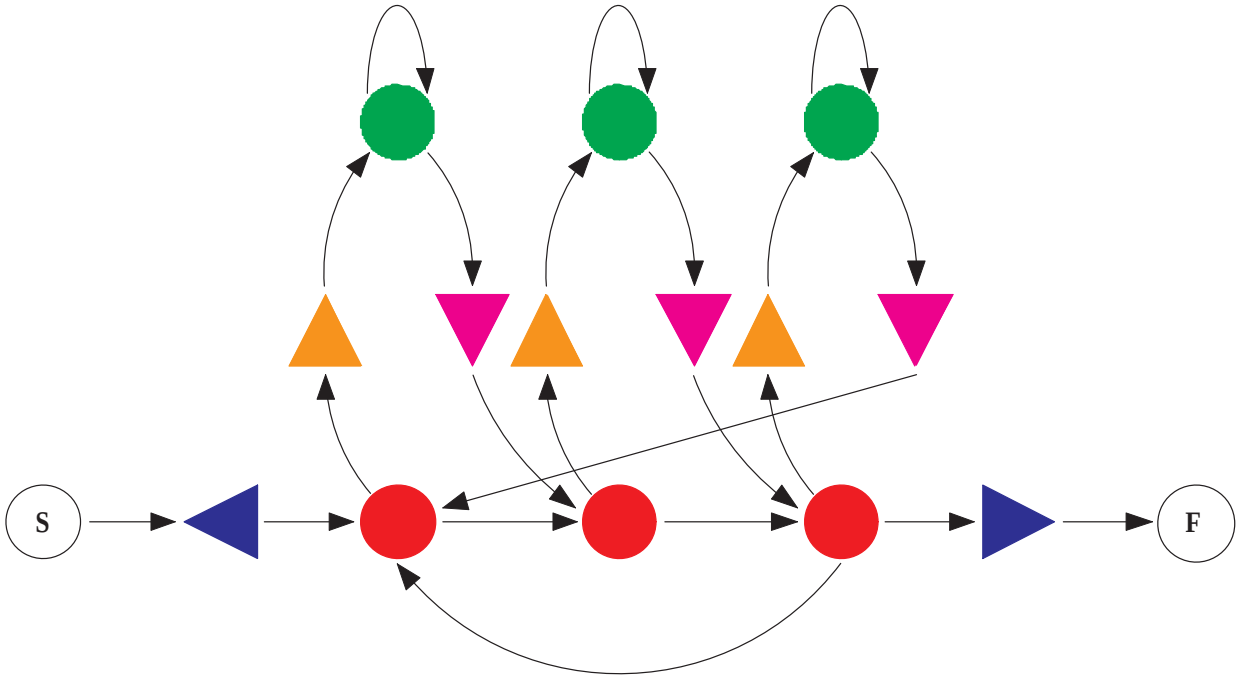


Figure 2.2: HMM used for homologous gene finding: State S is the silent initial state, and state F is the silent final state. The red states are the exon states, and the green states are intron states. The left triangle is the start signal submodel, and the right triangle is the stop signal submodel. The orange triangles are the donor signal submodels, and the pink triangles are the acceptor signal submodels.

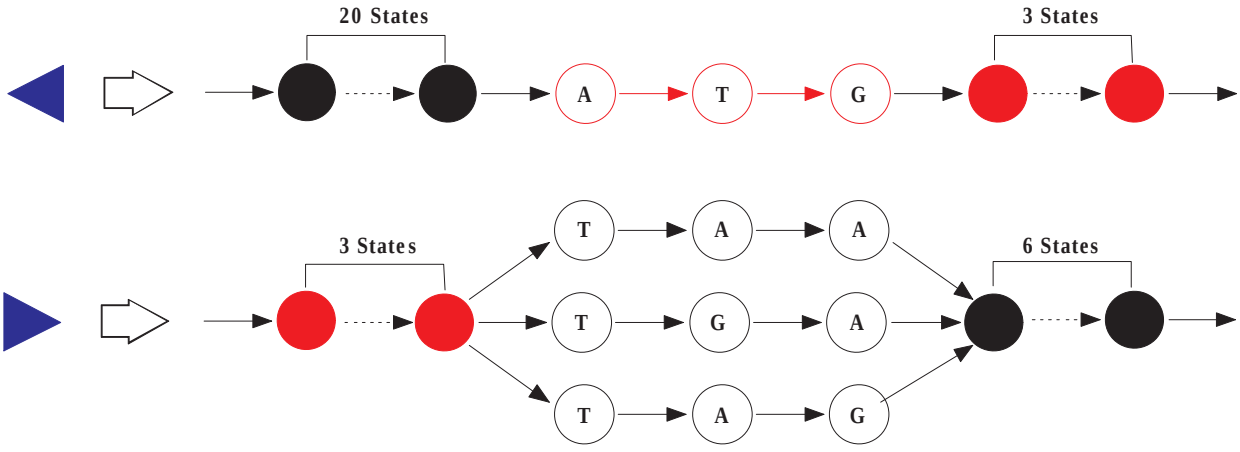


Figure 2.3: Start/stop signal submodels: The red states are the exon states, and the black states are intergenic/UTR states. The states, which show a single symbol inside, can emit only the nucleotide symbolized by that symbol.

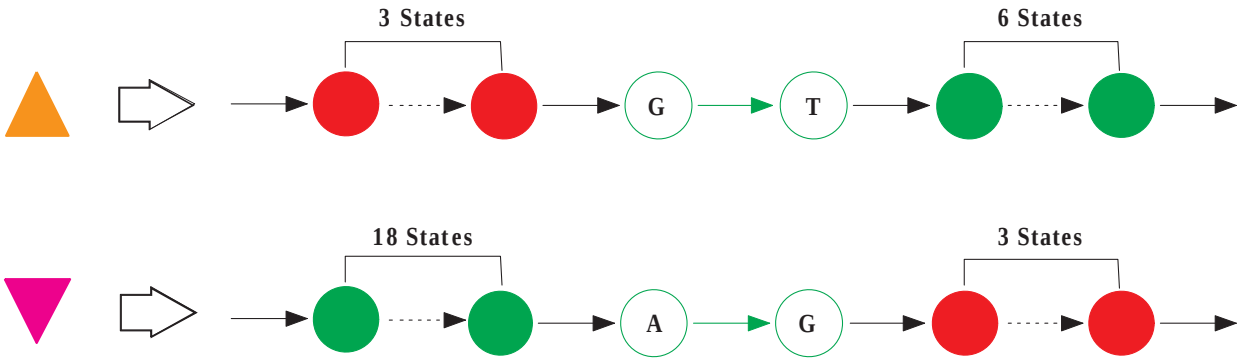


Figure 2.4: Donor/acceptor signal submodels: The red states are the exon states, and the green states are intron states. The states, which show a single symbol inside, can emit only the nucleotide symbolized by that symbol.

use states of order 2 in the donor and acceptor signal submodels.

2.3.2 Biased Training

Each gene has its own structural properties, such as distribution of nucleotides in the CDS. There is a good chance that many of these properties are conserved for homologous genes during evolution, as studied in Section 3.7. Based on this assumption, we use the query gene to train an HMM to be biased toward the query gene.

Let $P_t(x, y)$ be the transition probability from state x to state y, and $P_e(n, s)$ be the emission probability that nucleotide n is emitted by state s. Let $T(x, y)$ be the number of transitions from state x to state y observed in the training data set for the target species, and $T'(x, y)$ be the number of transitions from state x to state y observed in the query gene. Similarly, let $E(n, s)$ be the number of nucleotides n emitted by state s observed in training data set for the target species, and $E'(n, s)$ be the number of nucleotides n emitted by the state s observed in the query gene. Let M be the number of genes in the training data set, and Z be a weight. Let S be the set of all states in the HMM, and C be the pseudocount.

We set HMM probabilities as follows:

$$P_t(x,y) = \frac{T(x,y) + Z \cdot T'(x,y) + C}{\sum_{i \in S}(T(x,i) + Z \cdot T'(x,i) + C)}, \tag{2.2}$$

$$P_e(n,s) = \frac{E(n,s) + Z \cdot E'(n,s) + C}{\sum_{j \in \{A,C,G,T\}}(E(j,s) + Z \cdot E'(j,s) + C)}. \tag{2.3}$$

Intuitively, the biased training with above formulas is equivalent to training on a virtual training data set of M genes from the training data set and Z copies of the query gene. In this virtual training data set, the query gene weights $\frac{Z}{M+Z}$. To keep the query gene weight independent from M, we introduce another weight W, and set $Z = WM$. Thus, the query gene weights $\frac{Z}{M+Z} = \frac{WM}{M+WM} = \frac{W}{1+W}$ in the virtual training data set.

We use weight W to adjust how much the HMM is biased toward the query gene. If W is too big, we have overfitting, because the HMM probabilities are biased too much. If W is too small, the sensitivity is not improved much, because the HMM probabilities are not biased much. Therefore, we carefully choose W based on the distance between the query and the target species to avoid overfitting, and to maximize the sensitivity of the homologous gene finding, as studied in Section 3.8.

The idea of biased training is simple, but it improves the sensitivity of homologous gene finding significantly when the homologous genes are conserved, as shown in Section 3.8. Biased training builds the query gene information into the HMM without increasing the complexity of the model, and thus it does not slow down the gene finding algorithm. However, not all HMM probabilities should be biased. In our experiments described in Section 3.7, we observed that the transition probabilities may not be conserved. In addition, the emission probabilities of the intron states are not as well conserved as the emission probabilities of the exon states. For now, we bias only the emission probabilities of the exon states and signal states. In fact, our study on biased training is not complete. We are still looking for the most efficient way to train an HMM to be biased toward the query gene, and our future work in this area is described in Section 4.2.

2.3.3 Enhanced Viterbi Algorithm

We have already described how the Viterbi algorithm finds the optimal HMM state path that represents the best gene structure in Section 1.3.3.4. Here, we assume that there is a unique gene in the genomic sequence, and there is a unique initial state and a unique final state in the HMM. Let G be the genomic sequence, and L be the length of G. Let S be the set of all states in the HMM, and K be the number of states in S. Without loss of generality, we can say that S_1 is the initial state, and S_K is the final state. Let $P_{DP}(n, s)$ be the highest probability of a state path that emits the prefix $G_{1...n}$, and emits G_n in state s. The Viterbi algorithm computes $P_{DP}(n, s)$ from $n = 1$ to $n = L$, and the probability that G is emitted by the optimal state path is $P_{DP}(L, S_K)$. The optimal state path can be constructed by tracing the back pointers from $P_{DP}(L, S_K)$ to $P_{DP}(1, S_1)$. For our HMM (Figure 2.2), the Viterbi algorithm ideally starts a few nucleotides before the start codon, and finishes a few nucleotides after the stop codon. However, finding the start and stop codons is the most challenging part of finding candidate gene structures. Therefore, we cannot apply the Viterbi algorithm directly.

Recall that we describe the target region by a nucleotide (I_0) near the middle of the target region, and flexible ranges ($\lambda_{5'}$ and $\lambda_{3'}$) of the target region on the $5'$ and $3'$ ends. Here, we introduce a new dynamic programming algorithm that starts from the initiator, and proceeds in both directions. Without loss of generality, we can say that S_{I_0} is the exon state representing the codon frame of the initiator I_0. If $n < I_0$, let $P_{DP'}(n, s)$ be the highest probability of a state path that emits the genomic sequence $G_{n...I_0}$, and emits G_n in state s. If $n > I_0$, let $P_{DP'}(n, s)$ be the highest probability of a state path that emits the genomic sequence $G_{I_0...n}$, and emits G_n in state s. The enhanced Viterbi algorithm uses the following recurrence:

$$P_{DP'}(n,s) = \begin{cases} 1, \text{ if } n = I_0 \text{ and } s = S_{I_0}, \\ 0, \text{ if } n = I_0 \text{ and } s \neq S_{I_0}, \\ P_e(G_n, s) \cdot \max\limits_{i \in S}(P_{DP'}(n-1, i) \cdot P_t(i, s)), \text{ if } n > I_0, \\ P_e(G_n, s) \cdot \max\limits_{i \in S}(P_{DP'}(n+1, i) \cdot P_t(s, i)), \text{ if } n < I_0. \end{cases} \tag{2.4}$$

The enhanced Viterbi algorithm computes $P_{DP'}(n, s)$ from $n = I_0$ to $n = 1$, and from

$n = I_0$ to $n = L$. For convenience, if $n < I_0$, we can say that $H_{DP'}(n, s)$ is the optimal state path that emits G_n in state s, and emits the genomic sequence $G_{n...I_0}$ with the highest probability of $P_{DP'}(n, s)$. Similarly, if $n > I_0$, we can say that $H_{DP'}(n, s)$ is the optimal state path that emits G_n in state s, and emits the genomic sequence $G_{n...I_0}$ with the highest probability of $P_{DP'}(n, s)$. $H_{DP'}(n, s)$ can be constructed by tracing the back pointers from $P_{DP'}(n, s)$ to $P_{DP'}(I_0, S_{I_0})$, but we did not compute $H_{DP'}(n, s)$ in practice. We introduce $H_{DP'}(n, s)$ in order to simplify explanations, and we choose symbol H as the last character of "patH". In practice, we construct the optimal state path by concatenating $H_{DP'}(1, S_1)$ and $H_{DP'}(L, S_K)$. Therefore, we can imagine the enhanced Viterbi algorithm as the Viterbi algorithm to find the optimal state path subject to the additional constraint that G_{I_0} is emitted by S_{I_0}. Until now, we have not mentioned when to stop the ehnahced Viterbi algorithm. We introduce the stopping condition to determine when to stop the enhanced Viterbi algorithm in Section 2.3.6.

The runtime complexity of the enhanced Viterbi algorithm is $\theta(LK^2)$. The runtime complexity remains the same as that of the Viterbi algorithm, as long as there is a unique state that represents the codon frame of I_0, and is not in any signal submodel. If there are multiple states representing the same codon frame, we have to examine all possible choices of S_{I_0} to concatenate $H_{DP'}(1, S_1)$ and $H_{DP'}(L, S_K)$ properly. Therefore, the runtime complexity of the algorithm changes to $\theta(mLK^2)$, where m is the number of possible choices of S_{I_0}.

2.3.4 CDS Length Penalty

In this section, we introduce the use of the CDS length penalty, together with the enhanced Viterbi algorithm, to improve both the speed and the sensitivity of the homologous gene finding. The new CDS length penalty is based on the assumption that the CDS lengths of homologous genes are conserved during evolution. This assumption is supported by our study of the CDS lengths of homologous genes in Section 3.9.

To model the CDS length penalties, we adopt the approach of Christopher Burge in his program GENSCAN [8]. In his model, it is assumed that only a single codon insertion or deletion can occur in a generation, and the probabilities per generation of insertion and deletion are the same. Let n be the number of generations, p be the probability per generation

of insertion or deletion, and λ' be the observed exon length. The distribution of the lengths of homologous exons is approximately normal with mean $\mu \approx \lambda'$ and variance $\sigma^2 \approx 2np\lambda'$. Assuming that n is on the order of $1/p$, the variance of the distribution is on the order of λ'. If we assume that the evolution of each individual exon is independent of those of other exons, the distribution of the CDS lengths of homologous genes is also approximately normal. This is because the CDS length is the sum of the exon lengths, which are independent normally distributed variables, and the sum of independent normally distributed random variables is normal.

Recall that $\lambda_{5'}$ is the expected number of coding nucleotides on the $5'$ side of the initiator I_0. Let $\hat{\lambda}_{5'}$ be the number of coding nucleotides on the $5'$ side of the initiator I_0 of a gene candidate found. Based on the result of Christopher Burge [8], we approximate the distribution of $\hat{\lambda}_{5'}$ with a normal distribution with mean $\mu \approx \lambda_{5'}$ and variance $\sigma^2 \approx 2\lambda_{5'}$. Thus, we define the CDS length penalty $P_{CDS}(\hat{\lambda}_{5'}, \lambda_{5'})$ for the $5'$ end to be the probability density of CDS length $\hat{\lambda}_{5'}$ being observed as follows:

$$\begin{aligned} P_{CDS}(\hat{\lambda}_{5'}, \lambda_{5'}) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\hat{\lambda}_{5'}-\mu)^2}{2\sigma^2}} \\ &= \frac{1}{\sqrt{4\pi\lambda_{5'}}} e^{-\frac{(\hat{\lambda}_{5'}-\lambda_{5'})^2}{4\lambda_{5'}}}. \end{aligned} \tag{2.5}$$

We can similarly derive the CDS length penalty $P_{CDS}(\hat{\lambda}_{3'}, \lambda_{3'})$ for the $3'$ end. The study on our model is not complete, but in this thesis, we present some insights of this approach first. We will still improve improve the approach, as described in Section 4.2.

2.3.5 Scoring System

The single-gene HMM in Figure 2.2, with silent initial and final states, favors shorter gene structures. This is because we use Equation (1.2) to compute joint probability $P_{HMM}(G, H)$ that genomic sequence G is emitted by state path H, and shorter gene structures have fewer terms in Equation (1.2). We solve the problem by normalizing the probability that genomic sequence G is emitted by state path H, and denote the normalized probability $P'_{HMM}(G, H)$

as follows:

$$P'_{HMM}(G,H) = \frac{P_{HMM}(G,H)}{\sum_{i \in \pi(G)} P_{HMM}(G,i)}, \tag{2.6}$$

where $\pi(G)$ is the set of all state paths that can emit G. We compute $\sum_{i \in \pi(G)} P_{HMM}(G,i)$ with the enhanced forward algorithm, which is just the enhanced Viterbi algorithm with "max" replaced by "$\sum$" in Equation (2.5). Since the runtime complexities to find the maximum and to compute the sum are the same, the runtime complexity of the enhanced forward algorithm is the same as that of the enhanced Viterbi algorithm. Thus, the runtime complexity to find the optimal state path that maximizes the normalized probability $P'_{HMM}(G,H)$ is $\theta(LK^2)$.

For $i < I_0$, recall that $H_{DP'}(i, S_1)$ is the optimal state path that emits G_i in state S_1, emits G_{I_0} in state S_{I_0}, and emits genomic sequence $G_{i...I_0}$ with the highest probability of $P_{DP'}(i, S_1)$. Without loss of generality, we can say that $H_i = H_{DP'}(i, S_1)$ for $i < I_0$. Similarly, we can say that $H_j = H_{DP'}(j, S_K)$ for $j > I_0$. Let $\lambda(H)$ be the number of coding nucleotides in the gene structure represented by state path H. Let $H_{i,j} = H_i + H_j$ be the state path constructed by concatenating H_i and H_j. Recall that the CDS length penalty is the probability density of a particular CDS length being observed. We combine the normalized HMM probability and the CDS length penalty as our final probability of a gene structure as follows:

$$\begin{aligned}
P^*(G_{i...I_0}, H_i) &= P'_{HMM}(G_{i...I_0}, H_i) \cdot P_{CDS}(\lambda(H_i), \lambda_{5'}), & (2.7)\\
P^*(G_{I_0...j}, H_j) &= P'_{HMM}(G_{I_0...j}, H_j) \cdot P_{CDS}(\lambda(H_j), \lambda_{3'}), & (2.8)\\
P^*(G_{i...j}, H_{i,j}) &= P^*(G_{i...I_0}, H_i) \cdot P^*(G_{I_0...j}, H_j). & (2.9)
\end{aligned}$$

We did not find any algorithm that is capable of finding the state path maximizing Equation (2.9) directly. Instead, we use dynamic programming to compute $P'_{HMM}(G_{i...I_0}, H_i)$ for $i \in [1...I_0)$, and then find H_{i^*} such that

$$(G_{i^*...I_0}, H_{i^*}) = \arg \max_{i \in [1...I_0)} P^*(G_{i...I_0}, H_i). \tag{2.10}$$

Similarly, we use dynamic programming to compute $P'_{HMM}(G_{I_0...j}, H_j)$ for $j \in (I_0...L]$, and then find H_{j^*} such that

$$(G_{I_0...j^*}, H_{j^*}) = \arg \max_{j \in (I_0...L]} P^*(G_{I_0...j}, H_j). \tag{2.11}$$

Finally, we construct state path $H_{i^*,j^*} = H_{i^*} + H_{j^*}$ by concatenating H_{i^*} and H_{j^*}, and it is obvious that

$$(G_{i^*...j^*}, H_{i^*,j^*}) = \arg \max_{i \in [1...I_0), j \in (I_0...L]} P^*(G_{i^*...j}, H_i + H_j). \tag{2.12}$$

In practice, we only compute Equation (2.7) for $1 \leq i' \leq i < I_0$ and Equation (2.8) for $I_0 < j \leq j' \leq L$ to speed up the homologous gene finding without harming the accuracy. We dynamically choose the values of i' and j' by the stopping condition, as introduced in the next section.

Recall that the runtime complexity to find the optimal state path that maximizes the normalized probability $P'_{HMM}(G_{i...I_0}, H_i)$ is $\theta(LK^2)$. In practice, we remember the number of coding nucleotides in the dynamic programming algorithm, and thus when $P'_{HMM}(G_{i...I_0}, H_i)$ is computed, $\lambda(H_i)$ is computed. Assuming that $P'_{HMM}(G_{i...I_0}, H_i)$ and $\lambda(H_i)$ are computed, and the computation of $P_{CDS}(\lambda(H_i), \lambda_{5'})$ costs constant time, we can compute $P^*(G_{i...I_0}, H_i)$ in constant time. The runtime complexity to compute $P^*(G_{i...I_0}, H_i)$ is $\theta(LK^2) + O(LK^2) + O(1) = O(LK^2)$. Similarly, the runtime complexity to compute $P^*(G_{I_0...j}, H_j)$ is also $\theta(LK^2)$. Therefore, the runtime complexity to compute $P^*(G_{i...j}, H)$ is $\theta(LK^2) + O(LK^2) = O(LK^2)$.

In summary, instead of finding the optimal gene structure, we use the dynamic programming algorithm to find several homologous gene candidates, and then use the CDS length penalty to pick the candidate with the largest $P^*(G_{i...j}, H)$ from all candidates found in the target region.

2.3.6 Stopping Condition of the Enhanced Viterbi Algorithm

Using our CDS length penalties, we stop the dynamic programming when a higher state path probability can never be achieved. This means that our system dynamically chooses the target region size. In this section, only the stopping condition on the $5'$ end is introduced; the stopping condition on the $3'$ end is derived similarly.

For $i < I_0$, recall that H_i is the optimal state path that emits G_i in state S_1, emits G_{I_0} in state S_{I_0}, and emits $G_{n...I_0}$ with the highest probability of $P_{DP'}(i, S_1)$, and $\lambda(H_i)$ is the number of coding nucleotides in the gene structure represented by state path H_i. Our stopping condition is based on two simple observations:

$$P_{CDS}(\hat{\lambda}_1, \lambda_{5'}) \leq P_{CDS}(\hat{\lambda}_2, \lambda_{5'}), \text{ if } \hat{\lambda}_1 \geq \hat{\lambda}_2 > \lambda_{5'}, \tag{2.13}$$

$$P^*(G_{i...I_0}, H_i) \leq P_{CDS}(\lambda(H_i), \lambda_{5'}). \tag{2.14}$$

Observation (2.13) is true because the probability density function of a normal distribution is decreasing on the right side of the mean, and thus $P_{CDS}(\hat{\lambda}, \lambda_{5'})$ is decreasing in region $(\lambda_{5'}, +\infty)$. Observation (2.14) is true because $P'_{HMM}(G_{i...I_0}, H_i) \leq 1$, and thus $P^*(G_{i...I_0}, H_i) = P'_{HMM}(G_{i...I_0}, H_i) P_{CDS}(\lambda(H_i), \lambda_{5'}) \leq P_{CDS}(\lambda(H_i), \lambda_{5'})$. From these two observations, we can prove the following claim.

Claim 3 *Assume that we have computed $P^*(G_{i...I_0}, H_i)$ for $i \in [i', I_0)$, and found H_{i^*} such that $(G_{i^*...I_0}, H_{i^*}) = \arg\max_{i \in [i'...I_0)} P^*(G_{i...I_0}, H_i)$. If both Equation (2.15) and Equation (2.16) hold for all state $s \in S$, there is no state path H_i such that $i < i'$ and $P^*(G_{i...I_0}, H_i) > P^*(G_{i^*...I_0}, H_{i^*})$.*

$$P_{CDS}(\lambda(H_{DP'}(i', s)), \lambda_{5'}) < P^*(G_{i^*...I_0}, H_{i^*}), \tag{2.15}$$

$$\lambda(H_{DP'}(i', s)) > \lambda_{5'}. \tag{2.16}$$

Proof. Let H_i be any state path such that $i < i'$. Without loss of generality, we can say that $H_{DP'}(i', s)$ is a suffix of H_i. Then,

$$\begin{aligned}
\lambda(H_i) &\geq \lambda(H_{DP'}(i', s)) \\
&> \lambda_{5'} && \text{(by Equation (2.16))} \\
P^*(G_{i...I_0}, H_i) &< P_{CDS}(\lambda(H_i), \lambda_{5'}) && \text{(by Observation (2.14))} \\
&\leq P_{CDS}(\lambda(H_{DP'}(i', s)), \lambda_{5'}) && \text{(by Observation (2.13))} \\
&< P^*(G_{i^*...I_0}, H_{i^*}) && \text{(by Equation (2.15))}
\end{aligned}$$

Therefore, there is no state path H_i such that $i < i'$ and $P^*(G_{i...I_0}, H_i) > P^*(G_{i^*...I_0}, H_{i^*})$. ■

Claim 3 indicates that the enhanced Viterbi algorithm should stop on the $5'$ end when both Equation (2.15) and Equation (2.16) hold for all state $s \in S$, and we do not compute $P^*(G_{i...I_0}, H_i)$ for $i \in [1, i')$. We can similarly derive the stopping condition on the $3'$ end. Therefore, with the CDS length penalty and the stopping condition, we speed up the homologous gene finding without harming the accuracy. Several study cases are demonstrated in Section 3.10.

There is no guarantee that the stopping condition is always reached. To make sure that the algorithm always finishes, we add a second stopping condition: the gene length of homologous gene candidates is at most L'. We set $L' = 1,400,000$, which is a little longer than the longest known gene (with gene ID NM_013988 [35]) on human chromosome 6.

2.4 Assessment of Gene Candidates

The ultimate result of the previous step is a list of target gene candidates for each of the target regions. The purpose of this step is to choose the best candidate as the target gene in each target region, and then rank these target genes.

Since we define homologous genes as the genes encoding homologous proteins, we assess each of the target gene candidates using protein alignments. First, we translate target gene candidates to protein sequences, and align them to the protein translated from the query gene by the Needleman-Wunsch algorithm [27]. If a candidate is a gene homologous to the query gene, the protein alignment should have a high alignment score. After we choose the candidate with the highest protein alignment score in each of the target regions, we rank all of the candidates by their scores. A performance comparison between our approach of using protein alignments and an alternative approach of using CDS alignments is given in Section 3.11.

Chapter 3

Experiments

We implemented our algorithm to find homologous genes in Java, and then performed several experiments to validate the algorithm. Our implementation starts from HSPs found by BLAST [25, 40]. In this chapter, we first introduce the data sets used in our experiments, and then present the overall performance of our homologous gene finding algorithm, as well as the assessments of the individual steps of the algorithm.

3.1 Data Sets

In this section, we first describe three data sets that we used: the human genome training data set, the training data set of homologous genes, and the testing data set of homologous genes. The human genome training data set contains the reference sequences and the CDS annotations of 1070 human genes for training the HMM probabilities, as explained in Section 2.3.2. The training data set of homologous genes contains the reference sequences and the CDS annotations of 200 homologous gene groups for studying the parameters of our homologous gene finding algorithm. These parameters include the HSP filtering threshold, the number of target regions per target gene, the number of initiators per target region, and weight W. The testing data set of homologous genes contains the reference sequences and the CDS annotations of 400 homologous gene groups for evaluating the performance of our homologous gene finding algorithm. We also describe the target genomic sequence that we used in this section.

Number of Genes	1070
Average Gene Length	6476.75
Average Complete CDS Length	1195.63
Average Number of Exons per Gene	5.31

Table 3.1: Overview of the human genome training data set

3.1.1 Human Genome Training Data Set

The human genome training data set is based on the training data set of gene finder AUGUSTUS [34]. The original data set contained 1284 human genes retrieved from NCBI GenBank in October 2002. We removed 214 genes that did not satisfy various technical requirements:

- At least 20 nucleotides before the start codon in each reference sequence.
- At least 6 nucleotides after the stop codon in each reference sequence.
- At least 7 nucleotides in each exon.
- At least 29 nucleotides in each intron.
- The nucleotides "GT" right after each donor sites.
- The nucleotides "AG" right before each acceptor sites

The basic statistics of the remaining 1070 genes as our human genome training data set are shown in Table 3.1. We used the human genom training data set to train the HMM probabilities, as introduced in Section 2.3.2.

3.1.2 Training and Testing Data Sets of Homologous Genes

The training and testing data sets of homologous genes are based on NCBI HomoloGene 49.1 released on April 28, 2006 [26], which is a database of groups of homologous genes of several completely sequenced eukaryotic genomes. We restricted our data sets to three species: *H. sapiens* (human), *M. musculus* (house mouse) and *C. elegans*. We followed the following steps to prepare our training and testing data sets:

	H. sapiens	*M. musculus*	*C. elegans*
Number of Genes	200	200	200
Average Gene Length	33284.57	24780.59	2974.52
Average Complete CDS Length	1337.30	1326.15	1285.89
Average Number of Exons per Gene	10.70	10.43	5.67

Table 3.2: Overview of the training data set of homologous genes

	H. sapiens	*M. musculus*	*C. elegans*
Number of Genes	400	400	400
Average Gene Length	36536.54	28120.29	3004.86
Average Complete CDS Length	1337.23	1322.96	1311.36
Average Number of Exons per Gene	10.44	10.36	5.87

Table 3.3: Overview of the testing data set of homologous genes

1. We filtered all homologous gene groups in NCBI HomoloGene, and retained those ones that contain at least one gene from each of *H. sapiens*, *M. musculus*, and *C. elegans*. We removed the genes from other than the three species.

2. We used GMAP [39] to map each gene to its genome. After we checked if the mapped CDS annotation is consistent with the CDS annotation in NCBI GenBank, we removed the genes that have inconsistent annotations. Here, two annotations were consistent if and only if they had the same number of exons, and each pair of corresponding exons had the same length.

3. We randomly selected 200 homologous gene groups as the training data set, and 400 homologous gene groups as the testing data set. The training and testing data sets do not overlap.

The basic statistics of the training and testing data sets are shown in Tables 3.2 and 3.3, respectively. Although we expected to see paralogous genes, each homologous gene group in the training and testing data sets contains only one gene from each of the three species. Thus, we performed experiments on only orthologous genes for now, and we will perform more experiments on paralogous genes in the short future.

In all experiments that we performed with our implementation, we used human target genes, and mouse or *C. elegans* genes as query genes. Without further notice, we refer the

training data set as the training data set of homologous genes, and the testing data set as the testing data set of homologous genes in this chapter. When studying the parameters of our homologous gene finding algorithm, we used homologous genes in the training data set. When evaluating the performance of our homologous gene finding algorithm, we used homologous genes in the testing data set.

3.1.3 Target Genomic Sequence

In all experiments requiring the target genomic sequence, we used the NCBI human genome 36.1 released in March 2006 as the target genomic sequence. We downloaded the complete reference sequences with a total size of approximately $3.14GB$ from UCSC genome browser [35]. Repeats in the genomic sequences were masked by RepeatMasker [32] and Tandem Repeats Finder [4].

3.2 Overall Performance

Since we do not use an HMM that involves gene structure information as much as the HMMs used by GeneWise and Projector, it is unfair to compare the accuracy of our homologous gene finding algorithm to that of GeneWise or Projector. Thus, we evaluated the accuracy of our homologous gene finding algorithm, and compared the results to TBLASTN [25, 40]. We used Eval [16] to compute the sensitivity and specificity of the homologous gene finding.

To evaluate the overall performance of our homologous gene finding algorithm, we used the homologous gene groups in the testing data set, and designed experiment as follows: First, we found HSPs between the protein sequence encoded by the query gene and the target genomic sequence, and filtered HSPs with a threshold of 55. Second, we located three target regions for each target gene, and chose three initiators from each target region. Third, we set $W = 0.5$, and found a gene candidate for each initiator. Finally, we chose the best homologous gene from all gene candidates using protein alignment.

To compare the sensitivity and specificity to those of TBLASTN, we found HSPs between the protein sequence encoded by the query gene and the target genomic sequence. Then, we filtered HSPs with a threshold of 55, and chose the best target region as the highest scoring

	Gene SN / SP	Exon SN / SP	Nucleotide SN / SP
M. musculus	33.25% / 33.25%	74.45% / 76.80%	82.29% / 83.51%
C. elegans	0.25% / 0.27%	13.65% / 18.46%	26.03% / 29.98%

Table 3.4: Sensitivity and specificity of our homologous gene finding implementation

	Gene SN / SP	Exon SN / SP	Nucleotide SN / SP
M. musculus	0% / 0%	6.73% / 5.35%	90.67% / 82.92%
C. elegans	0% / 0%	1.27% / 1.75%	49.41% / 77.04%

Table 3.5: Sensitivity and specificity of homologous gene finding using TBLASTN

HSP cluster. Here, we simply used HSPs in the best target region as exons of the target gene, and HSP boundaries as splice sites.

The experiment results are shown in Tables 3.4 and 3.5. Table 3.4 shows that our homologous gene finding implementation has good sensitivity and specificity when the query and target genes are from close species. However, the sensitivity and specificity drops significantly when the query and target genes are from distant species. This is because homologous genes are more likely to be conserved in close species than those in distant species. When using mouse query genes, and human target genes, our implementation achieves an exon sensitivity of 74.45% and an exon specificity of 76.80%. Table 3.5 shows that TBLASTN is not sufficient to find splice sites and gene structures, because the gene level and exon level sensitivities and specificities are very low. Comparison of Tables 3.4 and 3.5 shows that gene and exon sensitivities and specificities of our homologous gene finding implementation are more accurate than that of TBLASTN. We observed that finding homologous genes with TBLASTN has higher nucleotide sensitivity than our homologous gene finding implementation. This is because we did not use the alignment information during the process of gene finding, and we will model the alignment information into the gene finding algorithm.

3.3 Finding HSPs

In this experiment, we compared two HSP finding approaches: using the protein sequence encoded by the query gene and the target genomic sequence (the protein-nucleotide alignment approach), and using the CDS of the query gene and the target genomic sequence (the

	BLASTN	TBLASTN
M. musculus	100%	100%
C. elegans	37.0%	91.5%

Table 3.6: Sensitivity of finding relevant HSPs (BLASTN v.s. TBLASTN)

nucleotide alignment approach). We used TBLASTN [25, 40] as an implementation of the protein-nucleotide alignment approach, and BLASTN [25, 40] as an implementation of the nucleotide alignment approach. For both approaches, we computed the sensitivity that at least one of the HSPs overlap at least one of the exons of the target gene, and compared the sensitivitys between the two approaches.

To compare the two approaches, we used the homologous gene groups in the training data set, as explained in Section 3.1.2. We first used the mouse genes, and then the *C. elegans* genes in the training data set as query genes to find HSPs. We found the HSPs between the protein sequence encoded by the query gene and the target genomic sequence with TBLASTN, and the HSPs between the CDS of the query gene and the target genomic sequence with BLASTN. Since we do not allow gaps in HSPs, as stated in Section 2.1, we used "-nogaps" to disable gaped alignments in TBLASTN and BLASTN. In this experiment, we simply used all HSPs found by TBLASTN and BLASTN without filtering.

Table 3.6 shows that when using mouse query genes, both TBLASTN and BLASTN found HSPs that overlap all target genes. The situation is not so good in case of *C. elegans* query genes. TBLASTN still found relevant HSPs with high sucess rate (91.5%). However, BLASTN failed to find relevant HSPs in most cases (true positive rate only 37.0%). This means that BLASTN is not sufficient to find relevant HSPs to locate accurate target regions, because we are not able to find at least one of the HSPs that overlaps at least one of the exons of the target gene in most cases.

The experiment results suggest two things. First, the protein-nucleotide approach is sufficient to find at least one of the HSPs that overlap at least one of the exons of the target gene. Second, when the query and target genes are from close species, the nucleotide alignment approach is as sufficient as the protein-nucleotide alignment approach, which is much slower. Therefore, we use the protein-nucleotide approach in our homologous gene finding algorithm in the following experiments for better accuracy.

Threshold	25	35	45	55	65	75	85
M. musculus	1.06	1.42	1.75	1.80	1.76	1.71	1.67
C. elegans	1.75	1.46	1.76	1.79	1.76	1.71	1.67

Table 3.7: Choosing a threshold to maximize the value of Equation (2.1)

3.4 Filtering HSPs

In this experiment, we studied the effects of filtering HSPs before locating target regions. Let a useful HSP be an HSP that overlaps at least one of the exons of the target gene, and a noise HSP be an HSP that does not overlap any exon of the target gene. We first chose a threshold in the way described in Section 2.1, and then evaluated the effects of filtering on the sensitivity that the predicted target region contains the target gene. Here, we say a target region contains the target gene if and only if there is a useful HSP in the target region.

Recall that we want to choose a threshold to maximize the value of Equation (2.1), so that most of noise HSPs are filtered out, and most of useful HSPs are retained. Thus, we used the homologous gene groups in the training data set, as explained in Section 3.1.2, and filtered HSPs with thresholds from 0 to 110 with a step size 5. For each threshold, we counted the numbers of useful HSPs and the number of noise HSPs before and after filtering, and computed the value of Equation (2.1). Some results are shown in Table 3.7. From the table, we chose a threshold of 55, which maximized the value of Equation (2.1).

Table 3.8 shows that the numbers of useful HSPs and the number of noise HSPs before and after filtering with a threshold of 55. When using mouse query genes, 85.28% of noise HSPs were filtered out, and 96.70% of useful HSPs were retained. Before filtering, useful HSPs weighted only 0.79% over all HSPs. After filtering, useful HSPs weighted 5.02%. Thus, we increased the weight of useful HSPs significantly from 0.79% to 5.02%. When using *C. elegans* query genes, 86.85% of noise HSPs were filtered out, and 89.00% of useful HSPs were retained. We increased the weight of useful HSPs significantly from 0.73% to 4.73%. Since the weight of useful HSPs was increased significantly, it was more likely to find the correct target region.

To study the effects of filtering on locating target regions, we predicted a target region using the highest scoring cluster before and after filtering with thresholds of 0, 45, 55, 65

	Before Filtering		After Filtering	
	#Useful HSPs	#Noise HSPs	#Useful HSPs	#Noise HSPs
M. musculus	2365	294116	2287	43296
C. elegans	1610	219344	1433	28834

Table 3.8: Filtering HSPs with a threshold of 55

Threshold	0	45	55	65	110
M. musculus	89.0%	92.0%	93.0%	93.0%	90.5%
C. elegans	70.5%	72.0%	70.5%	72.0%	61.5%

Table 3.9: Sensitivity of filtering HSPs with different thresholds

and 110. Then, we computed the sensitivity that the predicted target region contains the target gene.

Table 3.9 shows that before filtering (threshold = 0), we located 89.0% of the target regions correctly when using mouse query genes, and 70.5% of the target regions correctly when using *C. elegans* query genes. After filtering with a threshold of 55, we located 93.0% of the target regions correctly when using mouse query genes, and 70.5% of the target regions correctly when using *C. elegans* query genes. These numbers did not change much when thresholds of 45 and 65 were used, but dropped to 90.5% and 61.5% when a threshold of 110 was used.

Therefore, we can conclude that filtering HSPs increases the accuracy of target region locating when the query and target genes are from close species. Recall that we want to keep most of the useful HSPs so that we can use these HSPs in the gene finding step to improve the accuracy in the future. The experiments results show that choosing the filtering threshold to maximize the value of Equation (2.1) does not harm the accuracy of locating target regions much. In the following sections, we filter HSPs with a threshold of 55 as a default.

3.5 Using Multiple Target Regions

In this experiment, we studied the effects of using multiple target regions. We used the homologous gene groups in the training data set, as explained in Section 3.1.2, and clustered the filtered HSPs. Then, we located 1, 3, 5, 7 and 9 target regions from the highest scoring

#Target Regions	1	3	5	7	9
M. musculus	93.0%	98.5%	99.0%	99.0%	99.5%
C. elegans	70.5%	87.5%	88.5%	88.5%	89.5%

Table 3.10: Sensitivity of using different number of target regions

#Initiators	1	2	3	4	5
M. musculus	97.5%	98.5%	98.5%	98.5%	98.5%
C. elegans	86.0%	87.5%	87.5%	87.5%	87.5%

Table 3.11: Sensitivity of using different number of initiators

HSP clusters, as explained in Section 2.2. Finally, we computed the sensitivity that we found the true target region containing the target gene.

Table 3.10 shows that when we used only the highest scoring cluster to locate a target region, we located 93.0% of the true target regions for mouse query genes, and 70.5% of the true target regions for *C. elegans* query genes. The sensitivity was increased to 98.5% for mouse query genes, and 87.5% for *C. elegans* query genes, when the number of clusters was increased to three. In general, the sensitivity increases slowly with the increasing number of target regions.

The experiment results suggest that using multiple target regions improves the accuracy of locating target regions. To achieve a good accuracy without slowing down the computation much, we used three target regions as a default in our implementation and following experiments.

3.6 Using Multiple Initiators for Each Target Region

In this experiment, we studied the effects of using multiple initiators for each target regions by computing the sensitivity that at least one initiator hits the target gene. We used the homologous gene groups in the training data set, as explained in Section 3.1.2, and located three target regions from the top three scoring HSP clusters. Then, we chose 1, 2, 3, 4 and 5 initiators for each target region, as explained in Section 2.2.

Table 3.11 shows that when we chose a single initiator for each target region, 97.5% of the initiators hit the target gene for mouse query genes, and 86.0% of the initiators hit the

target gene for *C. elegans* query genes. These numbers were increased to 98.5% and 87.5% when we chose two initiators for each target region, and remain the same when we chose more than two initiators for each target region. When using a single initiator, we observed five cases where we chose incorrect initiators within correct target regions.

The experiment results suggest that using multiple initiators may improve the accuracy of locating target regions. In the following sections, we use three initiators for each target region as a default in order to achieve a good accuracy without slowing down the computation much.

3.7 HMM Probabilities to be Biased

In this section, we study which HMM probabilities should be biased. If biasing the HMM probabilities of a state toward the query gene boosts the HMM probabilities of target genes, we should bias the HMM probabilities of the state. Otherwise, we should not bias the HMM probabilities of the state. In the next section, we study if the biased training improves the accuracy of homologous gene finding.

For each pair of query and target genes, we first trained the HMM probabilities with the target gene, and then with the query gene. We say that the HMM probabilities trained with the target gene are the true HMM probabilities we seek to estimate, and the HMM probabilities trained with the query gene are the estimations of the true HMM probabilities, or estimated HMM probabilities. In fact, the true HMM probabilities are likely the best HMM probabilities to find the target gene. We should bias the HMM probabilities of a state, if we are likely to bias the HMM probabilities toward the true HMM probabilities by biasing the HMM probabilities toward the estimated HMM probabilities.

To study which HMM probabilities should be biased, we computed the root mean square error (RMSE) of the estimated HMM probabilities using the homologous gene groups in the training data set, as explained in Section 3.1.2. Since the emission probabilities are independent of the transition probabilities, we computed the RMSE of the emission probabilities and the RMSE of the transition probabilities separately for each state. In order to increase numerical reliability, we used the logarithms of the HMM probabilities, and thus we computed RMSEs of the logarithms of the HMM probabilities. If the computed RMSE is small,

	M. musculus		*C. elegans*	
	AVG	STDEV	AVG	STDEV
exon[1]	0.05	0.27	1.00	0.85
exon[2]	0.08	0.35	0.98	0.84
exon[3]	0.07	0.31	0.65	0.60

Table 3.12: RMSE of logarithms of transition probabilities of exon states

	M. musculus		*C. elegans*	
	AVG	STDEV	AVG	STDEV
intron[1]	0.33	0.70	2.81	1.71
intron[2]	0.34	0.71	2.59	1.65
intron[3]	0.35	0.72	2.39	1.42

Table 3.13: RMSE of logarithms of transition probabilities of intron states

the estimated HMM probabilities are good estimations of the true HMM probabilities. This also implies that if we bias the HMM probabilities toward the estimated HMM probabilities, we will successfully bias the HMM probabilities toward the true HMM probabilities. Otherwise, we will move the HMM probabilities away from the true HMM probabilities. Therefore, if both the average (AVG) and standard derivation (STDEV) of the computed RMSEs of a state are small, we are likely to bias the HMM probabilities of the state toward the true HMM probabilities by biasing the HMM probabilities toward the estimated HMM probabilities.

The experiment results are shown in Table 3.12 to Table 3.4. Comparisons of the numbers in these tables show which HMM probabilities are more likely to be conserved between homologous genes. Generally, the HMM probabilities with smaller average and standard derivation of RMSEs are more likely to be conserved, and the HMM probabilities with larger average and standard derivation of RMSEs are less likely to be conserved. However, we cannot conclude that the emission probabilities of higher order states are more likely to be conserved than those of lower order states based on only the comparisons of the average and standard derivation of RMSEs. This is mainly because the higher order states have more emission probabilities, and the RMSEs of more emission probabilities of higher order states tend to be larger and more unreliable.

Tables 3.12 and 3.13 show that coding exon and intron lengths of mouse genes are more

	M. musculus		*C. elegans*	
	AVG	STDEV	AVG	STDEV
exon[1]	0.61	0.10	0.80	0.10
exon[2]	0.53	0.11	0.77	0.10
exon[3]	0.62	0.11	0.82	0.12

Table 3.14: RMSE of logarithms of emission probabilities of exon states

	M. musculus		*C. elegans*	
	AVG	STDEV	AVG	STDEV
intron[1]	0.73	0.30	0.96	0.38
intron[2]	0.69	0.34	0.90	0.40
intron[3]	0.77	0.25	1.10	0.30

Table 3.15: RMSE of logarithms of emission probabilities of intron states

likely to be conserved to their homologous human genes than *C. elegans* genes. Comparison of Tables 3.12 and 3.13 shows that intron lengths are not as well conserved as coding exon lengths, because the average and standard derivation of RMSEs of logarithms of transition probabilities of introns states are much higher than exon states.

Tables 3.14 and 3.15 show that coding exons and introns of mouse genes are more likely to be conserved to their homologous human genes than *C. elegans* genes. We also observed that the probabilities of the nucleotides in the first codon frame are more likely to be conserved than those of the nucleotides in other codon frames between mouse genes and their homologous human genes. However, the probabilities of the nucleotides in the third codon frame are more likely to be conserved than those of the nucleotides in other codon frames between *C. elegans* genes and their homologous human genes. Comparison of Tables 3.14 and 3.15 shows that introns are not as well conserved as coding exons, because average and standard derivation of RMSEs of logarithms of emission probabilities of introns states are much higher than exon states.

Figures 3.1 and 3.2 show that the signals within coding exons around acceptor splice sites are more likely to be conserved than the signals within introns, and the signals near splice sites are more likely to be conserved than the signals distant from splice sites. We also observed that for splice site signal states, AVG RMSE of logarithms of emission probabilities for mouse query genes is usually smaller than that for *C. elegans* query genes, but STDEV

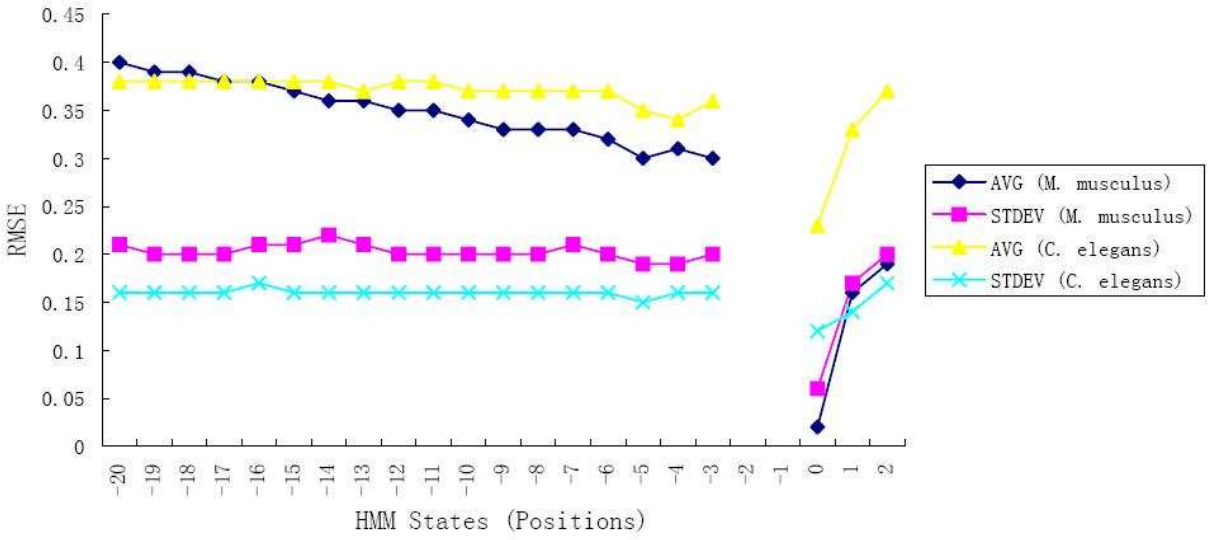


Figure 3.1: RMSE of logarithms of emission probabilities of acceptor signal states

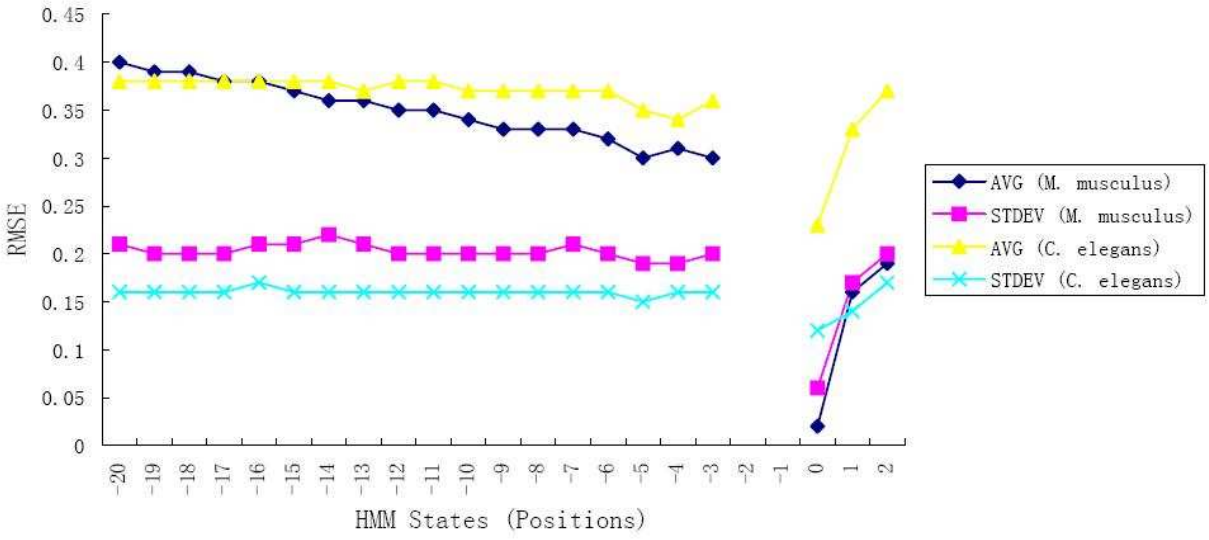


Figure 3.2: RMSE of logarithms of emission probabilities of donor signal states

RMSE of logarithms of emission probabilities for mouse query genes is usually larger than that for *C. elegans* query genes.

Figures 3.3 and 3.4 show that the signals within coding exons around start and stop sites are more likely to be conserved than the signals within introns, and the signals near start and stop sites are more likely to be conserved than the signals distant from start and stop sites. We also observed that for start and stop signal states, AVG RMSE of logarithms of emission probabilities for mouse query genes is usually smaller than that for *C. elegans* query genes, but STDEV RMSE of logarithms of emission probabilities for mouse query genes is usually larger than for *C. elegans* query genes.

Tables 3.12 and 3.13 suggest that we should not bias transition probabilities, because the biased training tends to unreliable (large average and standard derivation of RMSEs) when the query and target genes are from distant species. Tables 3.14 and 3.15 suggest that we should not bias emission probabilities of intron states, because they are not as well conserved as those of exon states. In fact, our research on biased training is very preliminary, and there is obviously a lot of work to do. We are considering other approaches to study which HMM probabilities should be biased, such as entropy. We are also trying to model the correlations between the true HMM probabilities and the estimated HMM probabilities into the biased training. In the current implementation of our homologous gene finding algorithm, we bias only the emission probabilities of the exon and signal states.

3.8 Biased Training

In this experiment, we evaluated the effects of the biased training introduced in Section 2.2. We used the homologous gene groups in the testing data set, as explained in Section 3.1.2. In order to avoid the effects of incorrect initiators, we used the ideal initiator of the true target region to find homologous genes. As stated earlier, the ideal initiator is located in the middle of the CDS of the target gene in the target genomic sequence. Then, we set the weight constant W, as introduced in Equations (2.2) and (2.3), to the values 0 (unbiased), 0.25, 0.5 and 0.75, and found gene candidates, as described in Section 2.3. Finally, we used the program Eval [10] to compute the sensitivity (SN) and specificity (SP) of our homologous gene finding (see Section 1.2 for the definitions of these metrics).

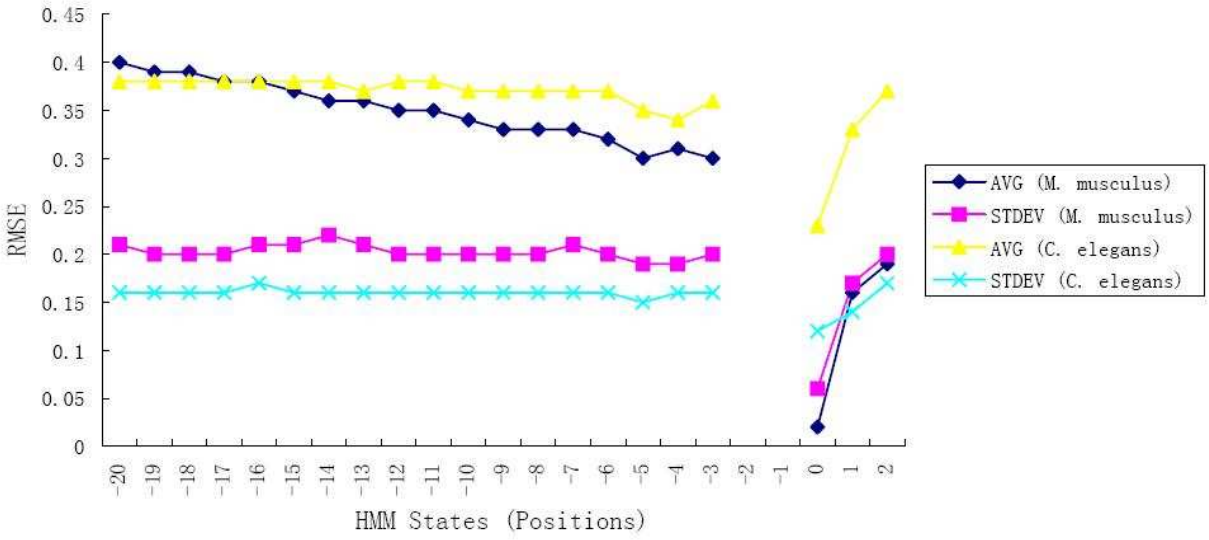


Figure 3.3: RMSE of logarithms of emission probabilities of start signal states

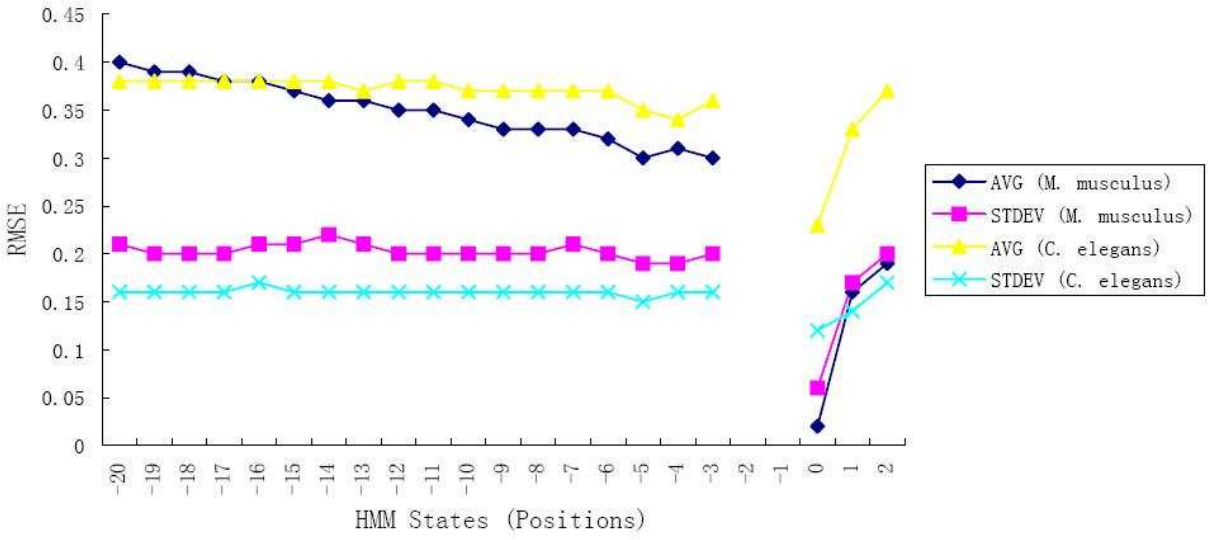


Figure 3.4: RMSE of logarithms of emission probabilities of stop signal states

W	Gene SN / SP	Exon SN / SP	Nucleotide SN / SP
0 (unbiased)	14.50% / 14.50%	42.07% / 52.35%	55.86% / 56.65%
0.25	42.00% / 42.00%	75.74% / 80.85%	83.22% / 83.07%
0.5	47.75% / 47.75%	79.69% / 82.87%	86.27% / 86.63%
0.75	47.00% / 47.00%	79.93% / 82.06%	86.65% / 86.96%
1.0	44.25% / 44.25%	78.28% / 80.58%	85.04% / 85.46%

Table 3.16: Sensitivity and specificity of the biased training with different weights (*M. musculus*)

W	Gene SN / SP	Exon SN / SP	Nucleotide SN / SP
0 (unbiased)	14.50% / 14.50%	42.07% / 52.35%	55.86% / 56.65%
0.25	5.00% / 5.00%	30.77% / 38.03%	49.56% / 50.32%
0.5	3.50% / 3.50%	19.42% / 25.55%	37.16% / 38.14%
0.75	1.75% / 1.75%	10.97% / 16.36%	25.34% / 26.23%
1.0	0.75% / 0.75%	6.59% / 11.27%	17.26% / 18.07%

Table 3.17: Sensitivity and specificity of the biased training with different weights (*C. elegans*)

Since we used only the very simple HMM shown in Figure 2.2, the sensitivity and specificity of our homologous gene finding algorithm is not as good as other single-genome gene finding algorithms, such as GENSCAN [9] and AUGUSTUS [34]. Using the 178 human genes in the data set sag178 [34], AUGUSTUS has an exon sensitivity of 78% and an exon specificity of 71%, while GENSCAN has an exon sensitivity of 68% and an exon specificity of 45%. Table 3.16 shows that our homologous gene finding algorithm without biased training has an exon sensitivity of only 42.28% and an exon specificity of only 52.54%. However, this is not a major concern, because we are interested in studying the effects of the biased training, not in finding the best HMM. Certainly, we will improve the HMM used for homologous gene finding in the future.

Table 3.16 shows that the sensitivity and specificity are improved significantly after using mouse genes in the biased training. When using $W = 0.5$, the best exon sensitivity of 79.69% and the best exon specificity of 82.87% are archieved. Table 3.16 also shows that the benefit of the biased training is not maximized when using small values of W, and overfitting occurs when using large values of W. Table 3.17 shows that the sensitivity and specificity drop after using *C. elegans* query genes in the biased training. Even worse, while we increase the

effects of the biased training by increasing W, the sensitivity and specificity decrease.

The experiment results suggest that when the query and target genes are from close species, the biased training improves the sensitivity and specificity of homologous gene finding. Moreover, the sensitivity and specificity can still be improved using more advanced HMMs, such as the ones used by GENSCAN [9] and AUGUSTUS [34]. However, the experiment results show that when the query and target genes are from distant species, our current biased training approach does not improve the sensitivity and specificity of homologous gene finding. We are still looking for other ways to bias the HMM probabilities toward the query genes for the query and target genes from distant species.

3.9 Complete CDS Length v.s. Gene Length

In this section, we study how the CDS lengths and the gene lengths differ between homologous genes. Thus, we computed the CDS length difference and the gene length difference between each pair of homologous genes in the training data set.

The results are shown from Figure 3.5 to Figure 3.8. Figures 3.5 and 3.6 show that the CDS length of homologous genes are conserved, and the distribution fits the normal distribution well. Comparison of Figures 3.5 and 3.6 shows that the CDS length of homologous genes from distant species are not as well conserved as those from close species. Figures 3.7 and 3.8 show that the gene length of homologous genes is not conserved, and human genes are generally longer than mouse and *C. elegans* genes.

Recall that we approximate the distribution of the CDS length of the target gene with a normal distribution with mean $\mu \approx \lambda_{5'}$ and variance $\sigma^2 \approx 2\lambda_{5'}$, where $\lambda_{5'}$ is the CDS length of the query gene. The results suggest that the approximation with a normal distribution is reasonable. However, the results also suggest that the variance of the normal distribution should depend on the distance between the query and target species. Thus, we will use this information to adjust the variance in the future.

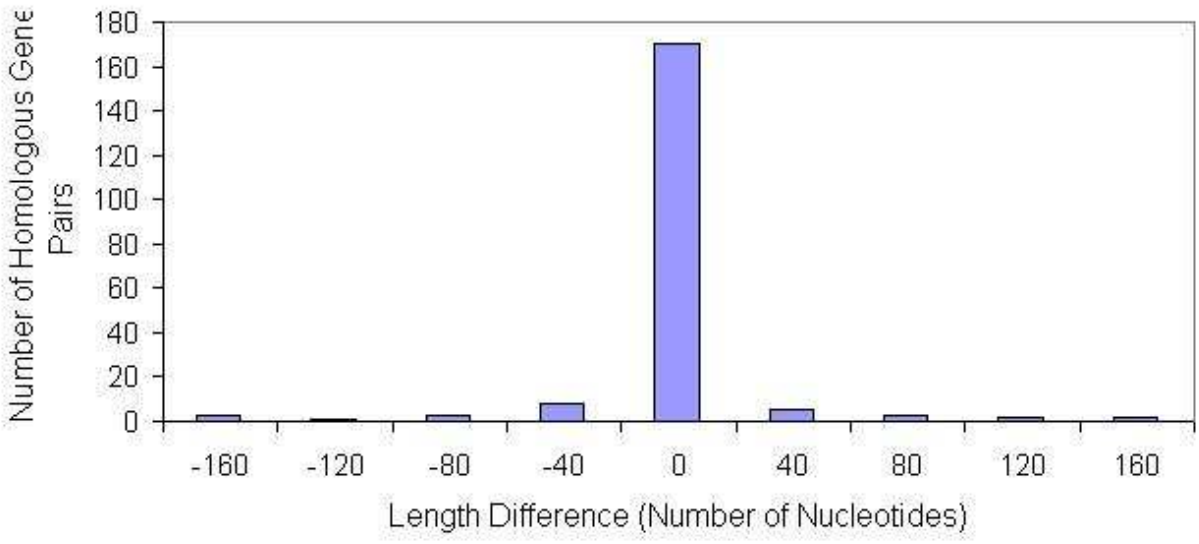


Figure 3.5: Complete CDS length difference distribution (*H. sapiens* v.s. *M. musculus*)

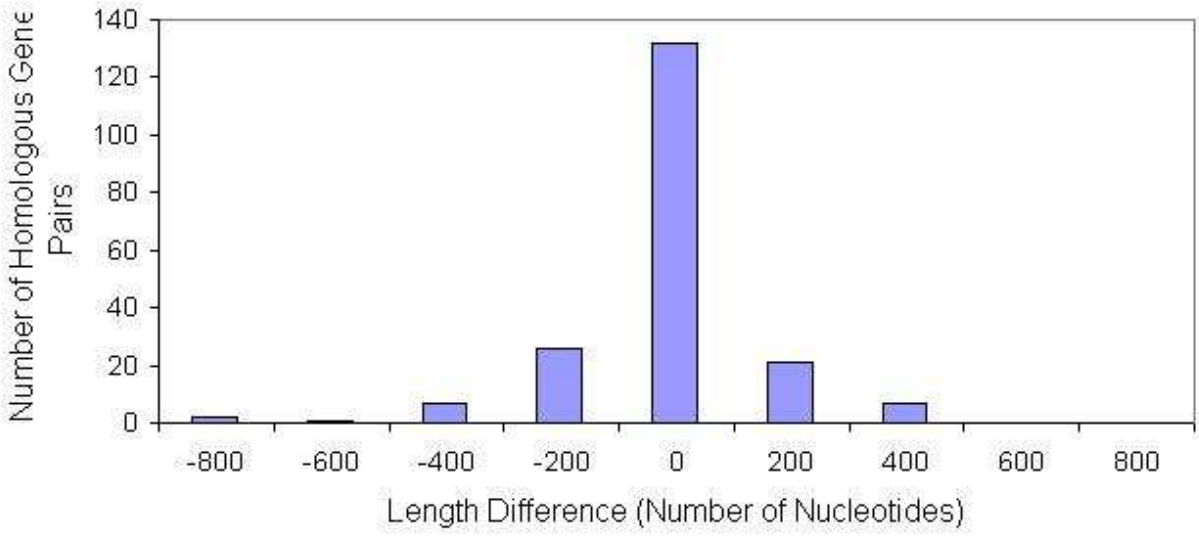


Figure 3.6: Complete CDS length difference distribution (*H. sapiens* v.s. *C. elegans*)

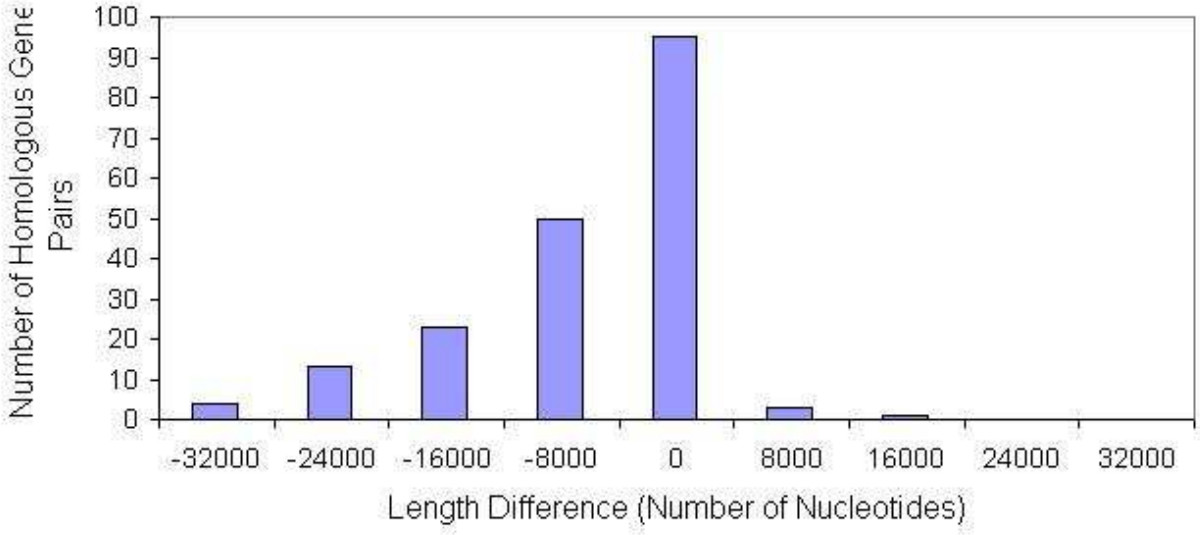


Figure 3.7: Gene length difference distribution (*H. sapiens* v.s. *M. musculus*)

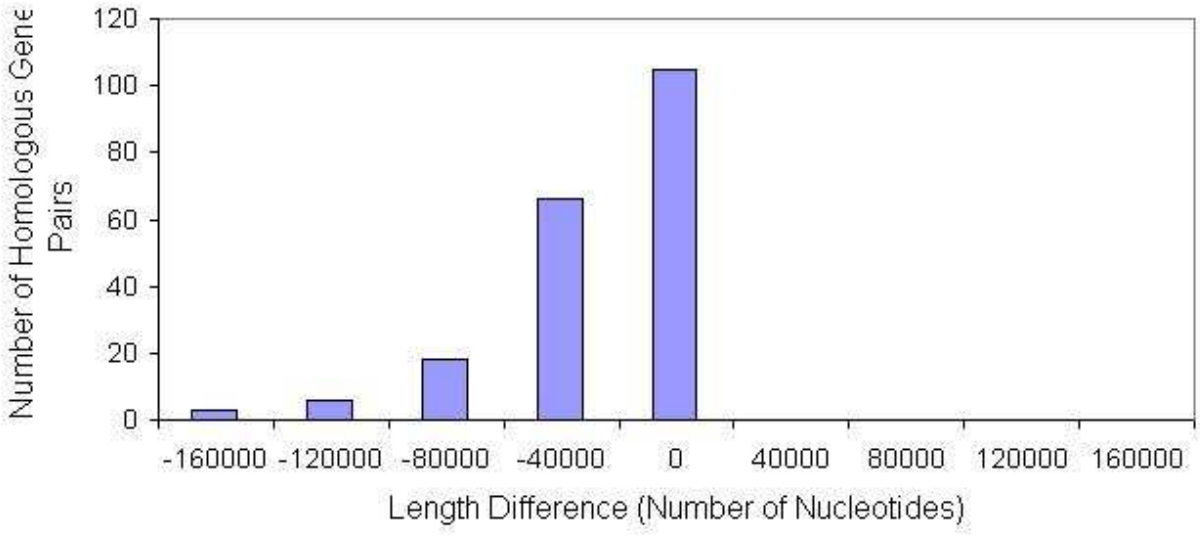


Figure 3.8: Gene length difference distribution (*H. sapiens* v.s. *C. elegans*)

		Exon SN / SP	Nucleotide SN / SP	Runtime
with CDSLP	*M. musculus*	79.69% / 82.87%	86.27% / 86.63%	32.19s
	C. elegans	19.42% / 25.55%	37.16% / 38.14%	35.94s
without CDSLP	*M. musculus*	1.48% / 7.38%	8.72% / 69.72%	51.06s
	C. elegans	0.72% / 3.53%	7.30% / 64.45%	43.65s

Table 3.18: Sensitivity and specificity of the CDS length penalty and the stopping condition

3.10 CDS Length Penalty and Stopping Condition

To study the effects of the CDS length penalty (CDSLP) and the stopping condition, we used homologous gene groups in the testing data, as explained in Section 3.1.2. We used the ideal initiator and $W = 0.5$ to find a homologous gene candidate for each query gene with and without the CDS length penalty and the stopping condition. We run our homologous gene finding implementation on a computer with a 1.4GHz Intel Xeon CPU, and recorded the average running time (CPU time) per query gene. The running time does not include the time required to find HSPs between the protein sequence encoded by the query gene and the target genomic sequence. Finally, we used Eval [16] to compute sensitivity SN and specificity SP of homologous gene finding.

The experiment results in Table 3.18 show that the sensitivity and specificity of homologous gene finding without the CDS length penalty are very poor. We observed that our HMM favors short gene structures. As a result, when using mouse query genes without the CDS length penalty, the average length of predicted genes is 465, while the true average length is 36537. Table 3.18 also shows that using the stopping condition, the average running time to find homologous genes is reduced 37.0%.

The experiment results suggest that using the CDS length penalty and the stopping condition helps to achieve better sensitivity and specificity, and to reduce running time of our homologous gene finding algorithm.

3.11 Gene Candidate Assessment

In this experiment, we compared two gene candidate assessment approaches using the homologous gene groups, as described in Section 3.1.2. The first approach is to choose and

		Gene SN / SP	Exon SN / SP	Nucleotide SN / SP
Protein Alignment	*M. musculus*	33.25% / 33.25%	74.45% / 76.80%	82.29% / 83.51%
	C. elegans	0.25% / 0.27%	13.65% / 18.46%	26.03% / 29.98%
Nucleotide Alignment	*M. musculus*	33.25% / 33.25%	74.45% / 76.80%	82.29% / 83.51%
	C. elegans	0.25% / 0.27%	13.65% / 18.46%	26.03% / 29.98%

Table 3.19: Sensitivity and specificity of gene candidate assessment (protein alignment v.s. nucleotide alignment)

rank gene candidates using the global alignment between the proteins encoded by the query gene and the gene candidate (the protein alignment approach). The second approach is to choose and rank gene candidates using the global alignment between the CDS of the query gene and the gene candidate (the nucleotide alignment approach). We used Eval [16] to compute the sensitivity and specificity of the highest ranked homologous gene.

Table 3.19 shows that both approaches have the same sensitivity and specificity. In our homologous gene finding algorithm and implementation, we choose and rank gene candidates using the global alignment between the proteins encoded by the query gene and the gene candidate.

Chapter 4

Conclusion and Future Work

In this thesis, we introduced the homologous gene finding problem and our new algorithm to solve this problem. The algorithm takes a query gene and a target genomic sequence as input, and then finds all homologous genes of the query gene in the target genomic sequence. We implemented and evaluated the performance of our homologous gene finding algorithm. We also showed that the homologous gene finding problem can be solved by applying and extending existing seed-based homology search algorithms and HMM-based gene finding algorithms.

4.1 Comparison to GeneWise [5] and Projector [24]

Compared to GeneWise [5] and Projector [24], our proposed homologous gene finding algorithm has three strengths. First, we train a simple single-genome gene finding HMM to be biased toward the query gene. Using the biased training, we can achieve comparative sensitivity and specificity without using complicated and computationally intensive pair-HMMs. Second, we use the approximate CDS length distribution of the target gene to further improve the sensitivity and specificity. In fact, finding correct start and stop codons is the most challenging part of gene finding, and the approximated CDS length distribution is designed directly to conquer this challenge. Third, we locate target regions which might contain target genes, and then find target genes in the target regions. This approach is more efficient than filtering homologous genes after finding target genes on the entire target genomic sequence.

This is because seed-based homology search algorithms are not as computationally intensive as gene finding algorithms, and we can locate target regions very fast using HSPs found by a seed-based homology search algorithm.

4.2 Future Work

Our homologous gene finding algorithm works well when the query and the target genes are from close species, but there is still space for improvements.

First, we can certainly improve the sensitivity and specificity of homologous gene finding by using a more advanced HMM.

Second, we will further study which HMM probabilities should be biased, and model the correlations between the true HMM probabilities and the estimated HMM probabilities into the biased training.

Third, we will bias the HMM probabilities toward all possible complete CDSs that encodes the protein encoded by the query gene. This might improve the sensitivity and specificity of homologous gene finding, when the query and target genes are from distant species.

Fourth, we will model the distance between the query and target species into the CDS length penalty, as the experiment results suggested in Section 3.9.

Finally, we will model the alignment information into the HMM-based gene finding process. Current research on dual-genome gene finding algorithms [19, 11, 29] shows that adding alignment probabilities into HMM probabilities can improve both the sensitivity and specificity of gene finding.

Bibliography

[1] M. Alexandersson, S. Cawley, and L. Pachter. SLAM: cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Research*, 13:493–502, 2003.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[3] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997.

[4] G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, 27:573–580, 1999.

[5] E. Birney, M. Clamp, and R. Durbin. GeneWise and GenomeWise. *Genome Research*, 14:988–995, 2004.

[6] B. Brejová, D. G. Brown, M. Li, and T. Vinař. ExonHunter: a comprehensive approach to gene finding. In *ISMB (Supplement of Bioinformatics)*, pages 57–65, 2005.

[7] B. Brejová, D. G. Brown, and T. Vinař. Optimal spaced seeds for homologous coding regions. *Journal of Bioinformatics and Computational Biology*, 1(4):595–610, January 2004.

[8] C. Burge. *Identification of genes in human genomic DNA*. PhD thesis, Stanford University, 1997.

[9] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94, 1997.

[10] M. Bursetb and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 33:353–367, 1996.

[11] P. Flicek, E. Keibler, P. Hu, I. Korf, and M. R. Brent. Leveraging the mouse genome for gene prediction in human: from whole-genome shotgun reads to a global synteny map. *Genome Research*, 13:46–54, 2003.

[12] L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Research*, 8(9):967–974, 1998.

[13] S. S. Gross and M. R. Brent. Using multiple alignments to improve gene prediction. In *Research in Computational Molecular Biology*, pages 374–388, 2005.

[14] R. Guigo, P. Agarwal, J. F. Abril, M. Burset, and J. W. Fickett. An assessment of gene prediction accuracy in large DNA sequences. *Genome Research*, 10:1631–1642, 2000.

[15] L. Hunter. Molecular biology for computer scientists, 1993.

[16] E. Keibler and M. R. Brent. Eval: A software package for analysis of genome annotations. *Bioinformatics*, 4, 2003.

[17] W. J. Kent. BLAT: the BLAST-like alignment tool. *Genome Research*, 12(4):656–664, April 2002.

[18] D. Kisman, M. Li, B. Ma, and L. Wang. tPatternHunter: gapped, fast and sensitive translated homology search. *Bioinformatics*, 21(4):542–544, February 2005.

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

[20] M. Li, B. Ma, D. Kisman, and J. Tromp. PatternHunter II: highly sensitive and fast homology search. *Journal of Bioinformatics and Computational Biology*, 2:417–439, 2004.

[21] D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, March 1985.

[22] B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, March 2002.

[23] I. M. Meyer and R. Durbin. Comparative *ab initio* prediction of gene structures using pair HMMs. *Bioinformatics*, 18(10):1309–1318, 2002.

[24] I. M. Meyer and R. Durbin. Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Research*, 32:776–783, 2004.

[25] BLAST program selection guide *http://www.ncbi.nlm.nih.gov/blast/producttable.shtml*.

[26] NCBI HomoloGene *http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=homologene*.

[27] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[28] L. Pachter, M. Alexandersson, and S. Cawley. Applications of generalized pair hidden Markov models to alignment and gene finding problems. In *Research in Computational Molecular Biology*, pages 241–248, 2001.

[29] G. Parra, P. Agarwal, J. F. Abril, T. Wiehe, J. W. Fickett, and R. Guigo. Comparative gene prediction in human and mouse. *Genome Research*, 13:108–117, 2003.

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

[31] A. C. Siepel and D. Haussler. Computational identification of evolutionarily conserved exons. In *Research in Computational Molecular Biology*, pages 177–186, 2004.

[32] A. Smit, G. Glusman, and R. Hubley. RepeatMasker *http://www.repeatmasker.org/*.

[33] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[34] M. Stanke and S. Waack. Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19(2):215–225, 2003.

[35] UCSC genome bioinformatics *http://genome.ucsc.edu/*.

[36] N. Volfovsky, B. J. Haas, and S. L. Salzberg. Computational discovery of internal micro-exons. *Genome Research*, 13:1216 – 1221, 2003.

[37] WebLogo *http://weblogo.berkeley.edu/*.

[38] S. J. Wheelan, D. M. Church, and J. M. Ostell. Spidey: a tool for mRNA-to-genomic alignments. *Genome Research*, 11(11):1952–1957, 2001.

[39] T. D. Wu and C. K. Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21(9):1859–1875, May 2005.

[40] WU-BLAST *http://blast.wustl.edu/*.

[41] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning DNA sequences. *Journal of Molecular Biology*, 7(1-2):203–214, 2000.