# Mathematical Methods for Comparative *Ab* Initio Gene Prediction

Irmtraud Margret Meyer
Trinity College
Cambridge

August 2002

A dissertation submitted for
the degree of Doctor of Philosophy
at the University of Cambridge

# Preface

The work presented in this dissertation was carried out at the Sanger Institute in Cambridge between January **2000** and August 2002. This dissertation is the result of my own work and includes nothing which is the outcome **of** work done in collaboration except where specifically indicated in the text. **No** part of this dissertation nor anything substantially the same **has** been or is being submitted for any qualification at any other university.

# Summary

This dissertation introduces two novel methods for the comparative prediction of protein coding genes in eukaryotic genomes. The first method, implemented in a program called **DOUBLESCAN,** is an ab initio method which simultaneously predicts the gene structures and the alignment of two evolutionarily related input **DNA** sequences from the sequence of their A, C, G, T bases only. The second method, implemented in a program called **PROJECTOR,** is a homology based method which predicts gene structures in one **DNA** sequence according to the known gene structures of a related **DNA** sequence and which simultaneously aligns the two **DNA** sequences. Both methods employ a probabilistic pair Hidden Markov model and are capable of predicting partial, complete and multiple genes **as** well **as** pairs of genes which are related by events of exon-fusion or exon-splitting. Predictions are generated using two different algorithms: the Hirschberg algorithm whose predictions are generated in linear memory and quadratic time and a new algorithm, called the Stepping Stone algorithm, whose memory and time requirements scale both linearly with the length of the input sequence. This work describes the theoretical concepts underlying the two novel methods and their implementation into computer programs and demonstrates the validity and generality of the approach by evaluating the performance of the gene prediction on a test set of mouse (Mus *musculus*) and human (*Homo* sapiens) **as** well **as** **Caenorhabditis *elegans*** and Caenorhabditis briggsae **DNA** sequence pairs.

# Acknowledgements

First of all, I would like to thank my supervisor, Richard Durbin, for his advice, support and encouragement. I thank Kevin Howe, Matthew Pocock, Raphael Leplae, Aaron Levine, Marc Sohrmann, Ashwin Hajarnavis, Lachlan Coin, Thomas Down and all the other members of the Wellcome Trust Genome Campus who have made both science and life on the campus interesting and enjoyable.

I am grateful to Trinity College, Cambridge, for an External Research Studentship and to the Wellcome Trust for a Prize Studentship.

# Contents

**Summary** ii

**1 Introduction** 1

- **1.1** Motivation . . . . 1
- **1.2** Biological background . . . . 6
- **1.3** Existing non-comparative methods . . . . 9
  - **1.3.1** Types of evidence . . . . 9
  - **1.3.2** Methods . . . . 9
  - **1.3.3** Summary . . . . 15
- **1.4** Existing comparative methods . . . . 15
  - **1.4.1** Conservation detection methods . . . . 16
  - **1.4.2** Methods for comparative functional prediction . . . . 17
  - **1.4.3** Summary . . . . 21
- **1.5** Theoretical background . . . . 23
  - **1.5.1** Pair hidden Markov models . . . . 24
  - **1.5.2** Alignment algorithms . . . . 24

**2 The pair HMM of** DOUBLESCAN **and** PROJECTOR **30**

- **2.1** Introduction and motivation . . . . 30
- **2.2** States **and** transitions of the pair HMM . . . . 32
- **2.3** Determination of the pair HMM's parameters . . . . 37
- **2.4** The Stepping Stone algorithm . . . . 42

**3 *Ab*** initio **prediction of mouse and human genes** **46**

- **3.1** Introduction and motivation . . . . 46

3.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
3.3 Prediction of conserved subsequences . . . . . . . . . . . . . . . . . . . . . . 51
3.4 Validation of the Stepping Stone algorithm . . . . . . . . . . . . . . . . . . . 51
3.5 Summary and discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

**4 Prediction of mouse and human genes 55**

4.1 Introduction and motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
4.1.1 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
4.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.3 Summary and discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59

**5 Prediction of C.** *elegans* **and C.** briggsae **genes 67**

5.1 Introduction and motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
5.2 Training of the pair HMM's parameters . . . . . . . . . . . . . . . . . . . . . 68
5.3 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
5.3.1 Performance on test set 1 . . . . . . . . . . . . . . . . . . . . . . . . 72
5.3.2 Performance on test set 2 . . . . . . . . . . . . . . . . . . . . . . . . 74
5.3.3 Comparison of the performance of DOUBLESCAN and FGENESH . . . . 83
5.4 Summary and discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86

**6** DOUBLEBUILD **88**

6.1 Introduction and motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
6.2 Special transitions within DOUBLEBUILD . . . . . . . . . . . . . . . . . . . . 89
6.3 Special emissions within DOUBLEBUILD . . . . . . . . . . . . . . . . . . . . 93
6.4 The main classes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
6.4.1 The `Pairhmm` class . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
6.4.2 The `Pairhmm_State` class . . . . . . . . . . . . . . . . . . . . . . . . 95
6.4.3 The Sequence class . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.5 Alignment algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
6.5.1 The Viterbi algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . 102
6.5.2 The Hirschberg algorithm . . . . . . . . . . . . . . . . . . . . . . . . 102
6.5.3 The Stepping Stone algorithm . . . . . . . . . . . . . . . . . . . . . . 102

**7 Conclusion 104**

**A Mouse human training and test sets** **107**

A.1 The training set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

A.2 The test set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

A.3 Post-processing of the predicted mouse and human genes . . . . . . . . . . . 110

**B Mouse human parameter tables** **111**

**C *C. elegans* C. *briggsae* training and test sets** **121**

C.1 The training set. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121

C.2 Test set 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122

C.3 Test set 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122

**D *C. elegans* C. *briggsae* parameter tables** **126**

**E The DOUBLESCAN web-server** **132**

**Bibliography** **142**

# Chapter 1

# Introduction

## 1.1 Motivation

**All** the inherited information that determines the physiology of an organism is encoded in a genome which is present in each cell of the organism. The genome of an organism typically consists of one or more molecules, each consisting of a linear succession of four **DNA** (deoxyribonucleic acid) bases symbolised by the four letters A, C, G and T. The complexity of an entire organism is thus encoded in a few long molecules of apparently striking simplicity.
Recent systematic genome sequencing efforts have determined the complete sequence of A, C, G and T letters for a number of organisms. Knowledge of the complete **DNA** sequence gives **us** the opportunity to study the genetics of organisms both at a fundamental molecular level and on a global scale. Since the first genome of a multi-cellular organism, the nematode ***Caenorhabditis elegans,*** was sequenced in 1998 [eSC98] comprising about 100 Mb (million bases), the genome sequence of the fruit fly ***Drosophila melanogaster*** (120 Mb) [ea00] and of the plant ***Arabidopsis thaliana*** (125 Mb) [Ini00] have been determined. We have good quality draft sequences of the human ***Homo sapiens*** [Con01] and the mouse ***Mus musculus*** genomes [Con02] (both around **3000** Mb) which will be completed in the near future. We are thus for the first time in the possession of the blueprints of several organisms, but without knowing how to understand the **DNA** text's contents.
The life of an organism depends on a variety of molecules that carry out specific tasks, one of the major groups being proteins. Each protein consists of a linear sequence of amino-acids which is encoded in a subsequence of the genome, called a gene. One of the most important challenges is to find the sections of the **DNA** which encode proteins, i.e. to find protein coding

genes, and to determine the amino-acid sequence of the encoded protein.

With the sequencing of the mouse genome soon to be finished, we *can* compare the human DNA sequence to that of the evolutionarily related mouse genome. By comparing the DNA sequences of two related organisms, we can not only study the large scale organisation of their genomes, but can also try to use pairs of related subsequences to predict genes. This is a promising approach because related organisms have similar proteins which are encoded by conserved genes in the DNA of the genomes. By finding and comparing subsequences which are conserved between two genomes we can try to predict protein coding genes.

The main goal of my work presented here is to develop a method for the comparative prediction of protein coding genes in pairs of related genomes. This task can be compared to the invention of a method for the automatic deciphering of the Rosetta stone. This stone contains one text in three different languages (Egyptian hieroglyphics, Demotic and Greek), see Figure **1.1,** and was carved in **196** BC in Egypt. When it was found in **1799,** only one of the three languages (Greek) was known. Manual comparative analysis of the three texts was completed in **1822** and led to the first understanding of both Egyptian hieroglyphics and Demotic. The task of comparatively deciphering the DNA texts of several related organisms is similar to the Rosetta stone deciphering in that we know that subsections of the texts are in close relation to ***each*** other, but it is complicated by the following:

- We do not have a DNA text which we completely understand, i.e. a text corresponding to the Greek text of the Rosetta stone does not exist in the DNA deciphering problem.
- The different sections of related DNA texts are not necessarily collinear or have a one-to-one correspondence. Figure **1.2** shows an example. There are **22** chromosomes plus the X and Y chromosomes in the human genome and **19** chromosomes plus the X and Y chromosomes in the mouse genome and the current estimate is that about **99 %** of the mouse genes have a corresponding human gene. We thus have to deal with rearrangements within the related DNA texts and cannot expect the contents of the texts (e.g. genes) to have a one-to-one correspondence.
- The DNA texts can be very long (around 3000 million A, C, G, T letters each for both the mouse and the human genome) and cannot be manually compared on a global scale and in a reproducible and efficient way.
- We do not know all the functional entities within DNA texts (like chapters, paragraphs,

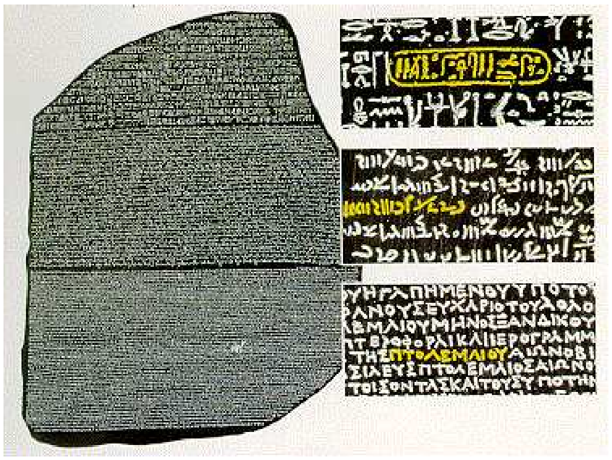


Figure 1.1: Photograph of the Rosetta stone. The upper text is written in Egyptian hieroglyphics, the text in the middle in Demotic and the lower text in Greek. The enlarged sections of the text show corresponding parts of the three texts. The highlighted word reads ‘Ptolemy’.

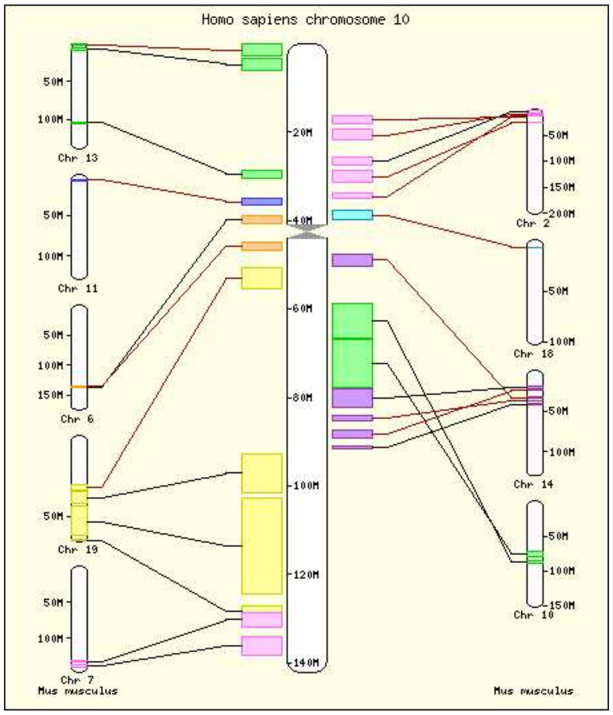


Figure 1.2: Large scale correspondence between human chromosome 10 (in the middle) and several mouse chromosomes (on the left and right) [Ens]. Note that most, but not all, of the human chromosome has a corresponding related region in the mouse genome. Also note that corresponding regions may appear in a rearranged region in the other organism (see human chromosome 10 and mouse chromosome 2) and that there is no collinearity between the human and mouse chromosomes at the large scale (see human chromosome 10 and mouse chromosomes 10 and 14).

```
...ctcagccttgtgtgagttgaggggaggtgtcacatccagctggagtcctttctaagcagc
   cacagcctgatcctcccacttcctcccccaagaaaacattgt~tgatggccataccc
   tgaggttctggtccaaatcggactttctatgaccttctgggtctctagtgaaaactaaag
   actcctctccagaaaaaaacatttggtttctaatgaggcctggaatcttattcttgacct
   ggggagcggaatccctttttgcagtactcccgggccctctgttggggcctccccttcctc
   tccagggtggagtcgaggaggcggggctgcgggcctccttatctctagagccggccctgg
   ctctctggcgcggggccccttagtccgggctttttgccATGGGGTCTCTGTTCCCTCTGT
   CGCTGCTG'ITRTMTGGCGGCCGCCTACCCGGGAGTTGGGAGCGCGCTGGGACGCCGGA
   CTAAGCGGGCGCAAAGCCCCAAGGGTAGCCCTCTCGCGCCCTCCGGGACCTCAGTGCCCT
   TCTGGGTGCGCATGAGCCCGGAGTTCGTGGCTGTGCAGCCGGGGAAGTCAGTGCAGCTCA
   ATTGCAGCAACAGCTGTCCCCAGCCGCAGAATTCCAGCCTCCGCACCCCGCTGCGGCAAG
   GCAAGACGCTCAGAGGGCCGGGTTGGGTGTCTTACCAGCTGCTCGACGTGAGGGCCTGGA
   GCTCCCTCGCGCACTGCCTCGTGACCTGCGCA~AAAACACGCTGGGCCAC~CCAGGA
   TCACCGCCTACAgtgagggacaggggctcggtcccggctggggtgaggggagggggctgg
   aagaggtgggggaagggtagttgacagtcgctctatagggagcgcccgcggacctcactc
   agaggctccccccttgccttagAACCGCCCCACAGCGTGATTTTGGAGCCTCCGGTCTTAA
   AGGGCAGGAAATACACTTTGCGCTGCCACGTGACGCAGGTGTTCCCGGTGGGCTACTTGG
   TGGTGACCCTGAGGCATGGAGCCGGGTCATCTATTCCGAAAGCCT~AGCGCTTCACCG
   GCCTGGATCTGGCCAACGTGACCTTGACCTACGAG~G~~TGGACCCCGCGACTTCT
   GGCAGCCCGTGATCTGCCACGCGCGCCTCAATCTCGACGGC~GGTGGTCCGCAACAGCT
   CGGCACCCATTACACTGATGCTCGgtgaggcacccctgtaaccctggggactaggaggaa
   gggggcagagagagttatgaccccgagagggcgcacagaccaagcgtgagctccacgcgg
   gtcgacagacctccctgtgttccgttcctaattctcgccttctgctcccagCTTGGAGCC
   CCGCGCCCACAGCTTTGGCCTCCGGTTCCATCGCTGCCCTTGTAGGGATCCTCCTCACTG
   TGGGCGCTGCGTACCTATGCAAGTGCCTAGCTATGAAGTCCCAGGCGtaaagggggatgt
   tctatgccggctgagcgagaaaaagaggaatatgaaacaatctggggaaatggccataca
   tggtggctgacgcctgtaatcccagcactttgggaggccgaggcaggagaatcgcttgag
   cccaggagttcgagaccagcctggacaacaacatagtgagaccccgtctatgcaaaaaataca
   caaattagcctggtgtggtggcccgcacctgtggtcccagctacccgggaggctgagttg
   ggaggatcctttgagccctgaaagtcgaggttgcagtgagccttgatcgtgccactgcac
   tccagcctgggggacagagcacgaccctgtctccaaaaataaaataaaaataaaaataaa
   tattggcgggggaaccctctggaatcaataaaggcttccttaaccagcctctgtcctgtg
   acctaagggtccgcattactgcccttcttcggaggaactggtttgtttttgttgttgttg
   ttgtttttgcgatcactttctccaagttccttgtctccctgagggcacctg~ttcct
   cactcagggcccacctggggtcccgaagccccagactctgtgtatccccagcgggtgtca
   cagaaacctctccttctgctggccttatcgagtgggatcagcgcgggccggggagagcca
   cgggcaggggcggggtggggttcatggtatggctttcctgattggcgccgccgccaccac
   gcggcagctctgattggatgttaagtttcctatcccagccccaccttc~accctgtgct
   ttcctggaggccaaacaactgtggagcgagaactcatctccacgctg
   gagtgagaccacgaatggtggggaggggagggtcccacggacatattgagggacgtggat
   acgcagaagaggtatccatgtggtggcagccgggaaggggtgatcagatggtccacaggg
   aatatcacaaactcgaattctgacgatgttctggtagtcacccagccagatgagcgcatg
   gagttgggggtggggggtgtcaaagcttggggcccggaagcggagtcaaaagcatcaccc
   tcggtcccttgttctcgcgtggatgtcagggcccccacccaccgagcagaaggcggactc
   aggggcgctccagggtggctcgagctcacacacgctgagtagacacgtgcccgctgcacc
   ctgggtaaatacagacccggagccgagcggattctaatttagacgcccgcgaacgctgcg
   cgcacgcacacgtgtcctcggctcgctggcactttcgtcccgccccctccgtcgcgtgcg
   ggagctgacccggaggggtgcttagaggtatggctccgcggggtcaaaaggagaaggatc
   agtgagagaggcatccccacaccctccc...
```

Figure 1.3: A short subsequence of DNA from the human ***Homo*** sapiens genome comprising one protein coding gene which is evolutionarily related to the gene contained in the mouse Mus ***musculus*** sequence shown in Figure 1.4. The protein coding parts of the gene are highlighted by capitalisation. They correspond to the protein coding exons which are shown **as** hashed boxed in Figure **1.5.** The DNA sequence of the whole ***Homo*** sapiens genome would correspond to about one million pages.

```
...gagtgtcttgtgagtttgtgtacagtcatcacatcagttaggcaaagccctaaggactgc
   cgactcccataatgcctcagggttgtctggtaacctaaccctaactctgagtctgtggat
   caggttggtccccacccccacccccttctttttttgagacaggttctctttgtggccatg
   gatgtcctgaaatctgctatgtggaatgggctggccttgacttcacaaagat~ccaac
   ctgtctcctgaatgctaggactaaatgacaaagccactgccatgtct~aaaatctacg
   ttagatagacagggtttcccagtgtagatcaggatggccttgaacttacagagatctgcc
   tccctgggagtgctgggatcaaaggcatgtgccatcaccaagcgttattttatttttttaa
   tttttaaagacttcttggggcttacgtaaaaactaaagagcaggtccagaactgtgcaat
   ggcttttggttgattgtagggtctgatgggaggggaggcaggtatcttcatcagggccgg
   ccgaggcccattctggggcgtggccagggtgccttcttatctcctgcggccagcctaaac
   tccctggcgttccgcccgcacttcagcgcgggctttgtgccATGGAGTCTGCCCTTCTGC
   TCCCGTCGCTTTTGCTGGTGGCTGCCTATCCGAGGGGTGGGAGCCCCCAGCAAGAGTGGA
   TGCAAAGTCCTCCCGCGCCTTCCGTGACCTCAGCACCI'TTCTGGGTGCGTCTTAATCCAG
   AGCTAGAGGCCGTGCCTCCCGGG~TCAGCGTG~AACTGCAGCCACAACTGCCCCC
   TGCCGGTGCATTCCAGCCTCGCACCCAACTGCGGCAGGGAAAGATAG~AATGGATCCG
   GCTGGGTATCTTACCAGCTACTGGATGTGAGGGCCTGGAATTCCAAGGTGCGCTGCGTCG
   TCACTTGCGCAGGAGAAACCCGAGAGGCCACCGCCAGGATCACTGCTTACAgtgagggag
   accggggctcaggccgggctggggtgaggggagaggggtggaggaagcggatagatggta
   attgctttaaggggtgcctgtgggccttatctctcttgccttagAACGGCCCAGAAGCGT
   GATCTTGGAGCCTCCGGTCCAGTGGGCCACAAGTACACTCTGCGAT~ATGTGACACA
   CGTGTTCCCAGTGGGATGTGGTGAGCCTGAGAAGAGGT~CGAGTGATTTATCA
   TGAAAGCCTGGAGCGCTTCACCGGTTCAGATTTGGCTAATGTCACI'TTGACCTACGTGAT
   GCGGGCCGGACTCAACGACC~~AGCCACTCACCTGCCATGCGCGCCTCAATCTCGA
   CGGGCTAGTGGTGCGCAGCAGCTCGGCACCTGTTATGTTGACAGTCCTCGgtgaggcatc
   ctgtaatcccagggaatgggtgcgggagaggggatgttgccactccaagggggcctgcag
   aacaggcgtgggctccacgcttggcggtaacctcctcagacctcctagttcctgattttcactcc
   tgcccacagCTTTAAGCCCAGCCTCTATAGCCTTGGCCTCTACCTCCATCGCAACCCTGG
   TGGGGATCCTCCTGGCTGTGGGGGCTGTCTACGTGCGCAAGTACCTGGCTGTGCAGACTt
   agttatagatctgttttcgatgcctgacaagagggaggggaaaagaacttcagagtaatt
   aattcagagactcttattgaaacaataaagtcttcctcctcagtctctgccttacggttc
   ttggagaaagtggtttctttttttaaggtaccttactttttccaaattccttacgtagggg
   ctgaagattagtagattagaggtagtactggaggaaacaacaccttgaaatttctccttc
   aaggccagcatggggtcctagaacccgagttcctctgcgtagagttttgttagctttatt
   tgtgcggggcagaaagactaaactgacctcccctccagggctgactcttggtatggcttt
   ttctgattggctccgctgatacaggccggagctctgattggaggctaagtttcccttctc
   ctccctccttttccactacggagcctgtgcgttactagagaaggccagcgggtggagcta
   gacctgattccccaaggttatcattaattgggggggggggggggaggtagaaacactcgag
   taggcggggccttcttcaagtagtagaggaagcggctaactágataggaaatctagcata
   gcaacaagttaagagatgattgttcaggccacgtgagctgtcacagacttgcttcctggc
   gttgtgcttgttgtctccgagtctggtatgtatgtagagagggatgtcaaagctggggtc
   aaagtgtccccagttgatcttttggtccagcgtgaattgcagaatctcgcactagttacc
   cagtagaggcggccacactcctggcgaggagggcgcagaagctctgctgagagactagac
   acacaacagcgttgtagacacattcccgctgcactctgggtaaataaagatcgggtgccg
   gagtcgactctaatttagaagcctgcgaacgctgcgcacacgcacacgtgtccgagtctt
   gctggcacttgatccccctcttccttcgccgcgtgcgcggag...
```

Figure 1.4: A short subsequence of DNA from the mouse *Mus musculus* genome comprising one protein coding gene. The protein coding parts of the gene are highlighted by capitalisation. They correspond to the protein coding exons which are shown as hashed boxed in Figure 1.5. The DNA sequence of the whole *Mus musculus* genome would correspond to about one million pages.

sentences and words) and how they structure the text hierarchically. We know for example of protein coding genes and promoters and other small entities, but do not know very much about how they are grouped into larger functional entities.

As is apparent from viewing only small pieces of DNA data such as those shown in Figure 1.3 and Figure 1.4, computational methods which can be applied to large amounts of data in a reproducible way have much potential for helping to unravel the text of genomes by proposing answers to biologically interesting questions which can be experimentally verified.
This introductory chapter provides the biological background, an overview of already existing methods for gene prediction and the theoretical background on which my work is built. Chapter 2 presents the pair HMM underlying DOUBLESCAN and PROJECTOR, two new methods which can be used for the comparative prediction of genes, as well as a new algorithm, called the Stepping Stone algorithm, by which genes can be predicted with essentially linear time and memory requirements, thus enabling large scale analyses. Chapter 3 demonstrates that DOUBLESCAN can be used to predict genes in mouse and human DNA. Chapter 4 presents a variant of DOUBLESCAN, called PROJECTOR, by which genes which are known in one organism can be used to find related genes in another related organism as exemplified on a set of mouse and human DNA sequences. Chapter 5 demonstrates that DOUBLESCAN and PROJECTOR can be easily adapted to analyse other pairs of related genomes by showing their performance for predicting genes in C. *elegans* and C. *briggsae* DNA sequences. Chapter 6 introduces a library of C++ classes by which large and complex projects such as DOUBLESCAN and PROJECTOR can be implemented in a short time.

## 1.2 Biological background

In eukaryotes, a subsequence of the genomic DNA is linked to its functional expression as a protein by a series of steps which can be roughly grouped into [HRS+87]:

- transcription of a DNA subsequence into an RNA (ribonucleic acid) sequence
- modification of the RNA sequence to produce a mature messenger RNA
- translation of the messenger RNA sequence into a protein sequence
- modification of the protein sequence

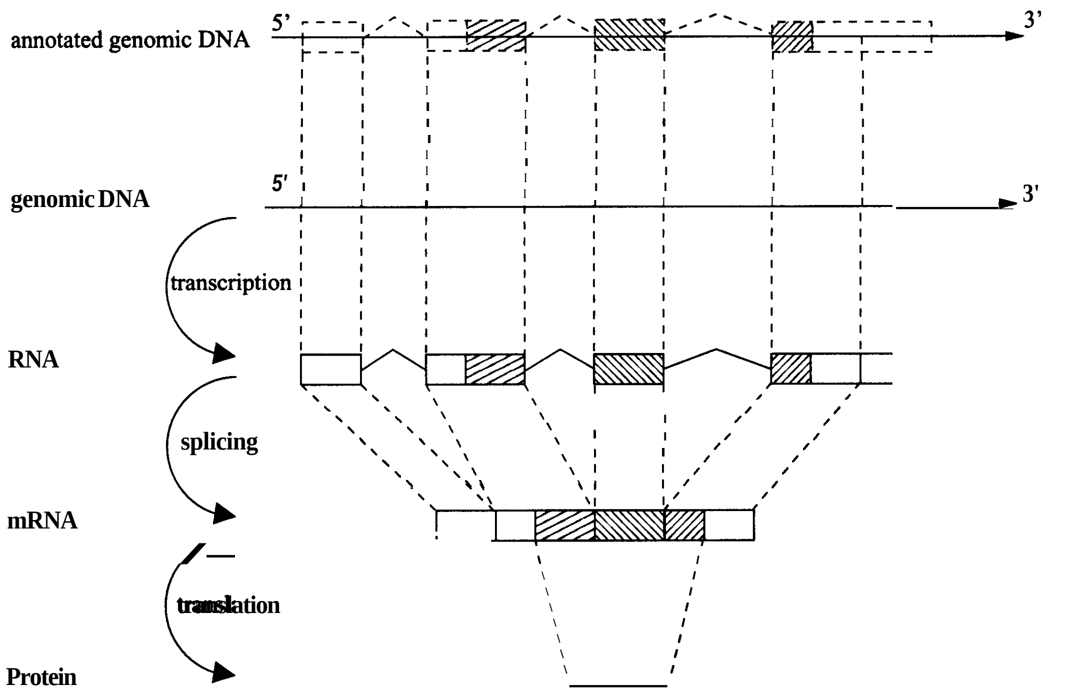


Figure 1.5: Schematic view of the steps by which a subsequence of the genomic DNA is linked to the amino-acid sequence of the protein it encodes. Each box (see boxes at RNA level) represents an exon and each kinked line an intron. The protein coding parts of each exon are hashed. See the text for a description of the processes.

See Figure 1.5 for a schematic view of the events in which the sequences involved are simplistically represented as linear molecules. A subsequence of the genomic DNA is transcribed into an RNA molecule by RNA polymerase after the genomic DNA has been prepared for transcription. The RNA molecule then undergoes a variety of modifications such as the modifications of its ends (as shown in Figure 1.5 for the 3' end) and splicing. During splicing small nuclear ribonucleoproteins (snRNPs) excise intronic sequences and join the exons into a shorter messenger RNA molecule (mRNA). This mRNA molecule is then transported from the nucleus to the cytoplasm where the continuous segment of exons is translated into the corresponding sequence of amino-acids by the ribosome and a variety of tRNA molecules. Depending on the final location of the protein, the amino-acid sequence may undergo modifications such as the cleavage of signal sequences. The protein coding part of a gene starts at the 5' end with a start codon and finishes on the 3' end before a stop codon. Splice sites are the 5' and 3' ends of introns.

The aim in *ab* initio gene prediction is to find genes and to infer their gene structures from a given DNA sequence of A, C, G and T letters only. The assignment of functional information to the DNA sequence is called annotation. By knowing the annotation of the DNA sequence i.e. the exon-intron structure of its genes as shown in Figure 1.5, we can directly infer the amino-acid sequence of the corresponding protein. In principle, it suffices to known the protein coding parts of the exons of a gene to derive the amino-acid sequence of the encoded protein. For the rest of this dissertation, the term gene refers to protein coding genes and the term exon to the protein coding part of an exon unless stated otherwise.

Traditionally, *ab* initio gene prediction methods for eukaryotes deal with one DNA sequence at a time. Methods for comparative gene *ab* initio gene prediction exploit the fact that related proteins have similar amino-acid sequences which are encoded in genes of similar exon-intron structure and that the exons of related genes are typically much more conserved than the introns which do not encode protein information.

As the method presented in this dissertation annotates two DNA sequences simultaneously, gene prediction methods which deal with only one sequence are only briefly reviewed and only those that are of relevance to this work are presented.

## 1.3 Existing non-comparative methods for *ab initio* gene prediction

There have been numerous studies with the aim to predict the intron and exon structure of eukaryotic genes given the DNA sequence as the only input information. Each of them typically consists of one or more programs which employ one or more methods to finally arrive at a prediction. The discussion is grouped by the methods employed rather than by the different ways in which they are combined into one program to emphasise the different underlying concepts.

### 1.3.1 Types of evidence

When trying to annotate a sequence of DNA we can make use of a variety of sequence signals which indicate the presence of functional elements or which mark a boundary between them. The principal measures used are:

**Coding measures** Exons and introns exhibit a different usage of nucleotide patterns. One statistically significant measure of difference found [CB86, JMCB90, FLS92, FT92] was that of relative frequencies of six nucleotide words, so-called hexamers.

**Sequence signals** Besides a compositional bias between exons and introns, their boundaries can be detected by certain sequence signals, as for example the acceptor and donor splice sites at the 5' and 3' sides of introns. Other signals include the translation initiation signal around the start codon (Kozak consensus [Koz81]), the translation terminal signal around the stop codon, the poly-adenylation signal and promoter sequences.
The **individual** statistical significance of any of these measures is not sufficient to reliably predict the exon and intron structure of a given DNA sequence [BEK91, CA96]. Only by combining several signals into a valid gene structure can we attempt to successfully predict genes.

### 1.3.2 Methods

This section concentrates on the description of those methods for integrating sequence signals which have turned out to be the most successful in ***ab* initio** gene prediction, namely neural networks, discriminants and hidden Markov models.

Besides these methods, other methods, such as rule-based methods (as used in the program GENEID [GKDS92]), linguistic methods (see for example the program GENLANG [DS94]) and decision trees (employed by the program MORGAN [SDFH97]) have been proposed to predict exons or genes.

### Neural networks

Neural networks in gene prediction are typically used for combining signals from numerous sources, as for example from sequence motifs and nucleotide frequencies, into one score.
A neural network consists of an input layer of so-called neurons which accept the input values, i.e. the scores. The input signals propagate from the neurons of the input layer to the neurons of one or more layers of hidden neurons until the propagated signal finally reaches the output neuron. The final result depends on the architecture of the neural network as well as on the function with which each neuron merges several incoming scores into one outgoing score. These functions typically depend on multiple parameters which are given some initial values without knowing their optimal values. These parameters and even the architecture of the neural network can be adjusted by training it with a representative data set for which the correct outcome is known. The trained neural network can then be used on unknown data sets. Signal propagation in the neural network is unidirectional though the training of the parameters need not be. A general overview on neural networks can be found in [Bis95].
Neural networks are used in the GRAIL program [UM91] to identify exons which are later assembled into genes. The program GENEPARSER [SS93] exemplifies how neural networks can be used in conjunction with other methods to predict the intron and exon structure of a DNA sequence. The neural network is used to combine the scores of different sources of information such as codon usage, compositional complexity, length distributions, k-tuple frequencies and splice site signals into one score under the hypothesis that an exon or intron is found at a certain position in the DNA sequence. Dynamic programming is then used to assemble these potential exons and introns at different positions along the DNA sequence into a gene structure with a valid splicing pattern which maximises the overall score.

### Discriminant methods

Linear discriminants are another approach to the classification of signal sequences that was used in the programs HEXON and FEX [SSL96]. Here an optimal separating plane is obtained between the true and false examples viewed as points in a multi-dimensional space, under the assumption that the true and false distributions are both gaussian with the same covariance matrix, but different means. The program MZEF [Zha97] employs quadratic discriminants [McL92] to predict independent internal coding exons in genomic DNA sequences. The characteristic features (e.g. splice site scores) of true and false internal exons are assumed to be described by two multinomial distributions in a multi-dimensional space which may have different means and different covariances. Quadratic discriminants can model the boundary between these two distributions and thus distinguish between true and false exons more effectively than linear discriminants as they are not limited to separating the two distributions by hyper-planes.

### Hidden Markov models

Hidden Markov models (HMMs) are a mathematical method to linearly label a sequence with labels from a finite set of states. The states of the finite set can be defined to reflect our knowledge of the biological problem and classify the letters of the input sequence into mutually exclusive classes, as for example 'intron' and 'exon' and other labels used for an annotation.

A hidden Markov model can be imagined as a finite set of states which are connected by directional transitions. Each transition connects two states and has a transition probability associated with it. Each state has a predefined action, for example it reads one letter from the input sequence and thereby assigns the state's label to it. From that state one can pass to one of the states to which it is connected. By thus walking along a state path in the Markov model, the letters of the input sequence are successively labelled with state labels.

The following paragraphs give some definitions and explain Markov models by giving a simple example.

**Definitions** A ***Markov*** model or ***Markov*** chain associates every random variable of a discrete time stochastic process $x_1, x_2, \ldots$ with a state from a finite set of states. A Markov chain is said to be of order $n$ if the probability of a transition from state $s_i$ at time $i$ to state $s_{i+1}$ at time $i + 1$ depends only on the $n$ previous states $(s_{i-n}, \ldots s_i)$. If the transition probabilities are independent of time, the Markov chain is said to be homogeneous, and inhomogeneous otherwise.

The notion of Markov chains can be extended by associating a probability distribution $\tau_s$ with every state s in order to model the time the Markov chain spends in this state. This type of Markov chain is called a semi Markov model [How71].
Another way to extend the notion of Markov chains is to define a so-called hidden Markov model *(HMM)* for which every step $x_i$ of the Markov process consists itself **of** a stochastic process that generates the observed values $y_i$. Also this inner stochastic process can take values from another finite state space [BP66, Rab89]. If the underlying Markov process is a semi Markov Model, the model is called an explicit state duration hidden Markov model, generalised hidden Markov model or hidden semi Markov model (HSMM) [Rab89, KHRE96]. A text book on the application of Markov models in the context of biology is [DEKM98].

Example **of** a simple **Markov** model The above definitions can be illustrated by the example of a very simple Markov model which is entirely trivial, but which helps to make the distinction between a Markov model and a hidden Markov model clear.
As already mentioned, a Markov model can be imagined **as** a finite set of states which are connected by transitions. Each state corresponds to one of the four observable bases of the DNA alphabet (**i.e.** A, C, G and T). The states are connected by directional transitions which each have a transition probability associated with them. $t_x(y)$ denotes the transition probability for going from state $x$ to state y. By reading the letters of an input DNA sequence $X = (x_1, x_2, \ldots, x_Z)$ of length $Z$, $x_j \in \{A, C, G, T\}$, the Markov model assigns the following probability to the sequence:

$$P(X) = \prod_{i=l}^{Z-1} t_{x_i}(x_{i+1})$$

As each state of this Markov model corresponds to one of the four possible observables A, C, G and T, the state path in this Markov model simply corresponds to the sequence **of** letters in the input DNA sequence.

Turning the **Markov** model into **a** hidden **Markov** model The above Markov model can be turned into a hidden Markov model by separating the states from the observables and by introducing emission probabilities e, for each state s. The emission probability of state s for reading letter $x_j$ at position $j$ in the sequence is denoted $e_s(x_j)$.
For a hidden Markov model, the above formula for the probability which the hidden Markov

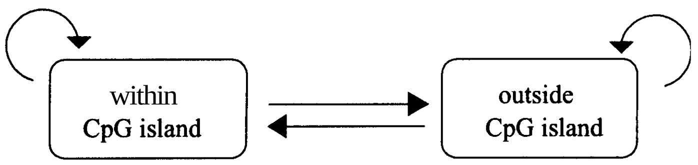


Figure **1.6:** Example of a simple HMM which classifies the letters of an input **DNA** sequence into those within and those outside CpG islands.

model assigns to an input **DNA** sequence $X = (x_1, 22, \ldots, x_Z)$ and a chosen state path $S = (s_1, s_2, \ldots, s_Z)$ of length $Z$ then reads:

$$P(X,S) = e_{s_1}(x_1) \cdot \prod_{i=1}^{Z-1} t_{s_i}(s_{i+1}) \cdot e_{s_{i+1}}(x_{i+1})$$

The use of emission probabilities in hidden Markov models facilitates the definition of states which closely represent biologically motivated classes. To give an example, see Figure **1.6:** when searching a **DNA** sequence for CpG islands [Bir87], we can encounter situations where a C in the **DNA** sequence can either be a frequently occurring C within a CpG island (read by the state within *CpG* island and thereby labelled **C** within *CpG* island) or a C outside a CpG island (read by the state outside *CpG* island and labelled **C** outside *CpG* island) which we would expect to encounter only rarely. The emission probabilities of these two states can be defined to distinguish between these two classes, e.g. the state within *CpG* island might have a high emission probability for reading a C and the state outside *CpG* island have a lower emission probability for reading a C. CpG islands are typically several thousand bases long and are better modelled using first order emission probabilities $e_{\text{state}}(x_i|x_{i-1})$ so that the probability of reading letter $x_i$ at position $i$ in the sequence depends on the letter $x_{i-1}$ at the previous position $i - 1$.

The action of states in HMMs can be extended to read zero, one or more letters from the input sequence.

**Using an HMM to predict an annotation** Once the states and transitions of a hidden Markov model have been defined to capture the features of the biological system which one wishes to describe and its emission and transition probabilities have been set, it can be used to assign a probability to a given input sequence and a chosen state path. In general, there exist a multitude of possible state paths and it is not clear a priori which state path to choose.

As the state path can be translated into an annotation of the input sequence, the aim is to select the state path that corresponds to the correct annotation of the sequence. The task is therefore to find a method which retrieves the desired state path for a given input sequence and a given HMM.

The above formula for $P(X, S)$ expresses the probability which the hidden Markov model assigns to a given input sequence X and a chosen state path S as function of the transition and emission probabilities encountered on the state path. If the emission and transition probabilities of the HMM have been chosen appropriately, we assume that the state path with the highest probability $P(X, S)$, denoted $S_{opt}$, corresponds to the correct annotation. The task is then to find this *optimal state path*, $S_{opt}$, which maximises $P(X, S)$. This optimal state path can be retrieved using the *Viterbi algorithm* [Vit67] and is therefore also called the *Viterbi path*. Once the optimal state path has been determined, its sequence of states can be translated into an annotation of the input sequence.

**HMM based gene prediction programs** The program GENSCAN [BK97] employs an explicit state duration HMM which models the length distribution of exons. It is capable of predicting complete, partial and multiple genes and simultaneously predicts genes on the forward and the reverse complemented strand of the input DNA sequence. GENSCAN's HMM has separate states for the exon of single exon genes and for initial, intermediate and terminal exons, as well as for a promoter, the 5' untranslated region, the 3' untranslated region and the poly-A signal. The HMM integrates information about several sequence signals such as splice sites, promoters, poly-A signals and start codons. GENSCAN's parameters are chosen according to one of four GC contents intervals [DMG95, ConOl] in which the GC contents of the input sequence falls. Initially, GENSCAN was trained to predict human genes, but its performance at nucleotide or exon level on genes of rodent (mostly mouse and rat DNA sequences) and non-mammalian vertebrates (fish, amphibian, reptilian and avian DNA sequences) is not much lower than that for primate genes, see [Bur97, pp. 106–1071. GENSCAN is one of the reference programs for the *ab initio* prediction of human genes.

A program which combines an HMM with neural networks is GENIE [KHRE96]. The program HMMGENE [Kro97] is also based on an HMM, but uses series of identical states to model length distributions for exons whereas DOUBLESCAN employs an explicit state duration HMM. For a given input sequence, HMMGENE reports the best labeled state path using a heuristic method (N-best method) rather than the most likely state path using the exact Viterbi algorithm as is done by **GENSCAN.**

### 1.3.3 Summary

The methods presented above can be grouped in two main groups: methods which can have a probabilistic interpretation and those which cannot. Those based on Markov models are the most amenable to a probabilistic interpretation. With some effort, rule-based methods, linguistic methods, decision trees and discriminants can also be provided with a probabilistic framework, see [SH94]. Neural networks lack this feature as probability tags attached to the input cannot be propagated to the output. This lack of statistical accessibility does not mean that they have an inferior performance with respect to other methods, but it limits the amount we can learn about how they produce results, how they can be trained and why they may fail to perform well.

Some features of the above methods, for example rule-based methods or decision trees, can be captured by hidden Markov models. The advantage of the latter method is that it can simultaneously work on splice sites in a way a decision tree method might do, and at the same time keep track of more global features such as the exon phase. If a hidden Markov model is set up correctly, it will by *definition* retrieve a valid state path. There is, for example, no need to go through a set of single exons and to decide how to combine them into one gene, a partial gene, several genes or maybe even no gene at all.

These are the main reasons why we chose to work with Markov models. Comparisons of the performance of different gene predicion program on a variety of data sets can be found in [BG96, Cla97, RHH+00].

## 1.4 Existing comparative methods

Research in the area of ***ab* initio** prediction of genes has so far focused on methods that take one DNA sequence and predict its gene structure, e.g. [BK97], and comparative gene prediction methods have only recently started to emerge. They use the same types of evidence as non-comparative methods, see Section **1.3.1,** together with similarity information from evolutionarily conserved subsequences and gene structures. The following paragraphs present the different methods which are used to combine these types of evidence into a prediction.

### 1.4.1 Conservation detection methods

Many of the earlier comparative methods do not aim to identify functional elements such as exons or entire gene structures, but only report subsequences which are conserved between two input DNA sequences without explicitly assigning a functional annotation to them.

**Dot plots** This method is based on the simple idea that two sequences can be compared by drawing one sequence along one axis and the other sequence along the other axis of a two-dimensional matrix and by assigning a 1 to a matrix element if the two corresponding letters of the sequences match, and a 0 where they do not match. This gives rise to a two dimensional matrix with 1s and 0s. Two identical subsequences give rise to a diagonal of 1s, whereas nonmatching subsequences correspond to areas with randomly distributed 1s. This is the basic principle upon which dot-plots are based. These plots can be refined by averaging over a selected diagonal and by applying some threshold value as done in the DOTTER program [SD96].
Dot plots do not predict a functional annotation as the underlying method does not know about exons, introns and valid gene structures.

**Percent identity plots** In this method, a gapped alignment is made between two sequences, say A and B. The percent identity plot is made by showing one of the two sequences, say sequence A, along the horizontal axis with the vertical axis showing how similar this part of sequence A is to the section of B which this is aligned to. Conserved regions show a high value of percent identity, non-conserved regions a low value.
A program called PIP was used in [OMM+97] to gain a first overview of the level of similarity between two DNA sequences. The authors refined their analysis by searching for gapped alignments using the SIM program [HHM90] in which the user can specify the penalty for a non-match and the two parameters for affine gap-penalties. These alignments were then transformed into precent identity plots relative to the positions in one of the two sequences. Similarly to dot plots, also percent identity plots do not give a functional annotation and do not predict genes.

**Block aligner** A pair hidden Markov model called DBA ('DNA block aligner') was introduced in [JBD99] to divide **DNA** subsequences into segments of different levels **of** percent identity. However, its states do not capture the different types of conservation between protein coding and non coding subsequences. DBA thus does not try to identify exons, introns or gene structures and was used in [JBD99] to study non protein coding **DNA** sequences in orthologous mouse and human gene pairs.

### 1.4.2 Methods for comparative functional prediction

The following methods have all emerged in the last few years and aim to make use of comparative information in two **DNA** sequences to predict functional elements such as exons or entire gene structures (refer to Figure **1.7** for an overview).

**Prediction of protein coding subsequences** The first attempt towards a comparative prediction of pairs of exons in two evolutionarily related **DNA** sequences was made in [KZ00] by introducing the program **WABA** ('wobble aware bulk aligner'). The underlying pair hidden Markov model (pair HMM, see Section **1.5.1** for an introduction) can distinguish between the different types of conservation between conserved protein coding and non coding **DNA** subsequences. It identifies and aligns subsequences which may be protein coding. However, as the pair HMM neither includes special states for splice sites nor uses scores from a splice site prediction program, the identification of the exact exon boundaries is not attempted.

**Incorporating similarity information into non-comparative hidden Markov models** Cross species similary can be incorporated into non-comparative methods such as hidden Markov models which operate on one **DNA** sequence only. [KFDB01] proposed an extension **of** the **GENSCAN** program [BK97], called **TWINSCAN,** which integrates cross-species similarity at **DNA** level into the probabilities of a non-comparative model. In the first step, a local alignment is generated between the target sequence (which is the **DNA** sequence to be annotated) and the informant sequence (which is a **DNA** sequence which is similar to the target sequence). This local alignment is then converted into a conservation sequence which indicates for every nucleotide in the target sequence one of three possible levels of conservation. Using the target sequence of **DNA** letters and the conservation sequence, the state path which maximises the joint probability of observing both the nucleotide and the conservation sequence is derived using the same optimisation algorithm as in **GENSCAN.** This joint probability is the product of the **DNA** sequence's probability and the conservation sequence's probability. The latter is calculated according to a conservation model which is defined for every state in the HMM of

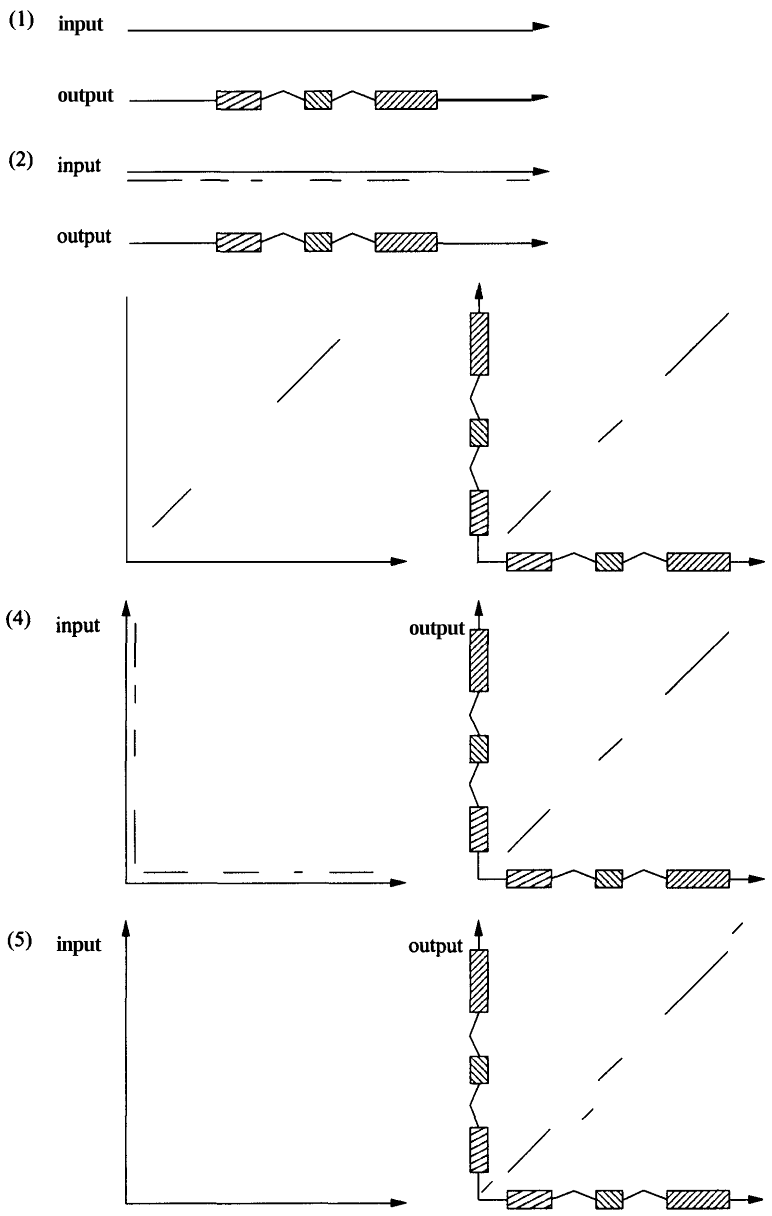


Figure 1.7: Overview over different types of ab initio gene prediction methods: **(1)** non-comparative gene prediction (e.g. **GENSCAN**), (2) non-comparative gene prediction which integrates homology information from a local alignment (e.g. **TWINSCAN**), (3) comparative gene prediction which is based on a global alignment (e.g. **GLASS** and **ROSETTA**), **(4)** comparative gene prediction which is based on a local alignment (**e.g. CEM** and **SGP-1**) and **(5)** comparative gene prediction where both alignment and genes are simultaneously predicted (**DOUBLESCAN** used with the Hirschberg algorithm). Refer to the text for a detailed description of the methods.

GENSCAN and which is based on fifth order Markov chains. TWINSCAN is not symmetric in the two input sequences and typically uses repeat-masked input sequences.

**Comparative gene prediction in both sequences using a multi-step approach** Any method for comparative ***ab* initio** gene prediction has to solve two problems: that of aligning the two input sequences and that of predicting gene structures for each of the two input sequences. One can try to solve these two problemes simultaneously (as is attempted in this dissertation, *see* Chapter **2,** Chapter **3** and Chapter ***5)***, but one can also try to solve them with some level of independence.
The latter approach was taken in the following studies:
[BPM+00] predicts gene structures in a two step approach by first globally aligning the two input DNA sequences using the program GLASS and by then identifying coding exons in both sequences and by merging them into identical gene structures using the program ROSETTA. The gene structures in the two sequences are assumed to have the same number of exons. The program does not deal with the two strands of each sequence simultaneously, but generates two independent gene predictions, one for each strand. The program works throughout with repeat-masked sequences which are used both for generating the global alignment and also the final gene structures.
Another multi-step program, called CEM ('conserved exon method'), is presented in [BH00]. In the first step, a local alignment of the two repeat-masked input sequences is generated using one of the existing programs. The next steps are executed for every match of the local alignment separately: a set of putative conserved exons is identified for every sequence separately. Next, only those pairs of putative conserved exons are retained that contain the match. The optimal alignment between the start point of each exon pair and the midpoint of the match, as well as that between the midpoint of the match and the end point of each exon pair is calculated using full dynamic programming. These alignments are converted into a set of alignments between every start and every end point which each have the score associated to it that was calculated in the dynamic programming. For every match of the initially generated local alignment, we then have a set of n-tuples each consisting of a start and end point, an alignment between them and a score. Complete gene structures are then built from this set of all n-tuples using dynamic programming with the assumption that the correct orthologous gene structures have the highest overall alignment score. CEM is capable

of predicting partial, complete and multiple genes. It predicts genes on both strands by running once on the forward and once on the reverse complemented strand and then merging the results into one set of genes which can lie on both strands. The prediction steps in CEM rely very much on the matches returned by the local alignment. If an exon pair is not hit by a match in the local alignment, it will be missing in the predicted genes. Similar genes are assumed to have the same number of exons.

Another example for the multi-step approach to comparative gene prediction is introduced in a program called SGP-1 (‘syntenicgene prediction’) [WGJMOG01]. In the first step, a local pairwise alignment is computed with one of the available programs. The matches of the local alignment may then be post-processed to reduce noice, if desired. In the second step (which is completely independent of the first step), a list of potential exons is generated for ***each*** of the two sequences separately. In the third step, the results of the first two steps are merged by retaining only those exons that are compatible with the alignment. This generates pairs of potential exons. In the fourth step, each exon pair is a assigned a score which is the sum **of** a similarity score and a sequence signal score. Finally, the list of exons is assembled into gene structures for each sequence independently. SGP-1 can deal with genes on both strands **as** well **as** with the partial and multiple genes. Further, **as** it is based on a local alignment of the two sequences, the genes do not have to appear collinearly within the two sequences. As the gene structures within each sequence are assembled independently **of** the other sequence, a one-to-one relationship between the genes in the two sequences or a one-to-one correspondence between the exons of two related gene structures are not automatically guaranteed. SGP-1 relies on the initial local alignments only for the definition of potential exons, but does not use similarity information to predict similar gene structures in the two sequences simultaneously (though this is what is likely to happen effectively if the sequences are well conserved and the local alignments coincide with the global alignment). There are thus very few steps involved in SGP-1 which depend quadratically on the length of the input sequence and which **are** thus time and memory consuming (a problem which **GLASS** and **ROSETTA** face to some extent and which limits the use of the fully pair HMM based programs DBA and WABA to rather short sequences). On the other hand, the similarity information is there and it should be possible to make good use of it in the simultaneous prediction of gene structures in the two sequences.

***Ab* initio comparative gene prediction using pair HMMs** Pair hidden Markov models (pair HMMs) are the natural generalisation of hidden Markov models (see Section 1.3.2) to two input sequences. They provide a fully probabilistic framework and the pair HMM's states, transitions and parameters have an intuitive interpretation. Pair HMMs are introduced in Section 1.5.1. As the mathematical concept of Markov models was very successfully applied to non-comparative ab initio gene prediction as shown by the program GENSCAN, see Section 1.3.2, it is tempting to try to use pair HMMs for the comparative *ab* initio prediction of genes.

[NGM01] present a method called PRO-GEN which they evaluate on a set of human-mouse, human-Xenopus and human-Drosophila *melanogaster* gene pairs. The underlying pair HMM can deal also with pairs of genes that are related by events of exon-fusion or exon-splitting, but it assumes each of the two input DNA sequences to contain exactly one complete gene. It predicts genes only in the input sequences, but not simultaneously in their corresponding reverse complemented strands. In a first step, potential splice sites and translation start and end sites are predicted for each of the two input DNA sequences separately. The Viterbi matrix is then calculated taking into into account the constraints imposed by the potential splice sites and translation start and end sites. The Viterbi algorithm is then used to derive the optimally scoring state path through the pair HMM with memory and time requirements which depend quadratically on the length of the input sequence. Pairs of codons within exons are scored using scores from the PAM120 matrix, whereas introns are not scored on a nucleotide by nucleotide basis, but rather by a fixed constant which is independent of the intron's length. A prediction generated by PRO-GEN consists of a complete gene in each DNA sequence as well as an alignment between corresponding exons. Note that the program does not predict conserved subsequences within introns or intergenic regions.

The pair HMM presented in this dissertation has been accepted for publication [MD02]. Recently, similar strategies were proposed by other authors [PAC01], but no implementation and evaluation has yet been published.

### 1.4.3 Summary

The comparative methods above can be subdivided into those that try to predict functional elements and those that do not. The latter provide some sort of alignment between two input DNA sequences which subdivides every sequence into subsequences of different levels of conservation without explicitly giving them a functional prediction. The findings in [JBD99] suggest that the length and percent identity of these conserved subsequences are generally not sufficient to allow their reliable functional annotation.

In order to attempt the comparative prediction of genes, sequence signals such as splice sites and start codons have to be integrated into the prediction process and single functional elements such as potential exons have to be grouped into valid gene structures. As mentioned earlier, any method for comparative *ab initio* gene prediction has to solve both an alignment and a gene prediction problem. These two problems can be solved simultaneously or sequentially, with some independence (see Figure 1.7).

Except for the method presented in [NGM01] (which assumes the presence of one complete gene in each input DNA sequence), the method proposed in [PAC01] and the method presented in this dissertation, the methods developed so far align and predict gene structures sequentially. This approach has the advantage that the time and memory requirements of most steps in the prediction process scale linearly with the length of the input DNA sequence as they are applied to each sequence independently. The multi-step methods for comparative *ab initio* gene prediction are therefore naturally well suited for applications on large DNA sequences. However, as the final gene prediction steps rely on the initial local or global alignment between the two input sequences, errors in the initial alignment may propagate to the gene prediction step which then has difficulties correcting for them. These methods thus assume that a fairly accurate local or global alignment can be made between the two input sequences. Furthermore, the prediction of the final gene structures in the two sequences is either done independently and thus does not make maximal use of the similarity information (as e.g. in the program SGP-1) or is done in very close dependence (as e.g. in the program CEM) which is probably best suited for pairs of closely related genes.

As opposed to the multi-step methods which try to solve the alignment and gene prediction sequentially, pair HMM based methods are suited to solve both in one step. As is shown in this work (see Chapter 2), the states and transitions of a pair HMM can be set up to simultaneously align the two input DNA sequences and to predict gene structures in both of the two sequences. The aim is to thereby obtain both, improved gene predictions and an improved global alignment which should also highlight conserved subsequences of yet unknown function, see (5) in Figure 1.7. The mathematical concept of pair HMMs can be used in a fully probabilistic way and sequence signals such as splice site scores and start codon scores

can be fully integrated.
Depending on the definition of states, the pair HMM can also be set up to be able to align more diverged pairs of genes which are related by events of exon-fusion or exon-splitting. The main advantage of pair HMM based methods is that the gene prediction process is not separated from the alignment process and the similarity information between the sequences is fully used to aid the gene prediction process and vice versa. The heuristical ideas and assumptions used in the multi-step methods which may impose unjustified prejudices and restrictions on the gene finding procedure, are essentially not needed within pair HMMs and the prediction process relies only on a few very basic assumptions. However, the integrated alignment and gene prediction approach has the disadvantage that the time and memory requirements of the prediction process scale quadratically with the length of the input sequence which limits the applicability to rather short DNA sequences or makes the implementation technically challenging. We solve this problem by introducing the Stepping Stone algorithm whose memory and time requirements scale linearly with the length of the input sequence, see Chapter **2.**
It remains to be seen how well multi-step methods perform in comparison to each other and to pair HMM based methods, how each method performs on more diverged pairs of genes and how readily it can be adapted to successfully analyse other pairs of genomes. It will be crucial to see how the performance of each method scales when going from nucleotide level to gene level **as** this should be a good indicator of how well and in which way similarity information is utilised within each method (except for TWINSCAN and PRO-GEN, the gene level performance of the above mentioned comparative methods is not reported).

## 1.5 Theoretical background

Traditionally, ***ab*** initio gene prediction deals with one DNA sequence at a time. Among the most successful methods are hidden Markov models **as** exemplified by GENSCAN [BK97] and HMMGENE [Kro97]. In order to extend gene prediction to work on two DNA sequences simultaneously, we employ an extension of hidden Markov models, called pair hidden Markov models (pair HMMs) [DEKM98, KZOO].

### 1.5.1 Pair hidden Markov models

In analogy to the previously introduced Markov models, see Section 1.3.2, the definition of a state can be extended to deal with two sequences instead of only one sequence. All previously given definitions and remarks which were made for variants of Markov models in Section 1.3.2 also apply to pair hidden Markov models.

The difference between a pair HMM and an HMM is that a pair HMM deals with two input sequences instead of only one. The states of a pair HMM read letters from only one of the two input sequences or from both of them. As for HMMs, a state of a pair HMM assigns an emission probability to the letters it reads. The pair HMM then passes to one of the states to which the current state is connected by directed transitions and assigns a transition probability to this action. This procedure is repeated until all letters of both sequences have been read. The sequence of states passed through is called the state path.

Each state assigns labels to the letters it reads, as for example 'intron' or 'exon'. A state path can therefore be translated into annotations for both DNA sequences.

### 1.5.2 Alignment algorithms

Once the transition and emission probabilities of the pair HMM have been specified, the pair HMM can be used to predict an annotation for the two input DNA sequences by finding the optimal state path, $S_{opt}$. To any chosen state path S and a given pair of sequences X and Y, the pair HMM assigns the following probability:

$$P(X,Y,S) = e_{s_1}(k_1,p_1) \cdot \prod_{i=1}^{Z-1} t_{s_i}(s_{i+1}) \cdot e_{s_{i+1}}(k_{i+1},p_{i+1})$$

The sequence of states encountered on the state path is S $= (s_1, s_2, \ldots, s_Z)$, $Z$ being the length of the chosen state path. $t_{s_i}(s_{i+1})$ is the transition probability to go from the i-th state $s_i$ to the i $+$1-th state $s_{i+1}$. $e_s(k,p)$ is the emission probability of state $s$ to read $\Delta_x(s)$ letters from sequence $X$, namely letters $x_{k-\Delta_x(s)}, \ldots, x_{k-1}$, and to read $\Delta_y(s)$ letters from sequences Y, namely letters $y_{p-\Delta_y(s)}, \ldots, y_{p-1}$. After the i-th step in the state path, we are therefore in state $s_{i+1}$ at position $k_{i+1}$ in sequence X and at position $p_{i+1}$ in sequence Y. At the end of the state path, i.e. after $Z$ steps in the pair HMM, all letters of the two sequences have been read.

As with a single sequence HMM, it is clear that there is a multitude of possible state paths for a given pair HMM and a given pair of input sequences. The aim is to find the state path which corresponds to a correct annotation for both sequences. The assumption is that, with appropriately chosen emission and transition probabilities, the state path with the highest probability $P(X, Y, S)$, denoted $S_{opt}$, corresponds to a correct annotation. The task is then to find this optimal state path, $S_{opt}$, that maximises $P(X, Y, S)$.
The basic method to retrieve the optimal state path is the Viterbi algorithm [Vit67]. We also introduce here the Hirschberg algorithm [Hir75] which finds the optimal state path with linear memory requirements.

### The Viterbi algorithm

The optimal state path can be found using the Viterbi algorithm [Vit67]. This algorithm solves the optimisation problem in two steps. In the first step, the elements of a three dimensional matrix, the ***Viterbi matrix,*** are iteratively calculated. In the second step, a traceback process through the matrix retrieves the optimal state path.
Let $N$ be the number of states and $T$ the number of transitions in the pair HMM and $L_x$ and $L_y$ the lengths of the two input sequences $X$ and $Y$, respectively. The value of each element in the Viterbi matrix, denoted $v(s, i, j)$, corresponds to the probability of a state path which ends in state $s$ and which has so far read $i$ letters from sequence $X$ and $j$ letters from sequence $Y$. By definition, every state path starts in the ***begin*** state, $s = 0$, and finishes in the ***end*** state, $s = N - 1$.
The elements of the Viterbi matrix are calculated as follows:

- Initialisation step:

  Set $v(0, 0, 0) = 1$ and all other $v(s, i, j) = 0$. This forces every state path to start in the ***begin*** state, $s = 0$.

- Recurrence relation:

  The $v(s, i, j)$ are iteratively calculated by looping over all $i \in \{1, \ldots, L_x\}$, all $j \in \{1, \ldots, L_y\}$ and all states $s \in \{1, \ldots, N - 2\}$ (the ***begin*** state, $s = 0$, and the ***end*** state, $s = N - 1$, need not be considered as they are only used at the start and end of each state path):

$$v(s,i,j) = \max_{s'\in\{1,\ldots,N-2\}} \{v(s', i - \Delta_x(s), j - \Delta_y(s)) \cdot t_{s'}(s) \cdot e_s(i,j)\}$$

where

- $t_{s'}(s)$ is the transition probability to go from state $s'$ to the state $s$.
- $e_s(i,j)$ is the emission probability of state s to read $\Delta_x(s)$ letters from sequence $X$ , namely letters $x_{i-\Delta_x(s)}, \ldots, x_{i-1}$, and to read $\Delta_y(s)$ letters from sequences $Y$, namely letters $y_{j-\Delta_y(s)}, \ldots, y_{j-1}$.
- Note that instead of maximising over all states $s' \in \{1,\ldots,N-2)$ only those states $s'$ for which a transition to s exists have to be considered.

- Termination step:

  The constraint that every state path has to end in the ***end*** state, s $= N - 1$, is implemented by setting

  $v(N-1, L_x, L_y) = \max_{s'\in\{1,\ldots,N-2\}} \{v(s', L_x - \Delta_x(s), L_y - \Delta_y(s)) \cdot t_{s'}(N-1)\}$

  This probability can be shown [Vit67] to be equal to the probability of the optimal state path, $S_{opt}$.

At this state, the probability of the optimal state path is known, but the path itself has still to be retrieved. Once the elements of the Viterbi matrix have been calculated, the optimal state path, $S_{opt}$, is retrieved by starting at the matrix element $v(N-1, L_x, L_y)$ whose value is equal to $P(X, \mathbf{Y}, S_{opt})$ and by recursively determining the state from which the maximum at the current state was derived. Using this traceback method, the sequence of states of the optimal state path is retrieved. The annotations of the two DNA sequences as well as the conserved subsequences can be deduced from this state path.

For a pair HMM with $N$ states and $T$ transitions and two sequences of length $L_x$ and $L_y$, respectively, the memory requirement for the Viterbi algorithm is of order $O(N \cdot L_x \cdot L_y)$, as this is the number of elements in the Viterbi matrix. The time requirement is of order $O(T \cdot L_x \cdot L_y)$, which is essentially the time consumed to calculate the elements of the Viterbi matrix.

It is clear that the quadratic dependency on the sequence length imposes serious restrictions on the applicability of the Viterbi algorithm on long sequences. For example, two sequences of $10^3$ base pairs length and a pair HMM with 50 states would need about 400 MB memory (numbers saved in double format) to save the Viterbi matrix. The same pair HMM used on two sequences of $10^4$ base pairs length would need a hundred times more memory and time to complete the calculation of the Viterbi matrix.

### The Hirschberg algorithm

The dependency of the Viterbi's memory requirement on the product $L_x \cdot L_y$ imposes a serious constraint on the analysis of long sequences. The Hirschberg algorithm [Hir75] linearises the memory requirement while still retrieving the optimal state path.

The key idea is to make use of the following underlying symmetry: instead of starting the calculation at the start of the two sequences, i.e. sequence positions $(x_1, y_1)$, we may as well start it at their ends, $(x_{L_x}, y_{L_y})$. This can be done by using a mirrored model which is created from the original pair HMM by reversing the directions of all arrows and by permuting the ***begin*** and ***end*** state with respect to the original pair HMM. This reversed pair HMM does not admit a probability interpretation any more because the probabilities of the transitions emerging from each state do no longer add up to one (instead, the probabilities of the transitions leading into each state add up to one). In the following, this model is called the ***mirror model.***

The Hirschberg algorithm divides the Viterbi matrix, see (1) in Figure **1.8,** into two halves which can each be calculated independently. One sub-matrix is calculated using the pair HMM starting at $(x_1, y_1)$ and proceeding towards higher values of the sequence index $i$, the other sub-matrix is calculated using the mirror model starting at $(x_{L_x}, y_{L_y})$ and proceeding towards lower values of the sequence index $i$, see (2). Instead of storing the whole Viterbi matrix, only the values in a narrow strip like volume are stored because only these are needed to continue the calculation, see the hatched areas in (2). The minimum strip width is equal to the maximum number of letters which are read from a sequence by a state in the pair HMM plus one, to store the row of new values. The process is stopped when the two strips overlap, see **(3).** The probability of the optimal state path, $P(X, Y, S_{opt})$, is then found by multiplying the appropriate values in the two strips and by searching for their maximum which is equal to $P(X, Y, S_{opt})$. We then not only the know the probability of the optimal state path, $S_{opt}$, but also the coordinates $(s, i, j)$ where the optimal state path crosses the two superimposed strips, see **(4).** The same procedure is then applied to the two emerging

sub-matrices whose boundaries are now known, one with sequence coordinates from $(x_1, y_1)$ to $(x_i, y_j)$ and the other one from $(x_i, y_j)$ to $(x_{L_x}, y_{L_y})$, see (5) and (6), until the adjacent coordinates of the optimal state path, (7), are at most separated by a sub-matrix of some predefined maximum size, (8). These coordinates are then used as boundary conditions to run the Viterbi algorithm separately on all the small sub-matrices. In the end, the state paths of the small sub-matrices are concatenated into the optimal state path from start state $s = 0$ at $(x_1, y_1)$ to the end state $s = N - 1$ at $(x_{L_x}, y_{L_y})$.
Using the Hirschberg algorithm, the memory requirement reduces to $O(N \cdot min\{L_x, L_y\})$. As each iteration halves the volume of the matrices that have to be calculated, the time used by the Hirschberg algorithm is at most twice the time used by the Viterbi algorithm, i.e. still of order $O(T \cdot L_x \cdot L_y)$.
The benefits of the Hirschberg algorithm are:

- The memory requirement of the Viterbi algorithm can be reduced to $\mathcal{O}(N \cdot min\{L_x, L_y\})$, i.e. the memory required to save the two strips which each have length $min\{L_x, L_y\}$, minimal width and height $N$.
- As all sub-matrices have known boundary conditions, they can be calculated independently, possibly in parallel and on several computers.

To summarise, the Hirschberg algorithm finds the optimal state path and the optimal score with a memory requirement which scales only linearly with the sequence length and in a time at most twice the time needed by the Viterbi algorithm (as each iteration halves the area which has to be calculated, the time used by the Hirschberg algorithm is at most $t + t/2 + t/4 \ldots = 2 \cdot t$, i.e. twice the time $t$ taken by the Viterbi algorithm). However, this time requirement of the Viterbi algorithm scales with $L_x \cdot L_y$ and still imposes a serious constraint on the analysis of long DNA sequences. In Chapter 2, we introduce a new algorithm which reduces both time and memory requirements to effectively linear dependence on the sequence length.

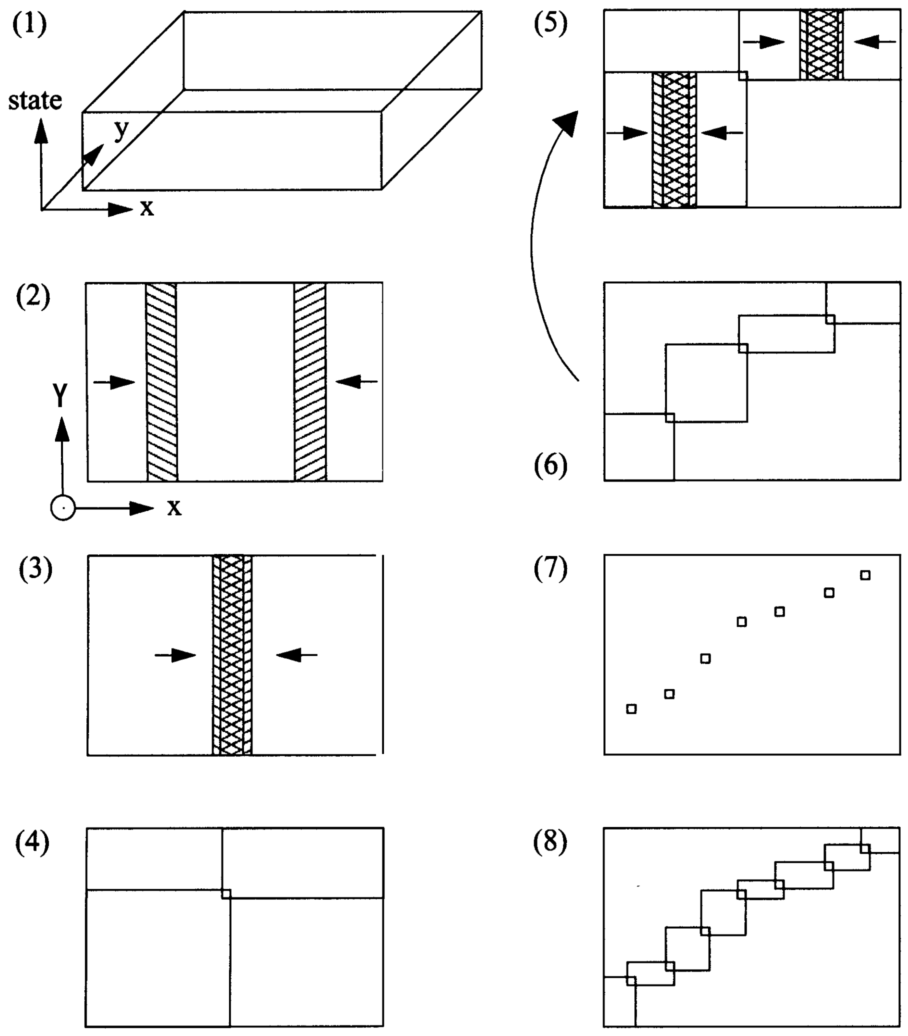


**Figure 1.8: The Hirschberg algorithm. (1) shows the three-dimensional Viterbi matrix. (2) to (8) show only its two-dimensional projection onto the plane spanned by the two sequences $X$ and $Y$. See text for details.**

# Chapter 2

# The pair HMM underlying DOUBLESCAN and PROJECTOR

## 2.1 Introduction and motivation

It is clear from previous comparative studies discussed in Section 1.4 that we cannot reliably infer the gene structures within two DNA sequences from a set of matching subsequences only. In order to comparatively predict gene structures, we would like a method which

- is symmetric with respect to the two input sequences
- keeps track of a valid splicing pattern simultaneously in each of the two DNA sequences
- does not assume that the input sequences contain a certain gene structure, for example one single complete gene
- can make use of different notions of similarity such as similarity at protein level as well as similarity at DNA level
- is able to incorporate information about sequence signals such as splice sites

As we wanted to see if two related DNA sequences would enable us to predict genes in an *ab initio* way, the method should

- use the two DNA sequences as the only input information and should in particular not have to know the amino-acid sequences they encode or how the two DNA sequences should be aligned.

The mathematical concept of pair hidden Markov models is well suited for achieving all of the above aims. It can treat each of the two input sequences on an equal footing. The pair HMM's states and transitions can be defined to enforce a valid splicing pattern in each of the two DNA sequences and to enable the prediction of a variety of different gene configurations which is not limited to predicting single complete genes. The different notions of similarity can be incorporated into the emission probabilities of the pair HMM's states. The strength of a variety of sequence signals such as translation start sites, splice sites and other functional elements can be translated into scores which are then used within the pair HMM to modify the nominal values of the transition probabilities. This is a generalisation of the standard form of pair HMMs as introduced in Section 1.5.1 which facilitates the efficient treatment of sequence dependent scores (see Chapter 6 for details).

In developing the states and transitions of the pair HMM underlying both DOUBLESCAN and PROJECTOR, we want the gene prediction to be mainly guided by the similarity information between the two DNA sequences. The different types of conservation between the two DNA sequences should, together with the constraint to produce a valid splicing pattern as imposed by the architecture of the pair HMM, enable the simultaneous comparative prediction of pairs of related genes.

The pair HMM can distinguish two different types of conservation as shown in Figure 2.1. The patterns of conservation between two DNA sequences can be different even if the overall percent identity between the two sequences is the same. It is this pattern of conservation which is used in order to distinguish conserved protein coding DNA from conserved non protein coding DNA. If the pattern of conservation has a three base pair periodicity and if the bases of the DNA can be grouped into triplets which could be interpreted as codons encoding the same or a chemically similar amino-acid, the DNA is likely to be protein coding and not protein coding otherwise.

We try to keep the number of assumptions on how a gene in isolation should look like, to a minimum of biologically well motivated assumptions and rather focus on implementing assumptions on the similarities which two related genes should exhibit. In particular we refrain from explicitly modelling the length distributions of exons and introns within a gene or the number of exons within a gene. Instead, we implement the assumption that two related genes should encode similar sequences of amino-acids which should be distributed onto the same or a similar number of exons of the same or similar length. This approach enables us

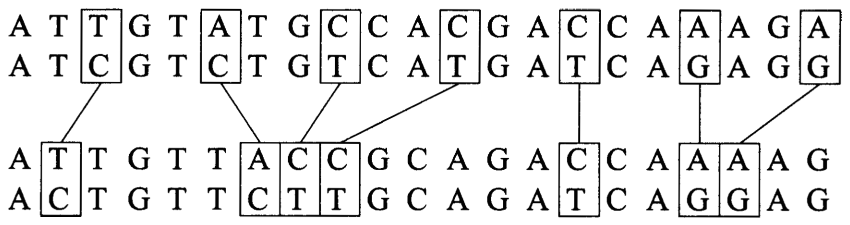

A T T G T A T G C C A C G A C C A A A G A
A T C G T C T G T C A T G A T C A G A G G
A T T G T T A C C G C A G A C C A A A A G
A C T G T T C T T G C A G A T C A G G A G

Figure **2.1:** Two alignments with the same percent identity, but different types of conservation. Boxes show un-conserved bases. The upper alignment exhibits a conservation pattern with a periodicity of three bases indicating pairs of codons which encode the same or a similar amino-acid, whereas the lower alignment shows no apparent pattern of conservation. The lower alignment is related to the upper alignment by permutations of the columns of aligned bases.

not only to detect novel genes whose amino-acid sequence is not yet known, but also to detect pairs of unusual genes. The main reason for choosing this approach is that genes, however unusual they might be, should be similarly unusual in a related organism and should therefore be detectable by our comparative method.

In the following, we first describe the pair HMM of DOUBLESCAN and PROJECTOR, its states and transitions, then explain how the parameters of the model are derived and conclude with a presentation of a new algorithm by which gene predictions are generated with essentially linear time and memory requirements.

## 2.2 States and transitions of the pair HMM

The aim in defining the states and transitions of the pair HMM is to be able to capture the most important configurations which can arise from the generic alignment of two homologous genes, see Figure 2.2. Suppose that one of the two DNA sequences, the first sequence in Figure 2.2, comprises all exons and introns which correspond to one protein. The homologous gene in the second DNA sequence, originating from the same or a different organism, may contain the same number of introns (a), an additional intron (b) or one intron less *(c)*. It can thus happen that the level of similarity between two DNA sequences is not high, even though they encode very similar amino-acid sequences.

The states and transitions of the pair HMM were defined so that the exons of two homologous genes can be aligned even if the genes are related by events of exon-fusion or exon-splitting. The pair HMM consists of 54 states. Every state of the pair HMM classifies every letter it

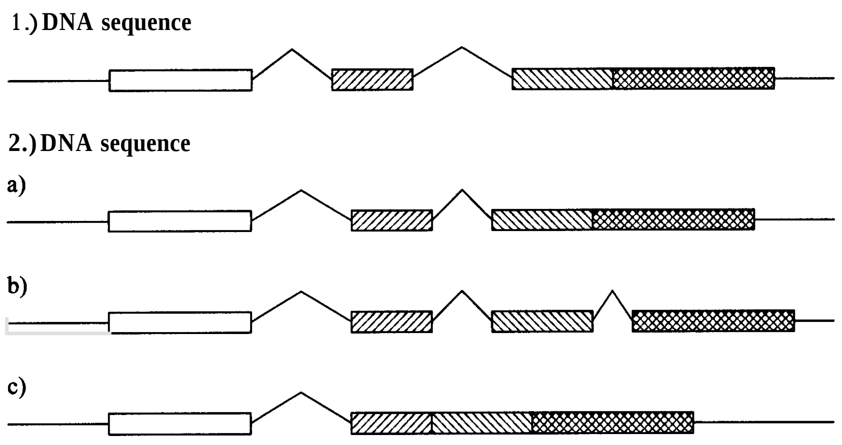


Figure 2.2: Different pairs of homologous genes. Boxes represent exons, kinked lines introns and straight lines intergenic sequences. Similar exons have the same hatching.

reads into four mutually exclusive classes: intergenic, non protein coding exon, intron and exon. Every ***match*** state reads the same number of letters from both sequences and assigns them to the same class, say exon DNA. We do not expect two homologous DNA sequences to exhibit the same features in the same length: even though the two encoded amino-acid sequences may be very similar, they may not have exactly the same length, see Figure 2.3. And we expect the non protein coding subsequences, e.g. introns and intergenic regions, to be more diverged than the exons. To be able to align two subsequences of the same class, but of different length, in addition to the ***match*** state we need two corresponding ***emit*** states which read non-matching letters from only one sequence at a time.
Even though the pair HMM can deal with the prevailing configurations which arise from the generic alignment of pairs of homologous gene structures, some configurations cannot be modeled with the pair HMM and would require the introduction of extra transitions or states into the pair HMM. To name just two examples which the pair HMM cannot model: (1) a pair of homologous genes in which one exon in one gene exhibits no homology to any exon of the other gene, (2) a pair of homologous genes in which the pairs of homologous exons do not appear in collinearity. In the default implementation of the model, only the pair of input DNA sequences, but not the pair of their corresponding reversecomplemented sequences, is analysed. The analysis of the reversecomplemented sequence pair requires a separate run. Simultaneous search for genes in both orientations could be obtained by essentially doubling the number of states in the pair HMM.

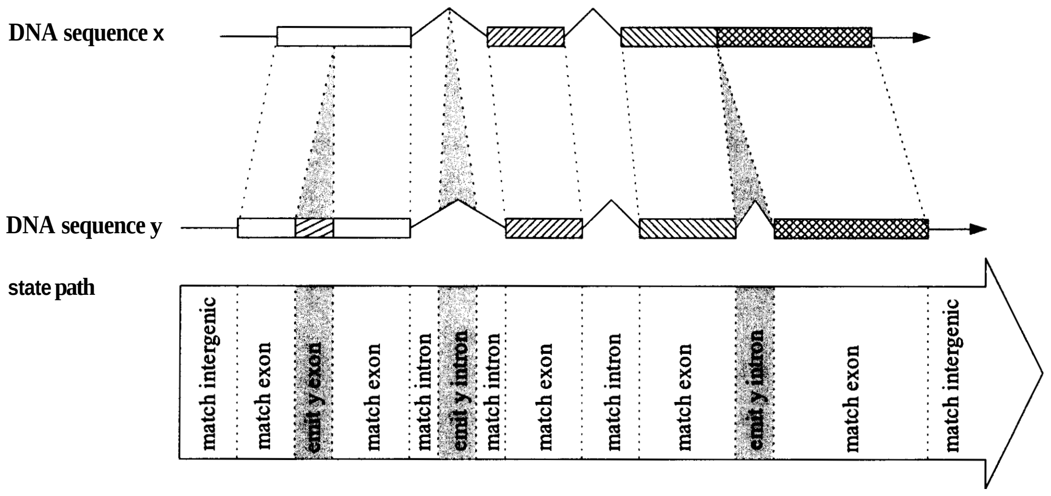


Figure 2.3: Sample alignment of two annotated DNA sequences with a possible state path. Boxes represent exons, kinked lines introns and straight lines intergenic sequences. Similar exons have the same hatching.

We now explain the different sets of states:

***begin* and *end* states** By definition, each state path begins in the ***begin*** state and ends in the ***end*** state. Both states are silent, i.e. they do not read any letters, and are used exactly once in every state path.

As we do not want to make assumptions on the annotations with which the two sequences start or end, the ***begin*** state is connected to every other state except for the ***end*** state. Likewise, the ***end*** state can be reached by every other state except for the ***begin*** state. The pair HMM can thus not only predict single complete genes, but also partial genes, no genes, multiple genes and other configurations of gene structures.

***START START* and STOP *STOP* states** We want to align pairs of genes and thus have a one-to-one correspondence between the start and the stop codons of the genes in the two DNA sequences. The start codons of the pair of initial exons can be aligned using the ***START START*** state and the stop codons of the terminal exons can be aligned using the STOP***STOP*** state.

All potential start codons, i.e. all ATG triplets, are scored using a weight matrix model of 21 base pairs width that starts 9 base pairs 5' to the potential start codon.

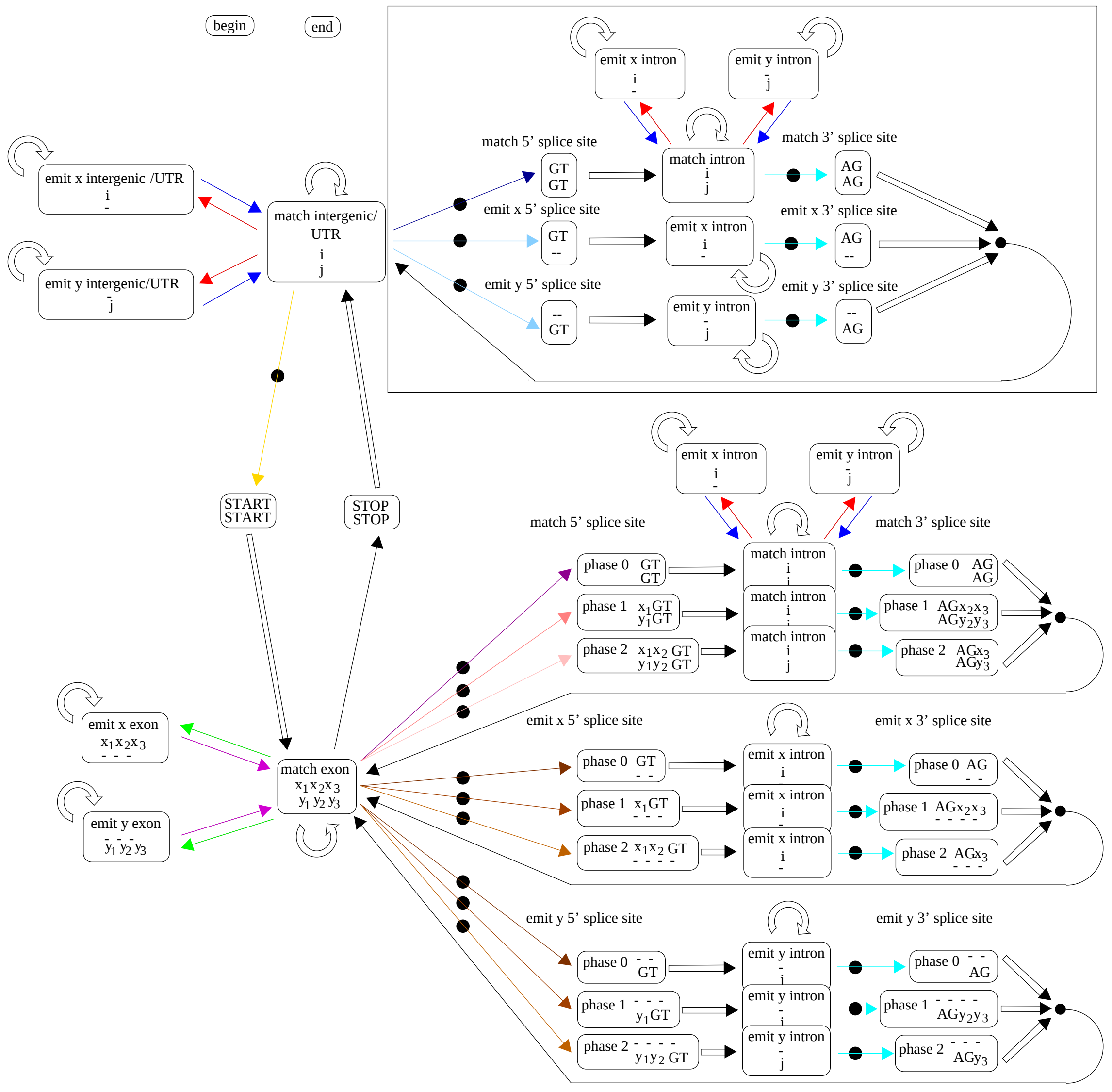


Figure 2.4: States and transitions of the pair HMM underlying DOUBLESCAN and PROJECTOR. States are shown as boxes with rounded corners, transitions as arrows. The *begin* state is connected to every state except for itself and the *end* state. Likewise, there are transitions to the *end* state from every state except for the *begin* state and itself. The arrows corresponding to these transitions are not shown for clarity. Each open arrow corresponds to a transition probability which is defined by the constraint that the probabilities of the transitions emerging from every state have to add up to one. Coloured arrows of the same colour correspond to transitions of the same probability. Arrows marked by a black dot are special transitions, see Section 2.3. The *match intron* states for phase 1 and 2 are connected to the corresponding *emit x* and *emit y* intron states as is the *match intron* state for phase 0. These states have been omitted for clarity (see Figure B.3 in Appendix B). The large box at the top right contains the states which model introns within untranslated regions (UTR-splicing).

**Exons** We expect evolutionarily related genes to encode similar amino-acid sequences which at **DNA** level correspond to similar sequences of codons. These codons *can* be aligned using the *match* exon state which reads one codon from each of the two DNA sequences at a time. The emission probability of the match exon state for aligning two codons which encode the same amino-acid is high compared to that for aligning two codons which encode chemically dissimilar amino-acids. The wobble position, the last (most **3')** position in a codon, has thus less importance in defining the level of similarity between two codons than the first (most **5')** codon position.
Closely related genes which encode similar proteins need not have the same number **of** amino-acids and thus need not correspond to the same number of codons at **DNA** level. This motivates the definition of the emit x exon and the emit y exon states which read a codon from only one **DNA** sequence at a time.
Closely related genes which encode similar proteins may not only have a different number of amino-acids, but these amino-acids may also be encoded on a different number of exons. These pairs of genes which are related by events of exon-fusion or exon-splitting *can* be aligned using the sets of emit x and emit y states of splice site and intron states.
Note that all match and emit exon states can read in-frame **ATG** codons encoding methionine, but that their emission probability for reading any of the three stop codons in frame is zero.

**Splice sites and introns within translated regions** Introns within protein coding regions can come in three different phases depending on where they are inserted into the codons. **As** we want to be able to align genes which are related by events of exon-fusion or exon-splitting, we have to take into account introns which are present in only one of the two genes. These introns can be modelled using the emit x or emit y sets of splice site states and intron states.
In the default implementation of the model, all splice sites are assumed to obey the **GT-AG** rule, stating that an intron should start with a **GT** at the 5' side and end with an **AG** at the **3'** side. This rule accounts for **99 %** of introns in the set of known mammalian **DNA** sequences **[BSSOO]. All** potential splice sites of the input **DNA** sequences are scored by a splice site prediction program [LD01] similar to that used in [BK97].

**Splice sites and introns within untranslated regions (UTR-splicing)** **A** special feature of our model is that it allows for introns within the untranslated regions of genes using a

set of states similar to those for introns within translated regions. The states for UTR-splicing are shown within the box in Figure **2.4.** The main reason for introducing introns within the untranslated regions is the observation that the model without them has difficulties to detect start codons properly. Some start codons were missing in the predictions and were hidden within internal exons. As there are true splice sites also to be found within the untranslated regions and as all potential splice sites – also those within the untranslated regions – are scored by the splice site predictor, the model without UTR-splicing had no means of selectively ignoring the high scoring splice sites within the untranslated regions and of taking only those within the translated regions into consideration. The addition of the UTR-splicing states handles this better and helps to detect both start and stop codons.
Unlike introns within translated regions, introns within untranslated regions do not have a phase. As for introns within protein coding regions, all splice sites are by default assumed to obey the GT-AG rule and are scored by the splice site prediction program.

**Intergenic/UTR states** We put the least constraints on the intergenic/UTR subsequences even though we know that they can have a rich functional structure, comprising for example promoters and sequences which bind molecules which determine the three-dimensional structure of the DNA sequence. We do not attempt to model these features with this pair HMM, as the ability to predict them is poor. If these functional elements are conserved, they will be predicted as conserved intergenic/UTR subsequences and they can be further investigated.

## 2.3 Parameters of the model and their determination

The parameters of the pair HMM can be subdivided into transition and emission probabilities. While the transition probabilities are the same for the different pairs of genomes investigated in this dissertation (with one exception, see Table D.1 in Appendix D), the emission probabilities are adapted to the pair of related organisms that is studied and are derived from a *training set* of known genes.

Transition probabilities and splice site scores Non-zero transition probabilities are represented by arrows in Figure 2.4. The *begin* state is connected to every other state except the *end* state. Likewise, the *end* state can be reached by all other states except the *begin* state. The corresponding arrows have been omitted for clarity. Every open arrow corresponds to a transition probability whose value is defined by the constraint that the probabilities of the transitions which emerge from every state must add up to one.

As we assume that there is no systematic bias in the number of exons or the length of exon, intron and intergenic sequences between the two organisms from which the two DNA sequences derive, the transition probabilities of the emit x states are the same as those of the corresponding emit y states.

As opposed to the emission probabilities which are derived from the training set, there is not straightforward way to derive the values of the transition probabilities. First, the training sets are generally too small to reliably estimate the transition probabilities, e.g. the probability for the transition from the match intergenic to the ***START START*** state, from the corresponding frequencies in the training set. Second, the probabilities for transitions between match and emit states can only be derived from a training set of pre-aligned sequences which is not available. We therefore derive only the relative probabilities for introns of phase zero, one and two from their respective frequencies within the training set. All other transition probabilities are set to estimated values which are then tuned by hand during the optimisation of the performance with the training set. All transitions emerging from the begin state have the same probability as well as those leading to the end state.

The values of the transition probabilities are generally fixed. This means that the probability of each transition is independent of the positions within the two sequences at which the transition in used in the pair HMM. However, there are sequence signals whose strength varies along the sequence and which cannot be adequately described by the emission probabilities of the pair HMM's states. One example for this are splice sites. In the pair HMM, they are modelled by states which recognise the consensus (GT in the case of a 5' splice site and AG for a 3' splice site). The emission probabilities of these states cannot take into account the splice site signal as it is wider than the window of letters that the splice site states read. In order to incorporate the splice site signal into the pair HMM, every potential splice site in the two DNA sequences is scored by a splice site predictor program [LD01] similar to that in [BK97]. These scores are transformed into posterior probabilities which modify the nominal transition probabilities leading into the splice site states. A potential splice site with a high score leaves the nominal transition probability almost unchanged, whereas a low score decreases it. The probabilities of all the other transitions emerging from the same state are rescaled accordingly by a common factor so that the sum of all transition probabilities emerging from that state

always remains one. This is an extension of the pair HMMs described in Section 1.5.1. We call transitions ***special*** if their value is affected by position dependent sequence signals. The implementation of special transitions is explained in detail in Chapter 6.
Similarly to splice sites, all potential start codons are scored using a weight matrix model of 21 base pairs width that starts 9 base pairs 5' to the potential start codon.

**Emission probabilities of the *match exon* state** The emission probabilities of the ***match exon*** state are derived from a training set, see Section A.1 in Appendix **A** and Section C.1 in Appendix C. The main idea is to base the emission probabilities of all states except for those of the ***START START*** and the *STOP STOP* state on the emission probabilities of the ***match exon*** state,

$$p(x_1, x_2, x_3, y_1, y_2, y_3),$$

where $(x_1, x_2, x_3)$ denotes the letters read from sequence $X$ an $(y_1, y_2, y_3)$ denotes those read from sequence $\boldsymbol{Y}$**.** These probabilities are derived from pairs of orthologous genes that have identical coding length, the coding length of a gene being defined **as** the sum of lengths of its exons. For every such pair of genes, the exons **of** each gene are concatenated into a continuous sequence of codons finishing in a stop codon. From each such pair of codon sequences, the aligned terminal stop codons are used to derive the emission probabilities of the *STOP* STOP state and the rest of the aligned codon pairs is used to derive the emission probabilities of the ***match exon*** state. One of our training sets, see Section A.1 in Appendix **A,** is not large enough to avoid zero counts for some legal codon pairs. We could thus not simply use the maximum likelihood method to estimate the emission probabilities of the ***match exon*** state. In order to be able to apply the same estimation method to training sets of variable size, we refrained from adding simple pseudo-counts and instead chose a Dirichlet distribution with the following posterior mean estimator $(i := (x_1, x_2, x_3),\ j := (y_1, y_2, y_3))$:

$$p(i,j) = \frac{n(i,j) + A \cdot q(i,j)}{\sum_{i,j} n(i,j) + A}$$

where $n(i,j)$ is the number of aligned, unordered codon pairs with codon $i$ and codon $j$ in the training set, $A$ is the number of unordered non-stop codon pairs, i.e. $61 \cdot 60/2 + 61 = 1891$, and $q(i,j) = (c(i) + c(j)) / \sum_{i,j}(c(i) + c(j))$, where $c(i)$ is the number of codons $i$ in the training set of aligned exons. This formula introduces a symmetry with respect to the two sequences $X$

and *Y* as $p(i,j) = p(j,i)$. The advantage of this method is that it scales well from rather small training sets for which the probability of rare events is $p(i,j) \approx q(i,j)$ to large sets for which this probability converges to the maximum likelihood result, i.e. $p(i,j) \approx n(i,j)/\sum_{i,j} n(i,j)$.
We have investigated whether the aligned concatenated exons of the training set give sensible emission probabilities for the *match exon* state by comparing the frequencies of pairs of amino-acids, see Figure B.1 in Appendix B and Figure D.1 in Appendix D. As one would expect, each codon is found to be preferentially aligned to codons which encode the same amino-acid. The DNA sequences of the training set are therefore evolutionarily well conserved.
Another question which we address is whether the codons in the pairs which encode the same amino-acid are likely to be the same or whether there exists a bias. The results of this study are shown in Figure B.2 in Appendix B and in Figure D.2 in Appendix D. Each diagram corresponds to one amino-acid and shows all possible *unordered* codon pairs and the frequency with which they are observed. It is apparent from the mouse human training set that the results for most amino-acids do not follow any simple rule: the frequency of pairs where the two codons are the same need not be higher than that of pairs where the two codons differ, as shown for example for isoleucine (I). However, the results for serine (S), which is encoded by six different codons which differ in any of the three codon positions, follow some basic rules: codon pairs where the two codons are the same, dominate, {TCC, TCC} $=$ 0.1980, {AGC, AGC} $=$ 0.1962, {TCT, TCT} $=$ 0.1058, {AGT, AGT} $=$ 0.0939, {TCA, TCA} $=$ 0.0870, and {TCG, TCG} $=$ 0.0239. Differences of the codons in the wobble position are tolerable (with observed frequencies ranging from {TCA, TCT} $=$ 0.0205 to {TCC, TCT} $=$ 0.0819) and pairings between codons which differ in their first codon position almost never occur (the frequencies of these events being lower than 0.0051).
Concerning the pairs of stop codons, codon pair {TGA, TGA} dominates, followed by the pairs {TAA, TAA} and {TAG, TAG} of approximately the same frequency.
It would be interesting to investigate if the codon pair frequencies shown in Figure B.2 in Appendix B and in Figure D.2 in Appendix D correspond to a deviation from the codon pair frequencies that would be obtained by assuming that the frequency of each codon pair is equal to the product of the two individual codon frequencies.

**Emission probabilities of the other states** The emission probabilities of the other states except for those of the *START START* and *STOP STOP* state are calculated using the above match exon emission probabilities $p(x_1, x_2, x_3, y_1, y_2, y_3)$. This is done by marginalising over one or more codon positions. The emission probabilities for the two emit exon states are given by:

$$p_{\text{emit x exon}}(x_1, x_2, x_3) = \sum_{(y_1, y_2, y_3)} p(x_1, x_2, x_3, y_1, y_2, y_3)$$

and the analogous expression for the emit y exon state.

The match intergenic and match intron states have the same emission probabilities. They are given by:

$$\begin{aligned} p_{\text{match intron}}(i, j) = \frac{1}{3}\Bigg( & \sum_{(x_2, x_3, y_2, y_3)} p(i, x_2, x_3, j, y_2, y_3) + \\ & \sum_{(x_1, x_3, y_1, y_3)} p(x_1, i, x_3, y_1, j, y_3) + \\ & \sum_{(x_1, x_2, y_1, y_2)} p(x_1, x_2, i, y_1, y_2, j)\Bigg) \end{aligned}$$

The *emit* intergenic states and the corresponding emit intron states also have the same emission probabilities. They are obtained by marginalising over one of the two indices of the match intron emission probabilities.

The emission probabilities for the three match 5' splice site states are obtained by:

$$\begin{aligned} p_{\text{phase 0}}(x_1, x_2, y_1, y_2) &= \delta_{x_1 G}\delta_{x_2 T}\delta_{y_1 G}\delta_{y_2 T} \\ p_{\text{phase 1}}(x_1, x_2, x_3, y_1, y_2, y_3) &= \delta_{x_2 G}\delta_{x_3 T}\delta_{y_2 G}\delta_{y_3 T} \sum_{(x_2', x_3', y_2', y_3')} p(x_1, x_2', x_3', y_1, y_2', y_3') \\ p_{\text{phase 2}}(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4) &= \delta_{x_3 G}\delta_{x_4 T}\delta_{y_3 G}\delta_{y_4 T} \sum_{(x_3', y_3')} p(x_1, x_2, x_3', y_1, y_2, y_3') \end{aligned}$$

In a similar way, the emission probabilities for the three match 3' splice site states axe determined using:

$$\begin{aligned}
p_{\text{phase 0}}(x_1,x_2,y_1,y_2) &= \delta_{x_1A}\delta_{x_2G}\delta_{y_1A}\delta_{y_2G}\\
p_{\text{phase 1}}(x_1,x_2,x_3,x_4,y_1,y_2,y_3,y_4) &= \delta_{x_1A}\delta_{x_2G}\delta_{y_1A}\delta_{y_2G}\sum_{(x_1',y_1')} p(x_1',x_2,x_3,y_1',y_2,y_3)\\
p_{\text{phase 2}}(x_1,x_2,x_3,y_1,y_2,y_3) &= \delta_{x_1A}\delta_{x_2G}\delta_{y_1A}\delta_{y_2G}\sum_{(x_1',x_2',y_1',y_2')} p(x_1',x_2',x_3,y_1',y_2',y_3)
\end{aligned}$$

The emission probabilities of the match 5′ splice site and match *3'* splice site states can then be used to derive the emission probabilities of the emit *5'* splice site and emit **3'** splice site states by summing over the relevant indices in the same way the emit *exon* state emission probabilities are obtained from the match exon state emission probabilities.
The emission probabilities of the *START START* state are simply given by:

$$p_{\text{START START}}(x_1,x_2,x_3,y_1,y_2,y_3) = \delta_{x_1A}\delta_{x_2T}\delta_{x_3G}\delta_{y_1A}\delta_{y_2T}\delta_{y_3G}$$

The emission probabilities of the *STOP STOP* state are determined from the training set in the same way the emission probabilities are determined for the match *exon* state by using a Dirichlet distribution. The observed frequencies for all possible pairs of stop codons in the training set are shown in Figure B.2 in Appendix B and Figure D.2 in Appendix D.
During the training of **DOUBLESCAN,** we implemented a fifth order Markov model in order to use hexamer frequencies which are frequently used to distinguish protein-coding from non-protein coding DNA [BK97, GF95], but abandoned the use of hexamer frequencies as they did not improve the performance.

## 2.4 The Stepping Stone algorithm

For a given pair HMM with $N$ states and T transitions and two input sequences $X$ and $Y$ of length $L_x$ and $L_y$, the optimal state path can be found with a memory requirement which scales only linearly with the length of the input sequence, $O(N\cdot min\{L_x,L_y\})$, and a time requirement which scales quadratically with the length of the input sequence, $O(T\cdot L_x\cdot L_y)$, using the Hirschberg algorithm, see Section 1.5.2. The memory requirement of the Viterbi algorithm can thus be linearised, but the time requirement is still quadratic and imposes a serious constraint on the analysis of long DNA sequences.

Our aim in introducing the Stepping Stone algorithm is to invent a method by which a nearly optimal state path can be found with time and memory requirements which scale linearly with the sequence length. The main idea is to first employ a simple local alignment program to search for subsequences of strong similarity between the two input DNA sequences and then to use these matches as guidelines to search for the optimal state path only in a sub-space of the Viterbi matrix.

In the first step, the local alignment program BLASTN [AGM+90] is used to search the two input DNA sequences for regions of high similarity. The set of matches returned by BLASTN is then turned into a set of mutually compatible constraints in the following way. We select the highest scoring match and define its middle point as its reference point. We then take this middle point to find the next highest scoring match whose middle point is compatible with it and repeat this scheme until no more compatible middle points can be added. A new middle point is compatible with an already selected set of middle points if their pairs of $(x, y)$ coordinates can be simultaneously ordered by their $x$ and y coordinates. Although we then have a set of $(x, y)$ constraints at which the two DNA sequences match, we do not know whether these matches correspond to exons, introns or intergenic regions as BLASTN does not assign any functional annotation to the matches. We allow for this uncertainty by allowing all states at the $(x, y)$ midpoint. The overlap between two adjacent sub-matrices is thus a line whose projection onto the $(X, Y)$ plane is a point at $(x, y)$. In particular, BLASTN does not know about codons and phases. It may thus happen that a match corresponds to aligned exons whose codons are out of phase. To allow DOUBLESCAN to correct for this phase difference, we increase the overlap at $(x, y)$ to a small 15 base pairs by 15 base pairs region around $(x, y)$. Two adjacent sub-matrices thus overlap in a small volume of 15 base pairs by 15 base pairs by $N$. The set of concatenated sub-matrices defines a continuous sub-space of the Viterbi matrix, see the hatched area in Figure 2.5, which is searched for the highest scoring state path in the following step.

In the next step, the optimal state path in the thus restricted sub-space of the Viterbi matrix is retrieved by first calculating the elements in the sub-space and by then applying a traceback procedure. The calculation is started at the lower left sub-matrix, see Figure **2.5,** using the Viterbi algorithm. During the calculation we keep only the values in a narrow strip like volume in memory with which the calculation can be continued. Once this sub-matrix has been calculated, only the values in the small volume where this sub-matrix overlaps the next one are used to initialise the calculation of the next sub-matrix. This process is iterated until the calculation of the upper right sub-matrix is finished and the ends of the sequences are reached. We then know the score of the highest scoring path that lies within the sub-space of the Viterbi matrix. The corresponding state path is retrieved by proceeding from the upper right to the lower left sub-matrix, recalculating each sub-matrix with now partially known boundaries either using the Viterbi algorithm, if there is sufficient memory, or the Hirschberg algorithm.

The benefits of the Stepping Stone algorithm are that the time and memory requirements are reduced with respect to the Viterbi algorithm. If we assume that there is a minimum number of **BLASTN** matches per sequence length, both memory and time requirements depend essentially linearly on the sequence length, i.e. are of order $U(N\cdot\sqrt{L_x^2+L_y^2})$ and $\mathcal{O}(T\cdot\sqrt{L_x^2+L_y^2})$, respectively, as the number of rectangles is expected to increase asymptotically as $\sqrt{L_x^2+L_y^2}$. The disadvantage of the Stepping Stone algorithm is that the state path which is optimal within the sub-space of the Viterbi matrix is not necessarily identical to the optimal state which would have been found by calculating the whole Viterbi matrix. In Section 3.4 we show for a test set of mouse and human DNA sequences that the Stepping Stone algorithm finds the true optimal state path in 81 % of all cases and that 97 % of the predicted genes are the same as those predicted by the Hirschberg algorithm. The Stepping Stone algorithm therefore provides a very good practical solution.

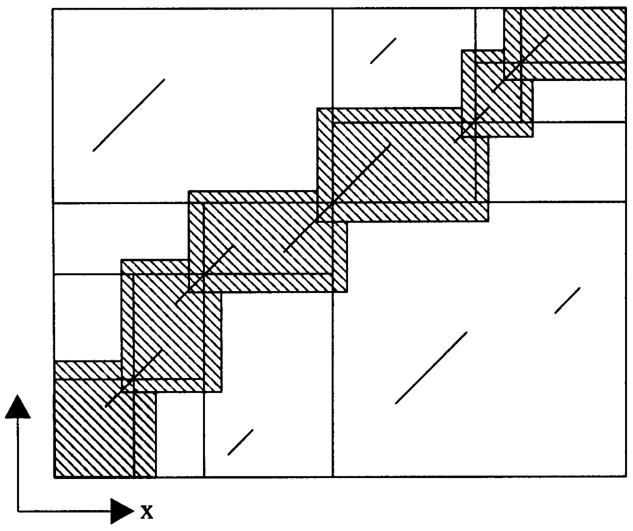


Figure 2.5: The Stepping Stone algorithm: Shown is the projection of the Viterbi matrix onto the (X,*Y*) plane spanned by the two sequences X and *Y*. Diagonals represent similar subsequences retrieved by BLASTN, hatched areas correspond to sub-matrices which are calculated using the Viterbi algorithm or the Hirschberg algorithm. Note that there is no restriction imposed on the third dimension, the state dimension.

# Chapter 3

# *Ab* initio **prediction of mouse and human genes with** DOUBLESCAN

## 3.1 Introduction and motivation

The architecture of the pair **HMM** underlying DOUBLESCAN **as** described in Chapter **2** is suited for the comparative prediction of pairs of genes in any pair of related eukaryotic organisms. Only by setting up the pair **HMM's** transition and emission probabilities with a training set of known pairs of genes, is DOUBLESCAN specialised for a certain purpose.

In this chapter, we show that DOUBLESCAN can be used to analyse pairs of mouse and human DNA sequences. For this, DOUBLESCAN'S emission probabilities were derived and its transition probabilities optimised with a training set of known mouse and human gene pairs [JBD99], *see* Section **2.3** for the derivation of parameters and Section A.l in Appendix A for a description of the training set. Once the parameters of the pair **HMM** have been set up, DOUBLESCAN is applied to a test set of 80 pairs of known orthologous mouse and human genes [**Pac99**], *see* Section A.2 in Appendix A. Note that the only input information to DOUBLESCAN are the letters of the two DNA sequences. In particular, the sequences are not masked for repeats.

This chapter evaluates the performance of DOUBLESCAN for finding genes and compares it to the performance of GENSCAN [BK97], a non-comparative *ab initio* gene prediction method. We then briefly discuss the alignments of the DNA sequences generated during the gene prediction and conclude the chapter **by** showing that the Stepping Stone algorithm provides a good solution for the analysis of long DNA sequences by comparing its predicted state paths and gene predictions to the optimal ones derived with DOUBLESCAN using the Hirschberg algorithm.

## 3.2 Results

The results of the comparison are shown in Table 3.1. In the following, the term prediction refers to the results retrieved by DOUBLESCAN or GENSCAN, whereas the term annotation refers to the gene structures in the data base from which the genes of the training and test sets have been derived.

GENSCAN is a non-comparative ab initio gene prediction method which employs an explicit state duration HMM. It is capable of predicting partial, complete and multiple genes. Its HMM contains separate states for the exon of a single exon gene and for initial and terminal exons, as well as states for promoter, 5' untranslated region, 3' untranslated region and the poly-A signal. It uses different parameter sets according to the GC contents of the input DNA sequence.

The first thing to note is that the pair HMM with states for UTR-splicing improves the overall performance of DOUBLESCAN, especially the sensitivity and specificity for stop codons, the specificity for start codons and exons and the sensitivity and specificity for genes as well as the rate of wrong genes. 16 % of the overlapping genes are turned into correctly predicted genes and 42 % of the wrong genes are completely removed when including UTR-splicing into the model, while only 5 % of the correctly predicted genes are turned into just overlapping genes. Instead of a correctly predicted start codon, these overlapping genes have a splice site in close vicinity 5' to the annotated start codon which is not predicted or their initial exons are completely missing in the prediction. Given its superior performance, DOUBLESCAN including the states for UTR-splicing will be taken as the reference model for DOUBLESCAN.

DOUBLESCAN including UTR-splicing still has a 14 % rate of wrong genes corresponding to 30 genes which are predicted in addition to those that overlap the annotated gene in each DNA sequence. 53 % of the wrong genes are short (less than 106 base pairs length) complete single exon genes, 13 % are complete two exon genes with a long intron (more than 656 base pairs length) and short coding length (less than 52 base pairs length), 7 % are complete two exon genes with a short intron (less than 17 base pairs length) and short coding length (less than 49 base pairs length) and the remaining 27 % are partial genes.

If we post-process DOUBLESCAN'S results as described in Section A.3 in Appendix A, all of

| | DOUBLESCAN without UTR-splicing | DOUBLESCAN | DOUBLESCAN including post-processing | GENSCAN |
|---|---|---|---|---|
| **Gene** | | | | |
| Sensitivity | 0.51 | 0.57 | 0.57 | 0.47 |
| Specificity | 0.35 | 0.43 | 0.50 | 0.46 |
| Genes overlapping | 0.42 | 0.44 | 0.46 | 0.53 |
| Genes missing | 0 | 0 | 0.01 | 0 |
| Genes wrong | 0.23 | 0.14 | 0.04 | 0.01 |
| **Start Codon** | | | | |
| Sensitivity | 0.77 | 0.78 | 0.75 | 0.73 |
| Specificity | 0.64 | 0.67 | 0.78 | 0.91 |
| **Stop Codon** | | | | |
| Sensitivity | 0.86 | 0.91 | 0.89 | 0.88 |
| Specificity | 0.70 | 0.74 | 0.86 | 0.97 |
| **Exon** | | | | |
| Feature Level | | | | |
| Sensitivity | 0.79 | 0.81 | 0.80 | 0.84 |
| Specificity | 0.68 | 0.74 | 0.79 | 0.82 |
| Exons overlapping | 0.16 | 0.15 | 0.15 | 0.12 |
| Exons missing | 0.03 | 0.03 | 0.05 | 0.03 |
| Exons wrong | 0.16 | 0.10 | 0.06 | 0.06 |
| Nucleotide Level | | | | |
| Sensitivity | 0.97 | 0.97 | 0.96 | 0.98 |
| Specificity | 0.97 | 0.98 | 0.99 | 0.94 |

Table 3.1: Performance figures for **DOUBLESCAN** without UTR-splicing, **DOUBLESCAN**, DOUBLESCAN including post-processing and **GENSCAN** on the test set. The predictions by DOUBLESCAN were generated using the Stepping Stone algorithm. Sensitivity is defined as the fraction of annotated features which are correctly predicted. Specificity is defined as the fraction of predicted features which match an annotated feature. For start and stop codons, sensitivity and specificity are shown at feature level, i.e. for entire codons. At feature level, sensitivity and specificity as well as the fraction of annotated exons which overlap a predicted exon (Exons overlapping), the fraction of annotated exons which do not overlap any predicted exon (Exons missing) and the fraction of predicted exons which do not overlap any annotated exon (Exons wrong) are given. At gene level, sensitivity and specificity are detailed as well as the fraction of annotated genes which overlap a predicted gene (Genes overlapping), the fraction of annotated genes which do not overlap any predicted gene (Genes missing) and the fraction of predicted genes which do not overlap any annotated gene (Genes wrong).

the wrong complete genes, corresponding to 73 % of the wrong genes, are removed. This post-processing steps also removes ten (10.7 %) of the overlapping genes. Six of them are complete short single exon genes which overlap an exon of the annotated gene. Two other overlapping genes which are removed in the post-processing step are complete two exon genes with a small coding length (less than 94 base pairs length) which have only a small overlap with the annotated genes. However, the post-processing step also removes two complete, overlapping multi-exon genes which overlap the annotated genes in most of their exons, but which each have one short intron (of 39 base pairs and 45 base pairs length, respectively) due to a mis-predicted exon. Overall, the post-processing step improves the performance considerably. It keeps the sensitivity at gene level unchanged while at the same time improving the specificity by 7 % and lowering the rate of wrong genes by 10 %. For start codons, it slightly lowers the sensitivity by 3 % while at the same time raising the specificity by 11 %. The same tendency is shown for stop codons where the sensitivity is lowered by 2 % while the specificity improves by 12 %. For exons, the performance at nucleotide level remains almost unchanged. At exon level, the sensitivity is lowered by 1 % while the specificity is increased by 5 %. Given the overall positive effect of the post-processing step, we discuss in the following parts of this chapter the results of DOUBLESCAN after post-processing unless otherwise stated.

Both for DOUBLESCAN and GENSCAN, the performance for stop codons is significantly higher than for start codons, the main reason being that in-frame start codons can be found both at the translation start as well as in frame within exons, while in-frame stop codons can only be found at the translation end. The sensitivity of DOUBLESCAN for start codons is 2 % higher than that of GENSCAN, but its specificity is 13 % lower than that of GENSCAN. DOUBLESCAN'S sensitivity for stop codons is slightly higher than that of GENSCAN, while its specificity is 11 % lower than that of GENSCAN. Unlike DOUBLESCAN, GENSCAN has dedicated states for a promoter and the 5' untranslated region which model the region 5' of the translation start. These extra states implement detailed knowledge about the upstream region of some genes and can therefore help to position the start codon correctly. In addition, GENSCAN is biased towards starting and finishing the predicted annotation within the intergenic state. Within GENSCAN, also the region 3' of the translation end has dedicated states which model the 3' untranslated region and a poly-A signal. However, without this extra information, DOUBLESCAN has a high sensitivity for both start and stop codons using only similarity information between the two DNA sequences.

DOUBLESCAN'S sensitivity for exons at nucleotide level is high, the sensitivity being 2 % lower and the specificity 5 % higher than those of GENSCAN. At exon level its sensitivity is 4 % and its specificity 3 % lower than GENSCAN. The difference in performance for exons between the nucleotide and exon level can be explained by cases in which two or more predicted genes overlap one annotated gene such that the overlap between the annotated and the predicted exons is large, but not perfect.

At gene level, DOUBLESCAN has a significantly higher sensitivity (10 %) and also higher specificity (4 %) than GENSCAN. Three of the gene pairs can not be predicted correctly by DOUBLESCAN as the configuration of annotated genes can not be modelled by the underlying pair HMM. One of the three gene gene pairs can not be modelled as the initial exons consist only of a start codon. The other two pairs of genes lie in pairs of sequences for which one sequence starts with intergenic subsequence 5' to the start codon and the other sequence starts directly with the start codon. Removing the corresponding three sequence pairs would improve the performance by up to 3 %. The 1 % rate of missing genes for DOUBLESCAN corresponds to one overlapping gene which is removed in the post-processing step.

In order to see whether or not DOUBLESCAN and GENSCAN preferentially detect different types of genes, we have compared the genes which were correctly predicted by one of the two methods to those predicted by the other method. About half (44%) of the genes which were found by DOUBLESCAN were incorrectly predicted by GENSCAN. Conversely, 32 % of the genes found by GENSCAN were not correctly predicted by DOUBLESCAN. By far the most common reason why a gene is correctly predicted by one method and incorrectly predicted by the other one is that the start codon is not found correctly or not found at all (accounting for 55 % of the genes found by DOUBLESCAN and not correctly predicted by GENSCAN, and for 58 % of the genes found by GENSCAN and not correctly predicted by DOUBLESCAN). The next common causes are incorrect splicing (accounting for 30 % of the genes found by DOUBLESCAN and not correctly predicted by GENSCAN, and for 21 % of the genes found by GENSCAN and not correctly predicted by DOUBLESCAN) and the wrong or missing prediction of the stop codon (accounting for 23 % of the genes found by DOUBLESCAN and not correctly predicted by GENSCAN, and for 25 % of the genes found by GENSCAN and not correctly predicted by DOUBLESCAN). Interestingly, GENSCAN tends to miss out whole terminal exons whereas DOUBLESCAN only gets the 3' end of the terminal exon wrong by introducing a 5' splice site in close vicinity 5' to the annotated stop codon. Overall, DOUBLESCAN and

GENSCAN complement each other, but we could not identify a pattern as to which genes tend to be correctly predicted by which method.

It is known that the density of genes as well as some of their features, e.g. intron length, depend on the GC contents of the DNA sequence [DMG95, Con01]. To test whether the performance of the methods depends on the GC contents of the input DNA sequences, we subdivided the test set into the following four subsets according to the GC contents intervals defined in [Ber89]. As the GC contents of the two DNA sequences of each pair are well correlated, the DNA sequences were sorted by GC contents in pairs. The four intervals are gc1 $= [0, 0.43)$, comprising four sequence pairs, gc2 $= [0.43, 0.51)$, comprising 22 sequence pairs, gc3 $= [0.51, 0.57)$, comprising 26 sequence pairs, and gc4 $= [0.57, 1]$, comprising 28 sequence pairs. Considering the DOUBLESCAN results without the post-processing step, the sensitivity and specificity for start codons, stop codons, exons and genes show no dependency on the GC contents of the DNA sequences and are the same within statistical errors. The same independence of GC contents was found for GENSCAN. However, in GENSCAN this independence is explicitly established by choosing the model's parameters according to the GC contents of the input DNA sequence, whereas DOUBLESCAN's performance is independent of the GC contents without using GC dependent parameters.

## 3.3 Prediction of conserved subsequences

DOUBLESCAN without the post-processing step retrieves 69 % of the intergenic subsequences, 48 % of the intron subsequences and 99 % of the exon subsequences as conserved subsequences. The level of conservation in the intergenic subsequences is higher than one would expect for long intergenic subsequences, but can be explained by the fact that the intergenic subsequences of the test set are close to the translation or transcription start and end of the genes where a higher density of conserved subsequences is expected [JBD99].

## 3.4 Validation of the Stepping Stone algorithm

The Stepping Stone algorithm has been developed in order to accelerate the prediction process as both its time and memory requirement scale essentially linearly with the length of the input sequence. Since it is not guaranteed to find an optimal state path, we compared both the state paths and annotations retrieved by DOUBLESCAN using the Stepping Stone algorithm to those retrieved by **Doublescan** using the Hirschberg algorithm on the test set. For these purposes we consider the **Doublescan** results without post-processing as they correspond to the state paths which are to be compared. For 81 % of the DNA sequence pairs, the Stepping Stone algorithm finds the optimal state path (this state path need not be the same as the optimal state path retrieved by the Hirschberg algorithm as there are generally several optimally scoring state paths). Comparing the predicted annotations, 97 % of the predicted genes are the same for both algorithms. The agreement for start codons is 100 % and 98 % for stop codons. At nucleotide level, the agreement for exons is 100 % and 99.8 %, respectively, i.e. close to perfect.

Compared to the annotation, the performance of the Hirschberg algorithm is the same as that of the Stepping Stone algorithm except for a 1 % improvement of the exon sensitivity at exon level and the corresponding 1 % decrease of the rate of overlapping exons.

The average length of the sequences in the test set is around 3300 base pairs and there is on average a **Blastn** match every 380 base pairs. If we constrain the Stepping Stone algorithm and Hirschberg algorithm to use the same maximum amount of memory, the prediction process using the Stepping Stone algorithm is on average four times faster than using the Hirschberg algorithm. To give an example, the analysis of one pair of DNA sequences of 9604 base pairs and 10373 base pairs length, respectively, took about 126340 CPU seconds and about 400 MB memory on an Alpha processor with the Hirschberg algorithm, while the analysis with the Stepping Stone algorithm took about 13313 CPU seconds using the same amount of memory.

We have used **Doublescan** with the Stepping Stone algorithm on pairs of sequences of more than $10^5$ base pairs length. As the maximum memory to be used can be set by the user, the memory requirement can be traded for the time requirement and vice versa.

Assuming that the density of **Blastn** matches is independent of the sequence length, the gain in time using the Stepping Stone algorithm increases with the length of the DNA sequences to be analysed.

## 3.5 Summary and discussion

The analysis of a test set of 80 pairs of orthologous mouse and human DNA sequences shows that **Doublescan** performs well at gene level and significantly outperforms **Genscan**, the reference non-comparative *ab initio* method. **Doublescan's** performance at nucleotide level is high, its sensitivity being 2 % lower and its specificity being 5 % higher than **Genscan's**.

At feature level, DOUBLESCAN'S sensitivity for start and stop codons is slightly higher than GENSCAN's, but its specificity is 11 % and 13 %, respectively, lower specificity than GENSCAN'S. Besides the extra states that help GENSCAN recognise the region 5' of the translation start, it also has an inherent bias towards starting and finishing the state path in intergenic regions and is thus biased towards detecting complete genes comprising start and stop codons. As our test set is entirely composed of DNA sequences which each contain one complete gene, we expect this to help GENSCAN. At exon level, DOUBLESCAN'S sensitivity and specificity are 4 % and 3 %, respectively, lower than GENSCAN's. At gene level, DOUBLESCAN outperforms GENSCAN's sensitivity by 10 % and its specificity by 4 %. One gene which is predicted by DOUBLESCAN and which overlaps the annotated gene is removed in the post-processing step which corresponds to a 1 % rate of missing genes. DOUBLESCAN and GENSCAN agree in more than half of their correctly predicted genes. 72 % of all annotated genes are correctly predicted by one or both of the two methods. DOUBLESCAN and GENSCAN thus complement each other. However, we could not find an obvious pattern that would allow us to predict which genes are correctly identified by which method.

It is interesting that the performance of DOUBLESCAN relative to GENSCAN increases progressively when going from fine scale (nucleotide level) to large scale (gene structure). It appears that long range constraints such as the exon-intron structure of genes can be captured well in the comparative model, even though the detailed modelling is simplified compared to GENSCAN.

The performance of DOUBLESCAN and GENSCAN as reported here for a test set of 80 pairs of orthologous mouse and human DNA sequences each comprising one single complete gene does not permit to conclude that the performance on other test sets, especially long DNA sequences comprising multiple genes, will be the same, see for example [GAA+00a] and [WGM00]. To investigate the performance of DOUBLESCAN on multi gene sequences, pairs of long homologous DNA sequences are needed in which the similarities between the two sequences appear in collinearity. This requirement implies that long semi-articifial DNA sequences comprising several single-gene sequences separated by randomly generated intergenic regions (see for example [GAA+00a]) are not likely to constitute an adequate test for comparative gene prediction methods such as DOUBLESCAN as the level and the patterns of conservation between two homologous intergenic subsequences will not necessarily be similar to those between two randomly generated intergenic subsequences.

Comparing the predictions of the Stepping Stone algorithm and the Hirschberg algorithm, for 81 % of the sequence pairs is the state path returned by the Stepping Stone algorithm optimal and 97 % of the predicted genes axe identical for the two algorithms. The performance of the Hirschberg algorithm is almost the same as that of the Stepping Stone algorithm, while the gain in time using the Stepping Stone algorithm is significant. This is especially important for the analysis of large genomic sequences for which the Stepping Stone algorithm provides a very efficient practical solution.

# Chapter 4

# Prediction of mouse and human genes with PROJECTOR

## 4.1 Introduction and motivation

In Chapter 3 we have shown that **DOUBLESCAN** can predict related genes given two un-annotated **DNA** sequences as the only input information.
Sometimes, we know more about the pair of input **DNA** sequences than just their sequence of A, C, G and T letters. One typical example is that we know the genes in one of the two **DNA** sequences, but not in the other homologous sequence. We then want to find the genes in this sequence given the known genes in the other sequence, i.e. we want to project the annotation of one **DNA** sequence onto the other **DNA** sequence whose annotation is not known. To name another example, we may have a set of confirmed introns in both sequences and may want to predict genes in the two sequences under the hypothesis that these introns are true.
In our test set [Pac99], see Section **A.2** in Appendix **A,** the orthologous mouse and human genes are very similar not only at protein level, i.e. comparing their sequences of amino-acids, but also at **DNA** level. In **97 %** of the gene pain, the sequences of amino-acids are encoded on the same number of exons. In **42 %** of the gene pain, the sequences of amino-acids are partitioned in the same way into pairs of exons of the same length. For 55 % of the gene pairs, the number of exons is the same, but their lengths are slightly different. The exon-intron structure of related genes is thus very similar concerning the number of exons and their lengths. If we therefore know the gene structure of one input **DNA** sequence, the related gene in the other input **DNA** sequence is likely to have the same or a similar number of exons, and

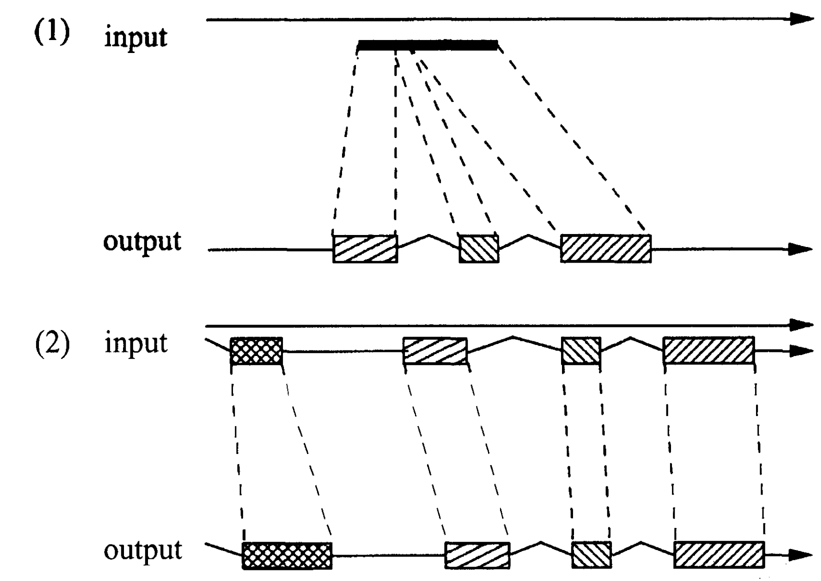


Figure **4.1**: Different types of homology based gene prediction methods: (1) gene prediction based on protein homology (e.g. GENEWISE), (2) gene prediction based on gene homology (e.g. PROJECTOR). Refer to the text for a detailed description of the methods.

the exons will be of the same or a similar length, i.e. the gene structures of the two genes should be very similar. If we knew only the amino-acid sequence of one gene and we would want to find the corresponding related gene in another DNA sequence, we could use programs such as GENEWISE [BD97, BD00] or PROCRUSTES [GMP96], but these methods would a priori not know where and if introns are inserted as they lack information on the gene structure. Implementing constraints into the pair HMM of DOUBLESCAN (see Chapter **2**), we construct PROJECTOR which can make use of the extra information on the gene structure of one gene to find the gene structure of a related gene. This approach should enable PROJECTOR to find more distantly related genes than is possible with protein based methods [GAA$^+$00b, YLB01].

### 4.1.1 Implementation

The parameters of the pair HMM according to which the optimal state path is defined, are its transition and emission probabilities, see Section **2.3**. By default, they have values which are independent of the positions in the two DNA sequences at which they are used. Some of the transitions within the pair HMM are special, i.e. their values depend on the positions in

the two DNA sequences at which they are used, for example the transitions to the splice site states or to the ***START START*** state, see Section **2.3** and Section **6.2** for a more extensive description. We can not only use special transition probabilities, but also special emission probabilities. A state whose emission probabilities are special, reads a score from each of the two sequences depending on the position within that sequence, transforms these scores into posterior probabilities and modifies the nominal value of the emission probability accordingly, *see* Section **6.3** for a detailed description. Special transition and emission probabilities provide the technical concept with which constraints defined by prior knowledge or hypotheses *can* be implemented into the gene prediction.
As an example of how prior knowledge about the input **DNA** sequences can be used within the pair HMM to predict genes, we show how mouse genes *can* be used to predict human genes and vice versa. For this, every state in the pair HMM underlying **PROJECTOR** is defined to have special emission probabilities. Each state can only read letters of the two sequences if the labels of the state match the labels of the sequence whose annotation is used as constraint. To give an example in the case where the annotation of the mouse sequence is used as a constrained to find that of the human sequence: the match exon state has only a non-zero emission probability for reading a pair of codons if the triplet of letters read from the mouse sequences is a codon in the correct phase. The value of the special emission probability thus depends on the letters and the annotation of the mouse DNA sequence at this position, but only the letters of the human DNA sequence. As we know the annotation of one of the two sequences, but not how the two sequences should be aligned, the pair HMM is free to use both match and emit states for finding the optimally scoring state path.

## 4.2 Results

We have used the mouse human test set of 80 sequence pairs described in Section **A.2** in Appendix A, but discarded three sequence pairs as their annotation cannot be found with the pair HMM. In two pairs, one sequence starts immediately with the start codon whereas the start codon of the other sequence is preceded by a intergenic subsequence. In the third pair, the initial exons consist of the start codon only which cannot be modelled by the states and transitions of the pair HMM. The thus reduced test set of 77 sequence pairs is analysed twice: once, using the human genes to find mouse genes and once using the mouse genes to find the human genes. The pair HMM of Chapter **2** is used, i.e. including the states for UTR-

| | Mouse annotation fixed | | Human annotation fixed | |
|---|---|---|---|---|
| | mouse | human | mouse | human |
| **Gene** | | | | |
| Sensitivity | 1 | 0.90 | 0.90 | 1 |
| Specificity | 1 | 0.90 | 0.90 | 1 |
| Genes overlapping | 0 | 0.10 | 0.10 | 0 |
| Genes missing | 0 | 0 | 0 | 0 |
| Genes wrong | 0 | 0 | 0 | 0 |
| **Start Codon** | | | | |
| Sensitivity | 1 | 0.99 | 0.99 | 1 |
| Specificity | 1 | 0.99 | 0.99 | 1 |
| **Stop Codon** | | | | |
| Sensitivity | 1 | 0.96 | 0.96 | 1 |
| Specificity | 1 | 0.96 | 0.96 | 1 |
| **Exon** | | | | |
| Feature Level | | | | |
| Sensitivity | 1 | 0.97 | 0.97 | 1 |
| Specificity | 1 | 0.96 | 0.97 | 1 |
| Exons overlapping | 0 | 0.02 | 0.03 | 0 |
| Exons missing | 0 | 0.003 | 0.01 | 0 |
| Exons wrong | 0 | 0.02 | 0.01 | 0 |
| Nucleotide Level | | | | |
| Sensitivity | 1 | 0.998 | 0.993 | 1 |
| Specificity | 1 | 0.995 | 0.999 | 1 |

Table 4.1: Performance figures for PROJECTOR on the mouse human test set. The predictions were generated using the Stepping Stone algorithm. See Table 3.1 for the definitions of rows.

splicing. The results are generated using PROJECTOR with the Stepping Stone algorithm and are shown in Table 4.1. Note that the predicted genes were not post-processed.

The first thing to note is that the performance for predicting entire genes is very high with a sensitivity and specificity of 90 %. The second thing to note is that the performance is symmetric with respect to the two sequences, i.e. it is as difficult to find a human gene given a related mouse gene as it is to find a mouse gene given a related human gene. The ability to detect start codons is almost perfect with a sensitivity and specificity of 99 %, whereas the performance for stop codons is slightly lower with a sensitivity and specificity of 96 %. The sensitivity and specificity for detecting whole exons is about 97 %. At nucleotide level, the performance for exons is almost perfect.

If we investigate the sixteen genes which were not correctly predicted in detail, we find that fourteen of them are found in pairs, i.e. the mouse gene could not be correctly predicted using the human gene as constraint and vice versa. Four incorrectly predicted genes are found in the two gene pairs for which the number of exons is not the same in the mouse and human gene. These two pairs correspond to the 3 % of the gene pairs in the test set whose genes are related by events of exon-fusion or exon-splitting. In both cases, PROJECTOR predicts the wrong number of exons, but not necessarily the same number of exons as in the annotated sequence see Figure 4.2 and Figure 4.3. PROJECTOR'S difficulty in correctly predicting genes which are related by events of exon-fusion or exon-splitting is not surprising as its parameters could not be reliably trained on the single pair of genes of this type within the training set, *see* Section A.1 in Appendix A. Another source of error for six of the sixteen incorrectly predicted genes is the incorrect prediction of a single splice site in an otherwise correctly predicted gene. These incorrectly predicted splice sites are close to the correct ones and introduce no phase shift into the exons, a typical example is shown in Figure 4.4. They may thus correspond to true alternative splice sites. This supposition is fortified by the fact that the incorrectly predicted splice sites are generally not due to PROJECTOR trying to approximate the length of the predicted exon to that of the annotated exon in the other sequence. Four out of the sixteen incorrectly predicted genes are due to a incorrect prediction of the stop codon as shown in Figure 4.5. In one of the sixteen incorrectly predicted genes is a wrong mini exon of 6 base pairs inserted into an otherwise correctly predicted gene, the corresponding pair of genes is shown in Figure 4.6. Two other incorrectly predicted genes are due to incorrectly predicted start codons, see Figure 4.7. In both cases is 'the length of the predicted exon shifted towards the length of the annotated exon in the other sequence without introducing a phase shift. This may be due to a mis-annotation of the start codon in one or other of the sequences, not a failure by PROJECTOR.

## 4.3 Summary and discussion

PROJECTOR can be successfully used to predict genes which are related to known genes and its sensitivity and specificity at gene level is 90 %. Start and stop codons as well as whole exons are predicted with a high reliability as sensitivity and specificity are higher than 95 %. About a third of the incorrectly predicted genes are due to a single splice site being predicted in close vicinity to the annotated splice site which does not introduce a phase shift into the exons. These cases may correspond to alternative splicing. PROJECTOR's performance could be further improved by training on an enlarged set of pairs of genes which are related by events of exon-splitting or exon-fusion as PROJECTOR so far has difficulty dealing with these cases.

```
------------------------------------------------------------------------------------------

Mm.U13921.MK13.1        1-4678  (4678)  forward

 Mm.U13921.MK13.  : |AGA-->--480-->--ACC|ATG|AGC-->--468-->--AAG|GTG-->--968-->--TAG|ATT-->--83-->--
 annotation       : |----->--480-->-----|SSS|000-->--468-->--000|----->--968-->-----|000-->--83-->--
 prediction       : |----->--480-->-----|SSS|000-->--468-->--000|----->--968-->-----|000-->--83-->--

 Mm.U13921.MK13.  : CAA|GTG-->--194-->--CAG|GTA-->--157-->--GAG|GTG-->--177-->--CAG|GAG-->--162-->--
 annotation       : 000|----->--194-->-----|222-->--157-->--222|----->--177-->-----|000-->--162-->--
 prediction       : 000|----->--194-->-----|222-->--157-->--222|----->--177-->-----|000-->--162-->--

 Mm.U13921.MK13.  : AAG|GTA-->--111-->--TAG|AGT-->--126-->--ATG|GTA-->--90-->--CAG|AAA-->--221-->--T
 annotation       : 000|----->--111-->-----|000-->--126-->--000|----->--90-->-----|000-->--221-->--0
 prediction       : 000|----->--111-->-----|000-->--126-->--000|----->--90-->-----|000-->--221-->--0

 Mm.U13921.MK13.  : AA|GTA-->--726-->--CAG|GAT-->--23-->--GAG|GTA-->--309-->--CAG|TCTCAG|GAAA|TAA|CA
 annotation       : 00|----->--726-->-----|222-->--23-->--222|----->--309-->-----|------|1111|111|11
 prediction       : 00|----->--726-->-----|----->--23-->-----|----->--309-->-----|222222|2222|SSS|--

 Mm.U13921.MK13.  : C-->--61-->--TAT|TAA|CTC-->--303-->--GGC|
 annotation       : 1-->--61-->--111|SSS|----->--303-->-----|
 prediction       : --->--61-->-----|---|----->--303-->-----|

..........................................................................................

Hs.AF049259.2   1-5575  (5575)  forward

 Hs.AF049259.2    : |GGA-->--511-->--ACC|ATG|AGC-->--492-->--AAG|GTG-->--1324-->--TAG|ATC-->--83-->-
 annotation       : |----->--511-->-----|SSS|000-->--492-->--000|----->--1324-->-----|000-->--83-->-
 prediction       : |----->--511-->-----|SSS|000-->--492-->--000|----->--1324-->-----|000-->--83-->-

 Hs.AF049259.2    : -CAA|GTG-->--199-->--TAG|GTA-->--157-->--GAG|GTG-->--190-->--CAG|GAG-->--162-->-
 annotation       : -000|----->--199-->-----|222-->--157-->--222|----->--190-->-----|000-->--162-->-
 prediction       : -000|----->--199-->-----|222-->--157-->--222|----->--190-->-----|000-->--162-->-

 Hs.AF049259.2    : -AAG|GTA-->--124-->--CAG|AGT-->--126-->--ATG|GTA-->--92-->--CAG|AAA-->--221-->--
 annotation       : -000|----->--124-->-----|000-->--126-->--000|----->--92-->-----|000-->--221-->--
 prediction       : -000|----->--124-->-----|000-->--126-->--000|----->--92-->-----|000-->--221-->--

 Hs.AF049259.2    : CAA|GTA-->--627-->--CAG|GAT-->--26-->--CAG|GTA-->--356-->--CAG|GAA-->--16-->--CC
 annotation       : 000|----->--627-->-----|----->--26-->-----|----->--356-->-----|222-->--16-->--22
 prediction       : 000|----->--627-->-----|222-->--26-->--222|----->--356-->-----|111-->--16-->--11

 Hs.AF049259.2    : G|TAG|CAC-->--85-->--CCT|TAA|ATC-->--775-->--CCA|
 annotation       : 2|SSS|----->--85-->-----|---|----->--775-->-----|
 prediction       : 1|111|111-->--85-->--111|SSS|----->--775-->-----|

------------------------------------------------------------------------------------------
```

Figure 4.2: One of the two pairs of mouse and human genes that are related by exon-fusion or exon-splitting. The gene of the mouse sequence, **Mm.U13921.MK13.1**, has eight exons whereas the gene of the human sequence, **Hs.AF049259.2**, has seven. The letters of the **DNA** sequence of the forward strand are shown in the upper row, the annotation in the middle row and the prediction generated by **PROJECTOR** in the lower row. Start and stop codons are denoted by **SSS**, letters within exons are denoted according to the exon's phase by 0, 1 or **2** and letters within intron or intergenic regions by -. The arrows, -->-- or --<--, indicate the orientation of the **DNA**. The numbers give the length of each segment between two separators (**|**) in base pairs.

```
-------------------------------------------------------------------------------------------

Mm.U16984.LT-beta.3     1616-4240       (2625)  forward

 Mm.U16984.LT-be : |CCA-->--500-->--TGG|ATG|GGG-->--159-->--CGG|GTG-->--366-->--CAG|GTT-->--109-->-
 annotation      : |----->--500-->-----|SSS|000-->--159-->--000|----->--366-->-----|000-->--109-->-
 prediction      : |----->--500-->-----|SSS|000-->--159-->--000|----->--366-->-----|000-->--109-->-

 Mm.U16984.LT-be : -CCC|GTC-->--135-->--CAG|GGG-->--72-->--TAG|GTA-->--338-->--CAG|GCG-->--440-->--
 annotation      : -000|000-->--135-->--000|000-->--72-->--000|----->--338-->-----|111-->--440-->--
 prediction      : -000|----->--135-->-----|111-->--72-->--111|----->--338-->-----|111-->--440-->--

 Mm.U16984.LT-be : GGG|TGA|CAG-->--500-->--GAG|
 annotation      : 111|SSS|----->--500-->-----|
 prediction      : 111|SSS|----->--500-->-----|

             .........................................................................

Hs.L11016.4     2264-5131       (2868)  forward

 Hs.L11016.4     : |AAC-->--508-->--TCA|ATG|GGG-->--159-->--CTG|GTG-->--396-->--CAG|GTA-->--46-->--
 annotation      : |----->--508-->-----|SSS|000-->--159-->--000|----->--396-->-----|000-->--46-->--
 prediction      : |----->--508-->-----|SSS|000-->--159-->--000|----->--396-->-----|000-->--46-->--

 Hs.L11016.4     : TGG|GTA-->--71-->--CAG|ATA-->--44-->--CCT|GTT-->--16-->--TAG|CCT-->--52-->--CAG|
 annotation      : 000|----->--71-->-----|----->--44-->-----|----->--16-->-----|----->--52-->-----|
 prediction      : 000|----->--71-->-----|111-->--44-->--111|----->--16-->-----|000-->--52-->--000|

 Hs.L11016.4     : GGT-->--72-->--TAG|GTA-->--395-->--CAG|GCG-->--452-->--GGG|TGA|GGG-->--651-->--T
 annotation      : 111-->--72-->--111|----->--395-->-----|111-->--452-->--111|SSS|----->--651-->---
 prediction      : 000-->--72-->--000|----->--395-->-----|111-->--452-->--111|SSS|----->--651-->---

 Hs.L11016.4     : CA|
 annotation      : --|
 prediction      : --|

-------------------------------------------------------------------------------------------
```

Figure 4.3: The second of the two pairs of mouse and human genes that are related by exon-fusion or exon-splitting. The gene of the mouse sequence, Mm.U16984.LT-beta.3, has three exons whereas the gene of the human sequence, Hs.L11016.4, has four. See Figure 4.2 for an explanation of the notation.

```
------------------------------------------------------------------------------------------

Mm.X72862.22     1-3438  (3438)  forward

 Mm.X72862.22     : |AGA-->--568-->--GAG|ATG|GCT-->--1160-->--CAG|GTA-->--463-->--AAG|GTT-->--37-->-
 annotation       : |----->--568-->-----|SSS|000-->--1160-->--000|----->--463-->-----|222-->--37-->-
 prediction       : |----->--568-->-----|SSS|000-->--1160-->--000|----->--463-->-----|----->--37-->-

 Mm.X72862.22     : -ACG|TGA|AGG-->--284-->--CAG|GAC-->--64-->--TTA|TAA|TGC-->--853-->--ATC|
 annotation       : -222|SSS|----->--284-->-----|----->--64-->-----|---|----->--853-->-----|
 prediction       : ----|---|----->--284-->-----|222-->--64-->--222|SSS|----->--853-->-----|

..........................................................................................

Hs.X72861.23     1-3683  (3683)  forward

 Hs.X72861.23     : |AGA-->--637-->--GGG|ATG|GCT-->--1202-->--CGG|GTA-->--763-->--AAG|GTT-->--37-->-
 annotation       : |----->--637-->-----|SSS|000-->--1202-->--000|----->--763-->-----|----->--37-->-
 prediction       : |----->--637-->-----|SSS|000-->--1202-->--000|----->--763-->-----|222-->--37-->-

 Hs.X72861.23     : -GTA|TGA|AGT-->--222-->--CAG|GGC-->--19-->--TCT|TAG|GCC-->--794-->--AAA|
 annotation       : ----|---|----->--222-->-----|222-->--19-->--222|SSS|----->--794-->-----|
 prediction       : -222|SSS|----->--222-->-----|----->--19-->-----|---|----->--794-->-----|

------------------------------------------------------------------------------------------

Mm.Y00848.26     1274-6554        (5281)  forward

 Mm.Y00848.26     : |GGG-->--500-->--GGG|ATG|GGC-->--217-->--ATA|GTG-->--1743-->--CAG|GCA-->--104-->
 annotation       : |----->--500-->-----|SSS|000-->--217-->--000|----->--1743-->-----|111-->--104-->
 prediction       : |----->--500-->-----|SSS|000-->--217-->--000|----->--1743-->-----|111-->--104-->

 Mm.Y00848.26     : --TCG|GTG-->--1800-->--CAG|GAT-->--331-->--CAG|GTG-->--80-->--GCC|TGA|GTG-->--35
 annotation       : --111|----->--1800-->-----|000-->--331-->--000|000-->--80-->--000|SSS|----->--35
 prediction       : --111|----->--1800-->-----|000-->--331-->--000|----->--80-->-----|---|----->--35

 Mm.Y00848.26     : 8-->--CAG|CTG-->--14-->--ACT|TAG|CAT-->--125-->--CAT|
 annotation       : 8-->-----|----->--14-->-----|---|----->--125-->-----|
 prediction       : 8-->-----|111-->--14-->--111|SSS|----->--125-->-----|

.......          .........................................................................

Hs.X14445.27     436-10092        (9657)  forward

 Hs.X14445.27     : |CGG-->--500-->--ACG|ATG|GGC-->--217-->--ACA|GTG-->--2289-->--TAG|GTA-->--104-->
 annotation       : |----->--500-->-----|SSS|000-->--217-->--000|----->--2289-->-----|111-->--104-->
 prediction       : |----->--500-->-----|SSS|000-->--217-->--000|----->--2289-->-----|111-->--104-->

 Hs.X14445.27     : --TCG|GTG-->--2838-->--CAG|TCA|GTG-->--2807-->--CAG|GAG|CAC-->--382-->--CCA|GTG-
 annotation       : --111|----->--2838-->-----|---|----->--2807-->-----|000|000-->--382-->--000|000-
 prediction       : --111|----->--2838-->-----|000|----->--2807-->-----|---|000-->--382-->--000|----

 Hs.X14445.27     : ->--8-->--CAC|TAG|CTG-->--224-->--CAG|GAA-->--20-->--GCC|TGA|GTG-->--253-->--TTT|
 annotation       : ->--8-->--000|SSS|----->--224-->-----|----->--20-->-----|---|----->--253-->-----|
 prediction       : ->--8-->-----|---|----->--224-->-----|111-->--20-->--111|SSS|----->--253-->-----|

------------------------------------------------------------------------------------------
```

Figure 4.5: The two pairs of genes whose stop codons were incorrectly predicted. The names of the mouse sequences start with Mm, those of the human sequences with Hs. See Figure 4.2 for an explanation of the notation.

```
------------------------------------------------------------------------------------------

Mm.K02781.28     63-1983 (1921)  forward

 Mm.K02781.28     : |TGA-->--500-->--AGC|ATG|GGC-->--117-->--AAG|GTA-->--103-->--AAG|AAC-->--327-->-
 annotation       : |----->--500-->-----|SSS|000-->--117-->--000|----->--103-->-----|000-->--327-->-
 prediction       : |----->--500-->-----|SSS|000-->--117-->--000|----->--103-->-----|000-->--327-->-

 Mm.K02781.28     : -CGG|GTA-->--387-->--TAG|TAC-->--9-->--AGA|TAA|CAG-->--472-->--ACC|
 annotation       : -000|----->--387-->-----|000-->--9-->--000|SSS|----->--472-->-----|
 prediction       : -000|----->--387-->-----|000-->--9-->--000|SSS|----->--472-->-----|

..........................................................................................

Hs.K02043.PND.29          70-2710 (2641)  forward

 Hs.K02043.PND.2 : |CAG-->--500-->--AGC|ATG|AGC-->--120-->--AAG|GTA-->--122-->--AAG|AAT-->--327-->-
 annotation      : |----->--500-->-----|SSS|000-->--120-->--000|----->--122-->-----|000-->--327-->-
 prediction      : |----->--500-->-----|SSS|000-->--120-->--000|----->--122-->-----|000-->--327-->-

 Hs.K02043.PND.2 : -CGG|GTA-->--330-->--TAG|CTGGGT|GTG-->--757-->--CAG|TAC|TGA|AGA-->--470-->--TTT|
 annotation      : -000|----->--330-->-----|------|----->--757-->-----|000|SSS|----->--470-->-----|
 prediction      : -000|----->--330-->-----|000000|----->--757-->-----|000|SSS|----->--470-->-----|

------------------------------------------------------------------------------------------
```

Figure 4.6: This prediction of the human gene, see sequence Hs.K02043.PND.29, contains a wrong mini exon of six base pairs length. The corresponding mouse gene, Mm.K02781.28, is also shown for comparison. See Figure 4.2 for an explanation of the notation.

# Chapter 5

# Prediction of *C. elegans* and C. *briggsae* genes using DOUBLESCAN and PROJECTOR

## 5.1 Introduction and motivation

The genome of the nematode Caenorhabditis elegans was the first multi-cellular organism to be sequenced in 1998 [eSC98]. This model organism has been studied in great detail: we know its developmental lineage to the cellular level and even the entire wiring diagram of its nerve cells. Databases are being maintained which aim at integrating all available information from experimental and theoretical studies into a single coherent picture of the organism [MBD97, SSD+01, Wor]. The sequencing of a related nematode, Caenorhabditis briggsae, is now near to completion. C. elegans and C. briggsae are estimated to have diverged from a common ancestor around 25–100 million years ago, see for example [KAA+93, BFV+97, VPS98] (these estimates vary greatly as they rely on a number of assumptions about mutation rates which cannot be verified as fossil records are not available [ABK96]). The comparative large scale analysis of these two genomes will deepen and maybe also revise our current understanding of the C. elegans genome and at the same time provide the C. briggsae genome with a first global annotation. These analyses will not only aim at detecting the protein coding genes of the two genomes, but will also investigate short conserved regions which may have regulatory functions as well as the large scale structure of the two genomes.

**Our** main motivation for studying C. elegans and C. briggsae DNA sequences is to test if the pair HMM underlying DOUBLESCAN and PROJECTOR can be easily adapted to successfully predict genes in other pairs of related genomes. The non-comparative ab **initio** gene prediction method GENSCAN which is the reference method for the ab **initio** prediction of human genes, **was** reported to have a 'rather poor performance for C. *elegans* genomic sequences' [Bur97, pp. 107] which was attributed to the difficulty of its gene model in dealing with nematode specific features such **as** trans-splicing. In designing the pair HMM underlying DOUBLESCAN and PROJECTOR, the main idea was to keep the gene model **as** general **as** possible so that it can be **used** on any pair of related eukaryotic genomes and to introduce the specialisation to a certain pair of genomes only through the parameters of the model which should be either robust or easily adaptable to new data.

## 5.2 Training of the pair HMM's parameters

The architecture of the pair HMM underlying DOUBLESCAN and PROJECTOR **as** described in Chapter 2 is suitable for the prediction of genes in any pair of related eukaryotic organisms. The specialisation for a certain pair of genomes is only introduced by setting its parameters accordingly.

The pair HMM was adapted to analyze DNA sequences of C. elegans and C. briggsae instead of mouse and human by implementing the following changes:

- Both the parametrisation of the transition probabilities and the values of the parameters (see Table B.1 and Table B.2 in Appendix B) are exactly the same **as** for the mouse human analysis. Only the prior for the transition from an intron state to a **3'** splice site was increased from 1/1000 to 1/100 **as** introns within C. elegans and C. briggsae are on average an order of magnitude shorter than in mouse and human introns (compare the values **for** Prior_AG in Table **B.3** in Appendix **B** and Table D.1 in Appendix D).
- The emission probabilities of the pair HMM were automatically derived from a training set [1] of known C. elegans C. briggsae gene pairs (see Section C.1 in Appendix C) in the same way **as** they were derived from a training set of known mouse human gene pairs, see Section 2.3 and Section 2.3 **for** a detailed description.

[1] **I thank Avril Coghlan, Trinity College, Dublin, for the preparation of the training set.**

- The splice sites scores and start codon scores described in Section 2.3 are generated by GENEFINDER [eSC98], a program which was trained on C. *elegans* genes, rather than by STRATASPLICE [LD01] which is used for the mouse and human analysis. GENEFINDER is used with default cutoff values (−2 for 3' splice sites and 0 for 5' splice sites and start codons) *so* that only splice sites and start codons which score above these cutoff values are taken into account by DOUBLESCAN and PROJECTOR, whereas for the mouse human analysis every potential translation start site and splice site is considered. In addition to the consensus splice sites GT at the 5' splice site and AG at the 3' splice site which are the only splice sites considered for the mouse human analysis, also GC is enabled for 5' splice sites as this type of 5' splice site occurs with a frequency of about 10 % in C. *elegans* introns. This frequency is similar for mouse and human introns, but we chose to model non-consensus splice sites only for nematodes as we expect the performance for nematodes to be *so* good that this effect will be relevant.

The above changes are the only changes made in order to transform the pair HMM underlying DOUBLESCAN and PROJECTOR from a mouse human into a C. *elegans* C. *briggsae* gene prediction program. In particular, its transition probabilities were not tuned by hand to further optimise the performance.

## 5.3 Results

DOUBLESCAN and PROJECTOR are run with the Stepping Stone algorithm on the two test sets of C. *elegans* and *C. briggsae* DNA sequences (see Section C.2 and Section C.3 in Appendix C). As for the analyses of mouse and human DNA sequences described in Chapter 3 and Chapter 4, the DNA sequences are not masked for repeats or anything else. The gene prediction is done three times. Once, using DOUBLESCAN to predict genes simultaneously in C. *elegans* and *C. briggsae* in an *ab initio* way, once keeping the annotation of the C. *elegans* sequences fixed to find C. *briggsae* genes using PROJECTOR and once keeping the annotation of the C. *briggsae* sequences fixed to find C. *elegans* genes. The predicted genes generated by DOUBLESCAN are compared to the annotated genes. The set of predicted genes is not post-processed. The results of the comparison are shown in Table 5.1. The columns labelled 'PROJECTOR' contain the performance on the joint set of C. *elegans* genes which are predicted by keeping the C. *briggsae* genes fixed and the C. *briggsae* genes which are predicted by keeping the C. *elegans* genes fixed.

| | Test set 1 | | Test set 2 | |
|---|---|---|---|---|
| | DOUBLESCAN | PROJECTOR | DOUBLESCAN | PROJECTOR |
| **Gene** | | | | |
| Sensitivity | 0.80 | 0.95 | 0.74 | 0.90 |
| Specificity | 0.71 | 0.95 | 0.62 | 0.90 |
| Genes overlapping | 0.23 | 0.05 | 0.28 | 0.10 |
| Genes missing | 0 | 0 | 0.01 | 0 |
| Genes wrong | 0.06 | 0 | 0.10 | 0 |
| **Start Codon** | | | | |
| Sensitivity | 0.96 | 0.99 | 0.96 | 0.99 |
| Specificity | 0.87 | 0.99 | 0.81 | 0.99 |
| **Stop Codon** | | | | |
| Sensitivity | 0.96 | 0.997 | 0.93 | 0.99 |
| Specificity | 0.89 | 0.997 | 0.82 | 0.99 |
| **Exon** | | | | |
| Feature Level | | | | |
| Sensitivity | 0.93 | 0.99 | 0.91 | 0.97 |
| Specificity | 0.90 | 0.98 | 0.89 | 0.97 |
| Exons overlapping | 0.06 | 0.01 | 0.07 | 0.02 |
| Exons missing | 0.004 | 0.003 | 0.02 | 0.003 |
| Exons wrong | 0.04 | 0.01 | 0.04 | 0.01 |
| Nucleotide Level | | | | |
| Sensitivity | 0.996 | 0.997 | 0.98 | 0.995 |
| Specificity | 0.991 | 0.998 | 0.99 | 0.998 |

**Table 5.1: Performance figures for DOUBLESCAN and PROJECTOR on the two C. *elegans* and C. *briggsae* test sets. The predictions by DOUBLESCAN and PROJECTOR were generated using the Stepping Stone algorithm. The table does not include the performance on the C. *elegans* and C. *briggsae* sequences separately as they are very similar. See Table 3.1 for the definitions of rows.**

| | Test set 1 | | | | Test set 2 | | | |
|---|---|---|---|---|---|---|---|---|
| **DOUBLESCAN** | | | | | | | | |
| incorrectly predicted genes | 139 (23 %) | | | | 268 (28 %) | | | |
| source of error | (1) | (2) | (3) | (4) | (1) | (2) | (3) | (4) |
| split genes | 36 | | 26 | 26 | 88 | | 33 | 27 |
| incorrect or missing start codons | 31 | 19 | 22 | 22 | 37 | 37 | 14 | 11 |
| incorrect or missing stop codons | 30 | | 22 | 22 | 69 | | 26 | 21 |
| incorrectly predicted splice sites | 24 | 20 | 17 | 17 | 70 | 64 | 26 | 22 |
| wrong exons | 14 | 11 | 10 | 10 | 39 | 36 | 15 | 12 |
| missing introns | 3 | 2 | 2 | 2 | 10 | 10 | 4 | 3 |
| missing exons | 2 | 2 | 1 | 1 | 8 | 8 | 3 | 3 |
| inserted introns | 0 | 0 | 0 | 0 | 4 | 4 | 1 | 1 |
| sum | 140 | | | 100 | 325 | | | 100 |
| **PROJECTOR** | | | | | | | | |
| incorrectly predicted genes | 36 (5 %) | | | | 112 (10 %) | | | |
| source of error | (1) | (2) | (3) | (4) | (1) | (2) | (3) | (4) |
| incorrectly predicted splice sites | 12 | 9 | 33 | 31 | 63 | 51 | 56 | 51 |
| wrong exons | 12 | 11 | 33 | 31 | 28 | 26 | 25 | 23 |
| incorrect start codons | 8 | 5 | 22 | 20 | 5 | 4 | 4 | 4 |
| missing introns | 3 | 3 | 8 | 8 | 13 | 13 | 12 | 10 |
| missing exons | 2 | 2 | 6 | 5 | 2 | 2 | 2 | 2 |
| incorrectly predicted stop codons | 2 | | 6 | 5 | 10 | | 9 | 8 |
| inserted introns | 0 | 0 | 0 | 0 | 3 | 2 | 3 | 2 |
| sum | 39 | | | 100 | 124 | | | 100 |

Table 5.2: Error analysis for the genes of the two *C. elegans* and *C. briggsae* test sets which are incorrectly predicted by **DOUBLESCAN** or **PROJECTOR**. Column (1) gives the number of incorrectly predicted genes with this type of error, column (2) gives the number of incorrectly predicted genes where this type of error does not lead to a phase shift, column (3) gives the percentage of incorrectly predicted genes with this error and column (4) the percentage of this error within all errors. To give an example: **PROJECTOR** predicts 36 genes incorrectly which corresponds to 5 % of the annotated genes in test set 1. 12 of the 36 incorrectly predicted genes have incorrectly predicted splice sites, but this leads in 9 out of 12 genes to no phase shift. 33 % of incorrectly predicted genes have an incorrectly predicted splice site and incorrectly predicted splice sites correspond to 31 % of the errors made. Note that the sum of numbers in column (1) need not be equal to the number of incorrectly predicted genes and the sum of numbers in column (3) is not necessarily 100 % as some genes are affected by more than one type of error.

The first thing to note is that the performance both of the ab initio gene prediction and the homology based prediction is very good. This is very promising, especially given the fact that the switch from the mouse and human to the C. eiegans and C. briggsae pair HMM which underlies **DOUBLESCAN** and **PROJECTOR** consists only of a few steps. The second thing to note is that the performance on test set **1** is significantly better than that on test set **2** both in terms of sensitivity and specificity. Note that the table does not include the performance on the C. elegans and C. briggsae sequences separately as they are very similar. **DOUBLESCAN** is not biased towards preferentially predicting C. eiegans or C. briggsae genes correctly.
As the two test sets have been generated in different ways, they are discussed separately.

### 5.3.1 Performance on test set 1

**Performance of the** ab initio **gene prediction with DOUBLESCAN** Though the sensitivity of the prediction is generally high with **80** % at gene level, the specificity on gene and feature level is generally significantly lower, but the sensitivity and specificity values converge when going from gene level to nucleotide level. **DOUBLESCAN** detects start and stop codons with **95** % specificity and its sensitivity and specificity for whole exons are above **90** %.
The set of **139** incorrectly predicted genes which overlap an annotated gene can be subdivided into subsets according to the error that was made, see Table **5.2** for an overview. There are three main errors.
The first type of error in **36** out of the **139** genes consists of splitting the gene into two (or three in six cases) genes which overlap the annotated gene. The overlap between the predicted genes and the annotated gene is generally very large and the split typically involves only two incorrectly predicted splice sites, see Figure **5.1** for an example.
The next common type of error present in **31** out of the **139** incorrectly predicted genes is a start codon which is incorrectly predicted or missing in the predicted gene. An incorrectly predicted start codon **(15** out of **31)** is typically close to the annotated one and does not lead to a phase shift **(13** out of **15).** A typical example is shown in Figure **5.2.** If the start codon is missing from the prediction **(16** out of **31**), there is usually a splice site predicted in close vicinity to the annotated start codon, but this splice site can **(10** out of **16)** or cannot **(6** out of **16)** lead to a phase shift. Figure **5.3** shows an example in which the missing start codon does not lead to a frame shift.
Another common type of error shown in **30** out of the **139** incorrectly predicted genes is a stop

| number of amino-acids | number of genes |
|---|---|
| 3 | 8 |
| 4 | 5 |
| 5 | 4 |
| 6 | 6 |
| 7 | 2 |
| 8 | 2 |
| 12 | 1 |
| 18 | 2 |
| 27 | 1 |
| 30 | 1 |

Table 5.3: Length distribution of the 32 wrong complete genes predicted by DOUBLESCAN on test set 1. All genes are single exon genes.

codon which is incorrectly predicted (2 out of 30) or missing (28 out of 30) in the predicted gene. As for missing start codons, a missing stop codon is usually due to a splice site being predicted close the the annotated stop codon. A typical example can be seen in Figure 5.4.
The rest of the errors found in the remaining 42 of the 139 incorrectly predicted genes are due to incorrectly predicted splice sites (24 cases), extra wrong exons being predicted (14 cases), an intron missing in the predicted gene (3 cases) or an exon missing in the predicted gene (2 cases). Twenty of the 24 genes in which a splice site is incorrectly predicted do not lead to phase shifts and the predicted splice site is close to the annotated one. These cases may thus be real splice sites which are used in alternative splicing. See Figure 5.5 for an example. The 14 genes in which an extra wrong exon has been predicted are mainly (11 out of 14) due to short exons which do not introduce a phase shift as their length is a multiple of three, see Figure 5.6.
The 6 % rate of wrong genes corresponds to 50 genes which do not overlap any annotated gene. They consist of 32 complete and 18 partial genes. The complete genes are typically very short, see Table 5.3, and are all single exon genes. The partial genes consist of a partial intron, an exon and the start or stop codon. 40 % of the wrong genes lie 5' to the annotated gene and 60 % lie 3' to the annotated gene.

**Performance of the gene prediction with PROJECTOR** The performance of **PROJECTOR** is very high with a sensitivity and specificity of **95** % at gene level. As the pair HMM underlying **DOUBLESCAN** and **PROJECTOR** predicts genes in pairs, the rate of missing and wrong genes for **PROJECTOR** is zero by construction. The low percentage of overlapping genes corresponds to **36** genes.

The two main sources of errors are incorrectly predicted splice sites and wrong intermediate exons of short length. There are **12** genes with incorrectly predicted splice sites which all do not lead to an overall phase shift. In nine out of **12** genes the incorrectly predicted splice sites do not changes the phase and in the other three the two incorrectly predicted splice sites follow each other and have no overall phase shifting effect. The incorrect splice sites are typically close to the annotated ones and may correspond to true splice sites which may be mis-annotated or used in alternative splicing. Twelve of the incorrectly predicted genes are due to the prediction of a wrong intermediate exon of short length. Almost all of them **(11** out of **12)** do not lead to a phase shift and thus correspond at protein level to the insertion of few amino-acids. The next common error present in eight out of the **36** incorrectly predicted genes are incorrectly predicted start codons. In five out of the eight cases there is no phase shift due to the incorrectly predicted start codon. The remaining errors are due to **missing** introns that do not alter the phase of the exons **(3** genes), missing exons **(2** genes) and incorrectly predicted stop codons **(2** genes).

### 5.3.2 Performance on test set 2

Test set **2** consists of more diverged pairs of genes (see Table **C.3** in Appendix **C)** whose genes have on average more exons and are longer than those of test set **1** (see Table **C.l** in Appendix **C).**

**Performance of the** ab **initio gene prediction with DOUBLESCAN** Sensitivity and specificity at gene level on test set **2** are generally lower than on test set **1,** the sensitivity of 74 % being **6** % lower and the specificity of **62** % being **9** % lower. Still, two thirds of the genes are perfectly predicted which is very high for an ***ab* initio** method. As **for** test set **1,** the values for sensitivity and specificity converge when going from gene level to nucleotide level performance where they are almost the same. Both sensitivity and specificity for whole exons are around 90 % and the sensitivity for detecting start and stop codons is even higher.

```
-----------------------------------------------------------------------------------------------

CE.C06G1.1.f    16658735-16668767       (10033) forward

 CE.C06G1.1.f    : |CGA-->--5433-->--CTG|ATG|ATT-->--61-->--CAA|GTG-->--58-->--CAG|ATA-->--158-->--
 annotation      : |----->--5433-->-----|SSS|000-->--61-->--000|----->--58-->-----|111-->--158-->--
 prediction      : |----->--5433-->-----|SSS|000-->--61-->--000|----->--58-->-----|111-->--158-->--

 CE.C06G1.1.f    : CAG|GTA-->--383-->--CAG|GTA-->--99-->--ACG|GTA-->--50-->--CAG|GGA-->--92-->--CAA
 annotation      : 111|----->--383-->-----|000-->--99-->--000|----->--50-->-----|000-->--92-->--000
 prediction      : 111|----->--383-->-----|000-->--99-->--000|----->--50-->-----|000-->--92-->--000

 CE.C06G1.1.f    : |GTG-->--165-->--CAG|AAC-->--181-->--CAG|GTA-->--70-->--CAG|GTT-->--115-->--TCG|
 annotation      : |----->--165-->-----|222-->--181-->--222|----->--70-->-----|000-->--115-->--000|
 prediction      : |----->--165-->-----|222-->--181-->--222|----->--70-->-----|000-->--115-->--000|

 CE.C06G1.1.f    : GTA-->--359-->--CAG|GAC-->--72-->--AAG|GTA-->--70-->--CAG|GAA-->--252-->--CCG|GT
 annotation      : ----->--359-->-----|111-->--72-->--111|----->--70-->-----|111-->--252-->--111|--
 prediction      : ----->--359-->-----|111-->--72-->--111|----->--70-->-----|111-->--252-->--111|--

 CE.C06G1.1.f    : A-->--200-->--CAG|CAC-->--128-->--CAG|GTA-->--97-->--CAG|CAA-->--126-->--GAT|GTA
 annotation      : --->--200-->-----|111-->--128-->--111|----->--97-->-----|000-->--126-->--000|---
 prediction      : --->--200-->-----|111-->--128-->--111|----->--97-->-----|000-->--126-->--000|---

 CE.C06G1.1.f    : -->--49-->--CAG|GTT-->--153-->--AAT|GTC-->--13-->--AAG|AT|G|TAG|GTA-->--780-->--
 annotation      : -->--49-->-----|000-->--153-->--000|000-->--13-->--000|00|-|---|----->--780-->--
 prediction      : -->--49-->-----|000-->--153-->--000|----->--13-->-----|00|0|SSS|----->--780-->--

 CE.C06G1.1.f    : CAG|GAC|ATG|CTG-->--96-->--GGT|TAA|AGT-->--48-->--GTG|ATG|AACG|GTG-->--700-->--T
 annotation      : ---|000|000|000-->--96-->--000|SSS|----->--48-->-----|---|----|----->--700-->---
 prediction      : ---|---|SSS|000-->--96-->--000|SSS|----->--48-->-----|SSS|0000|----->--700-->---

 CE.C06G1.1.f    : TT|
 annotation      : --|
 prediction      : --|

.............................................................................................

CB.gf.s146.9.r  35800-40603      (4804)  reverse

 CB.gf.s146.9.r  : |CAA-->--476-->--CAC|CGTT|CAT|CAC-->--56-->--TTT|TTA|ACC-->--96-->--CAC|CAT|GTC|
 annotation      : |-----<--476--<-----|----|---|-----<--56--<-----|SSS|000--<--96--<--000|000|000|
 prediction      : |-----<--476--<-----|0000|SSS|-----<--56--<-----|SSS|000--<--96--<--000|SSS|---|

 CB.gf.s146.9.r  : CTG-->--180-->--TAC|TCA|C|AT|CTT-->--13-->--AAC|ATT-->--153-->--GAC|CTG-->--52--
 annotation      : -----<--180--<-----|---|-|00|000--<--13--<--000|000--<--153--<--000|-----<--52--
 prediction      : -----<--180--<-----|SSS|0|00|-----<--13--<-----|000--<--153--<--000|-----<--52--

 CB.gf.s146.9.r  : >--TAC|ATC-->--126-->--TTG|CTG-->--96-->--TAC|CTG-->--128-->--GAG|CTG-->--50-->-
 annotation      : <-----|000--<--126--<--000|-----<--96--<-----|111--<--128--<--111|-----<--50--<-
 prediction      : <-----|000--<--126--<--000|-----<--96--<-----|111--<--128--<--111|-----<--50--<-

 CB.gf.s146.9.r  : -TAC|CAG-->--252-->--TTC|CTG-->--72-->--TAC|CTT-->--72-->--GGC|CTG-->--292-->--T
 annotation      : ----|111--<--252--<--111|-----<--72--<-----|111--<--72--<--111|-----<--292--<---
 prediction      : ----|111--<--252--<--111|-----<--72--<-----|111--<--72--<--111|-----<--292--<---

 CB.gf.s146.9.r  : AC|CTA-->--115-->--AAC|CTG-->--42-->--CAC|CTG-->--181-->--GTT|CTG-->--192-->--CA
 annotation      : --|000--<--115--<--000|-----<--42--<-----|222--<--181--<--222|-----<--192--<----
 prediction      : --|000--<--115--<--000|-----<--42--<-----|222--<--181--<--222|-----<--192--<----

 CB.gf.s146.9.r  : C|TTG-->--92-->--ACC|CTG-->--43-->--TAC|TGT-->--99-->--AAC|CTG-->--48-->--TAC|TT
 annotation      : -|000--<--92--<--000|-----<--43--<-----|000--<--99--<--000|-----<--48--<-----|11
 prediction      : -|000--<--92--<--000|-----<--43--<-----|000--<--99--<--000|-----<--48--<-----|11

 CB.gf.s146.9.r  : G-->--158-->--TGT|CTG-->--48-->--CAC|TTG-->--61-->--AAT|CAT|CAG-->--1586-->--AAG|
 annotation      : 1--<--158--<--111|-----<--48--<-----|000--<--61--<--000|SSS|-----<--1586--<-----|
 prediction      : 1--<--158--<--111|-----<--48--<-----|000--<--61--<--000|SSS|-----<--1586--<-----|

-----------------------------------------------------------------------------------------------
```

Figure 5.1: Example of a gene pair where the annotated gene is split into two genes predicted by DOUBLESCAN which overlap the annotated gene. The prediction also contains two partial genes which are wrong. The C. elegans sequence, CE.C06G1.1.f, is shown at the top, the corresponding C. *briggsae* sequence, CB.gf.s146.9.r, at the bottom. See Figure 4.2 for an explanation of the notation.

```
--------------------------------------------------------------------------------------------

CE.C25H3.9.r     5689738-5691493 (1756)  reverse

 CE.C25H3.9.r     : |CAA-->--864-->--CAA|TTA|CTG-->--154-->--TGC|CTG-->--45-->--TAC|CTC-->--140-->--
 annotation       : |-----<--864--<-----|SSS|222--<--154--<--222|-----<--45--<-----|000--<--140--<--
 prediction       : |-----<--864--<-----|SSS|222--<--154--<--222|-----<--45--<-----|000--<--140--<--

 CE.C25H3.9.r     : TCC|CTG-->--49-->--TAC|TTC-->--111-->--ATC|CTT-->--47-->--TAC|CTT-->--132-->--CG
 annotation       : 000|-----<--49--<-----|000--<--111--<--000|-----<--47--<-----|000--<--132--<--00
 prediction       : 000|-----<--49--<-----|000--<--111--<--000|-----<--47--<-----|000--<--132--<--00

 CE.C25H3.9.r     : C|CAT|TTT-->--15-->--CGC|CAT|GTT-->--190-->--AAG|
 annotation       : 0|SSS|-----<--15--<-----|---|-----<--190--<-----|
 prediction       : 0|000|000--<--15--<--000|SSS|-----<--190--<-----|

............................................................................................

CB.gf.s150.69.r 238584-240007   (1424)  reverse

 CB.gf.s150.69.r : |ATG-->--544-->--CAT|TCA|CTG-->--154-->--TGT|CTA-->--49-->--TAC|CTC-->--140-->--
 annotation      : |-----<--544--<-----|SSS|222--<--154--<--222|-----<--49--<-----|000--<--140--<--
 prediction      : |-----<--544--<-----|SSS|222--<--154--<--222|-----<--49--<-----|000--<--140--<--

 CB.gf.s150.69.r : TCC|CTG-->--50-->--TAC|CTC-->--111-->--GTC|CTG-->--51-->--TAC|CTT-->--132-->--AG
 annotation      : 000|-----<--50--<-----|000--<--111--<--000|-----<--51--<-----|000--<--132--<--00
 prediction      : 000|-----<--50--<-----|000--<--111--<--000|-----<--51--<-----|000--<--132--<--00

 CB.gf.s150.69.r : C|CAT|TTT-->--15-->--CGC|CAT|GGT-->--169-->--GTC|
 annotation      : 0|SSS|-----<--15--<-----|---|-----<--169--<-----|
 prediction      : 0|000|000--<--15--<--000|SSS|-----<--169--<-----|

--------------------------------------------------------------------------------------------
```

Figure 5.2: Example of a gene pair where the start codon of the gene predicted by **DOUBLESCAN** lies close to the annotated one and involves no phase shift. Note that the genes in this example lie on the reverse strand. The C. elegans sequence, CE.C25H3.9.r, is shown at the top, the corresponding C. briggsae sequence, CB.gf.s150.69.r, at the bottom. See Figure 4.2 for an explanation of the notation.

```
-------------------------------------------------------------------------------------------------

CE.F35G2.2.r     12207941-12209531         (1591)  reverse

 CE.F35G2.2.r     : |TTC-->--532-->--CAT|TTA|GAG-->--339-->--AAA|CTG-->--51-->--TAC|TTT-->--178-->--
 annotation       : |-----<--532--<-----|SSS|000--<--339--<--000|-----<--51--<-----|222--<--178--<--
 prediction       : |-----<--532--<-----|SSS|000--<--339--<--000|-----<--51--<-----|222--<--178--<--

 CE.F35G2.2.r     : GAC|CTG-->--57-->--TAC|GCG-->--77-->--AGC|CAT|CAC|CTG-->--348-->--CAA|
 annotation       : 222|-----<--57--<-----|000--<--77--<--000|SSS|---|-----<--348--<-----|
 prediction       : 222|-----<--57--<-----|000--<--77--<--000|000|000|-----<--348--<-----|

.......................................................................................

CB.gf.s6.24.f    71004-72828       (1825)  forward

 CB.gf.s6.24.f    : |AAA-->--630-->--CAG|ATC|ATG|GCT-->--77-->--TGC|GTA-->--50-->--CAG|GTC-->--178--
 annotation       : |----->--630-->-----|---|SSS|000-->--77-->--000|----->--50-->-----|222-->--178--
 prediction       : |----->--630-->-----|000|000|000-->--77-->--000|----->--50-->-----|222-->--178--

 CB.gf.s6.24.f    : >--AAG|GTA-->--54-->--TAG|TTC-->--339-->--CTC|TAA|ATT-->--488-->--AGT|
 annotation       : >--222|----->--54-->-----|000-->--339-->--000|SSS|----->--488-->-----|
 prediction       : >--222|----->--54-->-----|000-->--339-->--000|SSS|----->--488-->-----|

-------------------------------------------------------------------------------------------------
```

Figure **5.3:** Example **of** a gene pair where the start codons are missing in the genes predicted by **DOUBLESCAN. A** splice site has been introduced in close vicinity to the annotated start codon which does not lead to a phase shift. The C. *elegans* sequence, CE.F35G2.2.r, with the gene lying on the reverse strand is shown at the top, the corresponding C. briggsae sequence, CB.gf.s6.24.f, at the bottom. See Figure **4.2 for** an explanation **of** the notation.

```
-------------------------------------------------------------------------------------------------

CE.C06B8.8.f     15514309-15515518         (1210)  forward

 CE.C06B8.8.f     : |ACA-->--481-->--GCG|ATG|CCA-->--141-->--GCC|GTA-->--272-->--CAG|GAC-->--65-->--
 annotation       : |----->--481-->-----|SSS|000-->--141-->--000|----->--272-->-----|000-->--65-->--
 prediction       : |----->--481-->-----|SSS|000-->--141-->--000|----->--272-->-----|000-->--65-->--

 CE.C06B8.8.f     : CAA|G|TAG|ATC-->--244-->--GTT|
 annotation       : 000|0|SSS|----->--244-->-----|
 prediction       : 000|-|---|----->--244-->-----|

.................................................................................................

CB.gf.s219.10.f 53654-56301       (2648)  forward

 CB.gf.s219.10.f : |AGT-->--1786-->--ACA|ATG|CCA-->--141-->--GCC|GTG-->--553-->--CAG|GAC-->--65-->-
 annotation      : |----->--1786-->-----|SSS|000-->--141-->--000|----->--553-->-----|000-->--65-->-
 prediction      : |----->--1786-->-----|SSS|000-->--141-->--000|----->--553-->-----|000-->--65-->-

 CB.gf.s219.10.f : -CAA|G|TAG|ATT-->--96-->--AAT|
 annotation      : -000|0|SSS|----->--96-->-----|
 prediction      : -000|-|---|----->--96-->-----|

-------------------------------------------------------------------------------------------------
```

Figure **5.4:** Example **of** a gene pair where the stop codons are missing in the genes predicted by **DOUBLESCAN. A** splice site has been introduced in close vicinity to the annotated stop codon. The C. elegans sequence, CE.C06B8.8.f, is shown at the top, the corresponding C. briggsae sequence, CB.gf.s219.10.f, at the bottom. *See* Figure **4.2 for** an explanation **of** the notation.

```
-----------------------------------------------------------------------------------------------

CE.R10H10.5.f   10399662-10408729        (9068)  forward

 CE.R10H10.5.f   : |CCA-->--4532-->--CAA|ATG|GGT-->--115-->--TAG|GTA-->--1840-->--CAG|GAG-->--185--
 annotation      : |----->--4532-->-----|SSS|000-->--115-->--000|----->--1840-->-----|111-->--185--
 prediction      : |----->--4532-->-----|SSS|000-->--115-->--000|----->--1840-->-----|111-->--185--

 CE.R10H10.5.f   : >--GTG|GTG-->--94-->--AAG|AAA-->--15-->--CAG|CGT-->--155-->--ATA|GTA-->--372-->-
 annotation      : >--111|----->--94-->-----|----->--15-->-----|000-->--155-->--000|----->--372-->-
 prediction      : >--111|----->--94-->-----|000-->--15-->--000|000-->--155-->--000|----->--372-->-

 CE.R10H10.5.f   : -CAG|CTT-->--259-->--ACG|GTT-->--195-->--CAG|AAC-->--93-->--AAG|GTA-->--43-->--C
 annotation      : ----|222-->--259-->--222|----->--195-->-----|000-->--93-->--000|----->--43-->---
 prediction      : ----|222-->--259-->--222|----->--195-->-----|000-->--93-->--000|----->--43-->---

 CE.R10H10.5.f   : AG|GAC-->--122-->--TAG|GTA-->--630-->--CAG|GTC-->--124-->--TGT|TAA|ATA-->--288--
 annotation      : --|000-->--122-->--000|----->--630-->-----|222-->--124-->--222|SSS|----->--288--
 prediction      : --|000-->--122-->--000|----->--630-->-----|222-->--124-->--222|SSS|----->--288--

 CE.R10H10.5.f   : >--GAA|
 annotation      : >-----|
 prediction      : >-----|

...............................................................................................

CB.gf.s54.21.f  77164-84617      (7454)  forward

 CB.gf.s54.21.f  : |AAG-->--2668-->--TAA|ATG|GGT-->--115-->--TAG|GTA-->--1536-->--CAG|GAG-->--182--
 annotation      : |----->--2668-->-----|SSS|000-->--115-->--000|----->--1536-->-----|111-->--182--
 prediction      : |----->--2668-->-----|SSS|000-->--115-->--000|----->--1536-->-----|111-->--182--

 CB.gf.s54.21.f  : >--CGA|GTG|GTT-->--877-->--AAG|TAT-->--18-->--CAG|CGA-->--155-->--GTA|GTA-->--28
 annotation      : >--111|111|----->--877-->-----|----->--18-->-----|000-->--155-->--000|----->--28
 prediction      : >--111|---|----->--877-->-----|000-->--18-->--000|000-->--155-->--000|----->--28

 CB.gf.s54.21.f  : 9-->--CAG|TTT-->--259-->--ACA|GTA-->--46-->--CAG|AAC-->--93-->--AAG|GTA-->--243-
 annotation      : 9-->-----|222-->--259-->--222|----->--46-->-----|000-->--93-->--000|----->--243-
 prediction      : 9-->-----|222-->--259-->--222|----->--46-->-----|000-->--93-->--000|----->--243-

 CB.gf.s54.21.f  : ->--CAG|GAC-->--122-->--TCG|GTG-->--511-->--TAG|CTC-->--124-->--TGT|TAA|AAC-->--
 annotation      : ->-----|000-->--122-->--000|----->--511-->-----|222-->--124-->--222|SSS|----->--
 prediction      : ->-----|000-->--122-->--000|----->--511-->-----|222-->--124-->--222|SSS|----->--

 CB.gf.s54.21.f  : 207-->--TCC|
 annotation      : 207-->-----|
 prediction      : 207-->-----|
-----------------------------------------------------------------------------------------------
```

Figure *5.5:* Example of a gene pair predicted **by DOUBLESCAN** which has incorrectly predicted splice sites. The splice sites are close to the annotated ones and the mis-prediction does not introduce a phase shift. The C. elegans sequence, CE.R10H10.5.f, is shown at the top, the corresponding C. ***briggsae*** sequence, CB.gf.s54.21.f, at the bottom. ***See*** Figure 4.2 for an explanation of the notation.

```
------------------------------------------------------------------------------------------------

CE.Y38F1A.9.r   13003625-13008396        (4772)  reverse

 CE.Y38F1A.9.r   : |TTA-->--2928-->--AAT|TTA|GTC-->--123-->--GTT|CTG-->--912-->--GAC|ATT-->--24-->-
 annotation      : |-----<--2928--<-----|SSS|000--<--123--<--000|-----<--912--<-----|-----<--24--<-
 prediction      : |-----<--2928--<-----|SSS|000--<--123--<--000|-----<--912--<-----|000--<--24--<-

 CE.Y38F1A.9.r   : -TTT|CTG-->--198-->--TAC|CTT-->--91-->--AAC|CTA-->--44-->--TAC|CAT-->--107-->--A
 annotation      : ----|-----<--198--<-----|222--<--91--<--222|-----<--44--<-----|000--<--107--<--0
 prediction      : -000|-----<--198--<-----|222--<--91--<--222|-----<--44--<-----|000--<--107--<--0

 CE.Y38F1A.9.r   : AC|CAT|TCT-->--339-->--TTA|
 annotation      : 00|SSS|-----<--339--<-----|
 prediction      : 00|SSS|-----<--339--<-----|

.........................................................................................

CB.gf.s185.4.r  10405-14568      (4164)  reverse

 CB.gf.s185.4.r  : |ATA-->--2656-->--TAT|TTA|ATC-->--123-->--GTT|CTG-->--675-->--CAC|CTG-->--18-->-
 annotation      : |-----<--2656--<-----|SSS|000--<--123--<--000|-----<--675--<-----|-----<--18--<-
 prediction      : |-----<--2656--<-----|SSS|000--<--123--<--000|-----<--675--<-----|000--<--18--<-

 CB.gf.s185.4.r  : -CGG|CTC-->--89-->--TAC|CTT-->--91-->--AAC|CTG-->--97-->--CAC|CAT-->--107-->--AA
 annotation      : ----|-----<--89--<-----|222--<--91--<--222|-----<--97--<-----|000--<--107--<--00
 prediction      : -000|-----<--89--<-----|222--<--91--<--222|-----<--97--<-----|000--<--107--<--00

 CB.gf.s185.4.r  : C|CAT|TTT-->--302-->--ATC|
 annotation      : 0|SSS|-----<--302--<-----|
 prediction      : 0|SSS|-----<--302--<-----|

--------------------------------------------------------------------------
```

Figure *5.6:* Example of a gene pair with a wrong extra exon predicted by **DOUBLESCAN**. The extra exons are short and their length is a multiple of three base pairs thus not leading to a phase shift in the remaining correctly predicted gene structure. The *C. elegans* sequence, CE.Y38F1A.9.r, with the gene on the reverse strand is shown at the top, the corresponding *C. briggsae* sequence, CB.gf.s185.4.r, with the gene also on the reverse strand is shown at the bottom. See Figure 4.2 for an explanation of the notation.

As for test set 1, the set of 268 incorrectly predicted genes which overlap an annotated gene can be subdivided into subsets according to the type of error made in the prediction, see Table 5.2 for an overview.

The dominant type of error (accounting for 27 % of errors in this test set and 26 % of errors in test set 1) consists of splitting the gene into two or more genes which overlap the annotated gene. As for test set 1, the overlap between the predicted genes and the annotated gene is very large.

As for test set 1, the incorrect or missing prediction of stop codons is another common type of error accounting for 21 % of errors (22 % in test set 1). In most cases (52 out of the 69) is a splice site predicted close to the annotated stop codon and the stop codon is missing from the prediction, see Figure 5.4. Another common type of error accounting for 22 % of errors are incorrectly predicted splice sites. This type of error is less common in test set 1 where it accounts for only 17 % of the errors. The vast majority of incorrectly predicted splice sites (64 out of 70) does not lead to a phase shift. Though the predicted splice sites are not always in close vicinity to the annotated splice sites, at least some of them may correspond to splice sites which are used in alternative splicing.

Incorrectly predicted start codons or start codons which are missing in the predicted gene account for only 11 % of the errors in this test set, whereas this type of error was more prevalent in test set 1 (accounting for 22 % of errors). Of the 21 incorrectly predicted start codons, 17 cases are mis-predictions due to a shortened or enlarged initial exon, the cases typically look like Figure 5.2 and may be due to incorrectly annotated start codons. In 14 out of the 16 genes with missing start codon, a splice site is introduced in close vicinity to the annotated start codon, see Figure 5.3 for a typical example. However, as opposed to the errors made in test set 1, the incorrect or missing prediction of the start codon leads in no case to a phase shift, i.e. the overlap between the amino-acid sequence encoded in the predicted and the annotated gene is generally high.

The remaining errors are wrong exons (accounting for 12 % of errors in this test set and for 10 % of errors in test set 1), missing introns (3 % of errors in this test set and 2 % of errors in test set 1), missing exons (3 % of errors in this test set and 1 % of errors in test set 1) and inserted introns (1 % of errors in this test set and no errors in test set 1). The majority of wrong exons (36 out of 39 cases) entail no phase shift as does none of the missing or inserted introns or missing exons.

| number of amino-acids | number of genes | comments |
|---|---|---|
| 2 | 22 | |
| 3 | 12 | |
| 4 | 11 | |
| 5 | 7 | |
| 6 | 11 | |
| 7 | 7 | 2 two exon genes |
| 8 | 12 | |
| 10 | 1 | |
| 12 | 2 | |
| 13 | 1 | |
| 14 | 4 | |
| 15 | 1 | |
| 16 | 1 | |
| 18 | 2 | |
| 21 | 1 | |
| 26 | 1 | |
| 47 | 2 | |
| 110 | 1 | two exon gene |
| 113 | 1 | twoexoneene |

**Table 5.4: Length distribution of the 100 wrong complete genes predicted by DOUBLESCAN on test set 2.**

As opposed to test set **1** in which every annotated gene is overlapped by a predicted gene, eight genes in test set **2** are missing completely in the prediction. They correspond to four pairs of genes. The pairs of genes have the same number of exons **(3, 4, 5** and **6** exons, respectively), but except for one pair of genes the lengths of the exons in one pair of genes are generally not the same. Their difference in length ranges from **3** base pairs to **18** base pairs. Overall, the four gene pairs which are missing in the prediction do not have a distinctive feature which sets them apart from the other genes.

The **10 %** rate of wrong genes corresponds to **134** genes which do not overlap any annotated gene. One hundred of the **134** genes are complete genes comprising start and stop codon and the remaining **34** genes are partial genes. **82** out of the **100** complete genes encode less than ten amino-acids and almost all complete genes **(96** out of **100)** consist of a single exon gene. The length distribution of wrong complete genes is shown in Table **5.4.** The **34** partial genes typically consist of a short initial or terminal exon comprising the start or the stop codon and a partial intron. There is no clear bias towards the **5'** or **3'** side of the annotated gene: **52 %** of the wrong genes lie **5'** to the annotated gene and **48 % 3'** to the annotated gene.

**Performance of the gene prediction with PROJECTOR** The performance of PROJECTOR is high with a sensitivity and specificity of **90 %** at gene level. The **10 %** of overlapping genes corresponds to **112** genes.

About half of the incorrectly predicted genes **(63** out of **112)** are due to a mis-predicted splice site of one of the intermediate exons which in **51** of the **63** cases does not result in a phase shift. This type of error is much more common in this test set **(51** %) than in test set **1 (31** %). As for test set **1,** the incorrectly predicted splice sites are close to the annotated one and may be due to alternative splicing. The next most common source of errors are wrongly predicted intermediate exons. This type of error occurs in **28** out out the **112** incorrectly predicted genes and thus accounts for **23 %** of the errors **(31 %** of the errors in test set **1).** Incorrect start codons are only predicted in **5** of the **112** genes (corresponding to **4** %), whereas this type of error accounts for **22 %** of the incorrectly predicted genes in test set **1.** The mis-predicted start codon shortens or enlarges the initial exon, mostly without altering the phase within the exon. These cases may thus be due to a false annotation of the start codon rather than a false prediction by **DOUBLESCAN.** The rates of the other types of errors are similar to those in test set **1:** in **12 %** genes of the incorrectly predicted gene is an intron missing (in no case leading to a phase shift), **3** % of incorrectly predicted genes contain a wrong intron, **9** % have an incorrectly predicted stop codon and **2** % a missing exon (in all cases not leading to a phase shift). As opposed to genes with mis-predicted start codons for which the predicted and the annotated initial exons tend to have a large overlap, eight of the ten genes with an incorrectly predicted stop codon completely lack the annotated terminal exon.

### 5.3.3 Comparison of the performance of DOUBLESCAN and FGENESH

In order to see how well **DOUBLESCAN** does in comparison to other ab initio gene prediction programs, we compared its performance to that of **FGENESH** (Version **1.0,** nematode version of the model used with nematode parameters) [SS00]. **FGENESH** is a non-comparative ab initio gene prediction method which employs an HMM with an algorithm similar to that of **GENIE** [KHRE96] and **GENSCAN** [BK97]. **As GENSCAN, FGENESH** explicitly models the length distribution of exons and chooses its set of parameters according to the GC content of the input **DNA** sequence. Its parameters (transition and emission probabilities as well as length distributions) have been especially trained on a large set of known C. elegans genes and the underlying model has been modified to analyse nematode genes.

**FGENESH** is run on the two C. elegans and C. briggsae test sets (see Section **C.2** and Section **C.3** in Appendix **C)** and its performance compared to that of **DOUBLESCAN,** see Table **5.5.** **As** the performance for **FGENESH** is almost the same for the set of C. elegans and the set of C. briggsae genes (as is the case for **DOUBLESCAN),** Table **5.5** shows the performance only for the combined set of C. elegans and C. briggsae genes.

The first thing to note is that **FGENESH** has a significantly higher sensitivity and specificity on test set 1 than on test set **2.** When comparing the performance of **FGENESH** to that of **DOUBLESCAN** on test set **1, FGENESH** has a slightly higher sensitivity **(2** %) and a significantly higher specificity **(10** %) for correctly predicting entire genes. However, on test set **2** **FGENESH'S** sensitivity and specificity are both significantly **(15** % and **12** %, respectively) lower than on test set **1. DOUBLESCAN'S** sensitivity is significantly higher **(7** %) than that of **FGENESH** but its specificity is again much lower **(7** %) than that of **FGENESH. For** both test sets, **DOUBLESCAN** has a very low rate of missing genes (0 % and **1** %, respectively), whereas **FGENESH** misses out **0.4** % (test set **1,** corresponding to three genes) and **11** % (test set **2,** corresponding to **112** genes) of the annotated genes completely. DOUBLESCAN's high sensitivity **for** detecting annotated genes by predicting an exactly matching **or** overlapping

| | Test set 1 | | Test set 2 | |
|---|---|---|---|---|
| | DOUBLESCAN | FGENESH | DOUBLESCAN | FGENESH |
| **Gene** | | | | |
| Sensitivity | 0.80 | 0.82 | 0.74 | 0.67 |
| Specificity | 0.71 | 0.81 | 0.62 | 0.69 |
| Genes overlapping | 0.23 | 0.17 | 0.28 | 0.25 |
| Genes missing | 0 | 0.004 | 0.01 | 0.11 |
| Genes wrong | 0.06 | 0.02 | 0.10 | 0.06 |
| **Start Codon** | | | | |
| Sensitivity | 0.96 | 0.91 | 0.96 | 0.79 |
| Specificity | 0.87 | 0.94 | 0.81 | 0.86 |
| **Stop Codon** | | | | |
| Sensitivity | 0.96 | 0.96 | 0.93 | 0.83 |
| Specificity | 0.89 | 0.96 | 0.82 | 0.88 |
| **Exon** | | | | |
| Feature Level | | | | |
| Sensitivity | 0.93 | 0.94 | 0.91 | 0.82 |
| Specificity | 0.90 | 0.93 | 0.89 | 0.87 |
| Exons overlapping | 0.06 | 0.04 | 0.07 | 0.05 |
| Exons missing | 0.004 | 0.02 | 0.02 | 0.13 |
| Exons wrong | 0.04 | 0.03 | 0.04 | 0.08 |
| Nucleotide Level | | | | |
| Sensitivity | 0.996 | 0.987 | 0.98 | 0.87 |
| Specificity | 0.991 | 0.971 | 0.99 | 0.93 |

Table 5.5: Performance figures for DOUBLESCAN and FGENESH on the two *C. elegans* and *C. briggsae* test sets. The predictions by DOUBLESCAN were generated using the Stepping Stone algorithm. The table does not include the performance on the *C. elegans* and *C. briggsae* sequences separately as they are very similar. See Table 3.1 for the definitions of rows.

| | Combined test set | |
|---|---|---|
| | DOUBLESCAN | FGENESH |
| **Gene** | | |
| Sensitivity | 0.77 | 0.73 |
| Specificity | 0.65 | 0.74 |
| Genes overlapping | 0.26 | 0.22 |
| Genes missing | 0.005 | 0.07 |
| Genes wrong | 0.09 | 0.04 |
| **Start Codon** | | |
| Sensitivity | 0.96 | 0.84 |
| Specificity | 0.83 | 0.89 |
| **Stop Codon** | | |
| Sensitivity | 0.94 | 0.88 |
| Specificity | 0.85 | 0.91 |
| **Exon** | | |
| Feature Level | | |
| Sensitivity | 0.92 | 0.86 |
| Specificity | 0.89 | 0.89 |
| Exons overlapping | 0.07 | 0.05 |
| Exons missing | 0.01 | 0.09 |
| Exons wrong | 0.04 | 0.06 |
| Nucleotide Level | | |
| Sensitivity | 0.986 | 0.906 |
| Specificity | 0.993 | 0.944 |

Table 5.6: Performance figures for DOUBLESCAN and FGENESH on the combined *C. elegans* and *C. briggsae* test set (test set 1 and test set 2). The predictions by DOUBLESCAN were generated using the Stepping Stone algorithm. See Table 3.1 for the definitions of rows.

gene is counterbalanced by its higher rate of wrong genes (4 % higher on both test sets) with respect to FGENESH. As discussed in Section 5.3.1 and Section 5.3.2 and as shown in Table 5.3 and Table 5.4, these wrong genes are mainly short complete single exon genes which could be removed in a post-processing step.

DOUBLESCAN'S sensitivity for detecting start codons is significantly higher than that of FGENESH on both test sets (by 5 % and 17 %, respectively) and is the same for the two test sets, whereas its specificity is lower than that of FGENESH by 7 % and 5 %, respectively. For stop codons, DOUBLESCAN has almost the same sensitivity for both test sets (96 % and 93 %, respectively), whereas that of FGENESH decreases from 96 % on test set 1 to 83 % on test set 2. As for start codons, FGENESH has a higher specificity than DOUBLESCAN (7 % and 6 %, respectively), and both, DOUBLESCAN'S and FGENESH'S specificity decrease from test set 1 to test set 2.

At exon level, both DOUBLESCAN and FGENESH show a high sensitivity and specificity on test set 1 with FGENESH having a 3 % higher specificity. However, on test set 2 FGENESH'S sensitivity and specificity are significantly lower than on test set 1 (12 % and 6 %, respectively), whereas those of DOUBLESCAN almost stay the same (minus 2 % and minus 1 %, respectively). On test set 1, DOUBLESCAN misses almost no exons (0.4 %) and also FGENESH has a low rate of missing exons (2 %), but FGENESH'S rate rises to 13 % on test set 2, whereas that of DOUBLESCAN remains low (2 %). Note that also FGENESH'S rate of wrong exons changes from 3 % on test set 1 to 8 % on test set 2.

Table 5.6 shows the performance of DOUBLESCAN and FGENESH on the combined test set comprising test set 1 and 2.

## 5.4 Summary and discussion

Both DOUBLESCAN and PROJECTOR show very high sensitivity and specificity for predicting entire *C. elegans* and C. briggsae genes correctly and we therefore conclude that both methods, initially trained to analyse mouse and human DNA sequences, can be successfully adapted to analyse C. elegans and C. *briggsae* DNA sequences.

DOUBLESCAN has a higher sensitivity for genes, start and stop codons and exons and a significantly reduced rate of missing genes and exons compared to FGENESH, but shows a lower specificity for genes, start and stop codons. Given the fact that the training of DOUBLESCAN for C. elegans and C. briggsae involved no manual optimisation of the transition probabilities, the performance of **DOUBLESCAN** compares favorably with that of **FGENESH** and could probably be further improved.

When comparing the performances of **DOUBLESCAN** and **PROJECTOR** between the two test sets (see Table C.4 in Appendix C) and studying the sources of errors in detail (see Table 5.2), it is interesting ***to*** note that the main difference between the two test sets, namely the higher divergence of gene structures in the gene pairs of test set 2, and the difference in error rates, namely the highly increased rate of incorrectly predicted splice sites in test set **2,** may be linked.

One possible explanation is that test set **2** consists indeed of more diverged pairs of genes and that **DOUBLESCAN** and **PROJECTOR** have simply more difficulty predicting them correctly. However, another possible explanation is that the C. elegans and C. **briggsae** genes of test set 2 contain more mis-annotated splice sites and that the pairs of genes thus appear to be more diverged than they really are. This may be one of the reasons why the **BLASTN** matches covered only 95 % of the annotated exons (refer to Section **C.l** in Appendix C). In order to decide which of the two explanations holds, every gene predicted by **DOUBLESCAN** and **PROJECTOR** would have to be experimentally verified. However, one way for getting an indication **as** to which explanation is likely to be true, would be to verify whether the predicted genes are covered more by **BLASTN** hits than the annotated ones.

# Chapter 6

# DOUBLEBUILD

## 6.1 Introduction and motivation

The design of a pair HMM, i.e. its states and transitions, can be done rather quickly using pencil and paper. However, the implementation of the pair HMM into programming code and especially the implementation of alignment algorithms with which the pair HMM can be used to analyse sequences, can be a time consuming task.

[BD97] present a compiler called **DYNAMITE** with which a variety of pair HMMs can be defined and used with alignment algorithms to produce a prediction. The desired pair HMM is defined in a text file using the **DYNAMITE** language. This **DYNAMITE** file is then translated into C programming code using the **DYNAMITE** compiler and this C code must then be compiled with a standard C compiler before it *can* be executed. The **DYNAMITE** compiler shields the user from the underlying implementation into C programming code. This has the advantage of making the implementation of a pair HMM easy as the user only has to provide a short definition file using the **DYNAMITE** language. However, the introduction of an intermediate compiler makes it difficult for a user to understand and modify the underlying C source code.

My aim in programming **DOUBLESCAN** and **PROJECTOR** was to create a set of C++ classes, called **DOUBLEBUILD,** which can be used to define a variety of pair HMMs in a short time and which also provide sophisticated alignment algorithms so that the pair HMMs can be directly used for the analysis of data. In that respect, **DOUBLESCAN** and **PROJECTOR** can be seen as two sophisticated examples of what can be done with **DOUBLEBUILD.**

My main motivation for choosing C++ as a programming language was to use an object oriented language which provides all the features that I needed to realise projects such as

DOUBLESCAN and PROJECTOR. Using an object oriented language, there is no need to write an extra compiler to generate the final source code because the building blocks of the pair HMM correspond to the classes defined in DOUBLEBUILD with which the programmer can directly define and operate pair HMMs. This made things easier for me and hopefully also for the user who may not only wish to use the source code, but who may also wish to modify or extend it. C++ has the additional benefit that a freely available compiler (GNU compiler) exists and that an **ANSI** standard has already been defined which guarantees a high level of portability of the source code. All relevant data structures are provided by DOUBLEBUILD itself. In particular, the standard template library is not used.
This chapter first introduces the novel concept of special transitions and the concept of special emissions within pair HMMs and describe how they are implement within DOUBLEBUILD. It then presents the three main classes that form the foundation of DOUBLEBUILD, namely the **Sequence** class, the **PairhmmState** class and the **Pairhmm** class. Their description should also make clear how these classes interact. Finally, functions of special interest such **as** a variety of alignment algorithms are described.

## 6.2 Special transitions within DOUBLEBUILD

Special transitions are a new concept introduced in this dissertation. Special transitions within DOUBLEBUILD can be used to make any transition within a pair HMM dependent on position specific scores. These transitions are implemented in a way which conserves the probabilistic interpretation of the transition probabilities. The pair HMM underlying DOUBLESCAN and PROJECTOR is one example for a pair HMM which uses special transitions, ***see*** arrows with dots in Figure **2.4.** It uses special transitions to model the sequence signals around translation start sites and splice sites. These sequence signals are contained in a sequence interval which is too large to be easily incorporated into one state of the pair HMM, and the signal itself is too complex to be adequately modelled by the emission probabilities of a state. Before starting the gene prediction with algorithms such **as** the Hirschberg algorithm or the Stepping Stone algorithm, the two input sequences X and ***Y*** are first separately searched for potential translation start sites and splice sites by dedicated programs. **Each** potential translation start and splice site is assigned a score which is a measure of the likelihood for this site to be a true translation start or splice site. These sequence signal scores are stored for each sequence separately in its corresponding **Sequence** object. Once the two input sequences have been scanned for sequence signals, one of the algorithms is used to predict genes and align the two sequences. The sequence signal scores are used within the algorithm to modify the nominal values of the transition probabilities so that the transition has a high probability if it is supported by strong sequence signal scores at the given sequence positions.

Figure **6.1** shows a generic example with which the general concept of special transitions is elucidated in the following. Any alignment algorithm such as the Viterbi algorithm, Hirschberg algorithm or Stepping Stone algorithm derives the optimal state path according to the transition and emission probabilities encountered on the state paths through the pair HMM. The alignment algorithms work internally with scores which are derived from probabilities by **score** $= \log_2($**probability**$)$. The transition score for **a** transition from state **from** to state **to** at position **x_pos** in sequence X and position **y_pos** in sequence Y, see Figure 6.1, is calculated in the following way (description given in pseudo-code):

```
special_transition_score(from, to, X, x_pos, Y, y_pos) {
  if from $\rightarrow$ to special {
    return-score = score(special_transition_prob(from,to,X,x_pos,Y,y_pos))
  }
  else {
    if exists to' with from $\rightarrow$ to' special {
      returnscore = score(special_transition_prob(from,to,X,x_pos,Y,y_pos))
                    + score(scale_factor)
      where
      scale_factor = $\left( \frac{1 - \sum_{\substack{\text{to}' \\ \text{from} \rightarrow \text{to}' \text{ special}}} \text{special\_transition\_prob(from,to}',\text{X,x\_pos,Y,y\_pos)}}{\sum_{\substack{\text{to}' \\ \text{from} \rightarrow \text{to}' \text{ not special}}} \text{transition\_prob(from,to}')} \right)$
    }
    else {
      returnscore = transition_score(from, to)
    }
  }
  return(returnscore)
}
```

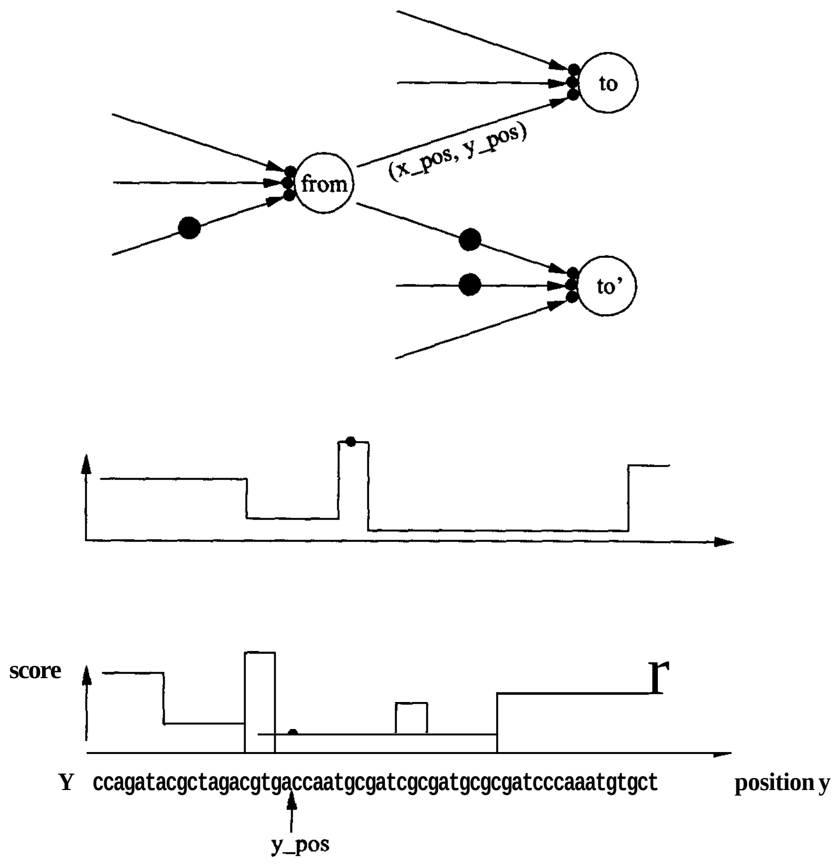


Figure **6.1:** Part of a pair HMM with special transitions. Special transitions correspond to the arrows marked with a big dot. Transitions belong to the state to which they are leading as indicated by a small dot between the tip of an arrow and its state. When calculating the probability for the transition from state `from` to state `to` which is not special, we have to take into account the position dependent values of all special transitions emerging from state `from` in order to ensure that the probabilities of all transitions emerging from state `from` always sum up to one. See the text for a detailed description.

If only non-special transitions are emerging from state from, the transition score **for** the transition from state from to state to at position x-pos and y-pos is equal to the nominal value of the transition score **for** going from state from to state to (transition_score(from, to)) which is independent of the positions in the input sequences. If the transition from state from to state to is special, the transition score depends on the positions in the input sequences and is equal to **score(special_transition_prob(from,** to, X, **x_pos,** Y, y_pos)), where the probability returned by **special_transition_prob(from,** to, X, **x_pos,** Y, y_pos) is calculated by:

```
special_transition_prob(from, to, X, x_pos, Y, y-pos) {
  return_prob = transition_prob(from, to)
  if from $\rightarrow$ to special {
    return_prob *= posterior_prob(prior, score)
    where
    prior = $\sqrt{\text{prior\_x} \cdot \text{prior\_y}}$ if to state of type EmitXY
            prior_x             if to state of type EmitX
            prior_y             if to state of type EmitY
    score = score-x + score-y   if to state of type EmitXY
            score_x             if to state of type EmitX
            Score-y             if to state of type EmitY
    prior_x = X.prior(from, to, x-pos)
    prior-y = Y.prior(from, to, y-pos)
    score_x = X.score(from, to, x-pos)
    score-y = Y.score(from, to, y-pos)
    and
    posteriorqrob(prior, score) = $\left(\frac{\text{prior}\cdot 2^{\text{score}}}{\text{prior}\cdot 2^{\text{score}}+1-\text{prior}}\right)$
  }
}
```

If the transition from state from to state to is not special, but if there **are** special transitions emerging from state from **(as** is shown in the example in Figure 6.1), the nominal value of the non special transition from state from to state to is adjusted so that the sum of all transition probabilities emerging from state **from** at any pair of sequence positions **(xpos , y-pos)** remains one. This is done by calculating a scaling factor **(scalefactor)** by which the value of the nominal transition probability is multiplied.

Generally, the priors for the special transitions may depend on the position within the sequence. To name an example, the value of the prior for the special transition between the match exon and the emit *x 5'* splice site phase 0 state in the pair HMM underlying DOUBLESCAN and PROJECTOR depends on whether this is a consensus GT or a non-consensus GC splice site (see Table D.1 and Figure 2.4).

The details of how special transitions are implemented into the C++ classes of DOUBLEBUILD are described in Section 6.4.

## 6.3 Special emissions within DOUBLEBUILD

Not only transition probabilities, but also emission probabilities can be made dependent on the sequence positions. States whose emission probabilities depend on the positions in the input sequences are called states with special emissions. In PROJECTOR (see Chapter 3), special emissions are used to implement constraints into the calculation of the optimal state path. Only those state paths are considered in the calculation of the optimal state path which reproduce the known annotation of one of the two input sequences. This way we can project the known genes of one input sequence onto the other input DNA sequence of yet unknown annotation. PROJECTOR is just one of many possible applications of special emissions within pair HMMs. The following paragraph illustrates the generality of the concept of special emissions.

Again, the algorithms internally employ scores, the logarithm of the probabilities, in order to avoid the numerical difficulties which arise when dealing with small probabilities. However, as scores and probabilities have a one-to-one correspondence, they can be easily converted into each other. For any given state **this** in a pair HMM, see Figure 6.2, the emission score is calculated in the following way:

```
special-emissionscore(this, X, x-pos, Y, y_pos) {
   return-score = emission-score(this, X, x-pos, y, y_pos)
   if this state has special emissions {
      returnscore += score
```

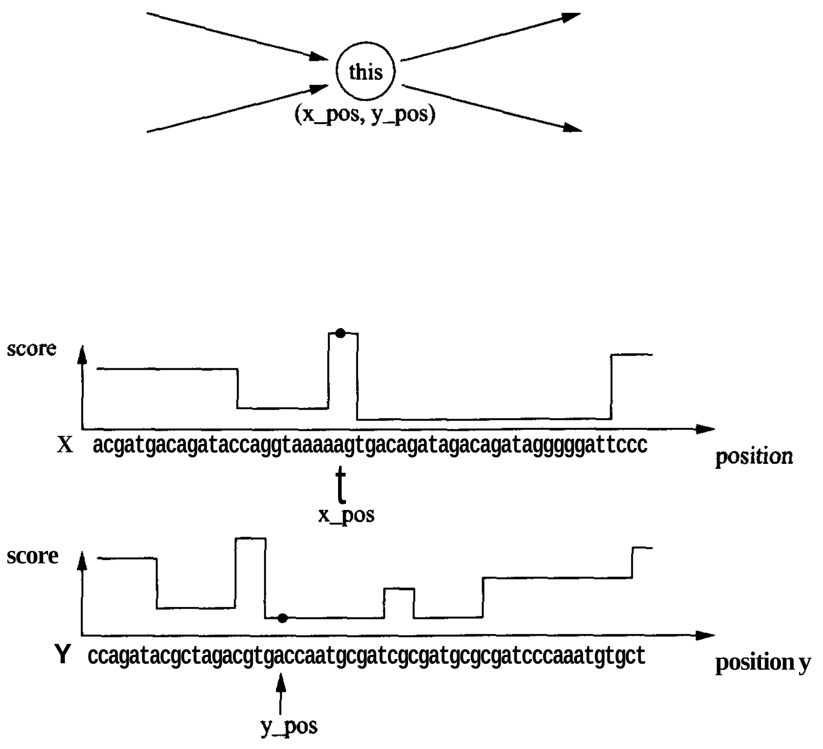

this
(x_pos, y_pos)
score
X acgatgacagataccaggtaaaaagtgacagatagacagatagggggattccc
position
x_pos
score
Y ccagatacgctagacgtgaccaatgcgatcgcgatgcgcgatcccaaatgtgct
position y
y_pos

Figure 6.2: State of a pair HMM with special emissions. The emission probability at positions (xpos, y_pos) not only depends on the letters read at these sequence positions, but also on the score at position x-pos in input sequence X and on the score at position ypos in input sequence Y. See the text for a detailed description.

```
    where
    score = score_x + score-y if this state of type EmitXY
            score_x           if this state of type EmitX
            score_y           if this state of type EmitY
    and
    scores = X.score(this, xpos)
    score-y = Y.score(this, y_pos)
  }
  return(return_score)
}
```

If **this** state does not have special emissions, the emission score (**emission_score(this, X, x-pos, Y, y-pos))** only depends on the ***letters*** read from the input sequences at the given sequence position **(x-pos, y_pos)**, but ***not the sequence positions.*** If **this** state has special emissions, the nominal value of the emission score **(emissionscore(this, X, x-pos, Y, ypos))** which only depends on the letters read is modified by a score which depends on the scores at the given sequence positions **(score-x** and **score-y).**

## 6.4 The main classes

The three main classes of **DOUBLEBUILD** are the **Sequence,** the **PairhmmState** and the **Pairhmm** class. Each of the three classes has a set of private variables whose values characterise every instance of each class.
We first introduce the private variables of each class in order to illustrate how the different classes interact within a pair HMM. The private variables are more important for the understanding of the concept of **DOUBLEBUILD** than the set of public functions by which the values of the private variables are accessed.

### 6.4.1 The Pairhmm class

**A Pairhmm** object knows the number of states it consists of **(int number-ofstates)** and has an array with pointers to each of its states (**Pairhmm_State* model).** It has private variables for storing a state path and provides private functions which are used as the building blocks of public functions such as the Stepping Stone algorithm and the Hirschberg algorithm.

### 6.4.2 The Pairhmm-State class

**PairhmmState** objects constitute the building blocks of a pair HMM and interact with **Sequence** objects. The definition of the **PairhmmState** class was motivated by the idea that each state should know about itself and its direct neighbours within the pair HMM (a direct neighbour being a state that can be reached within a single transition).
In its simplest variant, a **PairhmmState** object knows:

- **int number-ofstate** its number within the **Pairhmm**
- **int alphabet** the alphabet of letters it reads from an input sequence

- **int number_of_letters_to_read** the number of letters it **reads** from an input sequence
- **array<Labelseq> labels_of_letter_to_read** the labels it assigns to the letters **read**
- **array<Phase> phases_of_letters_to_read** the phases it assigns to the letters
- **State_type state_type** its own state type, e.g. if it is an **EmitXY**, **EmitX** or **EmitY** state or some other type of state
- **array<Prob> emissionprobs** the array of its emission probabilities
- **array<Prob> transitionprobs** the array of transition probabilities to states which **can** be reached from this state
- **int number-ofstates** the number of states in the **pair HMM** to which this state belongs
- **int number-ofnext-states** number of states which can be reached from this state
- **array<int> numbers-ofnext-states** array of the numbers of the states which *can* be reached from this state
- **int number-of-previous-states** number of states which have a transition to this state
- **array<int> numbers-of-previousstates** array of the numbers of the states which have a transition to this state

### PairhmmState objects with special emissions

If a state has special emissions, its emission probabilities depend both on the letters it **reads** and position specific sequence scores. This concept is .employed in **PROJECTOR** to predict an annotation for one of the two input sequences while keeping that of the other sequence fixed. Suppose we are dealing with the match exon state of **PROJECTOR** and are keeping the annotation of sequence $X$ fixed, see Figure **2.4** and Figure **B.3** in Appendix B. We want the match exon state to have non-zero emission probabilities only **for** letters whose annotation matches the labels and phases of sequence $X$. For a given position in sequence $X$, the emission probability within the match exon state is calculated by requesting the corresponding score for that position from object **Sequence X.** Instead of storing the information for every state with special emission probabilities and all sequence positions in **Sequence** X, the information is only stored for a few states from which the information of the remaining states with special emissions can be derived. The match exon state derives its information on the position specific emission probabilities of sequence X by requesting that of state emit x exon state as it suffices to know where this state is allowed in order to know where the match exon state is allowed. The private variables used to implement special emissions are:

- int special-emission indicates if this state has special emissions or not. Its value is one if this state has special emissions and zero otherwise.
- int number_of_child_state_x is the number of the state under which the position specific sequence scores for this state are stored in Sequence X, in the above example this would be the number of the state emit x exon
- int number_of_child_state_y is the number of the state under which the position specific sequence scores of this state are stored in Sequence Y (In the above example, special emission probabilities within the match exon state are only used for the position specific scores of sequence X as only the annotation of sequence X is kept fixed. In this case, int number-of_child_state_y would be set to zero.)

### PairhmmState objects with special transitions

If a state has special transitions, the probability of one or several transitions leading into the state depends on position specific scores within the input sequences. Both DOUBLESCAN and PROJECTOR use special transitions to improve the detection of splice sites and translation start sites. Dedicated programs score potential translation start sites and splice sites within the two input sequences separately. These position specific scores are then used within the pair HMM to modify the nominal values of the transition probabilities. If both sequences have strong signals for being 5' splice sites at the given sequence positions, the probability of transferring from the match exon to the match 5' splice site state is high and small otherwise, see Figure 2.4.

As for special emissions, the information about the position specific scores as well as the priors is stored within the Sequence objects of the two input sequences, Sequence X and Sequence Y. And as for special emissions, also special transitions are implemented in a memory efficient way by storing information only for a minimum of transitions from which the information of the remaining special transitions can be easily derived. The probability of the special transition from the match exon to the match 5' splice site phase 1 state, see Figure 2.4 and Figure B.3 in Appendix B, is derived from that of the special transition from the match exon

to the emit x 5' splice site phase 0 state for input sequence X and from that of the special transition from the match exon to the emit y 5' splice site phase 0 state for input sequence Y by considering the offset of one base pair. For long sequences, these tricks save a considerable amount of memory.
The private variables dealing with special transitions are:

- **int special** indicates if this state has special transitions or not. Its value is one if this state has special transitions and zero otherwise.
- **int number_of_special_transitions_to_previous_states** is the number of special transition leading into this state
- **array<int> special_flags_of_transitions_to_previous_states** one-dimensional array indicating for each transition leading into this state if it is special (array element has value one) or not (array element has value zero)
- **array<int> numbers_of_from_child_states_x_previous** one-dimensional array indicating for each transition leading into this state the number of the 'from' state to be used for deriving the special transition score for sequence X, if the transition is special. If an alternative transition is to be used, this number is the number of the 'from' state of the alternative transition. In the above example in which we are dealing with the match 5' splice site phase 1 state, the array element for the transition from the match exon state to the match 5' splice site phase 1 state is the number of the match exon state as this is the 'from' state of the alternative transition from the match exon state to the emit x 5' splice site phase 0 state which is used for deriving the position specific score of sequence X.
- **array<int> numbers_of_to_child_states_x_previous** onedimensional array indicating for each transition leading into this state the number of the 'to' state to be used for deriving the special transition score for sequence $X$, if the transition is special. If an alternative transition is to be used, this number is the number of the 'to' state of the alternative transition. In the above example in which we are dealing with the match 5' splice site phase 1 state, the array element for the transition from the match exon state to the match 5' splice site phase 1 state is the number of the emit x 5' splice site phase 0 state as this is the 'to' state of the alternative transition from the match exon state to the emit x 5' splice site phase 0 state which is used for deriving the position specific

score of sequence $X$.

- **array<int> offset_for_previous_states_x** onedimensional array indicating for each transition leading into this state the offset in base pairs to be used if the transition is special. In the above example in which we are dealing with the ***match 5' splice site phase 1*** state, the array element for the transition from the ***match exon*** state to the ***match 5' splice site phase 1*** state is one as this is the offset between the alternative transition from the ***match exon*** state to the ***emit x 5' splice site phase 0*** state which is used when dealing with sequence $X$ and the transition from the ***match exon*** state to the ***match 5' splice site phase 1*** state.

- **array<int> numbers-of_from_child_states_y_previous**

  same as **array<int> numbers-of_from_child_states_x_previous**, but for dealing with input sequence $Y$

- **array<int> numbers-of_to_child_states_y_previous**

  same as **array<int> numbers-of_to_child_states_x_previous**, but for dealing with input sequence $Y$

- **array<int> offset_for_previous_states_y**

  same as **array<int> offsetforprevious-states3** , but for dealing with input sequence $Y$

### 6.4.3 The Sequence class

In its most fundamental form, a **Sequence** object consists **of:**

- **Letter* sequence** the sequence of symbolic letters
- **int length_ofsequence** the length of the sequence
- **Sequence-Type sequence-type** the type of the sequence
- **int orientation** the orientation of the sequence
- **int start_position** and **end-position,** the start and end positions of the sequence
- **char* i d** the name of the sequence

### **`Sequence` objects for use with special emissions**

If the input sequence is to be used with a pair HMM whose states have special emissions, the private variables of the **`Sequence`** object have to be set up accordingly, refer to Section **6.4.2** to see how special emissions are dealt with in the **`Pairhmm-State`** class.
The private variables which store information on special emissions are:

- **`int number-ofspecial-emissions`** the number of states for which special emissions have to be implemented. In general, this is not the number of states in the pair HMM that have special emissions, but a smaller number of states from which the special emissions of all states with special emissions are derived. In the case of PROJECTOR, **`int number-ofspecial-emissions`** is **22,** but the number of states with special emissions is **52** (all non silent states of the pair HMM).
- **`array<int> indices-ofspecial-emissions`** one-dimensional array which assigns an index to every implemented state with special emissions. This index is used as the first of two indices (the second index indicating the position within the sequence) for the arrays **`array<Prob> posterior_probs_for_special_emissions`** and **`array<Score> scores--for_special_emissions`** to look up the value of the corresponding posterior probability or score.
- **`array<Prob> posterior_probs_for_special_emissions`** two-dimensional array with the posterior probabilities for all implemented states with special emissions (first index) and all sequence positions (second index), if the posterior probabilities are to be used instead of scores
- **`array<Score> scores_for-special_emissions`** two-dimensional array with the scores for all implemented states with special emissions (first index) and all sequence positions (second index), if scores are to be used instead of posterior probabilities

### **`Sequence` objects for use with special transitions**

If the input sequence is to be used with a pair HMM which has special transitions, the private variables of the **`Sequence`** object have to provide the information needed by the states which have special transitions, see Section **6.4.2.**
The private variables which store information on special transitions are:

- `int number-of special-transitions` the number of special transitions which are implemented. As for special emissions, this is generally not the number of special transitions in the pair HMM, but a smaller number of special transition from which the information of all remaining special transitions can be derived, see Section **6.4.2** for more information. For DOUBLESCAN, the number of special transitions is **25,** but they derive their information from only six special transitions (three for ***each*** sequence) and the information on only these three transitions has to be provided by the Sequence object of each input sequence.
- `array<int> indices-of -special-transitions` two-dimensional array which assigns an index to every implemented special transition. This index is used to refer to the transition within other arrays (arrays `array<Prob> posterior_probs_for_special_-transitions`, `array<Prob> priors-of special_transitions` and `array<Score>scores--for-special-transitions`). The fist index of this array is the state number of the 'from' state of the special transition and the second index the state number of the 'to' state of the special transition and the return value is the index which is to be used to refer to that transition within the previously mentioned arrays.
- `array<Prob> posterior_probs_for_special_transitions` two-dimensional array with the posterior probabilities for all implemented special transitions (first index) and all sequence positions (second index), if the special transitions are to be used with posterior probabilities rather than with priors and scores.
- `array<Prob> priorsnf -special-transitions` two-dimensional array with the priors for **all** implemented special transitions (first index) and all sequence positions (second index), if the special transitions are to be used with priors and scores rather than with posterior probabilities. Generally, the priors for special transitions can be dependent on the position within the sequence. To name an example, the value of the prior for the special transition between the match **exon** and the emit **x 5'** splice site phase *0* state in the pair HMM underlying DOUBLESCAN and PROJECTOR depends on whether this is a consensus GT or a non-consensus GC splice site (see Table D.1 and Figure **2.4).**
- `array<Score> scores_for_special_transitions` two-dimensional array with the scores for **all** implemented special transitions (first index) and all sequence positions (second index), if the special transitions are to be used with priors and scores rather than with posterior probabilities.

## 6.5 Alignment algorithms

The public functions of the **Pairhmm** class provide several algorithms by which the pair HMM can be used to analyse data.

### 6.5.1 The Viterbi algorithm

The Viterbi algorithm in its original form [Vit67] was mainly implemented to verify the other algorithms. It calculates the optimally scoring state path for the given pair HMM and two input sequences.

### 6.5.2 The Hirschberg algorithm

The Hirschberg algorithm, see Section **1.5.2** and Figure **1.8,** takes two **Sequence** objects and an integer value **(int max-area) as** the input and calculates the optimally scoring state path for the given pair HMM and the two input sequences in linear memory and quadratic time dependence. The value of **max-area** indicates the maximum size (in the two sequence dimensions) of sub-matrices which are directly calculated by the Viterbi algorithm. The smaller this value, the more iterations have to be performed within the Hirschberg algorithm before ***each*** sub-matrix is small enough to be calculated using the Viterbi algorithm.
As the Hirschberg algorithm works internally with two pair HMMs, the original pair HMM and the mirrored model of the original pair HMM, the mirrored version has to be provided **as an**-other input parameter. It is created by the public **Pairhmm** function **int get_mirrored(void)** before the Hirschberg algorithm is called.

### 6.5.3 The Stepping Stone algorithm

The Stepping Stone algorithm, see Section **2.4** and Figure **2.5,** takes two **Sequence** objects and a list of $(x, y)$ coordinates **as** input which are simultaneously ordered in their $x$ and y coordinates and derives the highest scoring state path in the thus restricted sub-space of the Viterbi matrix. The input values, **int x_margin** and **int y_margin**, indicate the size of the overlap in the two sequence dimensions that two adjacent sub-matrices shall have. The input parameter **int max-area** ***is*** the same **as for** the the Hirschberg algorithm. If a sub-matrix is smaller than its value, it is directly calculated by the Viterbi algorithm during the traceback process. Otherwise, it is calculated using a special variant of the Hirschberg algorithm.

As opposed to the previously described implementation of the Hirschberg algorithm whose calculation is initialised by the constraint that every state path has to start in the begin state $s = 0$ at sequence positions $(x, y) = (1, 1)$, the calculation within this variant of the Hirschberg algorithm is initialised by a small sub-matrix of already pre-calculated values (this sub-matrix corresponds to the volume by which the current sub-matrix overlaps its lower left neighbouring sub-matrix, see Figure 2.5). The internal use of the Hirschberg algorithm is the reason why the mirrored model of the original pair HMM has to be provided as input to the Stepping Stone algorithm.

# Chapter 7

# Conclusion

Since the work within this dissertation was started in January 2000, the initial sequencing of the human genome has been completed and the sequencing of the related mouse genome has been started and is now close to completion. Other pairs of evolutionarily related genomes such as the nematodes C. elegans and C. *briggsae* or the fruit fly ***Drosophila** melanogaster* and the mosquito ***Anopheles gambiae*** are emerging. The availability of these data opens the new opportunity to understand these genomes by comparing evolutionarily related genomes to each other. Comparative studies will greatly increase our understanding of genomes as both, regions which are conserved between the genomes and those which are not conserved will teach us something important.

Methods for comparative ab initio gene prediction have only started to emerge in 2000. The two novel methods presented in this dissertation are among the first to solve the gene prediction and sequence alignment problem simultaneously. They make use of the different types of conservation within two related DNA sequences in order to predict protein coding genes.

One method, **DOUBLESCAN,** simultaneously predicts the genes as well as the alignment of two related input DNA sequences by only knowing their sequence of A, C, G, T letters. This approach from first principles makes use of only a few and very basic assumptions on the general structure of eukaryotic genes and the ways in which two similar genes are related. It is capable of predicting partial, single and multiple genes as well as pairs of genes which are related by events of exon-fusion or exon-splitting. The underlying probabilistic pair hidden Markov model is parametrised in a simple way, and all parameters have a clear interpretation. The results presented in this dissertation show that **DOUBLESCAN** can be successfully used to predict mouse and human genes simultaneously and that its parameters can be easily adapted to analyse other pairs of genomes, as demonstrated in the analysis of C. *elegans* and C. *briggsae* sequence pairs. DOUBLESCAN has a high sensitivity for predicting entire known genes correctly and captures the long range constraints imposed by the similar exon-intron structures of related genes well in its comparative model. This is reflected in the performance of DOUBLESCAN relative to that of one of the reference non-comparative *ab* initio gene prediction methods, GENSCAN, which increases progressively when going from nucleotide (fine scale) to gene level (large scale).

The second method, PROJECTOR, can be used to find the gene structures of one DNA sequence when those of a related DNA sequence are already known. Similarly to DOUBLESCAN, PROJECTOR is capable of dealing with partial, single and multiple genes as well as genes which are related by events of exon-fusion or exon-splitting. It was presented here for the comparative prediction of mouse and human as well as C. elegans and C. *briggsae* genes. It is the first gene prediction method which makes use of homology directly at DNA level and also simultaneously predicts genes and an alignment. As it makes use of gene structure information, it should have a superior sensitivity especially for detecting remotely related genes with respect to gene prediction methods which employ protein homology information.

Both methods not only detect genes, but also comparatively predict conserved subsequences within their genomic context. This should highlight novel regulatory elements which cannot be reliably predicted by non-comparative methods which have a very low specificity for detecting these typically short subsequences. For example, PROJECTOR can be used with one DNA sequence containing known genes to find both, the related genes and conserved subsequences in another related DNA sequence of yet unknown annotation. Both, DOUBLESCAN and PROJECTOR are the first methods to comparatively predict conserved subsequences in their genomic context.

The two above methods not only introduce new theoretical concepts for comparatively predicting protein coding genes, but have also been implemented into efficient computer programs so that they can be applied to realistic large scale problems. The latter was achieved by introducing a new algorithm, the Stepping Stone algorithm, whose memory and time requirements both scale essentially linearly with the length of the input sequence. The predictions generated by the Stepping Stone algorithm were compared to those of the exact Hirschberg algorithm and shown to provide a very good practical solution for the analysis of long sequences.

As **DOUBLESCAN** and **PROJECTOR** both require that the pairs of input sequences exhibit similarities in collinearity, the application of **DOUBLESCAN** and **PROJECTOR** to entire genomes requires more care in the preparation of the input sequences than non-comparative methods which can essentially be given any genomic sequence **as** input. The genomes to be analysed with **DOUBLESCAN** and **PROJECTOR** first have to be partitioned into pairs of sequences in which sequence similarities appear in collinearity using simple alignment programs like **BLASTN** [AGM+90] or **DOTTER** [SD96]. The maximal length of these sequence pairs will vary not only between different pairs of genomes, but also within one pair of genomes and will depend on the local level of divergence. Concerning the performance, it is a **priori** not clear how the performance of **DOUBLESCAN** and **PROJECTOR** on multi gene sequences will compare to that on single gene sequences. The results in [GAA+00b] show that the specificity of **GENSCAN** on semi-artificial long genomic sequences is significantly lower than on single gene sequences while its sensitivity remains essentially unchanged, whereas the specificity of similarity based programs like GENEWISE and PROCRUSTES is not significantly altered and depends mainly on the strength of the similarity to a homologous protein. The change in performance of comparative ab **initio** gene prediction methods when analysing multi gene instead of single gene sequences has so far only been investigated in [WGJMOG01]. The authors evaluate **GENSCAN** and their comparative ab **initio** gene prediction program SGP-1 on one single gene set and several multi gene sets which are derived from different regions of two genomes. Whereas GENSCAN's specificity generally decreases when analysing the multi gene sets, that of **SGP-1** increases on some of the multi gene sets. And whereas GENSCAN's sensitivity only slightly decreases when analysing the different multi gene sets, that of **SGP-1** shows both positive and negative changes with a higher amplitude than **DOUBLESCAN.** The authors conclude that the performance of their comparative ab **initio** method depends more on the level of conservation between the regions of the two genomes from which the sequences are derived than the single or multi gene nature of the sequences. Although the behavior of DOUBLESCAN's or **PROJECTOR'S** performance on multi gene sequences remains to be investigated, we expect them to behave similarly.

# Bibliography

[ABK96] F. J. Ayala, E. Barrio, and J. Kwiatowski. Molecular clock or erratic evolution? A tale of two genes. *Proceedings of the National Academy of Sciences of the USA*, 93:11729–11734, 1996.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[BD97] E. Birney and R. Durbin. Dynamite: a flexible code generating language for dynamic programming methods used in sequence comparison. In Gaasterland et al. [GKK+97], pages 56–64.

[BD00] E. Birney and R. Durbin. Using genewise in the drosophila annotation experiment. *Genome Research*, 10:547–548, 2000.

[BEK91] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220(1):49–65, 1991.

[Ber89] G. Bernardi. The isochore organization of the human genome. *Annual Review of Genetics*, 23:637–661, 1989.

[BFV+97] J. G. Baldwin, L. M. Frisse, J. T. Vida, C. D. Eddleman, and W. K. Thomas. An evolutionary framework for the study of developmental evolution in a set of nematodes related to *Caenorhabditis elegans*. *Molecular Phylogenetics and Evolution*, 8:249–259, 1997.

[BG96] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.

[BH00] V. Bafna and D. H. Huson. The conserved exon method for gene finding. In R. Altman et al., editor, *Proceedings of the Eights International Conference on Intelligent Systems for Molecular Biology*, pages 3–12, Menlo Park, CA, 2000. AAAI Press.

[Bir87] A. Bird. CpG islands as gene markers in the vertebrate nucleus. *Trends in Genetics*, 3:342–347, 1987.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, UK, 1995.

[BK97] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[BP66] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[BPM$^+$00] S. Batzoglou, L. Pachter, J.P. Mesirov, B. Berger, and E.S. Lander. Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Research*, 10:950–958, 2000.

[BSS00] M. Burset, I. A. Seledtsov, and V. V. Solovyev. Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Research*, 28:4364–4375, 2000.

[Bur97] C. Burge. *Identification of genes in human genomic DNA.* PhD thesis, Stanford University, USA, 1997.

[CA96] J.-M. Claverie and S. Audic. The statistical significance of nucleotide position-weigth matrix matches. *Computer Applications in the Biosciences*, 12:431–439, 1996.

[CB86] J.-M. Claverie and K. Bougueleret. Heuristic informational analysis of sequences. *Nucleic Acids Research*, 14:179–196, 1986.

[Cla97] J.-M. Claverie. Computational methods for the identification of genes in vertebrate genomic sequences. *Human Molecular Genetics*, 6:1735–1744, 1997.

[Con01] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[Con02] Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome, accepted for publication. *Nature*, 2002.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, UK, 1998.

[DMG95] L. Duret, D. Mouchiroud, and C. Gautier. Statistical analysis of vertebrate sequences reveals that long genes are scarce in CG-rich isochores. *Journal of Molecular Evolution*, 40:308–317, 1995.

[DS94] S. Dong and D. B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23:540–551, 1994.

[ea00] M. D. Adams et. al. The genome sequence of *Drosophila melanogaster*. *Science*, 24:2185–2195, 2000.

[Ens] Ensembl Webpage at http://www.ensembl.org/.

[eSC98] The *C. elegans* Sequencing Consortium. Genome Sequence of the Nematode *C. elegans*: A Platform for Investigating. *Science*, 11:2012–2018, 1998.

[FLS92] R. Farber, A. Lapedes, and K. Sirotkin. Determintation of eukaryotic protein coding regions using neural networks and information theory. *Journal of Molecular Biology*, 226:471–479, 1992.

[FT92] J. W. Fickett and C. S. Tung. Assessment of protein coding measures. *Nucleic Acids Research*, 20:6441–6450, 1992.

[GAA$^+$00a] R. Guigó, 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.

[GAA$^+$00b] R. Guigó, 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.

[GF95] R. Guigó and J. W. Fickett. Distinctive sequence features in protein coding genic non-coding, and intergenic human DNA. *Journal of Molecular Biology*, 13:51–60, 1995.

[GKDS92] R. Guigó, S. Knudsen, N. Drake, and T. Smith. Prediction of gene structure. *Journal of Molecular Biology*, 226:141–157, 1992.

[GKK+97] T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia, editors. *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, 1997. AAAI Press.

[GMP96] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Sciences of the USA*, 93:9061–9066, 1996.

[HHM90] X. Huang, R. C. Hardison, and W. Miller. A space-efficient algorithm for local similarities. *Computer Applications in the Biosciences*, 6:373–381, 1990.

[Hir75] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18:341–343, 1975.

[How71] R. A. Howard. *Dynamic Probabilistic Systems Volume II: Semi-Markov and Decision Processes.* John Wiley & Sons, New York, 1971.

[HRS+87] N. H. Hopkins, J. W. Roberts, J. A. Steitz, J. D. Watson, and A. M. Weiner. *Molecular Biology of the Gene.* Benjamin Cummings, 4th edition, 1987.

[Ini00] The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 14:796–815, 2000.

[JBD99] N. Jareborg, E. Birney, and R. Durbin. Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Research*, 9:815–824, 1999.

[JMCB90] I. Sauvaget J.-M. Claverie and K. Bougueleret. K-tuple frequency analysis: from intron/exon discrimination to t-cell epitope mapping. *Methods in Enzymology*, 183:237–252, 1990.

[KAA$^+$93] B. P. Kennedy, E. J. Aamodt, F. L. Allen, M. A. Chung, M. F.P. Heschl, and J. D. McGhee. The gut esterase gene (ges-1) from the nematodes *Caenorhabditis elegans* and *Caenorhabditis briggsae*. *Journal of Molecular Biology*, 229:890–908, 1993.

[KFDB01] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 1:1–9, 2001.

[KHRE96] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. F. Smith, editors, *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 134–142, Menlo Park, CA, 1996. AAAI Press.

[Koz81] M. Kozak. Possible role of flanking nucleotides in recognition of the AUG initiator codon by eukaryotic ribosomes. *Nucleic Acids Research*, 9:5233–5252, 1981.

[Kro97] A. Krogh. Two methods for improving performance of a HMM and their application for gene finding. In Gaasterland et al. [GKK$^+$97], pages 179–186.

[KZ00] W.J. Kent and A.M. Zahler. Conservation, regulation, synteny, and introns in a large-scale *C. briggsae–C. elegans* genomic alignment. *Genome Research*, 10:1115–1125, 2000.

[LD01] A. Levine and R. Durbin. (unpublished data). 2001.

[MBD97] S. D. Martinelli, C. G. Brown, and R. Durbin. Gene expression and development databases for *C. elegans*. *Seminars in Cell and Developmental Biology*, 8:459–467, 1997.

[McL92] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York, USA, 1992.

[MD02] I. M. Meyer and R. Durbin. Comparative *ab initio* prediction of gene structures using pair HMMs. *Bioinformatics*, 18:1309–1318, 2002.

[NGM01] P. S. Novichkov, M. S. Gelfand, and A. A. Mironov. Gene recognition in eukaryotic DNA by comparison of genomic sequences. *Bioinformatics*, 17:1011–1018, 2001.

[OMM+97] J. C. Oeltjen, T. M. Malley, D. M. Muzny, W. Miller, R. A. Gibbs, and J. W. Belmont. Large-scale comparative sequence analysis of the human and the murine Bruton's tyrosine kinase loci reveals conserved regulatory domains. *Genome Research*, 7:315–329, 1997.

[Pac99] L. Pachter. *Domino Tiling, Gene Recognition, and Mice.* PhD thesis, Massachusetts Institute of Technology, USA, 1999.

[PAC01] L. Pachter, M. Alexandersson, and S. Cawley. Applications of generalized pair hidden markov models to alignment and gene finding problems. In *Proceedings of the Fifth Annual International Conference on Computational Molecular Biology RECOMB 2001*, 2001.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[RHH+00] M. G. Reese, G. Hartzell, N. L. Harris, U. Ohler, J. F. Abril, and S. E. Lewis. Genome annotation assessment. *Genome Research*, 10:483–501, 2000.

[SD94] E. Sonnhammer and R. Durbin. A workbench for large scale sequence homology analysis. *Computer Applications in the Biosciences*, 10:301–307, 1994.

[SD96] E. Sonnhammer and R. Durbin. A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. *Gene*, 167:GC1–10, 1996.

[SDFH97] S. Salzberg, A. Delcher, K. Fasman, and J. Henderson. A decision tree system for finding genes in DNA. *Technical report John Hopkins University*, 1997.

[SH94] G. D. Stormo and D. Haussler. Optimally parsing a sequence into different classes based on multiple types of evidence. In R. Altman, D. Brutlag, P. Karp, R. Lathrop, and D. Searls, editors, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 369–375, Menlo Park, CA, 1994. AAAI Press.

[SMS+98] G. Stoesser, M. Moseley, J. Sleep, M. McGowran, M. Garcia-Pator, and P. Sterk. The EMBL nucleotide sequence database. *Nucleic Acids Research*, 26:8–15, 1998.

[SS93] E. E. Snyder and G. D. Stormo. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucleic Acids Research*, 21:607–613, 1993.

[SS00] A. A. Salamov and V. V. Solovyev. Ab Initio Gene Finding in *Drosophila* Genomic DNA. *Bioinformatics*, 10:516–522, 2000.

[SSD+01] L. Stein, P. Sternberg, R. Durbin, J. Thierry-Mieg, and J. Spieth. Wormbase: network access to the genome and biology of *Caenorhabditis elegans*. *Nucleic Acids Research*, 29:82–86, 2001.

[SSL96] V. V. Solovyev, A. A. Salamov, and C. B. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Research*, 22:5156–5163, 1996.

[UM91] E. C. Uberbacher and R. J. Mural. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proceedings of the National Academy of Sciences of the USA*, 88:11261–11265, 1991.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, pages 260–269, 1967.

[VPS98] D. A. Voronov, Y. V. Panchin, and S. E. Spiridonov. Nematode phylogeny and embryology. *Nature*, 395:28, 1998.

[WGJMOG01] T. Wiehe, S. Gebauer-Jung, T. Mitchell-Olds, and R. Guigó. SGP-1: Prediction and validation of homologous genes bases on sequence alignments. *Genome Research*, 11:1574–1583, 2001.

[WGM00] T. Wiehe, R. Guigó, and W. Miller. Genome sequence comparisons: Hurdles in the fast lane to functional genomics. *Briefings in Bioinformatics*, 1:381–388, 2000.

[Wor] Wormbase Webpage at http://www.wormbase.org/.

[YLB01] R. Yeh, L. P. Lim, and C. B. Burge. Computational inference of homologous gene structures in the human genome. *Genome Research*, 11:803–816, 2001.

[Zha97] M. Q. Zhang. Identification of protein coding regions in the human genome by quadratic discriminant analysis. *Proceedings of the National Academy of Sciences of the USA*, 94:565–568, 1997.

# Appendix A

# Mouse human training and test sets

## A.1 The training set

The training set was derived from the data set in [JBD99] which consists of pairs of orthologous mouse and human **DNA** sequences which each comprise exactly one complete gene, i.e. comprising all protein coding parts of the gene. The data set in [JBD99] was derived from the **EMBL** nucleotide database (release **55)** [SMS+98] by searching human **DNA** sequences for orthologous mouse **DNA** sequences using **BLASTN** [AGM+90] and by then manually inspecting the BLASTN results with MSPCRUNCH and BLIXEM [SD94]. We discarded those sequence pairs from the data set in [JBD99] which had non-consensus start or stop codons or in-frame stop codons. The remaining sequence pairs were used *to* derive the emission probabilities according to Section **2.3.** The **36** pairs of genes with consensus GT–AG splice sites were used to train the transition probabilities of the pair HMM by manually optimising the performance, see Section **2.3.** This data set is referred to as the mouse human training set.

Table **A.1** shows the basic statistics of this training set. The human and mouse genome can be divided into long GC isochores according to their GC contents and the density of genes is correlated with the GC contents. The sequences of the training set are not evenly distributed into the four GC contents intervals as defined by [Ber89] as can be seen in Table **A.2.** Within each pair, the GC contents of the two **DNA** sequences are well correlated, see Table **A.3.** Table **A.4** shows the levels of conservation of gene structures within the pairs of the training set. For the majority of pairs **(61** %), the genes in a pair have the same number of exons, but a different coding length. **36 %** of the pairs consist of evolutionarily well conserved genes which have both the same number of exons and the same coding length and only **3 %** of pairs

| | min | max | mean ± standard deviation | unit |
|---|---|---|---|---|
| training set | | | | |
| number of exons per gene | **1** | **41** | **8.3 ± 7.6** | |
| coding length of gene | **318** | **5232** | **1250 ± 964** | base pairs |
| length of **DNA** | **1903** | **21911** | **7256 ± 4293** | base pairs |
| length of gene | **1032** | **21105** | **6071 ± 4320** | base pairs |
| GC contents | **0.40** | **0.66** | **0.52 ± 0.06** | |
| test set | | | | |
| number of exons per gene | **1** | **14** | **3.6 ± 2.8** | |
| coding length of gene | **276** | **2121** | **910 ± 477** | base pairs |
| length of **DNA** | **576** | **23076** | **3300 ± 2679** | base pairs |
| length of gene | **309** | **9033** | **2066 ± 1601** | base pairs |
| GC contents | **0.33** | **0.72** | **0.54 ± 0.07** | |

Table **A.l:** Statistics of the mouse human training and test set. The coding length of a gene is the sum of lengths of its exons, and the length of a gene is the distance in base pairs between the start codon and the stop codon.

consist of genes which are related by events of exon-fusion or exon-splitting.

## A.2 The test set

The test set was derived from the list of mouse human orthologs in [Pac99] by discarding all **DNA** pairs whose genes have non-consensus splice sites. This resulted in a set of **80** sequence pairs which is called the test set. Each **DNA** sequence in the test set comprises exactly one complete gene.

**As** can be seen by comparing the statistics of the training set to that **of** the test set (see Table **A.1**), the test set contains shorter genes with fewer exons in shorter **DNA** sequences. The sequences **of** the test set are more biased towards high GC contents than those in the training set, see Table **A.2. As** for the training set, **also** the GC contents of the genes within each pair of the test set are well correlated, see Table **A.3.** The test set has a higher proportion of pairs with well conserved gene structures **as 42 %** of the pairs consist of genes with the

| **GC contents**<br>GC contents | **training set**<br>training set | test set |
|---|---|---|
| [0.0, 0.43) | 0.06 | 0.05 |
| [0.43, 0.51) | 0.30 | 0.28 |
| [0.51, 0.57) | 0.47 | 0.32 |
| [0.57, 1.00] | 0.17 | 0.35 |

Table **A.2:** Distribution of GC contents in the mouse human training and test sets.

| | min | max | mean ± standard deviation |
|---|---|---|---|
| training set | | | |
| mean GC contents of pair | **0.40** | **0.64** | **0.52 ± 0.05** |
| difference in GC contents in pair | **0.002** | **0.09** | **0.03 ± 0.02** |
| test set | | | |
| mean GC contents of pair | **0.38** | **0.68** | **0.54 ± 0.07** |
| difference in GC contents in pair | **0.00** | **0.11** | **0.04 ± 0.03** |

Table **A.3:** Distribution of GC contents in the sequence pairs of the mouse human training and test sets.

| | training set | **test set**<br>test set |
|---|---|---|
| same coding length same number of exons | **0.36** | **0.42** |
| same coding length different number of exon | **0.00** | **0.00** |
| different coding length same number of exons | **0.61** | **0.55** |
| different coding length different number of exon | **0.03** | **0.03** |

Table **A.4:** Conservation of gene structures in the gene pairs of the mouse human training and test sets.

same number of exons and the same coding length (opposed to only 36 % in the training set), see Table A.4. As for the training set, also the majority (55 %) of the test set consists of pairs in which the genes have the same number of exons, but a different coding length. Only 3 % of the gene pairs are related by events of exon-fusion or exon-splitting.

Eight genes (10 %) of the genes of the test set are also found in the training set. When removing them from the test set, the performance of Table 3.1 remains almost unchanged with most positive and negative changes within 1 % and all within 3 %.

## A.3 Post-processing of the predicted mouse and human genes

In the post-processing step all predicted genes with introns of less than or equal to 50 base pairs length and or a total coding length of less than or equal to 120 base pairs length are removed.

# Appendix B

# Mouse human parameter tables

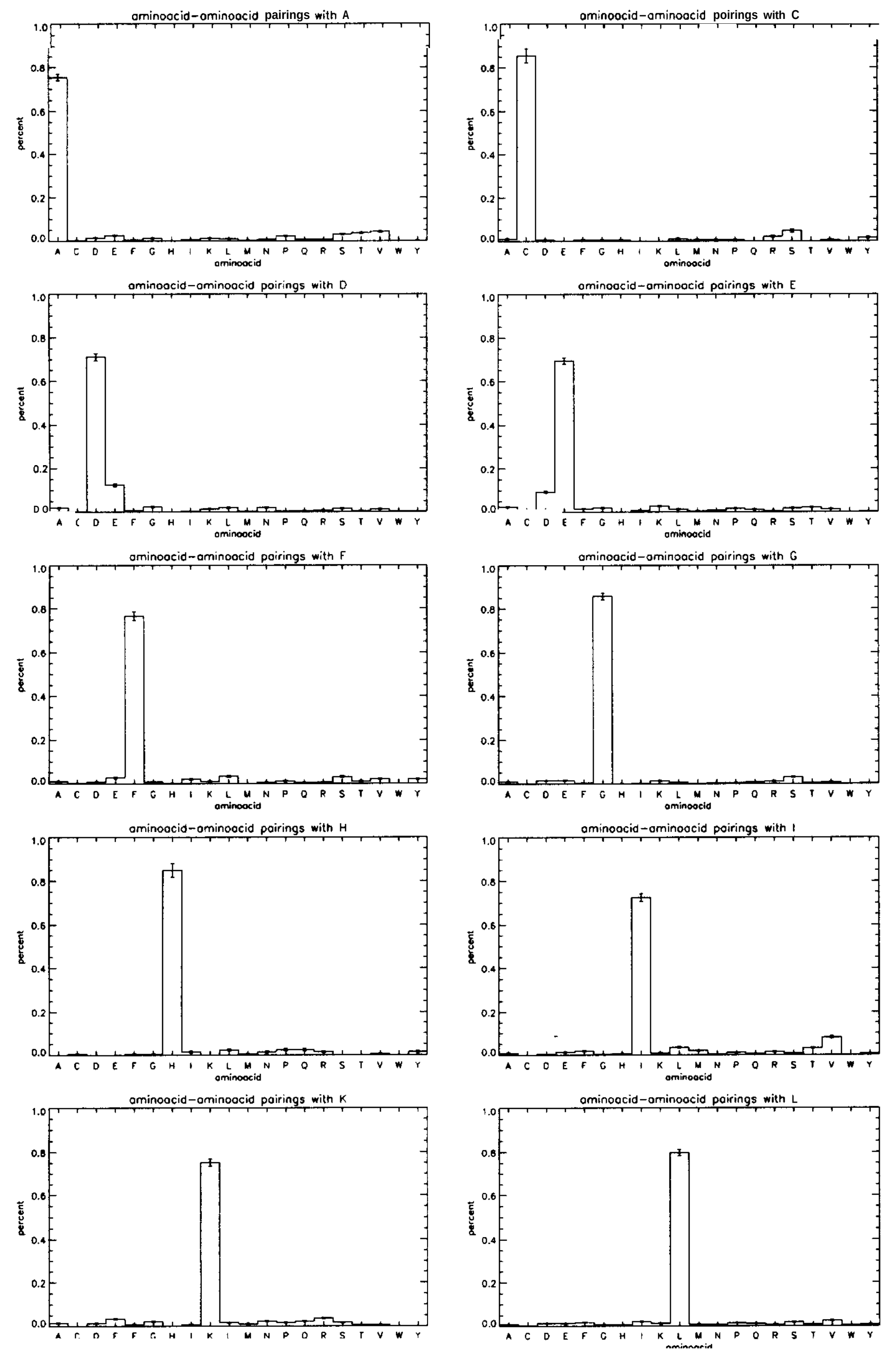
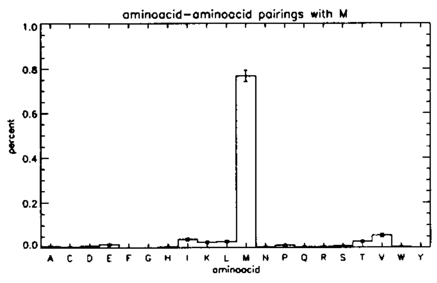
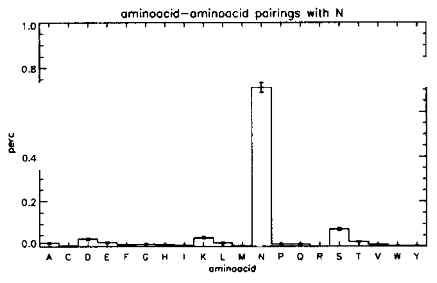
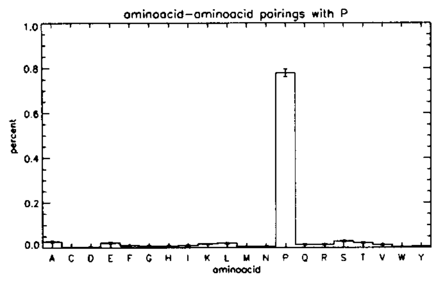
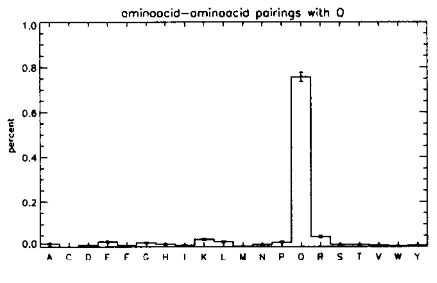


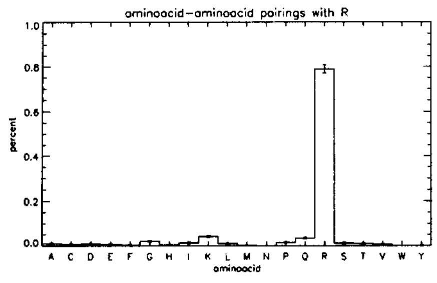
Figure B.1: **Amino-aid statistics derived from the emission probabilities** of **the** ***match*** exon **state as determined from the training set** of **mouse and human DNA. The error** bars **indicate the statistical errors.**

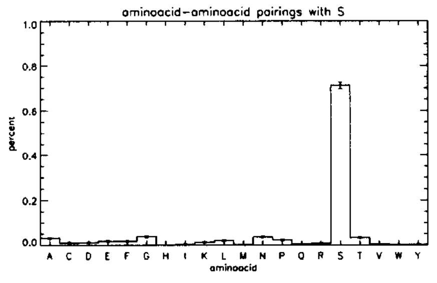

aminoacid–aminoacid pairings with M
percent
aminoacid

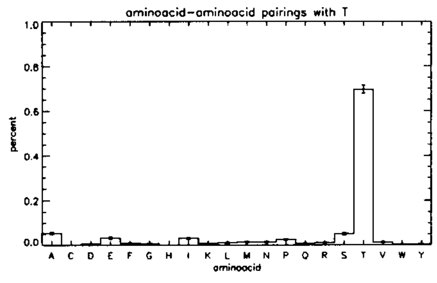

aminoacid–aminoacid pairings with N
perc
aminoacid

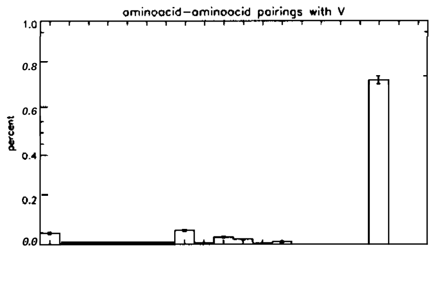

aminoacid–aminoacid pairings with P
percent
aminoacid

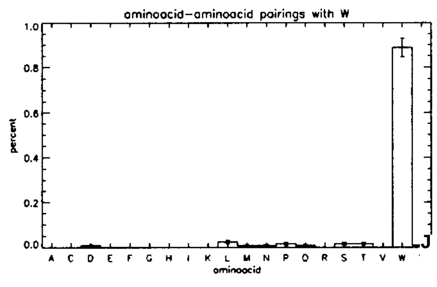

aminoacid–aminoacid pairings with Q
percent

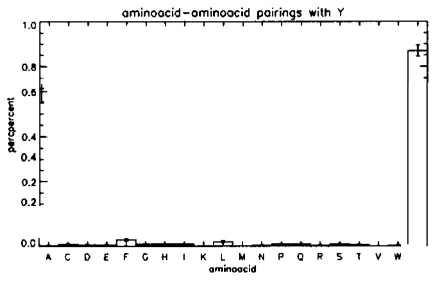

aminoacid–aminoacid pairings with R
percent
aminoacid


aminoacid–aminoacid pairings with S
percent
aminoacid


aminoacid–aminoacid pairings with T
percent
aminoacid


aminoacid–aminoacid pairings with V
percent


aminoacid–aminoacid pairings with W
percent
aminoacid


aminoacid–aminoacid pairings with Y
percpercent
aminoacid

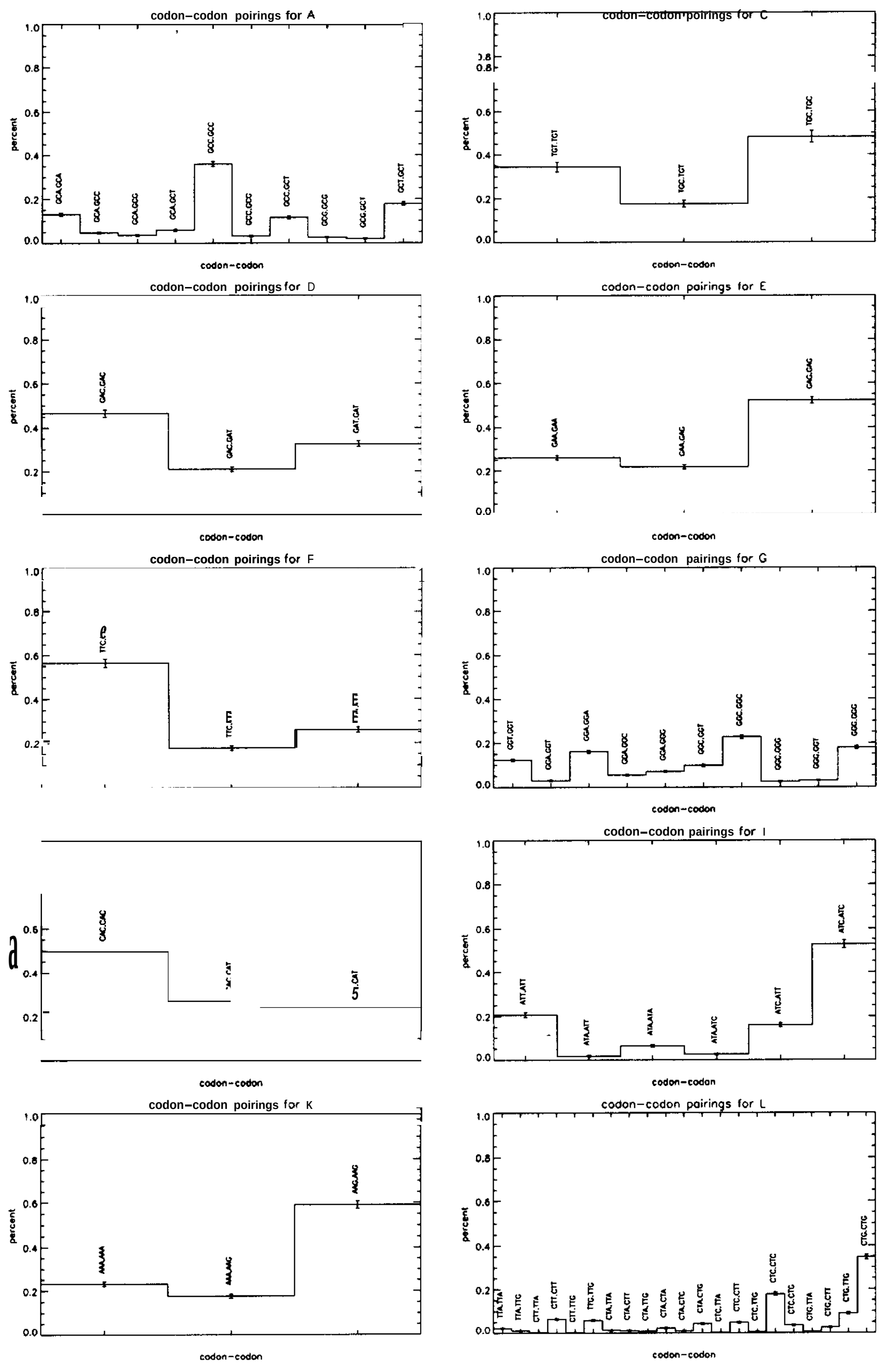
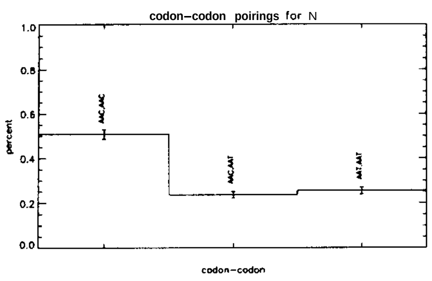
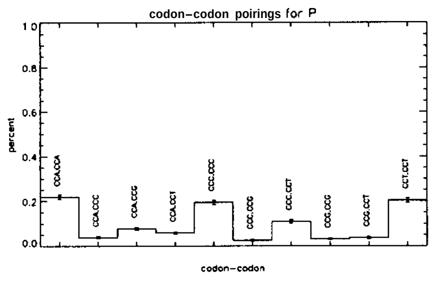
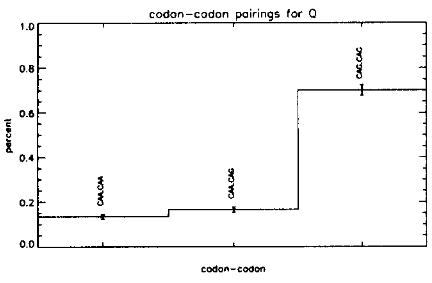
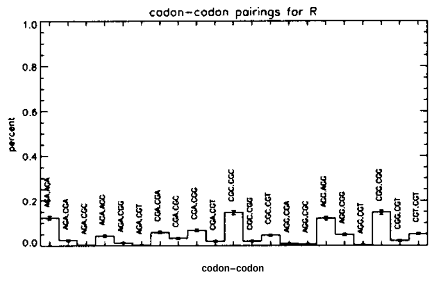


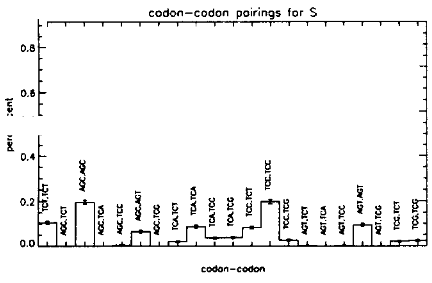
Figure **B.2**: Codon usage statistics derived from the emission probabilities of the *match exon* and the *STOP STOP* state as determined from the training set of mouse and human **DNA.** The error bars indicate the statistical errors.

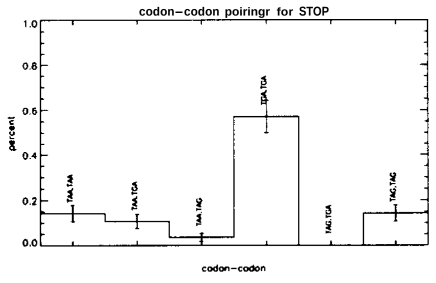

codon–codon poirings for N
percent
codon–codon
AAC,AAC
AAC,AAT
AAT,AAT

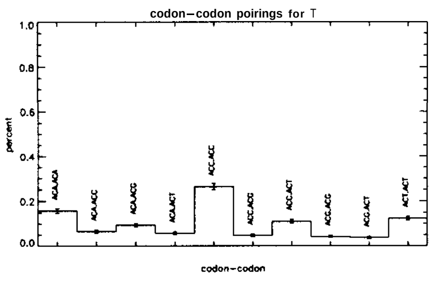

codon–codon poirings for P
percent
codon–codon

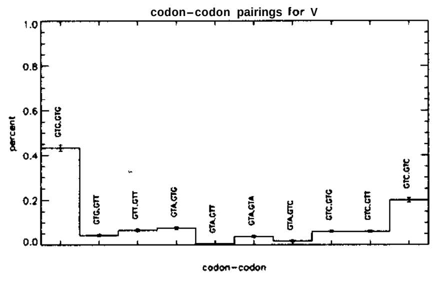

codon–codon poirings for Q
percent
codon–codon
CAA,CAA
CAA,CAG
CAG,CAG

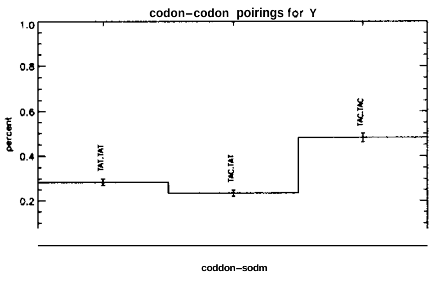

codon–codon poirings for R
percent
codon–codon


codon–codon poirings for S
percent
codon–codon


codon–codon poiringr for STOP
percent
codon–codon
TAA,TAA
TAA,TGA
TAA,TAG
TGA,TGA
TAG,TGA
TAG,TAG


codon–codon poirings for T
percent
codon–codon


codon–codon pairings for V
percent
codon–codon


codon–codon poirings for Y
percent
coddon–sodm
TAT,TAT
TAC,TAT
TAC,TAC

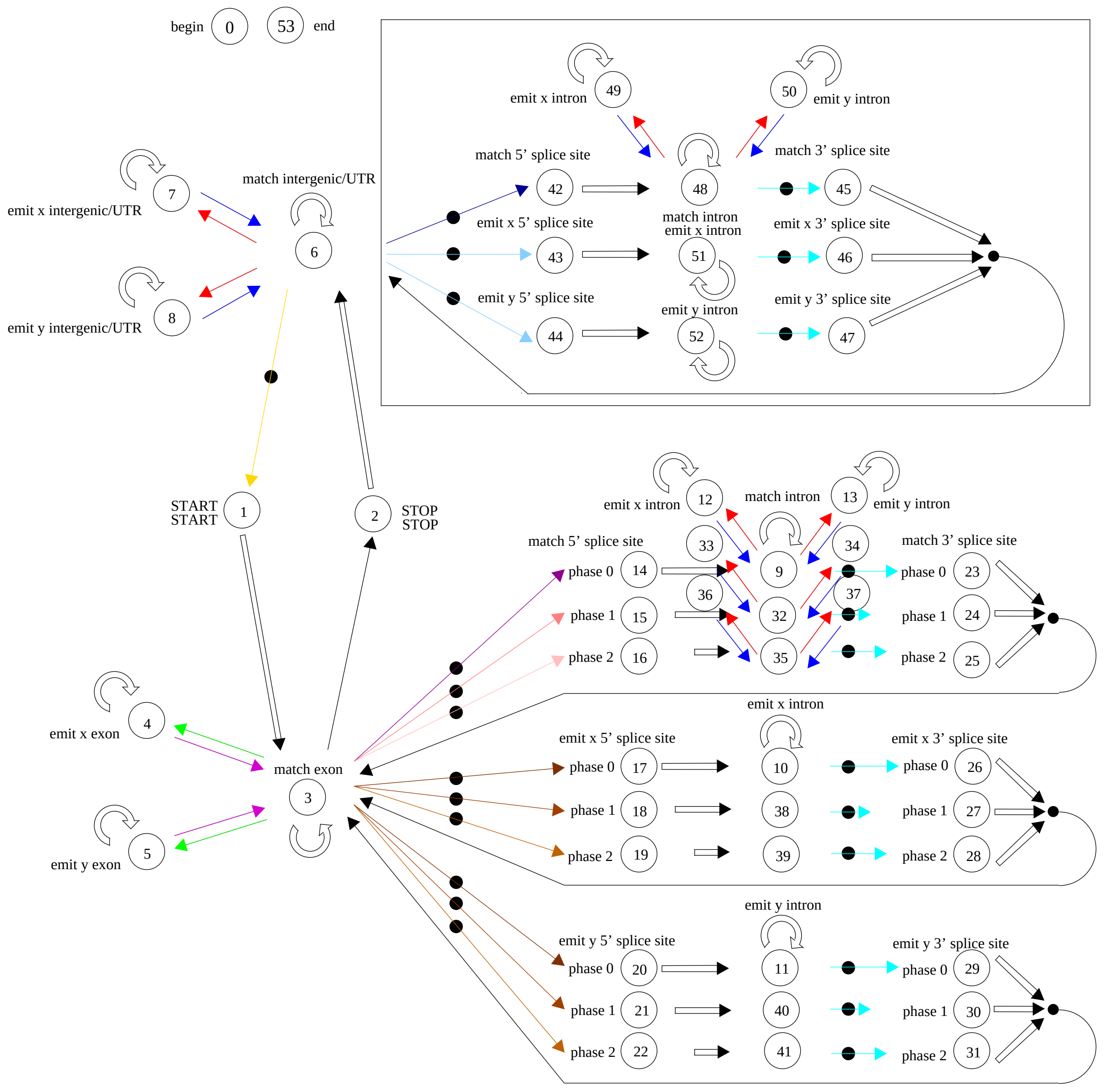


Figure B.3: States and transitions of the pair HMM underlying DOUBLESCAN and PROJECTOR. States are shown as circles, transitions as arrows. The *begin* state is connected to every state except itself and the *end* state. Likewise, there are transitions to the *end* state from every state except the *begin* state and itself. The arrows corresponding to these transitions are not shown for clarity. Each open arrow corresponds to a transition probability which is defined by the constraint that the probabilities of the transitions emerging from every state have to add up to one. Coloured arrows of the same colour correspond to transitions of the same probability. Arrows marked by a black dot are special transitions, see Section 2.3. The large box at the top right contains the states which model introns within untranslated regions (UTR-splicing).

| from state | to state | | derivation |
|---|---|---|---|
| *match exon* | *emit x exon* | | (Match_exon_to_emit_exon)/2 · (1 − To_end) |
| | *emit y exon* | | (Match_exon_to_emit_exon)/2 · (1 − To_end) |
| | *STOP STOP* | | (Match_exon_to_stop_exon) · (1 − To_end) |
| | *match 5' splice site phase 0* | * | Special_match_exon_to_intron · Phase0 · (1 − To_end) |
| | *match 5' splice site phase 1* | * | Special_match_exon_to_intron · Phase1 · (1 − To_end) |
| | *match 5' splice site phase 2* | * | Special_match_exon_to_intron · (1 − Phase0 − Phase1)· (1 − To_end) |
| | *emit x 5' splice site phase 0* | * | Special_match_exon_to_emit_intron/2 · Phase0· (1 − To_end) |
| | *emit x 5' splice site phase 1* | * | Special_match_exon_to_emit_intron/2 · Phase1· (1 − To_end) |
| | *emit x 5' splice site phase 2* | * | Special_match_exon_to_emit_intron/2· (1 − Phase0 − Phase1) · (1 − To_end) |
| | *emit y 5' splice site phase 0* | * | Special_match_exon_to_emit_intron/2 · Phase0· (1 − To_end) |
| | *emit y 5' splice site phase 1* | * | Special_match_exon_to_emit_intron/2 · Phase1· (1 − To_end) |
| | *emit y 5' splice site phase 2* | * | Special_match_exon_to_emit_intron/2· (1 − Phase0 − Phase1) · (1 − To_end) |
| | *match exon* | | (1 − Match_exon_to_stop_exon − Match_exon_to_emit_exon − Match_exon_to_match_5_splice_site − Match_exon_to_emit_5_splice_site)· (1 − To_end) |
| | *end* | | To_end |
| *match intergenic/UTR* | *emit x intergenic/UTR* | | Match_non_exon_to_emit_non_exon/2 · (1 − To_end) |
| | *emit y intergenic/UTR* | | Match_non_exon_to_emit_non_exon/2 · (1 − To_end) |
| | *START START* | * | Special_intergenic_to_start_exon · (1 − To_end) |
| | *match 5' splice site* | * | Special_match_exon_to_intron· 1/( Special_match_exon_to_intron + Special_match_exon_to_emit_intron)· (1 − To_end) |
| | *emit x 5' splice site* | * | Special_match_exon_to_emit_intron/2· 1/( Special_match_exon_to_intron + Special_match_exon_to_emit_intron)· (1 − To_end) |
| | *emit y 5' splice site* | * | Special_match_exon_to_emit_intron/2· 1/( Special_match_exon_to_intron + Special_match_exon_to_emit_intron)· (1 − To_end) |
| | *match intergenic/UTR* | | (1 − Match_intergenic_to_start_exon − Match_non_exon_to_emit_non_exon − Match_exon_to_match_5_splice_site − Match_exon_to_emit_5_splice_site)· (1 − To_end) |
| | *end* | | To_end |

Table B.1: Parametrisation of the transition probabilities within the pair HMM underlying Doublescan and Projector. The values of the parameters are given in Table B.2. $N = 54$ is the number of states in the pair HMM of Doublescan and Projector. Special transitions (see Section 6.2 for details) are indicated by an asterisk (*) in the third column. Note that the nominal values of the transitions emerging from a state do not have to add up to one if one or more of the transitions are special.

| from state | to state | | derivation |
|---|---|---|---|
| match intron | match intron | | (1 − Match_non_exon_to_emit_non_exon − Match_intron_to_match_exon) . (1 − To-end) |
| | | | same for states 9, 32, 35, 48 |
| | *emit x intron* | | Match_non_exon_to_emit_non_exon/2 · (1 − To_end) |
| | | | same for transitions 9 to 12, 32 to 33, 35 to 36, 48 to 49 |
| | *emit y intron* | | Match_non_exon_to_emit_non_exon/2 · (1 − To_end) |
| | | | same for transitions 9 to 13, 32 to 34, 35 to 37, 48 to 50 |
| | *match 3' splice site* | * | Special_intron_to_match_exon · (1 − To_end) |
| | | | same for transitions 9 to 23, 32 to 24, 35 to 25, 48 to 45 |
| | *end* | | To_end |
| | | | same for transitions 9 to 53, 32 to 53, 35 to 53, 48 to 53 |
| *emit x exon* | *match exon* | | Emit_exon_to_match_exon · (1 − To_end) |
| | *emit x exon* | | (1 − Emit_exon_to_match_exon) · (1 − To_end) |
| | *end* | | To_end |
| *emit y exon* | *match exon* | | Emit_exon_to_match_exon · (1 − To_end) |
| | *emit y exon* | | (1 − Emit_exon_to_match_exon) · (1 − To_end) |
| | *end* | | To_end |
| *emit x intergenic/UTR* | *match intergenic/UTR* | | Emit_non_exon_to_match_non_exon · (1 − To_end) |
| | *emit x intergenic/UTR* | | (1 − Emit_non_exon_to_match_non_exon) · (1 − To_end) |
| | *end* | | To_end |
| *emit y intergenic/UTR* | *match intergenic/UTR* | | Emit_non_exon_to_match_non_exon · (1 − To_end) |
| | *emit y intergenic/UTR* | | (1 − Emit_non_exon_to_match_non_exon) · (1 − To_end) |
| | *end* | | To_end |
| *emit x intron* | *emit x 3' splice site* | * | Special_intron_to_match_exon · (1 − To_end) |
| | | | same for transitions 10 to 26, 38 to 27, 39 to 28, 51 to 46 |
| | *emit x intron* | | (1 − Match_intron_to_match_exon) · (1 − To_end) |
| | | | same for states 10, 38, 39, 51 |
| | *end* | | To_end |
| | | | same for transitions 10 to 53, 38 to 53, 39 to 53, 51 to 53 |
| *emit y intron* | *emit y 3' splice site* | * | Special_intron_to_match_exon · (1 − To_end) |
| | | | same for transitions 11 to 29, 40 to 30, 41 to 31, 52 to 47 |
| | *emit y intron* | | (1 − Match_intron_to_match_exon) · (1 − To_end) |
| | | | same for states 11, 40, 41, 52 |
| | *end* | | To_end |
| | | | same for transitions 11 to 53, 40 to 53, 41 to 53, 52 to 53 |
| *match 5' splice site* | *match intron* | | (1 − To_end) |
| | | | same for transitions 14 to 9, 15 to 32, 16 to 35, 42 to 48 |
| | *end* | | To_end |
| | | | same for transitions 14 to 53, 15 to 53, 16 to 53, 42 to 53 |
| *emit x 5' splice site* | *emit x intron* | | (1 − To_end) |
| | | | same for transitions 17 to 10, 18 to 38, 19 to 39, 43 to 51 |
| | *end* | | To_end |
| | | | same for transitions 17 to 53, 18 to 53, 19 to 53, 43 to 53 |
| *emit y 5' splice site* | *emit y intron* | | (1 − To_end) |
| | | | same for transitions 20 to 11, 21 to 40, 22 to 41, 44 to 52 |
| | *end* | | To_end |
| | | | same for transitions 20 to 53, 21 to 53, 22 to 53, 44 to 53 |

| from state | to state | | derivation |
|---|---|---|---|
| ***begin*** | anv connected state | | $1/(N-2)$ |
| ***START START*** | match exon | | 1 − To-end |
| | end | | To_end |
| ***STOP STOP*** | match *intergenic/UTR* | | 1 − To-end |
| | *end* | | To-end |
| ***match 3' splice site*** | match exon or match *intergenic/UTR* | | (1 − To-end) |
| | | | same for transitions **23** to **3, 24** to **3, 25** to **3, 45** to **6** |
| | end | | To-end |
| | | | same for transitions **23** to **53, 24** to **53, 25** to **53, 45** to **53** |
| ***emit x 3' splice site*** | match exon or match *intergenic/UTR* | | (1 − To-end) |
| | | | same for transitions **26** to **3, 27** to **3, 28** to **3, 46** to **6** |
| | *end* | | To-end |
| | | | same for transitions **26** to **53, 27** to **53, 28** to **53, 46** to **53** |
| emit y 3' splice site | match exon | | (1 − To-end) |
| | | | same for transitions 29 to **3, 30** to **3, 31** to **3, 47** to **6** |
| | end | | To-end |
| | | | same for transitions 29 to **53, 30** to **53, 31** to **53, 47** to **53** |
| ***emit x intron of match intron*** | match intron | | **Emit_non_exon_to_match_non_exon** · (1 − To-end) |
| | | | same for transitions **12** to **9, 33** to **32, 36** to **35, 49** to **48** |
| | emit x intron of match intron | | (1 − Emit_non_exon_to_match_non_exon) . (1 − To-end) |
| | | | same for states **12, 33, 36, 49** |
| | *end* | | To-end |
| | | | same for transitions **12** to **53, 33** to **53, 36** to **53, 49** to **53** |
| ***emit y intron of match intron*** | match intron | | **Emit_non_exon_to_match_non_exon** . (1 − To-end) |
| | | | same for transitions **13** to **9, 34** to **32, 37** to **35, 50** to **48** |
| | emit y intron of match intron | | (1 − Emit_non_exon_to_match_non_exon) . (1 − To-end) |
| | | | same for states **13, 34, 37, 50** |
| | *end* | | **To_end** |
| | | | **same for transitions 13 to 53, 34 to 53, 37 to 53, 50 to 53** |

| parameter | value |
|---|---|
| Phase0 | 0.4387 |
| Phase1 | 0.387 |
| To-end | 0.0001 |
| Match-exon-tostop-exon | 0.003 |
| Match-exon-to-emit_exon | 0.02 |
| Match_exon_to_match_5_splice_site | 5e-06 |
| Match-exon-to-emit_5_splice_site | 5e-06 |
| Matchintergenic-tostart-exon | 0.0001 |
| Matchnon-exon-to-emitnon-exon | 0.08 |
| Matchintron-tomatch-exon | 1e-05 |
| Emit_exon_to_match_exon | 0.33333 |
| Emitnon-exon-tomatchnon-exon | 0.04 |
| Special_match_exon_to_intron | 1 |
| Special_intron_to_match_exon | 0.25 |
| Specialmatch_exon_to_emit intron | 0.06666 |
| Special_intergenic_to_start_exon | 0.1 |

Table B.2: Values of the parameters on which the transition probabilities depend.

| parameter | value |
|---|---|
| Prior-GT | 0.01 |
| Prior-GC | 0.0001 |
| PriorAG | 0.001 |
| PriorATG | 0.005 |

Table **B.3:** Values of the priors which are used with the special transition probabilities of the pair HMM underlying **DOUBLESCAN** and **PROJECTOR** **for** the analysis of mouse and human **DNA** sequences.

# Appendix *C*

# *C. elegans C. briggsae* training and test sets

The training set of C. elegans and C. briggsae gene pairs has been established by Avril Coghlan, Trinity College, Dublin.

## C.1 The training set

As described in Chapter 5, the training set was used only to derive the emission probabilities of DOUBLESCAN according to Section 2.3. In particular, it was not used to derive the values of the transition probabilities nor to fine-tune the performance, see Section 5.2. The test set comprises 910 pairs of C. elegans and C. briggsae DNA sequences, each comprising exactly one complete gene. The C. elegans genes are known genes of Wormbase release WS77 [SSD+01, Wor] and the *C. briggsae* genes are putative genes predicted by GENEFINDER [eSC98]. All pairs of genes were defined as being orthologous using BLAST [AGM+90]. The exons of the two genes were mutual best hits and hit each other with an Evalue a hundred times smaller than the second best hit and with an E-value of less than 0.1. Pairs of orthologous exons were covered by at least 95 % by BLAST hits. Only 16 out of 910 gene pairs (1.7 % of the training set) had splice sites which were not equal to the GT–AG consensus. Table C.1 shows some statistics of the training set. As opposed to the mouse and human genome which can be partitioned into long GC isochores according to their GC contents, the GC density within the C. elegans and C. briggsae genomes is uniform around 36 %, see Table C.2. However, as can be seen by comparing Table C.3 and Table A.3 in Appendix A, the GC contents of orthologous C. elegans and C. briggsae genes are **as** well correlated **as** those of orthologous mouse and human genes.

The gene structures of orthologous C. elegans and C. briggsae genes are more conserved than those of the mouse human training set (see Table A.4 in Appendix A) **as** can be seen from Table C.4. The majority **(53** %) of genes has the same exon number and coding length **as** its orthologous partner in the other genome and differences in the gene structures between orthologous genes are only due to a difference in coding length, but not in exon number.

## C.2 Test set 1

**As** the training set is only used to automatically derive the emission probabilities of the *match* exon and *STOP STOP* state, but not for the derivation of the transition probabilities nor the fine-tuning of the performance, we can use the same data **as** a test set. Test set 1 is **a** subset of the training set. It comprises **353** pairs of genes whose exons were entirely covered by **BLAST** hits (100 %) and which either have the consensus splice sites **GT-AG** or the non-consensus splice sites **GC-AG** (present in **3** out of **353** gene pairs). The statistics *can* be found in Table C.l. Genes in this test set are on average shorter than those of test set 2 and have fewer **exons.** The orthologous genes in this test set have better conserved gene structures and are thus more closely related than those of test set 2, *see* Table C.4.

## C.3 Test set 2

**Also** test set 2 **is** a subset of the training set. It comprises **535** pairs of genes whose exons were covered by at least **95 %** but less than 100 % by **BLAST** matches and which either have the consensus splice sites **GT-AG** or the non-consensus splice sites **GC-AG** (present in **8** out of **535** gene pairs). There is no intersection between test set 1 and test set 2. The statistics *can* be found in Table C.l. Table C.4 shows the level of conservation between the gene structures of orthologous genes.

| | min | max | mean ± standard deviation | unit |
|---|---|---|---|---|
| training set | | | | |
| number of exons per gene | 1 | 21 | 4.1 ± 2.1 | |
| coding length of gene | 150 | 5046 | 917 ± 606 | base pairs |
| length of **DNA** | 461 | 36529 | 3455 ± 2818 | base pairs |
| length of gene | 180 | 11594 | 1536 ± 1187 | base pairs |
| GC contents | 0.27 | 0.55 | 0.38 ± 0.04 | |
| test set 1 | | | | |
| number of exons per gene | 1 | 13 | 3.5 ± 1.7 | |
| coding length of gene | 150 | 2988 | 697 ± 435 | base pairs |
| length of **DNA** | 461 | 19253 | 2994 ± 2477 | base pairs |
| length of gene | 180 | 7759 | 1191 ± 930 | base pairs |
| GC contents | 0.27 | 0.51 | 0.38 ± 0.04 | |
| test set 2 | | | | |
| number of exons per gene | 1 | 21 | 4.5 ± 2.3 | |
| coding length of gene | 177 | 5046 | 1058 ± 665 | base pairs |
| length of **DNA** | 560 | 36529 | 3741 ± 2988 | base pairs |
| length of gene | 225 | 11594 | 1753 ± 1286 | base pairs |
| GC contents | 0.29 | 0.55 | 0.38 ± 0.04 | |

Table C.l: Statistics of the C. *elegans* C. *briggsae* training and test sets. The coding length of a gene **is** the sum of lengths of its exons and the length of a gene is the distance in base pairs between the start codon and the stop codon.

| GC contents | training set | test set 1 | test set 2 |
|---|---|---|---|
| [0.0, 0.43) | 0.923 | 0.91 | 0.933 |
| [0.43, 0.51) | 0.074 | 0.09 | 0.062 |
| [0.51, 0.57) | 0.003 | 0.00 | 0.005 |
| [0.57, 1.00] | 0.000 | 0.00 | 0.000 |

Table **C.2:** Distribution of GC contents in the *C. elegans* *C. briggsae* training and test sets.

| | min | max | mean ± standard deviation |
|---|---|---|---|
| training set | | | |
| mean GC contents of pair | 0.31 | 0.53 | $0.38 \pm 0.03$ |
| difference in GC contents in pair | 0.00 | 0.20 | $0.03 \pm 0.02$ |
| test set 1 | | | |
| mean GC contents of pair | 0.32 | 0.50 | $0.38 \pm 0.03$ |
| difference in GC contents in pair | 0.00 | 0.11 | $0.03 \pm 0.02$ |
| test set 2 | | | |
| mean GC contents of pair | 0.31 | 0.53 | $0.38 \pm 0.03$ |
| difference in GC contents in pair | 0.00 | 0.20 | $0.03 \pm 0.02$ |

Table **C.3:** Distribution of GC contents in the sequence pairs of the *C. elegans* *C. briggsae* training and test sets.

| | training set | test set 1 | test set 2 |
|---|---|---|---|
| same coding length same number of exons | 0.53 | 0.997 | 0.21 |
| same coding length different number of exon | 0.00 | *0.000* | 0.00 |
| different coding length same number of exons | 0.47 | 0.003 | 0.79 |
| different coding length different number of exon | 0.00 | *0.000* | 0.00 |

**Table C.4: Conservation of gene structures in the gene pairs of the C. *elegans* C. *briggsae* training and test sets.**

# Appendix D

# *C. elegans C. briggsae* **parameter tables**

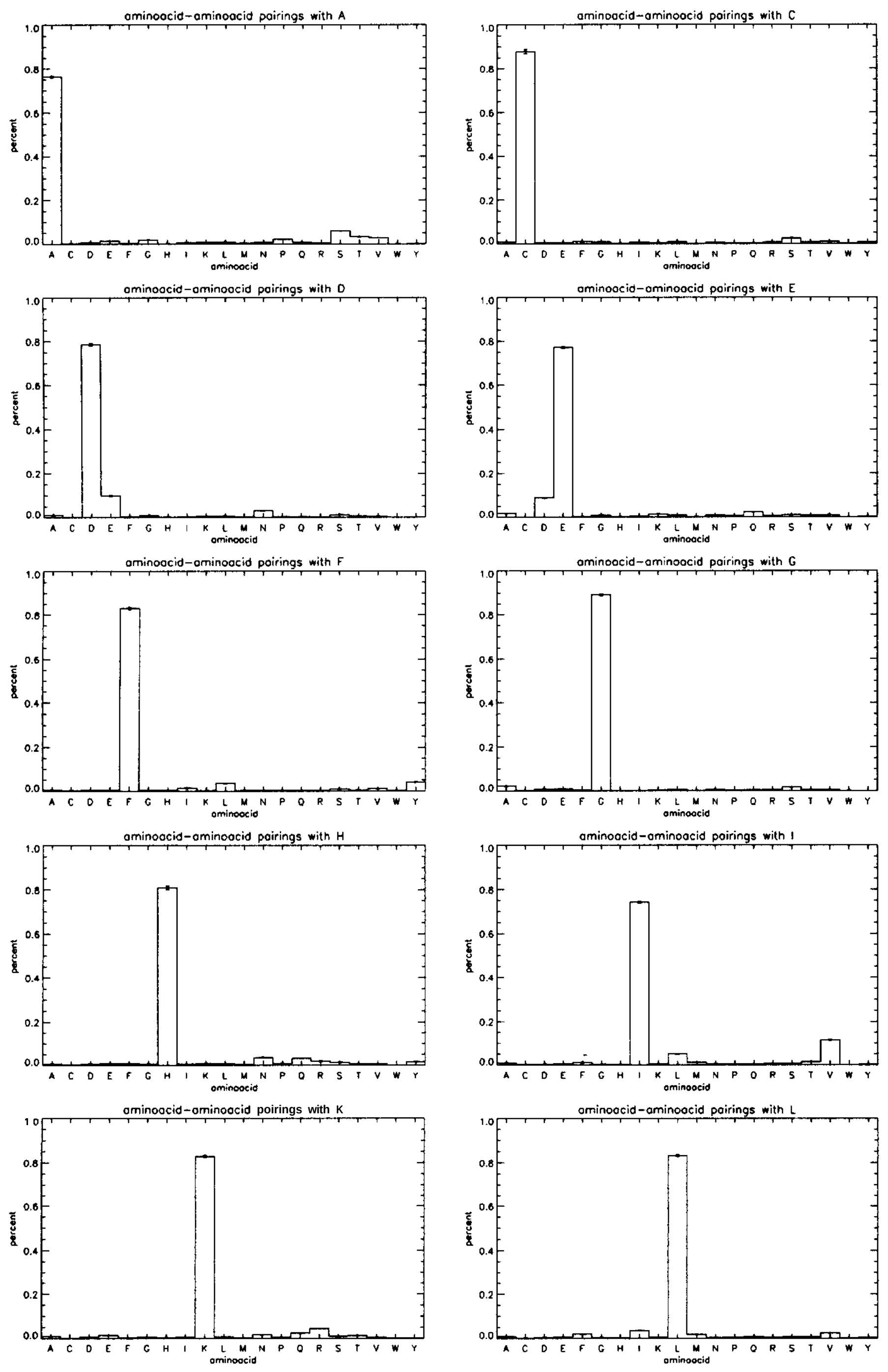
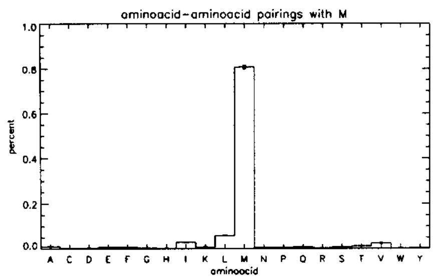
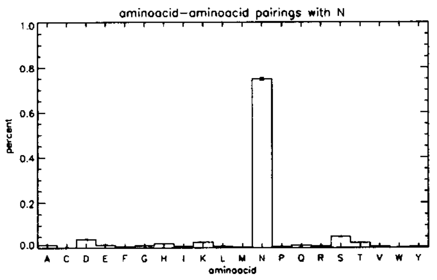
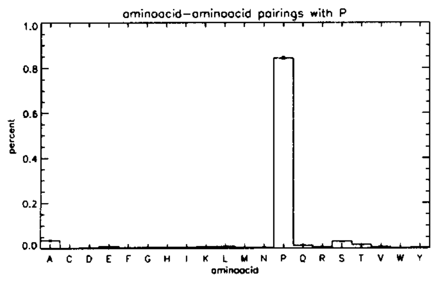
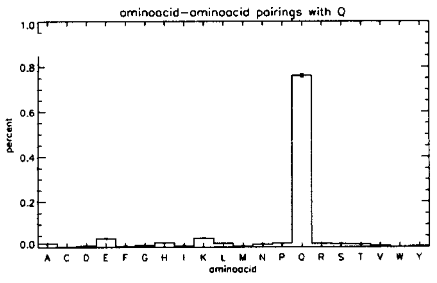


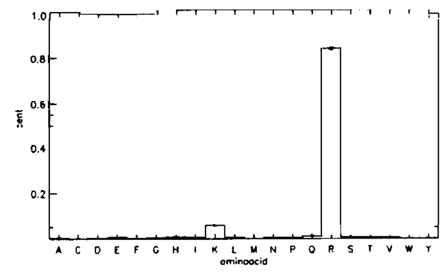
Figure D.1: Amino-acid statistics derived from the emission probabilities of the match *exon* state as determined from the training set of *C. elegans* and *C. briggsae* DNA. The error bars indicate the statistical errors.

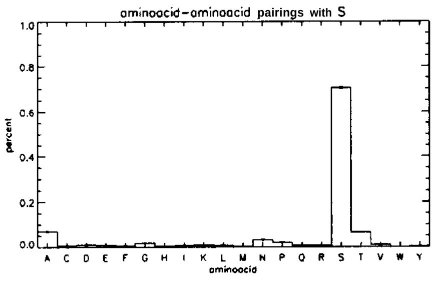

aminoacid–aminoacid pairings with M
percent
aminoacid

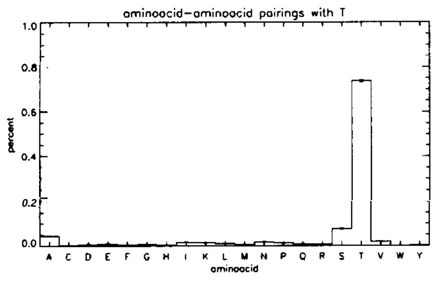

aminoacid–aminoacid pairings with N
percent
aminoacid

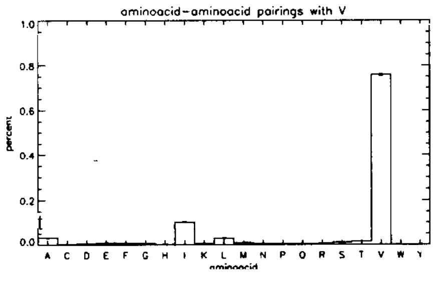

aminoacid–aminoacid pairings with P
percent
aminoacid

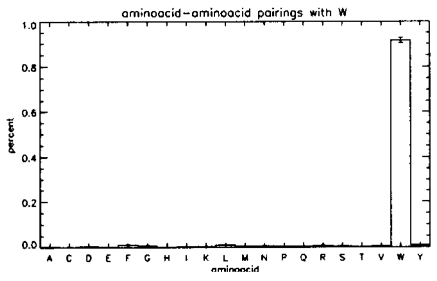

aminoacid–aminoacid pairings with Q
percent
aminoacid

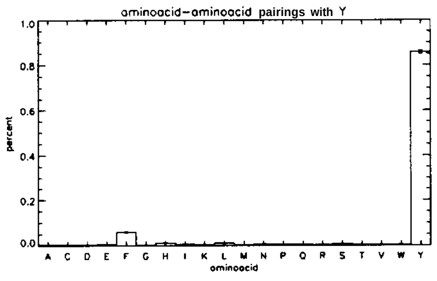

aminoacid


aminoacid–aminoacid pairings with S
percent
aminoacid


aminoacid–aminoacid pairings with T
percent
aminoacid


aminoacid–aminoacid pairings with V
percent
aminoacid


aminoacid–aminoacid pairings with W
percent
aminoacid


aminoacid–aminoacid pairings with Y
percent
aminoacid

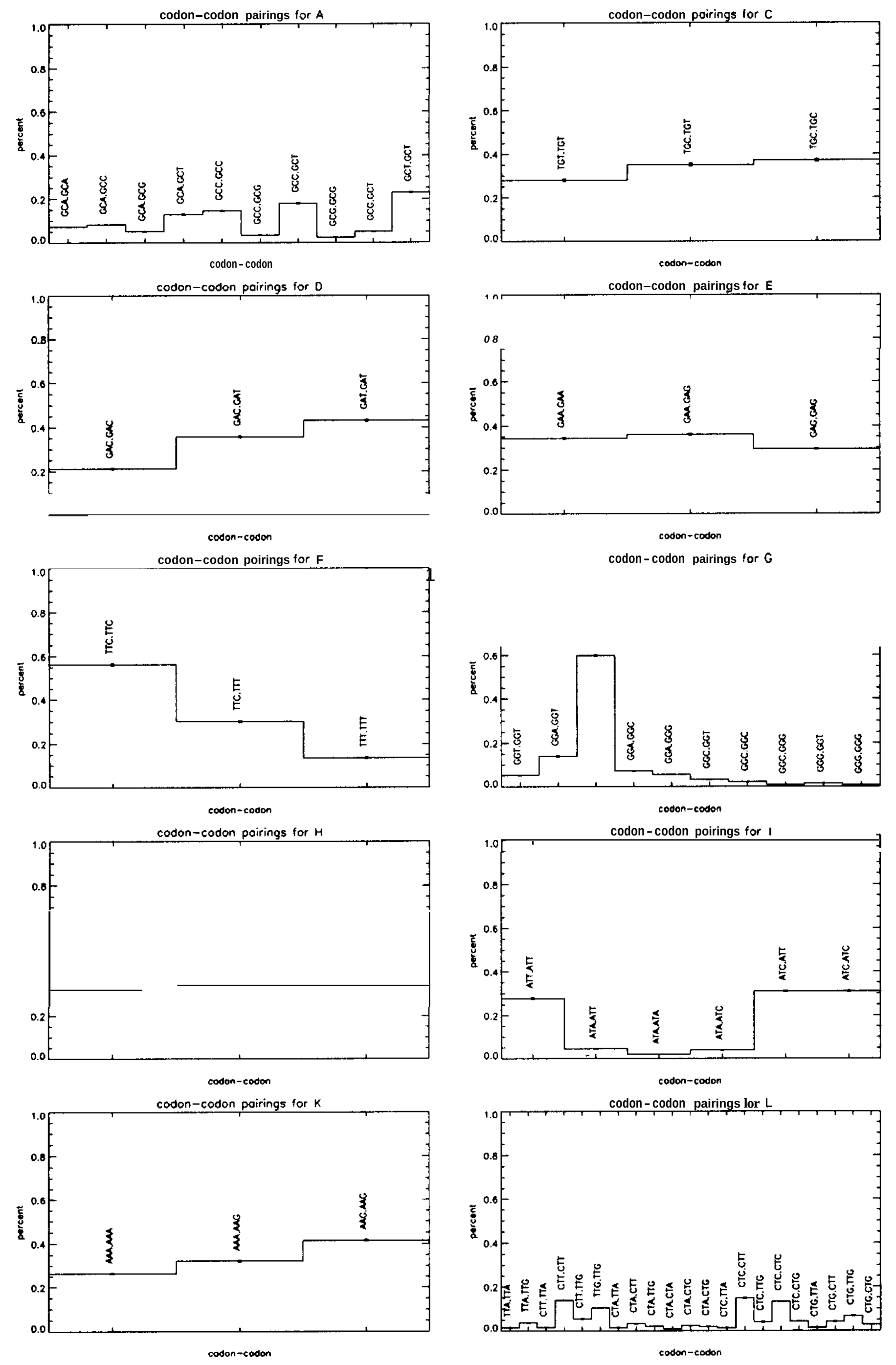
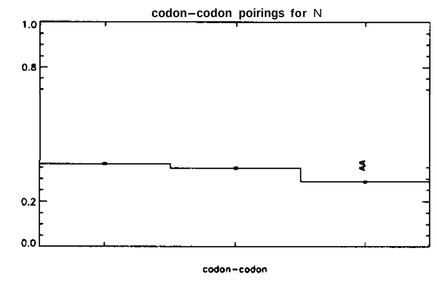
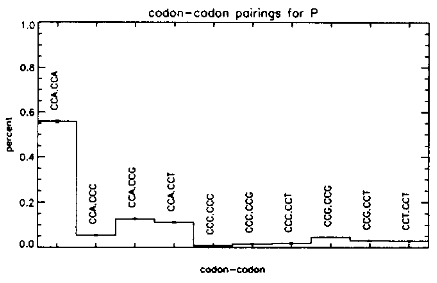
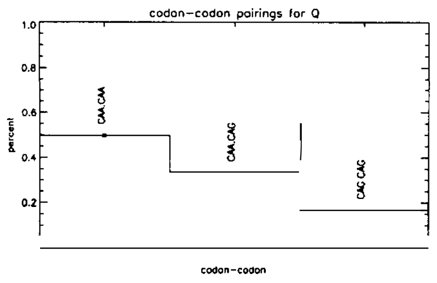
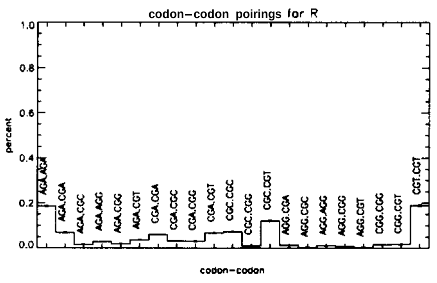


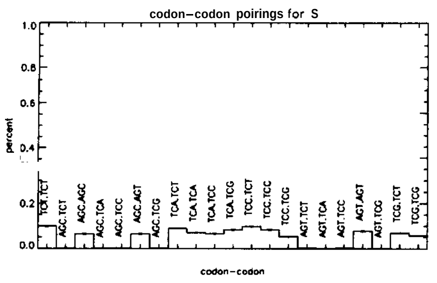
**Figure D.2: Codon usage statistics derived from the emission probabilities of the *match exon* and the *STOP STOP* state as determined from the training set of *C.* elegans and *C.* briggsae DNA. The error bars indicate the statistical errors.**

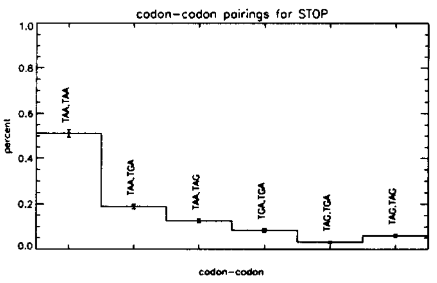

codon−codon pairings for N
1.0
0.8
0.2
0.0
AA
codon−codon

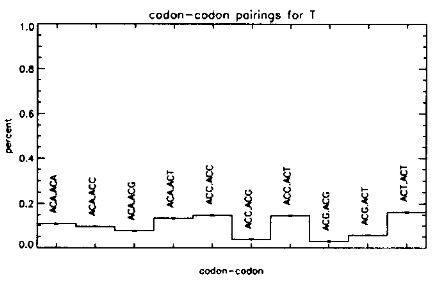

codon−codon pairings for P
1.0
0.8
0.6
0.4
0.2
0.0
percent
CCA.CCA
CCA.CCC
CCA.CCG
CCA.CCT
CCC.CCC
CCC.CCG
CCC.CCT
CCG.CCG
CCG.CCT
CCT.CCT
codon−codon

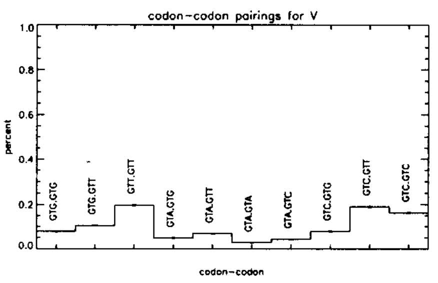

codon−codon pairings for Q
1.0
0.8
0.6
0.4
0.2
percent
CAA.CAA
CAA.CAG
CAG.CAG
codon−codon

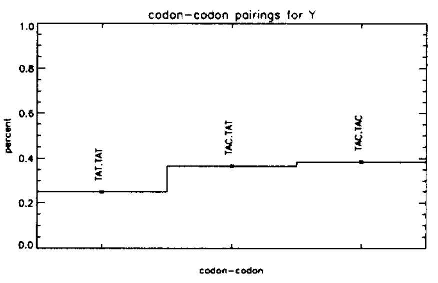

codon−codon pairings for R
1.0
0.8
0.6
0.4
0.2
0.0
percent
AGA.AGA
AGA.CGA
AGA.CGC
AGA.AGG
AGA.CGG
AGA.CGT
CGA.CGA
CGA.CGC
CGA.CGG
CGA.CGT
CGC.CGC
CGC.CGG
CGC.CGT
AGG.CGA
AGG.CGC
AGG.AGG
AGG.CGG
AGG.CGT
CGG.CGG
CGG.CGT
CGT.CGT
codon−codon


codon−codon pairings for S
1.0
0.8
0.6
0.4
0.2
0.0
percent
TCT.TCT
AGC.TCT
AGC.AGC
AGC.TCA
AGC.TCC
AGC.AGT
AGC.TCG
TCA.TCT
TCA.TCA
TCA.TCC
TCA.TCG
TCC.TCT
TCC.TCC
TCC.TCG
AGT.TCT
AGT.TCA
AGT.TCC
AGT.AGT
AGT.TCG
TCG.TCT
TCG.TCG
codon−codon


codon−codon pairings for STOP
1.0
0.8
0.6
0.4
0.2
0.0
percent
TAA.TAA
TAA.TGA
TAA.TAG
TGA.TGA
TAG.TGA
TAG.TAG
codon−codon


codon−codon pairings for T
1.0
0.8
0.6
0.4
0.2
0.0
percent
ACA.ACA
ACA.ACC
ACA.ACG
ACA.ACT
ACC.ACC
ACC.ACG
ACC.ACT
ACG.ACG
ACG.ACT
ACT.ACT
codon−codon


codon−codon pairings for V
1.0
0.8
0.6
0.4
0.2
0.0
percent
GTG.GTG
GTG.GTT
GTT.GTT
GTA.GTG
GTA.GTT
GTA.GTA
GTA.GTC
GTC.GTG
GTC.GTT
GTC.GTC
codon−codon


codon−codon pairings for Y
1.0
0.8
0.6
0.4
0.2
0.0
percent
TAT.TAT
TAC.TAT
TAC.TAC
codon−codon

| parameter | value | comment |
|---|---|---|
| Prior_GT | 0.01 | |
| Prior_GC | 0.0001 | |
| Prior_AG | 0.01 | PriorAG (mousehuman) = 0.001 |
| Prior_ATG | 0.005 | |

Table D.1: Values of the priors which are used with the special transition probabilities of the pair HMM underlying DOUBLESCAN and PROJECTOR for the analysis of *C. elegans C.* briggsae DNA sequences. The value of the prior for the 3' splice sites (PriorAG) is the *only* transition parameter which is different from the parameters used for the analysis of mouse and human DNA (see Table B.3 in Appendix B). The parametrisation of the transition probabilities as well as the values of the parameters for the analysis of *C. elegans C. briggsae* DNA sequences are the same as those for the analysis of mouse human DNA sequences (see Table B.1 and Table B.2 in Appendix B).

# Appendix E

# The DOUBLESCAN web-server

**DOUBLESCAN** can be accessed via a web-server at

www.sanger.ac.uk/Software/analysis/doublescan/

**DOUBLESCAN** needs as input two DNA sequences in a variant of the **FASTA** format which requires a modified header-line:

>name start_position–end_position orientation

(see also www.sanger.ac.uk/Software/analysis/doublescan/fasta_format.shtml) where

- name is the name of the sequence (example: Mm)
- start-position is an integer which is the position of the first character in the sequence (example: **100)** and its value has to be smaller to that of the end-position
- end-position is an integer which is the position of the last character in the sequence (example: **737** i.e. the sequence is **737-100+1** = **638** nucleotides long)
- orientation can be either 'forward' or 'reverse' depending on the strand which is to be analysed for genes. Note that the value of the orientation in the header line does not indicate the orientation of the sequence as the **FASTA** file should always give the sequence of the forward strand.
- the fields in the header line have to be tab-delimited

To give an example of an input file in the required Fasta format:

```
>Mm 100-737 forward
gggaatgaagtttttctgcaggatttaaatgtggtctttaagagacaccgcatgcaaaga
atagctggggcttgctagccaatgaaaacattcagattccaatgacgcatcctttttttct
ccaccccccttccaagacccggattcggaaaccccgcctaacgctctagttttcaaccagg
tccgcagaaggcctatttaaagggacgattgctgtctccctgctgtcataaccatgtctg
gacgtggcaagggtggtaaaggccttgggaaaggcggcgctaagcgccaccgtaaggttc
tccgcgataacatccagggcatcaccaagcctgccatccgccgcctggcccggcgcgggg
gagtgaagcgcatctccggcctcatctacgaggagacccgcggtgtgctgaaggtgttcc
tggagaacgtgatccgcgacgccgtcacctacacggagcacgccaagcgcaagaccgtca
ccgccatggacgtggtctacgcgctcaagcgccagggccgcactctctacggattcggcg
gttaatcgactaacaaacgattttccactgtcaacaaaaggccctttttcagggccaccca
caaattcctagaaggagttgttcacttaccgaagctt
```

Every analysis by Doublescan returns two output files:

- a file containing the predicted annotation of the two input DNA sequences in gtf format (see http://www.fruitfly.org/flyannot/format.html#GTF)
- a file containing the predicted annotation of the two input DNA sequences and the predicted conserved subsequences in a variant of the gtf format

The following example shows an output file in gtf-format which indicates the predicted annotation:

```
Mm  Doublescan  Start_Codon  234  236  .  +  0  gene_id  3;  transcript_id  3;  exon_number  1
Mm  Doublescan  CDS          234  542  .  +  0  gene_id  3;  transcript_id  3;  exon_number  1
Mm  Doublescan  Stop_Codon   543  545  .  +  0  gene_id  3;  transcript_id  3;  exon_number  1
Mm  Doublescan  Exon         234  545  .  +  .  gene_id  3;  transcript_id  3;  exon_number  1

Hs  Doublescan  Start_Codon  311  313  .  +  0  gene_id  7;  transcript_id  7;  exon_number  1
Hs  Doublescan  CDS          311  619  .  +  0  gene_id  7;  transcript_id  7;  exon_number  1
Hs  Doublescan  Stop_Codon   620  622  .  +  0  gene_id  7;  transcript_id  7;  exon_number  1
Hs  Doublescan  Exon         311  622  .  +  .  gene_id  7;  transcript_id  7;  exon_number  1
```

The corresponding output file in the modified gtf-format indicates the predicted annotation as well as the conserved subsequences:

```
Mm  Doublescan  Intergenic     1   61  .  +  .  conserved
Mm  Doublescan  Intergenic    62  103  .  +  .
Mm  Doublescan  Intergenic   104  133  .  +  .  conserved
Mm  Doublescan  Intergenic   134  154  .  +  .
Mm  Doublescan  Intergenic   155  187  .  +  .  conserved
Mm  Doublescan  Intergenic   188  209  .  +  .
Mm  Doublescan  Intergenic   210  233  .  +  .  conserved
Mm  Doublescan  Start_Codon  234  236  .  +  0  gene_id  3;  transcript_id  3;  exon_number  1;  conserved
Mm  Doublescan  CDS          237  542  .  +  0  gene_id  3;  transcript_id  3;  exon_number  1;  conserved
Mm  Doublescan  Stop_Codon   543  545  .  +  0  gene_id  3;  transcript_id  3;  exon_number  1;  conserved
Mm  Doublescan  Intergenic   546  551  .  +  .  conserved
Mm  Doublescan  Intergenic   552  560  .  +  .
Mm  Doublescan  Intergenic   561  568  .  +  .  conserved
Mm  Doublescan  Intergenic   569  571  .  +  .
Mm  Doublescan  Intergenic   572  609  .  +  .  conserved
Mm  Doublescan  Intergenic   610  631  .  +  .
Mm  Doublescan  Intergenic   632  637  .  +  .  conserved

Hs  Doublescan  Intergenic     1   26  .  +  .
Hs  Doublescan  Intergenic    27   74  .  +  .  conserved
Hs  Doublescan  Intergenic    75  115  .  +  .
Hs  Doublescan  Intergenic   116  180  .  +  .  conserved
Hs  Doublescan  Intergenic   181  216  .  +  .
Hs  Doublescan  Intergenic   217  248  .  +  .  conserved
Hs  Doublescan  Intergenic   249  307  .  +  .
Hs  Doublescan  Intergenic   308  310  .  +  .  conserved
Hs  Doublescan  Start_Codon  311  313  .  +  0  gene_id  7;  transcript_id  7;  exon_number  1;  conserved
Hs  Doublescan  CDS          314  619  .  +  0  gene_id  7;  transcript_id  7;  exon_number  1;  conserved
Hs  Doublescan  Stop_Codon   620  622  .  +  0  gene_id  7;  transcript_id  7;  exon_number  1;  conserved
Hs  Doublescan  Intergenic   623  665  .  +  .  conserved
Hs  Doublescan  Intergenic   666  844  .  +  .
Hs  Doublescan  Intergenic   845  859  .  +  .  conserved
```