

Large-Scale Comparative Annotation of Bacterial Genomes

Waqar Ali



A thesis submitted in partial fulfillment of requirements
for the degree of
MSc Computer Science

September 2007

Project Supervisors
Dr. Jotun Hein (Department of Statistics)
Dr. Peter Jeavons (OUCL)

ABSTRACT

The MSc thesis describes the design and evaluation of a comparative annotation system for prokaryotic genomes. The proposed system makes use of a combination of comparative and ab-initio gene finding techniques for accurate annotation. The three main modules described in the system are the markov-model based gene finding module, the alignment based profiler and the comparative module that combines the results from both modules. The system works by first creating a multiple sequence alignment of the input sequences and doing independent gene finding through a combination of markov models of order one to five. These two operations are carried out in parallel and there are no interdependencies. The multiple sequence alignment is then used to create a profile that emphasizes regions of high conservation within the sequences. The results from the markov model module are then processed through the profile to give a final comparative annotation.

The system was evaluated by carrying out a series of tests with an increasing number of sequences in each test set. The results obtained were quite encouraging, in that the gene of interest was found 100% of the time, with the false positives being removed very accurately due to the comparative technique. Moreover the results indicated as expected, that increasing the number of sequences in the test sets has a very positive impact on resulting predictions and confidence scores.

ACKNOWLEDGEMENT

The author wishes to express gratitude to Dr. Jotun Hein for his constant help and support throughout the project duration. Dr. Hein was always readily available for discussion and forthcoming with new ideas. The author would also like to thank Dr. Peter Jeavons for his suggestions and comments which helped a lot during the formulation of the project goals and the thesis write-up.

CONTENTS

CHAPTER1-INTRODUCTION	6
PROBLEM.....	6
PROJECT GOALS	7
PROJECT RELEVANCE TO THE MSC	7
BREAK-DOWN OF THE THESIS.....	8
CHAPTER2-LITERATURE REVIEW	9
PROKARYOTIC VS. EUKARYOTIC ANNOTATION.....	9
AB-INITIO GENE FINDING	10
HOMOLOGY BASED GENE FINDING	13
COMPARATIVE GENE FINDING.....	14
CHAPTER3-DESIGN AND ALGORITHMS	16
REQUIREMENTS ANALYSIS	16
ALIGNMENT.....	19
PROFILE CREATION.....	21
INDEPENDENT ANNOTATION VIA MARKOV MODELS	23
COMPARATIVE ANNOTATION	29
CHAPTER4-TESTING AND EVALUATION.....	31
SELECTION OF GENE	31
SELECTION OF TEST SET	31
TRAINING THE MARKOV MODEL	32
TESTING STRATEGY.....	32
NULL CASE.....	33
TEST 1.....	34
TEST 2.....	37
TEST 3.....	39
CHAPTER5-CONCLUSION AND FUTURE WORK.....	42
FUTURE WORK	43
APPENDIX A- BIOLOGY BACKGROUND.....	44
DNA	44
GENES.....	44
RNA	45
PROTEIN SYNTHESIS	46
TRANSCRIPTION.....	46
TRANSLATION	46
APPENDIX B- MAJOR MODULES FROM SOURCE CODE.....	48
BIBLIOGRAPHY	56

List of Figures

Fig 1: Typical Prokaryotic gene Structure. [TCGF].....	9
Fig 2: Typical Eukaryotic Gene Structure [MM]	10
Fig 3: A fifth order markov model. [BMI]	11
Fig 4: The architecture of (a) standard HMM and (b) generalised HMM. [RM]	12
Fig 5: A GPHMM for alignment and prediction of exons [LMS]	14
Fig 6: Use case diagram for the system	18
Fig 7: System Architecture	19
Fig 8: Profile Creation	22
Fig 9: Sequence Comparison with Profile	23
Fig 10: Length-5 context string ATGCG.....	25
Fig 11: Part of a context model showing conditional probabilities	26
Fig 12: Different reading frames with respective start and stop codons.....	27
Fig 13: Scoring each position with 5 models.....	28
Fig 14: The conservation pattern of a sample sequence.	30
Fig 15: BLAST hit for gene lepA	32
Fig 16: The NULL case. No gene was present	33
Fig 17: Test Case size 15(a).....	35
Fig 18: Test Case Size 15(b).....	36
Fig 19: Test Case Size 30(a).....	37
Fig 20: Test Case Size 30 (b).....	38
Fig 21: Test Case Size 70.....	40
Fig 22: No. of Sequences vs. Annotation Confidence Score.....	41
Fig 23: DNA Structure. [AEGL1].....	44
Fig 24: Gene Structure [AEGL2].....	45
Fig 25: RNA structure [RSP].....	46
Fig 26: Transcription and Translation. [BP].....	46
Fig 27: Protein Synthesis. [AEGL3]	47

Chapter1-INTRODUCTION

Problem

The genomic sequencing projects are moving into a new era. In recent years complete genomes from diverse eukaryotic organisms have been sequenced. The sequencing of closely related genomes both between and within organisms has now begun. Once a genome is sequenced, it needs to be annotated to make sense of it. Genome annotation is the process of identifying the locations of genes and all of the coding regions in a genome and determining what those genes do. Comparative genomics is the analysis and comparison of genomes from different species. The purpose is to gain a better understanding of how species have evolved and to determine the function of genes and non-coding regions of the genome. The coming years will see the number of known bacterial genomes go into the thousands, which gives a unique chance to apply comparative annotation on a massive scale, which will allow much more ambitious goals and also create computational challenges. For instance, researchers have learnt a great deal about the function of human genes by examining their counterparts in simpler model organisms such as the mouse.

Many different approaches for gene annotation exist in bioinformatics literature. A detailed review of these techniques is given in the background section. Most of these techniques fall into one of the three categories; Ab-initio techniques, similarity based techniques and comparative techniques. With the increase in number of sequenced genomes the comparative approaches are becoming the most relevant and popular of the three. However current comparative approaches suffer from the main problem of computational complexity. The techniques used in these approaches restrict them to two or at most three sequences, thus clearly failing to take advantage of the rapid pace of sequencing.

CHAPTER1-INTRODUCTION

The MSc project aims to address this particular problem. The aim therefore is to design a comparative annotation technique that can push the input size from two or three sequences to tens and possibly hundreds.

Project Goals

- Design and implement a comparative annotation system that is computationally efficient enough to process large sets of sequences.
- Test and evaluate the system on sets selected from prokaryotic sequence databases available online.

Project relevance to the MSc

The project was initially selected from a list of potential research topics suggested by Dr. Hein. Since the project was primarily involved with the field of bioinformatics, the *Bioinformatics and Computational Biology* course taken in Michaelmas term was highly relevant and beneficial. The main modules covered during the course were phylo-genetics and sequence alignment, which were directly related to the project focus.

Information Retrieval also turned out to be a very relevant module due to its focus on techniques of statistical machine translation and speech recognition. Specifically, the methods of hidden markov models discussed repeatedly throughout the course came in very handy during the implementation of the markov model for the ab-initio annotation module.

Throughout the design and implementation phase, the emphasis on formal analysis and specification stressed during virtually all MSc modules including *Concurrent and Distributed Programming* proved to be of great value in reasoning about the various tradeoffs between the complexity of technique and computational efficiency.

CHAPTER1-INTRODUCTION

Although the author did not take the final assignment for the *OOP* module, attending the classes for most part of the term helped a lot later on in the implementation phase, as the design involved the integration of several modules, each with a very well defined and separate set of operations.

Break-down of the thesis

Chapter 2 provides a background review of popular gene finding algorithms and techniques in literature. Some commonly used tools and their shortfalls are discussed.

Chapter 3 deals with the requirements analysis and algorithm design for the project. The complete set of requirements for the project is discussed along with a thorough explanation of the methods and algorithms designed to achieve the requirements.

Chapter 4 describes the testing and evaluation strategy and the results obtained from the project implementation.

Finally, Chapter 5 concludes the thesis and also highlights some potential extensions to the project for the future.

Chapter2-LITERATURE REVIEW

Gene Finding

Gene annotation refers to applying algorithmic techniques in order to identify regions in DNA sequences that are biologically functional. The primary motifs of interest in this respect are protein coding genes, although there may be others such as RNA genes and regulatory regions.

Prokaryotic vs. Eukaryotic annotation

Gene finding in prokaryotic DNA is a relatively much easier problem than the corresponding annotation of eukaryotic sequences. This is because prokaryotic DNA has very high coding density, the sequences are much smaller than eukaryotic DNA and most importantly the gene structure is much simpler since the protein coding regions are continuous (there are no introns). Typical prokaryotic genomes are around 0.5 Mbp (Million base pairs) to around 10 Mbp, and the gene density is $> 85\%$.

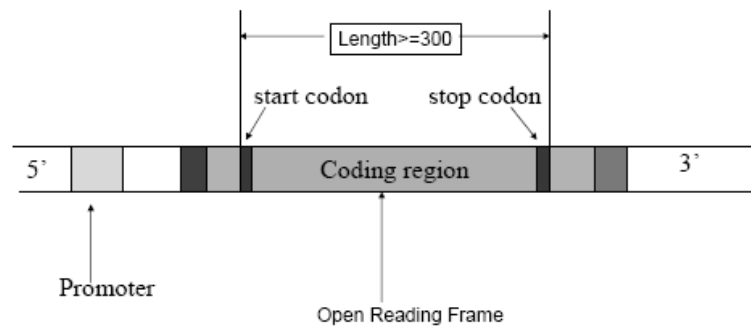


Fig 1: Typical Prokaryotic gene Structure. Each coding region is a continuous sequence. [TCGF]

The gene structure and the gene expression mechanism in eukaryotes are far more complicated than in prokaryotes. Typical eukaryotic genomes range from 2.9 Mbp to 4000 Mbp and the coding density is much lower than prokaryotes. For instance the

CHAPTER2-LITERATURE REVIEW

human DNA contains on average only around 1 gene per 40-45kbp. Additionally, in typical eukaryotes, the region of the DNA coding for a protein is usually not continuous. This region is characterized by alternating regions switching between *exons* and *introns*. During the process of transcription these regions are transcribed onto the RNA, preserving the order. After this, as part of the process of splicing the, intron sequences are removed from the RNA sequence. The remaining RNA segments relating to exons form the RNA.

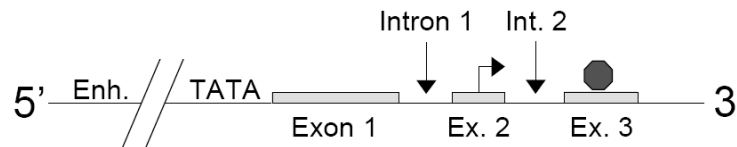


Fig 2: Typical Eukaryotic Gene Structure: Coding region interspersed with introns. [MM]

Ab-initio gene finding

Originally, ab-initio methods were defined for prokaryotic genomes. In prokaryotic sequences, genes are defined by sufficiently long coding regions which are characterized by the absence of stop codons. Perhaps one of the simplest techniques for gene finding therefore is to search for sufficiently long regions that do not contain stop codons. These regions between start and stop codons are known as Open Reading Frames.

In order to have more accurate representation of regions in sequences as functional or non-functional many other measures that take into account the difference in composition of the regions that are functional and those that are not have been introduced. Quite a lot of the recent gene-finding methods involve Markov models of coding and non-coding regions. A Markov model of order 'm' can represent conditional dependencies in oligomers of length m+1. For instance, fifth-order models can be used to represent dependencies in hexamers. The model may be homogeneous or non-homogeneous depending upon whether all positions are same, or whether the transition probabilities differ with the positions.

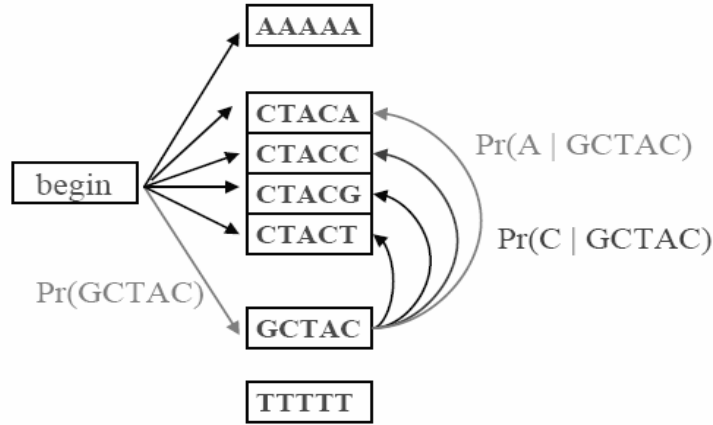


Fig 3: A fifth order markov model. [BMI]

The parameters for such models are usually estimated by making use of the maximum likelihood method, which takes into account the observed conditional frequencies of an appropriate training set of known genes to estimate the corresponding conditional probabilities. Recently, an improved approach using interpolated Markov Models has been proposed. In this approach, the conditional probabilities of several models of different orders are taken together for annotation.

Several recent methods involve explicit models of gene structure and require the components of a gene to follow a strict order. They also allow models for the length distributions of functional and non-functional regions. Hidden Markov Models (HMM) are a popular class of probabilistic tools that can generate sequence models. The approach, was originally used for gene finding by Haussler[HAUSS94] and colleagues and also used by Henderson[HEND97]. In HMM's the transitions between various components of a gene can be modeled as markov processes which are hidden. These hidden processes control the generation probabilities of nucleotides, which are the observables. The HMM architecture has been extensively applied to a myriad of problems in computational biology as well as other fields such as speech recognition.

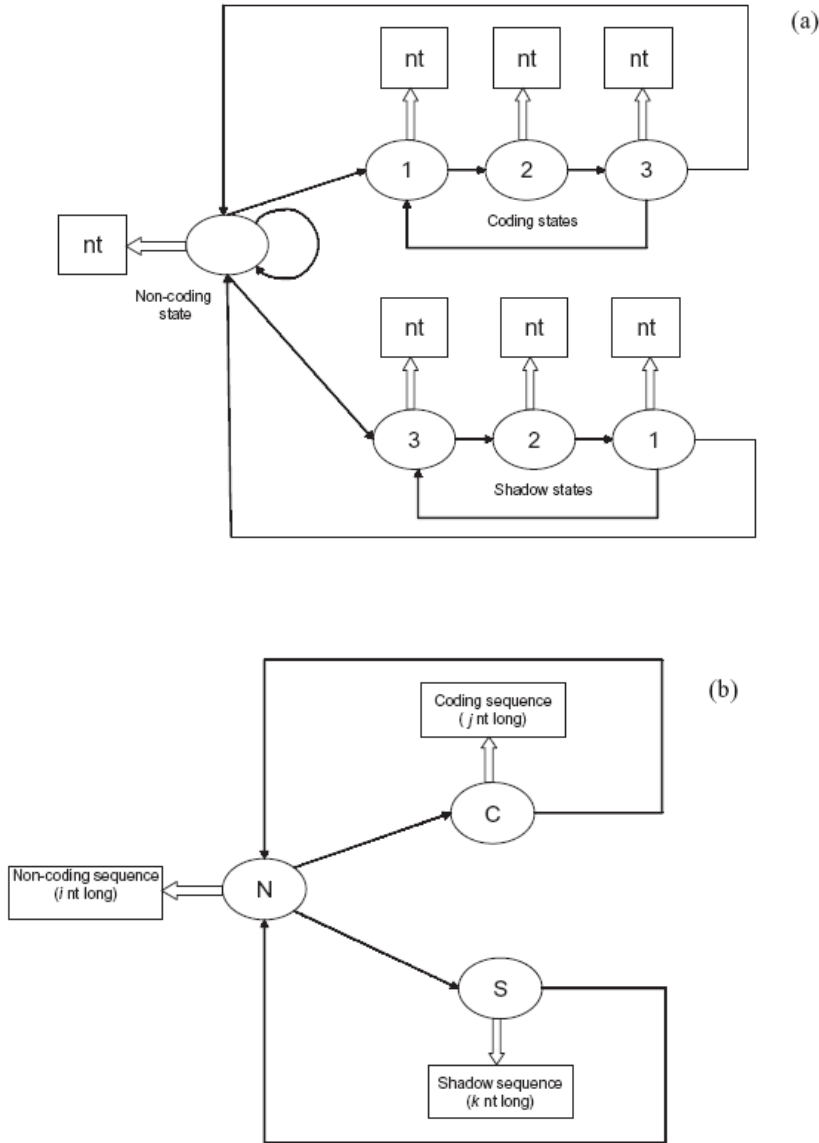


Fig 4: The architecture of (a) standard HMM and (b) generalised HMM. In the HMM each state emits a nucleotide and then transitions to the next state. In the GHMM each state emits a string of nucleotides and then transitions to another state. [RM]

The HMM approach is limited by the fact that it cannot very accurately model the exon and intron lengths since they do not follow geometric distributions. In the more general models used by GENSCAN[GENS97] and GENIE[GE96], DNA nucleotides are generated according to the Generalized Hidden Markov Model (GHMM). In this architecture, each state can generate strings of varying length. Thus the emitted string as well as its length at each state is a randomly chosen. Another improvement in annotation

CHAPTER2-LITERATURE REVIEW

techniques in this category is that programs like GENMARK[GM93] make use of double-stranded models and can model the existence of multiple genes, while most previous tools could only analyze a single strand at a time and assumed the existence of a single gene.

Homology based gene finding

Homology based methods exploit a sufficient similarity between a genomic sequence region and a protein or DNA sequence present in a database in order to determine whether the region is coding. In order to detect similarity between sequences, the primary tools are local alignment methods such as the optimal Smith–Waterman algorithm and heuristic approaches such as FASTA[FST90] and BLAST[BST97].

In general, comparison with three types of sequences is carried out. The most commonly used are protein sequences found in databases such as SwissProt or Pfam. It is estimated that almost 50% of the genes can be identified as a result of similarity score with a homologous protein sequence.

Several powerful gene-finding programs using homology information have been developed in recent years. The ORPHEUS[ORP98] program selects the ORFs with high similarity to proteins in databases and the sequences of these ORFs are used to train models of protein coding sequences. The CRITICA[CRIT99] algorithm computes a coding score for each codon using database search and combines this score with a di-codon frequency score. CRITICA uses an iterative procedure for learning the frequencies of di-codons. The Bio-Dictionary approach uses a database of patterns describing a sequence space of proteins. The patterns are called ‘seqlets’ and are generated from protein databases through the Teiresias algorithm. In order to do gene annotation, the seqlets are matched against amino acid translation of an ORF. If the number of matching seqlets significantly exceeds a certain limit, the ORF is predicted as a gene.

In general, the primary strength of similarity-based algorithms is that predictions rely on already collected biological data, with the possible negatives of low database quality.

Comparative gene finding

As more and more genomes from different species become available, a new class of tools has been developed to use genome comparisons for annotation. The primary assumption at the foundation of these techniques is that functional regions such as genes should be comparatively more conserved than non-functional regions between different organisms. Different programs have been proposed to exploit this idea in different ways. Some tools implement this idea as an extension of pairwise sequence alignment. The aim is to predict genes in each of the two genomes based on the alignment that gives a maximum score. More advance approaches use more sophisticated models of coding and non-coding DNA regions, in conjunction with sequence similarity. Programs such as SLAM[SLAM03] and DOUBLESCAN[DSCAN02] use the Generalized Pair HMM (GPHMM) approach, in which the gene prediction and sequence alignment are carried out simultaneously, instead of the prediction being based upon the alignment.

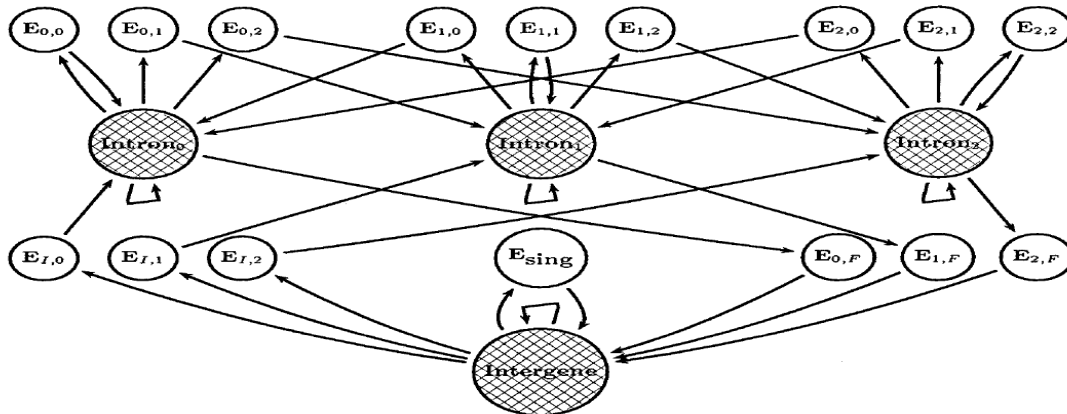


Fig 5: A GPHMM for alignment and prediction of exons using genomic DNA from two different organisms. [LMS]

In the Evolutionary Hidden Markov Models (EHMMs) approach, Hidden Markov Models are used in conjunction with phylogenetic trees to simulate the varying rate of evolutionary events between functional and non-functional sites. EHMMs have been used for eukaryotic sequence annotation with success [HEIN03].

CHAPTER2-LITERATURE REVIEW

Some other tools implement a separation between gene prediction and sequence alignment. The programs ROSETTA[*ROS00*] and SGP1[*SGP03*] are examples of this approach. All these programs start by aligning two regions and subsequently generate annotations that are in agreement with the alignment. This approach is based on the conservation of gene structure in the regions that are being aligned. With the rapid increase in the number of sequenced genomes from species that are at varying evolutionary distances to each other there is an ever increasing interest in techniques that predict genes based on the comparative analysis of much more than a couple of genomes.

The primary difference between homology based annotation and comparative annotation is that the former uses a sequence's similarity with already known databases of proteins to determine if it is coding or not. Comparative annotation however compares two sequences, both of which might be un-annotated, and uses the information about evolutionarily conserved regions to identify functional regions.

Chapter3-DESIGN AND ALGORITHMS

Requirements Analysis

As a result of the thorough review of current annotation techniques given in Chapter 2, the following conclusions were reached.

1: Homology based approaches depend too heavily on previously available biological information. In case of novel genes and poor database quality, this approach simply is not practical.

2: Ab-initio gene finding techniques might be applicable to novel genes however they generally tend to display a high rate of false positives. Additionally, they fail to make use of the extremely rich information source available in the form of large numbers of genomes from related species

3: Comparative approaches are the most promising and clearly the way forward, however current approaches are too computationally expensive to serve the purpose of this project; namely to comparatively analyze tens of prokaryotic genomes.

It is the last observation that the project really focuses on. The main aim therefore was to come up with a comparative technique that is extensible to potentially hundreds of genomes. The two main functional requirements identified for the proposed annotation system were:

- The system must use comparative analysis.
- The system must be capable of handling large input sets (> 50)

As there is a conflict between Requirements 1 and 2 ('comparative' equals more complexity), in order to fulfill both these requirements it was decided to do a trade-off by following a middle approach. This was achieved by including two separate modules in

CHAPTER3-DESIGN AND ALGORITHMS

the design; a simple and therefore computationally tractable comparative module and a separate ab-initio module. The results from both modules are combined in order to give very accurate as well as fast results. Another consideration as part of the requirement analysis was whether to do the alignment and annotation simultaneously. Again, for the sake of efficiency, it was decided to treat them separately. Furthermore, several highly efficient multi-sequence aligners exist in the literature, capable of aligning quite a large number of sequences of the lengths anticipated in this project in a reasonable time. It was therefore decided to make use of one of these instead of re-inventing the wheel and instead to focus on how to use the alignment for annotation. For the actual implementation therefore, the modules selected were: the profile creation and analysis module, the ab-initio module and the comparative module. The algorithms for these modules were designed and implemented from scratch.

The use-cases anticipated for the proposed project mainly related to sequence submission in multi-fast format and inputting a set of training sequences for the ab-initio module , as well as creating a profile , carrying out ab-initio annotation , and finally to do the comparative annotation.

The user therefore interacts with the system in five ways:

- Align Sequences
- Train the ab-initio model
- Do ab-initio annotation
- Create profile
- Do Comparative Annotation

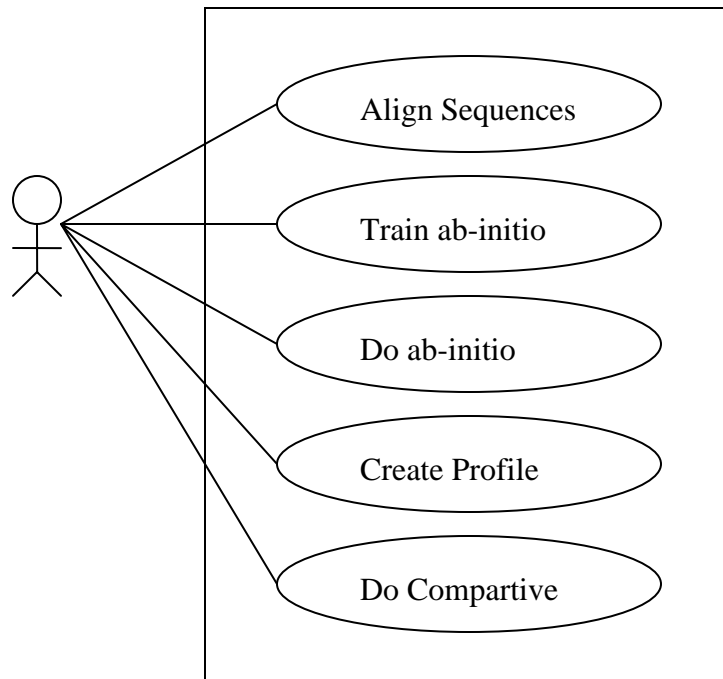


Fig 6: Use case diagram for the system

Based upon the requirements and use case analysis, the following set of goals was finally identified:

1: Design and implement a module that allows multiple sequence alignment of a large number of sequences as input and then extracts crucial information about sub-sequence conservation from the alignment. The emphasis here will be on efficient and accurate indication of common motifs.

2: Design and implement a module that carries out gene annotation on a set of bacterial genome sequences, treating each sequence independently. This module finds potential coding regions using an ab-initio technique, specifically markov models.

3 : Design and implement a module that can combine the results from the two modules above to give a much more reliable annotation. The module will use the conserved motif information from the profile creation module to refine the results of the independent annotation module.

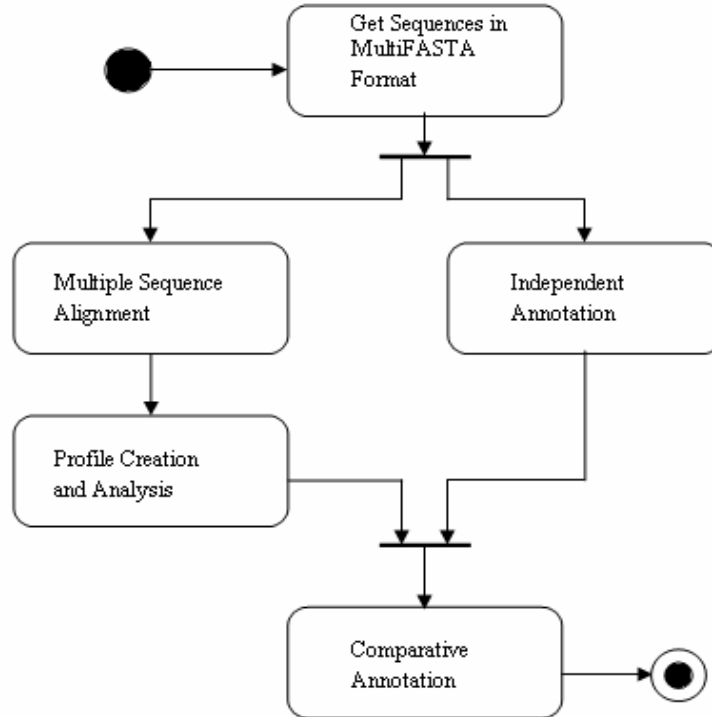


Fig 7: System Architecture

In the following sections, the algorithmic details of each of the main modules in the system are given.

Alignment

The first step in the comparative annotation pipeline was the multiple sequence alignment. There were several issues involved in this:

1: The alignment should be temporally efficient enough for the large number of sequences involved. This meant a straight out rejection of the standard dynamic programming model for multiple sequences, as this would involve an N dimensional table and take $O(N^k)$ time, where k is the sequence size.

2: The alignment should be able to preserve conserved blocks in the sequences as the entire annotation depends upon it. This would mean looking into both global alignment

CHAPTER3-DESIGN AND ALGORITHMS

and local alignment tools. Global alignments are good for sequences that are very closely evolutionarily related whereas local alignment. For this project, the bacterial genomes were assumed to be quite close evolutionarily, therefore the choice of local vs. global was not very important.

Based upon the above two requirements and a thorough review of the alignment literature, we chose the ClustalW [CW94] algorithm for the alignment. ClustalW is a popular, freely available multiple sequence alignment algorithm that aligns multiple sequences in an iterative manner. The algorithm proceeds as follows:

1. Calculate pair-wise alignments between all possible sequence pairs and find out the degree of similarity between them.
2. Find out a guide tree for the multiple alignment. This is done by first calculating a distance between each pair by counting the number of mismatches in each alignment. A distance matrix is created with these distances and this matrix is used to build the guide tree, using Neighbor-Joining method, where the branch lengths signify the distances. The root is placed in the middle of the longest chain of consecutive edges.
3. Combine the alignments from 1 in the order specified in 2 such that the gaps once added are always preserved.

As the algorithm follows a progressive approach, aligning the sequences pair-wise and then combining the pair-wise alignments, instead of simultaneously aligning all sequences, the dynamic programming approach can be used during the pair-wise alignment phase without making the computations infeasible.

The output from the multiple alignment module is in multi-fasta format which can easily be processed by the profile generation module.

Profile Creation

The alignment of a large set of genomes can be used to generate a profile for the entire set, which summarizes the information in the alignment, and can subsequently be used for database searches for homologous sequences that belong in the family.

The central idea for this module is that a profile can be used to score each sequence in the set, by running it against the profile and identifying regions that score relatively high based upon some scoring criteria. Since the profile is a sort of consensus sequence, close similarity with the profile would indicate regions that are conserved in a relatively high percentage of the sequence set. Under the assumption that the functional regions in related genomes are much more likely to be conserved due to evolutionary constraints, these regions of high similarity to the profile can then be marked as probable gene segments.

The profile itself is created in the form of a frequency matrix, calculated from the multiple alignment. The frequency matrix stores the probability of occurrence of each nucleotide for every column of the alignment.

The probability score for the nucleotides at any column in the alignment is,

$$P(x) = \frac{f(x)}{\sum_{y \in \{a,c,g,t\}} f(y)}$$

where $f(x)$ is the number of times nucleotide x appears in the column.

The profile is saved in the form of a linear array *probarr* that stores the relative frequencies of all four residues at each position in the alignment. This array is then used for the subsequent comparison with individual sequences.

CHAPTER3-DESIGN AND ALGORITHMS

Seq 1	G	T	T	T	C	T	T	T	A	T	G	G	T	T	A	G
Seq 2	G	C	T	A	C	A	T	C	T	T	G	T	C	.	.	C
Seq 3	A	T	A	G	T	A	T	A	T	T	A	T	A	.	.	T
Seq 4	A	A	T	T	C	C	T	T	T	T	A
Seq 5	G	A	T	G	A	T	C	G	C	T	A	C
Seq 6	C	T	A	C	C	C	C	G	G	C	C	T	T	.	.	T
Seq 7	A	T	G	T	A	A	A	A	C	T	C	C	T	.	.	T
Seq 8	A	A	G	C	G	T	A	A	C	T	A	C	C	.	.	C
Seq 9	C	A	T	G	T	T	C	T	T	T	G	T	G	A	G	T
Seq 10	C	G	A	C	G	A	A	C	T	T	G	T	T	.	.	G
<hr/>																
P (A)	0.4										0.0					
P (G)	0.3										0.0					
P (C)	0.3										0.1					
P (T)	0.0										0.9					

Fig 8: Profile Creation

Once the profile is created, each of the genome sequence in the set is then compared to it thereby indicating areas in the sequence that are potential candidates for genes.

The algorithm for this is:

For each sequence in the set:

For each position(x) in the sequence:

If (x=gapcharacter) then ignore

*Else ConservedScore(x) = $\sum_{i \in \{a,c,g,t\}} P(i) * \delta_i$*

where $P(i)$ are the positional probabilities calculated for the profile and δ_i represents a weight parameter that can be used to take into account cases such as conserved substitutions. In the simplest case the weights are defined as:

$$\delta_i = \begin{cases} 1 & \text{if } i = x \\ 0 & \text{otherwise.} \end{cases}$$

However these can be given different values, for instance penalizing A-T transitions much more than C-G ones or any other combination depending on the availability of extra information about the data set. The weights can be changed via entries in a substitution matrix.

P (A)	0.4	0.1	0.8	0.6	0.0
P (C)	0.3	0.2	0.2	0.3	0.0
P (G)	0.2	0.3	0.0	0.1	0.0
P (T)	0.1	0.4	0.0	0.0	1.0

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

Fig 9: Sequence Comparison with Profile

As a result of this, regions in the alignment which exhibit a high degree of conservation will stand out as segments with high probability scores. These segments will tend to stand out from the flanking regions in the genome sequence.

Columns in the profile that consist entirely of gap characters are ignored in the analysis, as they provide no usable information. Similarly, when there is a gap character at a particular position in the sequence being compared to the profile, while the profile may have non-zero probabilities for some nucleotides at that position, the column is not included in the analysis for that particular genome.

Independent Annotation via Markov Models

Markov chains and hidden markov models are very popular tools for pattern recognition and are especially prevalent in gene prediction literature. For the purposes of the

CHAPTER3-DESIGN AND ALGORITHMS

alignment-independent portion of the annotation pipeline, we decided to make use of the markov chain method.

A first order markov chain is defined as :

$$P(X_{n+1} = j | X_0 = i_0, \dots, X_n = i_n) = P(X_{n+1} = j | X_n = i_n)$$

That is, the conditional probability of any event is dependent only upon the previous event. In the context of DNA sequences, markov chains can model the dependency of a residue on a certain number of past residues. The sequence of past residues upon which the current residue is dependent is called the 'context sequence'. More specifically ,if we have a sequence of characters ACGTACGTACGT , then the length-2 context sequence for each residue would be the previous two residues and a second order markov would approximate the conditional probability of the 'C' at position 6 as :

$$P(C | ACGTA) = P(C | TA)$$

In general, by increasing the order of the model we get more precise approximation of the 'history' of an event and thus more predictive power. However high order markov chains suffer from two drawbacks:

1: State space explosion

As the order of a markov model is increased, the number of context sequences for which probabilities need to be estimated grows exponentially. In general for a markov model of order k, we need $4^{(k+1)}$ probabilities. For a 6th order markov model a total of $4^7 = 16384$ different probability values will be needed.

2 : Unavailability of data for long contexts

For long context sequences, the number of occurrences in the training set may be too less to get a reasonable estimate of the corresponding probability. For instance if we are using

markov models of order ten , then many of the possible length-ten context sequences will have close to zero occurrences in the training sequences and so their probabilities cannot be correctly estimated.

One way of avoiding the absence of training data for higher order markov models reported in literature is through the use of variable length markov chains. Variable length markov chains use a mixture of models from order zero up to a specific maximum. The selection of which models to use is based on the availability of sufficient training data in the training set for each model. Thus for building the context models, we can use longer contexts when they are present in the taining data, and shorter contexts when they are absent.

ATGCGTAAGGCTTTCACAGTATGCGAGTAAGCTGCGTCGTAAGG

Fig 10: Length-5 context string ATGCG

Step 1a: Generation of Context patterns with probabilities

The first step is to generate a context model, which stores all possible context sequences as well as the related probabilities from length 1 to length k, which in the case of this particular implementation is set to 5. These context sequences are used by markov models of the corresponding order. The context model is learned from training sequences, which were extracted from NCBI bacterial genome databases using a Blast search.

The conditional probability of observing a symbol after any context sequence is defined as the ratio between the number of times the symbol appears following the context sequence and the total number of times the context sequence appears in the training set.

Specifically, let $S_{x,i}$ denote the number of occurrences of string x followed by i in the training set, then the conditional probability of i following x is,

$$P(i | x) = \frac{S_{x,i}}{\sum_{r \in \{a,c,g,t\}} S_{x,r}}$$

The algorithm for generating the context model is:

For each context length 'L' from 1 to k
For each possible context pattern 'G' of length L
Find all occurrences of G in training set
Calculate and Store $P(i | G) = \frac{S_{G,i}}{\sum_{r \in \{a,c,g,t\}} S_{G,r}}$

	A	C	G	T
AAC	0.39	0.20	0.01	0.4
AAG	0.25	0.35	0.10	0.30
AAT	0.15	0.12	0.33	0.40
AAA	0.53	0.17	0.27	0.03

Fig 11:Part of a context model showing conditional probabilities for context sequences of length 3. The complete context model will contain probabilities for all possible context sequences of length 1 through k.

Step 1b: Extraction of long ORFs :

An open reading frame or ORF is a part of a genome that could potentially encode a protein. ORFs contain no stop codons. As a result, ORFs are searched using the start codon ATG and the stop CODONS TAA, TAG TGA. Although ORF's may also belong to non-coding areas, in prokaryotes sufficiently long ORFs (>100 bp) generally are a strong indication of the presence of a gene.

CHAPTER3-DESIGN AND ALGORITHMS

Since for double stranded DNA there are six possible reading frames, ORF's are searched for using each one of them. For example, the sequence of DNA in Fig(12) can be read in a total of six reading frames. Three of these frames are in the forward direction and the other three are in the reverse direction. In all the frames, the stop codons are indicated by a '*'. In this case frame 1 contains the longest open reading frame

```
5'                                     3'
                                     atgcccaagctgaatagcgtagaggggttttcatcatttgaggacgatgtataa

1 atg ccc aag ctg aat agc gta gag ggg ttt tca tca ttt gag gac gat gta taa
  M  P  K  L  N  S  V  E  G  F  S  S  F  E  D  D  V  *
2  tgc cca agc tga ata gcg tag agg ggt ttt cat cat ttg agg acg atg tat
  C  P  S  *  I  A  *  R  G  F  H  H  L  R  T  M  Y
3  gcc caa gct gaa tag cgt aga ggg gtt ttc atc att tga gga cga tgt ata
  A  Q  A  E  *  R  R  G  V  F  I  I  *  G  R  C  I
```

Fig 12: Different reading frames with respective start and stop codons.[BW]

To extract open reading frames from the set of sequences, the following algorithm was designed:

For each possible reading frame,
Read nucleotides in groups of 3 (codons) until stop codon
Read backwards from the stop codon and locate start codon
If (length(ORF) > threshold)
Store the orf in a permanent list along with its reading frame
Remove all orf's completely contained by others.

Step 2: Scoring the ORF's :

Once we have a list of potential coding regions through ORF search, we score these regions using markov chains. The score for each region is calculated as follows:

1: A position score is calculated for each position along the length of the ORF. This position score is a sum of the probability scores from each of the k-1 markov models.

$$\text{Position Score at Position } i = \sum_{r=i+1}^{i+k} P(S_r | S_i, \dots, S_{r-1})$$

where S_r is the residue at position r .

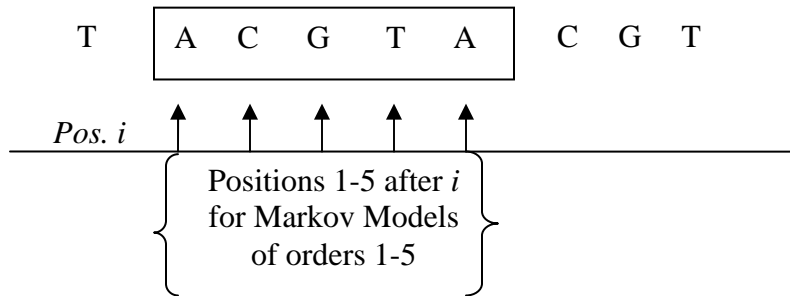


Fig 13: Scoring each position with 5 models

2: The position scores over the entire length of the ORF are multiplied to give the total score of the region. To avoid underflow due to very small probability scores, logarithmic calculations are used for this part.

$$\text{Total Score} = \sum_{i=1}^n \log(\text{pos.score}_i)$$

where n is the length of the ORF

The algorithm for computing the score for an ORF is given below:

```

For i = 1 to length
  Pos.Score = 0
  For r = i + 1 to i + k
    Pos.Score = Pos.Score + P(Sr | Si,...,Sr-1)
  Total.Score=Total.Score + log ( Pos.Score)

```

Based upon the scores calculated for all ORF's, a list of likely genes is created by removing ORF's scoring below a minimum threshold. This list of genes along with their coordinates is then used in conjunction with the comparative part of the project for annotation purposes.

Comparative annotation

The terminal module in the annotation is responsible for combining the results of the markov model and the profile builder. The module uses the information about the relative degree of conservation along the entire length of each genome sequence to filter out the results from the markov model. The markov model module will in general output a number of false positives along with regions that have a high coding potential. The real benefit of the comparative approach is utilized at this stage by removing all those regions which fail to satisfy a user selectable degree of conservation. Regions which exhibit a significantly higher conservations score as compared to the baseline score of the entire sequence, are then finally output as coding regions.

The algorithm for the comparative module is the last step in the annotation system. It finally outputs all the potential genes which have been found by the combination of profile analysis and ab-initio annotation. The algorithm is given below:

For each genome 'k' in the set ,

$$Avg.Score(k) = \frac{\sum_{i=1}^N ConservedScore(i)}{N} \quad [N \text{ is sequence length }]$$

For each ORF 'o' in the ORF list for the genome k,

$$Avg.Score(o) = \frac{\sum_{i=1}^L ConservedScore(i)}{L} \quad [L \text{ is ORF length }]$$

If $\frac{Avg.Score(o) - Avg.Score(k)}{Avg.Score(k)} \geq threshold$

Add ORF to gene list for genome k

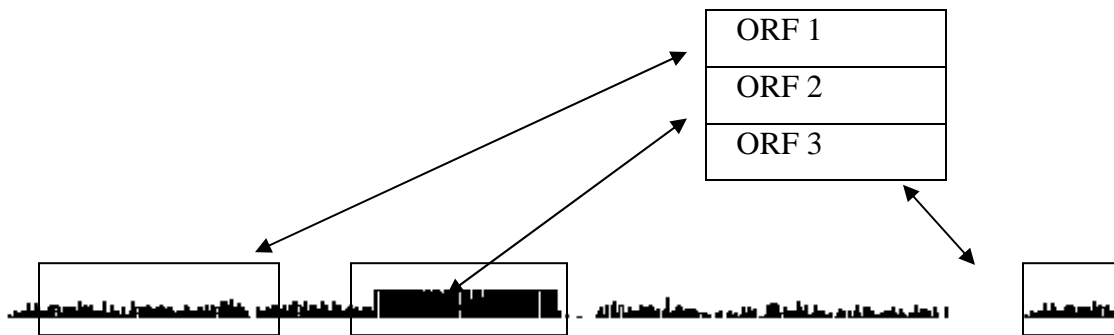


Fig 14: The conservation pattern of a sample sequence. The markov model module output three ORF's for this sequence , whereas the conservation pattern obtained from the profile module indicates that only one of these has coding potential.

Chapter4-TESTING AND EVALUATION

Selection of gene

The project aim was to explore comparative annotation at very large scales. As such, for testing purposes a very large number of genomes with a particularly common gene were needed. In order to collect this test set, therefore an analysis of the most common genes was carried out, and as a result of the analysis, the lepA gene was selected for testing. The lepA is one of the most conserved proteins and is present in all bacteria and mitochondria with the amino acid identity in bacteria being around 55-68%.

LepA Clostridium kluyveri DSM 555

```
mhterqkhir nfvahidh gkstladrli ektgtltere mnsqvldnme lekergitik
sqairliykr kdggeyvlnl idtpghvdfn yevsrslaac egailvvdatt qgiqaqtlan
cylamehdle ilpvinkidl psaraeevke eiediigiea seaplvsakt glnieqvlea
ivdkipspgg denaplkali fdsnydsykg vvchirvkeg nikpgdeikl matgkiyevt
etgifvpmfm praelragdv gyftasiknv rdarvgdtvt gahnqakepl kgyrvismv
ysgiypvdga kygelkeale klqvndaaln fepetsvalg fgfrcgflgl lhmdviqerv
erefnldiit tapsviykig ktdgtvvelt nptnmpvse ikymeepivk asiitpseyv
gavmelaqnr rgvfrdmqyi ettrvslnyd iplneiynf fdvlksrtrg yasldyelkg
yxsaklvrlld vllngdmvda lsmivpeera ydrgrgiaek lkgiiprqlf eipiqaaavgg
kviaretvka mrkdvlakcy ggdisrkrkl lekqkegkkr mrqvgtveip qeafmailkt
ee
```

Selection of Test Set

After selecting the gene, the set of bacterial genomes for the annotation was collected from the NCBI microbial database. A BLAST search was carried out using the protein sequence of LepA in clostridium kluyveri DSM 555, and the top 100 hits were then analyzed. The vast majority of these hits were bacterial genomes and thus they were readily included in the test set. A region of around 10Kbp, containing the hit was extracted from each genome.

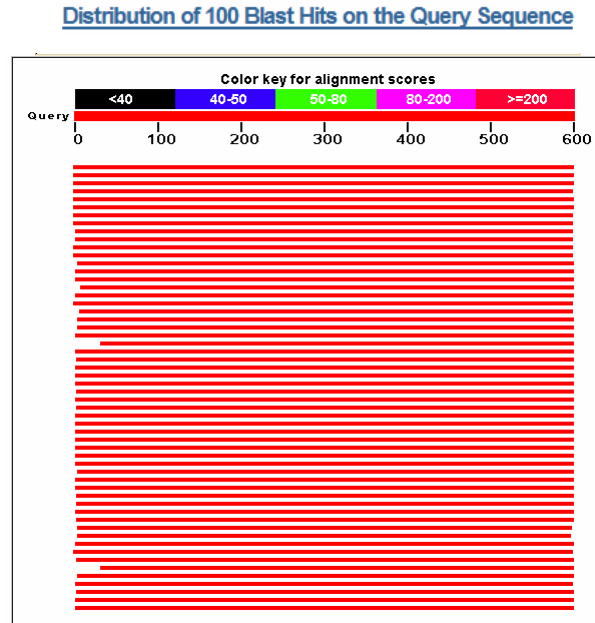


Fig 15: BLAST hit for gene lepA

Training the Markov Model

Before the markov model module could be used for ab-initio annotation, the probability parameters had to be learnt, through the use of a context module. The context model was generated using Eq(3) , and the training set for generating it was extracted from within the 100 sequences chosen initially as part of the test set. However the sequences used for training were not subsequently used in the actual tests(1-3).

Testing Strategy

To test the accuracy of the system and evaluate the actual advantage of the comparative approach, the following testing strategy was adopted:

1: Carry out three separate tests, with genome sets of sizes 15, 30 and 70 respectively. The tests for sizes 15 and 30 will have two sets each in order to check the robustness of the results.

CHAPTER4-TESTING AND EVALUATION

2: Compare the results from the comparative approach in tabular as well as graphical format. The graphs depict the conservation scores along the entire length of a sequence and the tables give the relative scores of the potential ORFs. The table scores are calculate by dividing the average conservation along the ORF by the average conservation of the entire sequence. A value greater than 1 would therefore indicate a region more conserved than the overall sequence.

Null Case

Before the testing on sequences that contained the gene, a null case test was also carried out of a genome sequence that did not contain the gene.

The markov model output the following potential genes in the sequence:

```
>Listeria welshimeri serovar b str.  
orf00001      1376      801  -3  
orf00002      2455     1418  -2  
orf00003      3759     2602  -1  
orf00004      3794     3940  +2
```

The graph for this null case however clearly shows the absence of any such coding regions. As can be seen from graph-0, there is no visible region of increased conservation throughout the sequence, and the scores are pretty much uniformly distributed throughout.

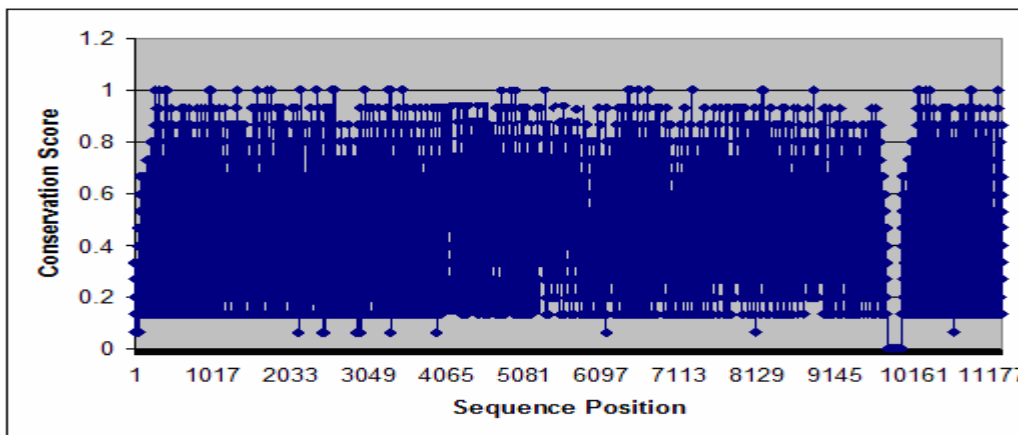


Fig 16: The NULL case. No gene was present

CHAPTER4-TESTING AND EVALUATION

Table 5.1 shows the comparative scoring of the outputs from the markov model , and it is clear that the regions show no increased conservation above the baseline (1).

Start	Stop	Score
1376	801	0.889555
2455	1418	0.912951
3759	2602	1.000867
3794	3940	0.915664

Table 1

Test 1

The first real test consisted of two different sets of 15 genomes each, chosen randomly from the set of 100 selected initially. Two different sets of the same size were selected in order to ensure that the results were robust. The markov module processed each of the sequences independently.

For illustration purposes, the output for only one genome from each set is provided here.

The Markov model module output for a single genome from the first set:

```
>Clostridium kluyveri DSM
orf00001      164      748 +2
orf00003     1044      778 -1
orf00004     1343     2317 +2
orf00005     3986     5809 +2
orf00006     3574     3921 +1
orf00007     2412     3527 +3
orf00008     5821     6972 +1
orf00009     7163     8203 +2
orf00010     8288     8959 +2
```

When the sequence was executed on the comparative module via profile scoring, the following result was obtained.

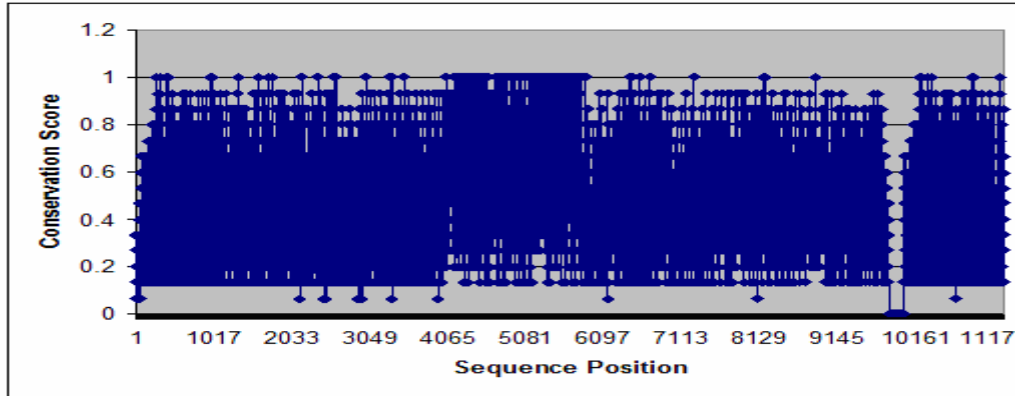


Fig 17: Test Case size 15(a)

As can be seen in the graph, the region where the gene is actually located (3986-5809) shows a slightly higher conservation score than the overall sequence. However the effect is not remarkable due to the comparative annotation making use of only 15 genomes.

The visual results are confirmed when the markov model outputs are scored using the conservation scores from the comparative module, as shown in Table 5.2. Here the actual gene region shows a significantly higher (nearly 20 %) score as compared to the overall sequence, the baseline score being 1.0

Start	Stop	Score
164	748	0.938894
1044	778	0.962106
1343	2317	0.933987
3986	5809	1.170224
3574	3921	0.956179
2412	3527	1.027014
5821	6972	0.9475
7163	8203	1.0947
8288	8959	1.077356

Table 2

Interestingly, some of the putative coding regions indicated by the markov model module exhibit scores that are actually lower than the base-line .This might be due to the baseline

CHAPTER4-TESTING AND EVALUATION

being too high in this case because the sequences are very closely related , so the average scores for even non-coding regions are quite high , by pure chance.

The markov model output for a genome from the second set is:

```
>Lactococcus lactis subsp. lactis Il
```

```
orf00001      8391      24  +1
orf00002       119     727  +2
orf00005     1588     803  -2
orf00008     2938    1589  -2
orf00009     3047    2901  -3
orf00010     3049    3714  +1
orf00012     4001    5824  +2
orf00014     5936    6526  +2
orf00016     6740    6621  -3
```

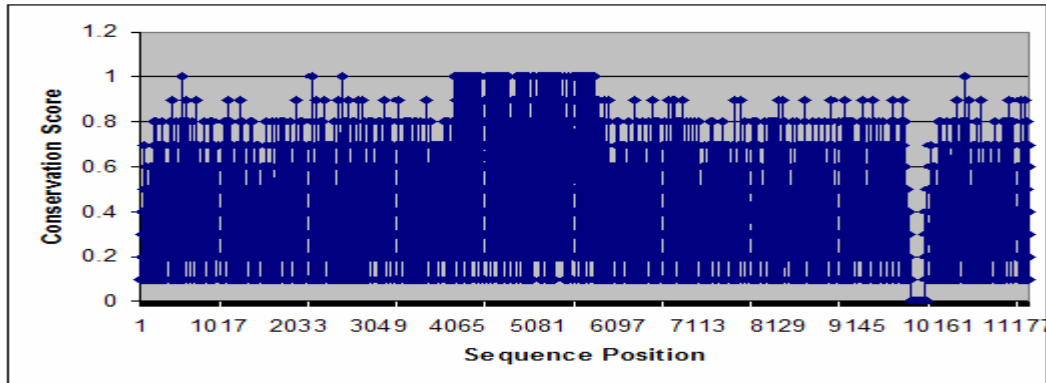


Fig 18:Test Case Size 15(b)

As in the case of the first set, both the graph and the table show significantly higher scores for the gene true area.

Start	Stop	Score
8391	24	0.949325
119	727	0.920434
1588	803	0.903332
2938	1589	0.914897
3047	2901	0.956815
3049	3714	0.934261
4001	5824	1.218595
5936	6526	0.862071
6740	6621	0.903174

Table 3

Test 2

The second test consisted of two sets of 30 genomes each. For comparison purposes, both these sets contained at least 1 genome in common with the sets in test 1.

.The markov model output for the genome in the set first was:

```
>Geobacter sulfurreducens PCA
orf00001      322      690  +1
orf00002      744     1682  +3
orf00005     3795     2458  -1
orf00006     3962     3840  -3
orf00009     4933     3962  -2
orf00010     4886     5803  +2
orf00011     7496     6489  -3
orf00012      354     7653  -3
```

The comparative annotation graph shows a remarkable improvement in the accuracy and confidence of results. As can be seen from the graph, the gene area is now much more distinct as a region of high scores. This is also confirmed from the scoring table of the markov model outputs. In this case the gene region shows an almost 35% greater average score than the baseline.

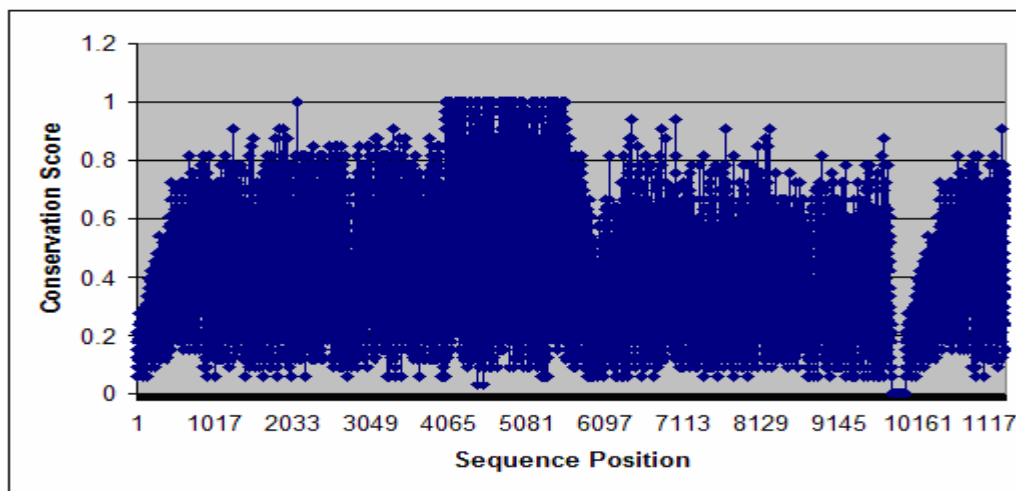


Fig 19: Test Case Size 30(a)

CHAPTER4-TESTING AND EVALUATION

Start	Stop	Score
322	690	0.906224
744	1682	0.938796
3795	2458	1.326481
3962	3840	0.89176
4933	3962	0.910605
4886	5803	0.923016
7496	6489	0.923132
354	7653	0.929897

Table 4

The markov model output for a genome from the second set was:

```
>Streptococcus pyogenes MGAS
orf00001    9899    770    +3
orf00002    767     1108   +2
orf00003   1065    1760   +3
orf00004   1882    2103   +1
orf00005   2144    2341   +2
orf00006   2405    2653   +2
orf00007   2699    3832   +2
orf00008   3816    3941   +3
orf00009   4001    5908   +2
orf00010   6158    5952   -3
orf00011   6147    7361   +3
```

The graph and table for the second set of size 30 also show marked improvement in the identification accuracy.

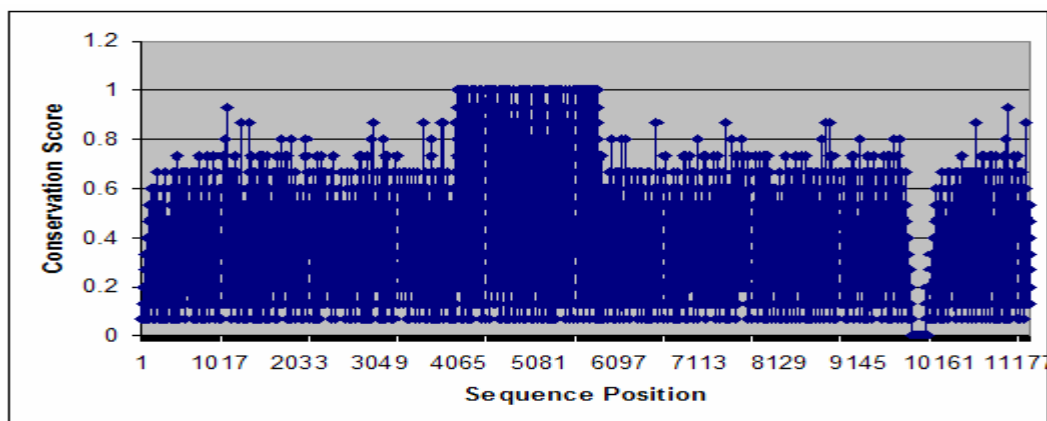


Fig 20: Test Case Size 30 (b)

CHAPTER4-TESTING AND EVALUATION

Start	Stop	Score
9899	770	1.930276
767	1108	0.914141
1065	1760	0.903166
1882	2103	1.351354
2144	2341	0.894183
2405	2653	0.910138
2699	3832	0.925398
3816	3941	0.95724
4001	5908	0.953147
6158	5952	1.016318
6147	7361	1.009457

Table 5

Test 3

The third test consisted of a single set of 70 genomes. This set also contained some genomes from both tests 1 and test 2. The result from the markov model for a genome from this set is:

```
>Myxococcus xanthus DK
orf00001    9361    469  +2
orf00003    3275    3730 +2
orf00004    3322    1607 -2
orf00005    3976    5813 +2
orf00006    6818    7072 +2
orf00007    6859    3920 -2
orf00009    7484    7206 -3
orf00010    9194    9361 +2
orf00011    9238    7460 -2
```

By now the trend is obvious and increasing the number of genomes has resulted in still higher resolution of the genome area. The main difference from tests 1 and 2 is that the flanking regions have now slightly lower average scores than before. This is to be expected, since as the number of genomes in the test set grows, the ratio of sequences with the same residue at positions which are not part of coding regions will tend to decrease.

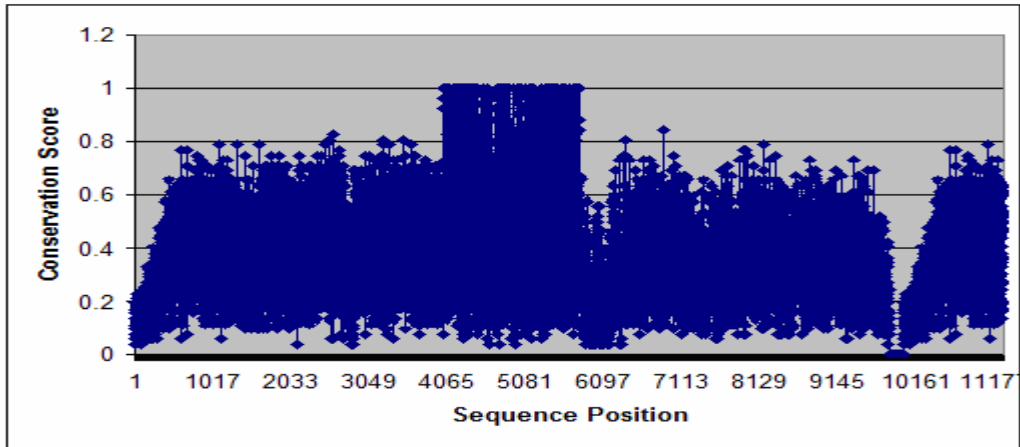


Fig 21: Test Case Size 70

In this case, the gene region is showing an average score that is about 50% greater than the baseline.

Start	Stop	Score
9361	469	0.900022
3275	3730	0.922108
3322	1607	1.062207
3976	5813	1.529063
6818	7072	0.886775
6859	3920	0.877678
7484	7206	0.878202
9194	9361	0.889328
9238	7460	1.015565

Table 6

By analyzing the results of the three test sets, it is clear that the prediction strength of the method increase as more and more sequences are added to the comparison set. A plot of the confidence score versus the number of sequences in the training set as given in Fig(5.8), confirms this. In general though , the graph should tend to flatten out after a critical number of sequences are present in the set , after which adding more sequences to the set does not have any significant change in annotation quality. This critical number of

CHAPTER4-TESTING AND EVALUATION

sequences would of course differ from gene to gene, alignment quality and selection of the comparison set.

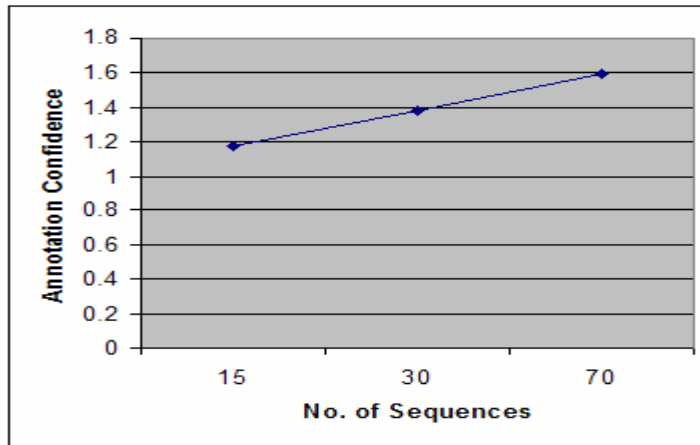


Fig 22: No. of Sequences vs. Annotation Confidence Score

Chapter5-CONCLUSION AND FUTURE WORK

The MSc project explored the potential of using a large number of genome sequences for annotation purposes. The availability of a large number of bacterial genomes was exploited to design an annotation system that uses information about conserved motifs in multiple sequence alignments in conjunction with ab-initio gene prediction for accurate annotation.

The major aim of the project was to carry out successful annotation of a single particular gene in regions of around 10 Kbp from the sequences. This aim was achieved by creating a design that employed both ab-initio and comparative gene finding techniques for better prediction and accuracy. Initial estimates of probable coding regions are received from a markov model based annotation module. These estimates are then processed by a comparative module that makes use of motif conservation information obtained from a multiple sequence alignment to refine the results and provide a confidence score.

The annotation system was tested out on several test sets of bacterial genomes ranging from 10 to 90 sequences. The results were very encouraging for the proposed method and the annotation exhibited a definite increase in the accuracy and confidence score of the gene prediction, as the number of sequences in the test set was increased. This agreed with the theoretical expectations since as the number of sequences grows, only the evolutionary constrained regions in the genomes are expected to show signs of conservation and the ratio of similarity in non-functional regions should go down.

Based upon the results of the project, large scale comparative approaches to gene annotation system definitely are feasible and can be particularly useful in conjunction with purely ab-initio techniques.

Future Work

Although the project achieved its main goals of comparative annotation, in order to make it applicable to more realistic and practical settings, several extensions are possible. Some of the potential future directions for the approach include:

1: Simultaneous alignment and annotation. The current approach of first obtaining a multiple alignment of the sequences and then using it to extract regions of conservation depends heavily upon the quality of alignment. In cases where the alignment fails to identify correctly the conserved areas the entire process of annotation will suffer. One way of overcoming this would be to carry out the alignment and annotation simultaneously. This could be done by integrating both these operations within one probabilistic framework such as GPHMM. However the GPHMM approach is currently barely tractable for just a pair of sequences. Applying this to the number of sequences of the order encountered in this project would be a huge computational challenge.

2: One simplifying assumption for the project was that the selected genome areas contained only a single gene. This assumption would of course need to be relaxed in order to extend the project to more realistic settings. Especially in the case of prokaryotic genomes, where the gene density is very high, models of multiple gene identification might be another possible extension for the project.

3: The project chose large scale annotation of prokaryotic genomes because of the availability of a large number of microbial sequences in the bioinformatics databases. However as more and more eukaryotic sequences become available due to improving sequencing technology , the focus could be shifted to these genomes as they are much harder and more interesting. For this the annotation module will have to be much more complex compared to the present implementation as the eukaryotic genomes have a very complex structure as compared to prokaryotes.

APPENDIX A- BIOLOGY BACKGROUND

This section gives a brief description of the main biological terms related to the project.

DNA

Deoxyribonucleic acid (DNA) is the molecule containing the genetic instructions used in the development and functioning of all known living organisms. The primary role of DNA is the storage of genetic information. DNA holds the information on how to create other components such as proteins. DNA carries this information in regions called genes.

In its chemical composition, DNA consists of long polymer nucleotides, and a backbone made of sugars and phosphate atoms. Base molecules from a total of four types are attached to the sugars. The sequence of bases along the helical backbone holds the genetic information.

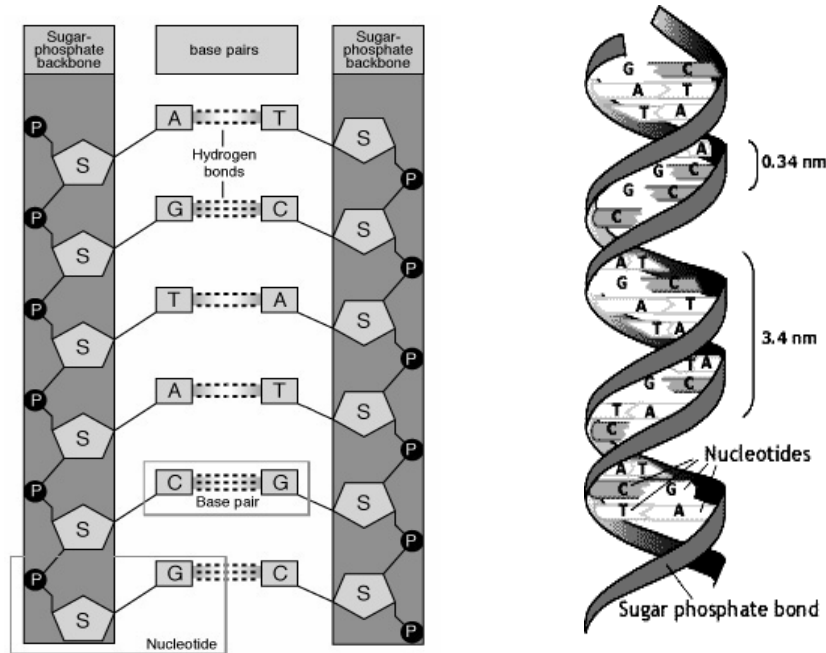


Fig 23: DNA Structure. [AEGLI]

Inside cells, DNA is organized inside chromosomes and the genome consists of the entire set of chromosomes in a cell. The process of chromosome duplication is called DNA replication. In Eukaryotic organisms DNA is stored inside the cell nucleus, whereas in prokaryotes it resides in the cell's cytoplasm.

Genes

APPENDIX A-BIOLOGY BACKGROUND

A gene is a collection of nucleic acid regions that encodes the information required for producing functional RNA products. Genes contain the sequence of the RNA products as well as regulatory regions that control the creation of the products.

In cells, genes consist of a long strand of DNA that contains a promoter, which controls the activity of a gene, and a coding sequence, which determines what the gene produces. When a gene is active, the coding sequence is copied in a process called transcription, producing an RNA copy of the gene's information. This RNA can then direct the synthesis of proteins via the genetic code. However, RNAs can also be used directly, for example as part of the ribosome. These molecules resulting from gene expression, whether RNA or protein, are known as gene products.

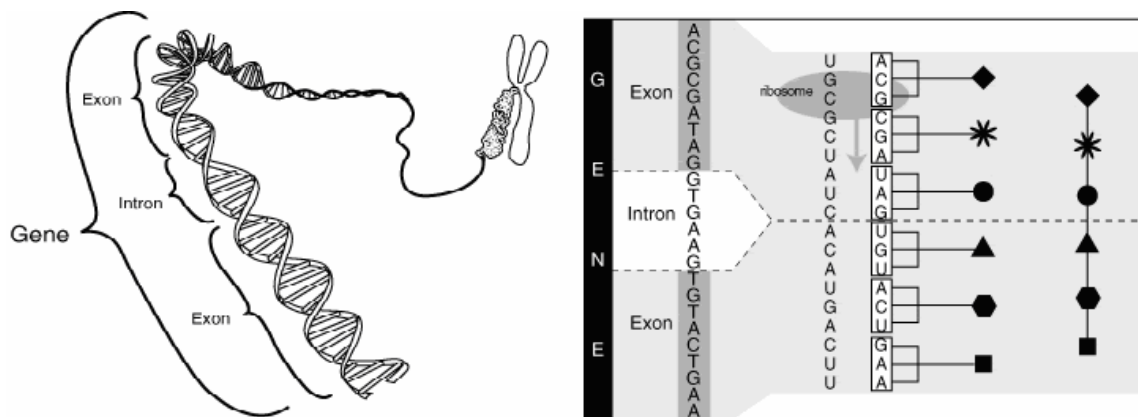


Fig 24:Gene Structure [A EGL2]

Most genes contain non-coding regions that do not code for the gene products, but regulate gene expression. The genes of eukaryotic organisms can contain non-coding regions called introns that are removed from the messenger RNA in a process known as splicing. The regions that actually encode the gene product, which can be much smaller than the introns, are known as exons. One single gene can lead to the synthesis of multiple proteins through the different arrangements of exons produced by alternative splicings.

RNA

Ribonucleic acid or RNA is a polymer consisting of ribonucleoside-phosphates. RNA plays an important part during the process of transfer of genetic information from DNA into proteins.

RNA is different from DNA in several aspects, including the fact that RNA uses uracil in place of thymine.



Fig 25: RNA structure [RSP].

Protein Synthesis

Protein synthesis is the process by which DNA encodes for the production of amino acids and proteins. It is carried out in two steps :

- 1 Transcription
- 2 Translation

Transcription

Transcription is the process of creation of complementary RNA by DNA. During transcription genetic information is transferred from DNA to RNA. In the case of protein-encoding DNA, transcription culminates in the creation of a protein. The stretch of DNA that is transcribed into an RNA molecule is called a transcription unit.

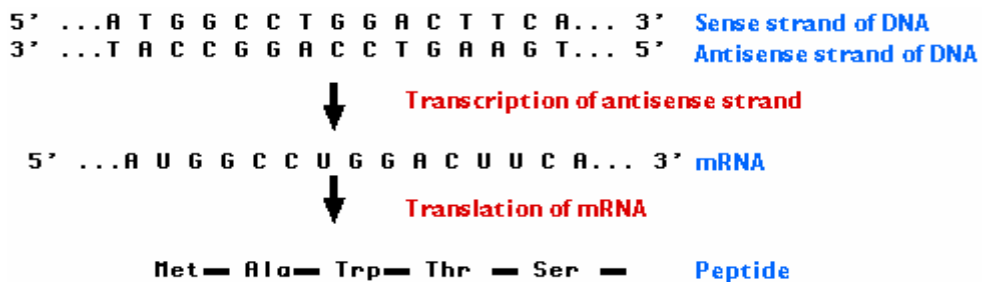


Fig 26: Transcription and Translation. [BP]

Translation

APPENDIX A-BIOLOGY BACKGROUND

Translation is the process that takes the information passed from DNA as messenger RNA and turns this into a series of amino acids bound together with peptide bonds. It really is a translation from one code, nucleotide sequence, to another code, amino acid sequence. The process is carried out through the help of ribosome's which bind to messenger RNA at specific locations and gets amino acids from new tRNA molecules. The ribosome eventually releases the chain of amino acids and the mRNA at a stop sequence.

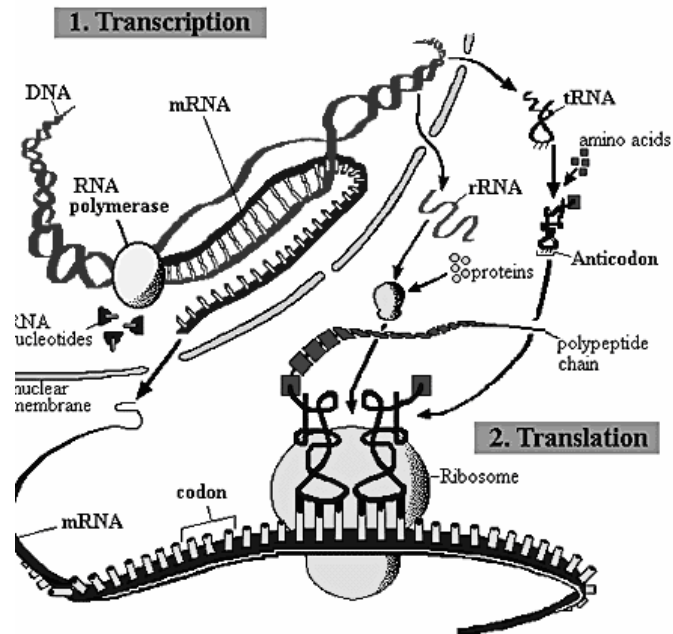


Fig 27: Protein Synthesis. [A EGL3]

APPENDIX B- MAJOR MODULES FROM SOURCE CODE

Only a few main modules related to the actual algorithms are included. Most of the source code related to sequence file handling and module interfaces to each other is not included for clarity.

```
//////////profile related//////////////////////////////////////////
static void profile creation(const string arr[], File outf,
                           int thresholdold, int number)
{
    const char *p;
    int ns;
    int mlen;
    int len;
    int i;
    int j;
    int dummy;
    int holder;
    int x;
    int px;
    int looper;
    int nucleotide;
    int maxscore;
    int score;
    int *matrix[AZ+2];
    int stlen;
    int k;
    int gene;
    int ns;
    int comparison;
    int index;
    float probarr[100][10000];
    int st;
    int stop;
    float summation;
    int ns=number;
    if (ns<2)
        exit(0);

    for(j=0;j<ns;++j)
    {
        p = arr[j];
        dummy=strlen(p);
        Printf(outf,"\n");
        for ( i=0;i<dummy;i++)
        {
            nucleotide=25;
            if ((p[i]=='A')||(p[i]=='a'))
            { nucleotide=0;}

            else if((p[i]=='C')||(p[i]=='c'))
            { nucleotide=2;}

            else if((p[i]=='G')||(p[i]=='g'))
            { nucleotide=6;}

            else if((p[i]=='T')||(p[i]=='t'))
            { nucleotide=19;}

            Printf(outf,"%d\t",matrix[nucleotide][i]);
            if (nucleotide!=25)
            { probarr[j][holder]=(float)matrix[nucleotide][i]/(float)ns;
              Printf(outf,"%2d\t%2d\t%5f\n",j,holder,probarr[j][holder]);
              holder++;
              if (holder%1==0){
```


APPENDIX B-MAJOR MODULES FROM SOURCE CODE

```

    Stop [i] = 0;

    if (T < 3)
        return;

    Cod = Ch_Mask (X [1]) << 4 | Ch_Mask (X [2]);

    for (i = 3; i <= T; i++)
    {
        Codon = (Codon & SHIFT_MASK) << 4;
        Codon |= Ch_Mask (X [i]);

        if (Is_Forward_Stop (Codon))
            Stop [i % 3] = TRUE;
        if (Is_Reverse_Stop (Codon))
            Stop [3 + i % 3] = TRUE;
    }

    return;

//////////////////////////////////ORF//////////////////////////////////
static void Fsr
    (const string & s, int len, vector <bool> & hs)

{
    Codon_t codon;
    int fm_ss; // fm subscript
    int which;
    int i;

    hs . resize (7);
    for (i = 0; i < 7; i++)
        hs [i] = false;

    fm_ss = 1;

    for (i = len - 1; i >= 0; i--)
    {
        codon . Shift_In (s [i]);

        if (codon . Must_Be (Fstpat, which))
            hs [fm_ss] = true;
        if (codon . Must_Be (Rev_Stop_Pattern, which))
            hs [fm_ss + 3] = true;

        if (fm_ss == 2)
            fm_ss = 0;
        else
            fm_ss++;
    }

    return;
}

static void Roverlp
    (vector <Orf_Iv_t> & iv, int molp)

// remove open reading frames contained by others

{
    vector <int> hg;

    int i, j, n;

    n = iv.size();
    hg.resize (n);

    // set values in hg and right_wrap

```

APPENDIX B-MAJOR MODULES FROM SOURCE CODE

```

for (i = 0; i < n; i++)
{
    if (i == 0)
        hg [i] = iv [i] . hi;
    else
        hg [i] = Max (hg [i - 1], iv [i] . hi);
}

hg [0] = iv [0] . hi;
for (i = 1; i < n; i++)
{
    for (j = i - 1; 0 <= j; j--)
    {
        if (hg [j] <= iv [i] . lo + molp)
            break; //

        if (molp < Isze (iv [j] . lo, iv [j] . hi,
            iv [i] . lo, iv [i] . hi))
            iv [j] . dltd = iv [i] . dltd = true;
    }
}

for (i = j = 0; i < n; i++)
    if (! iv [i] . dltd)
    {
        if (i != j)
            iv [j] = iv [i];
        j++;
    }

iv . resize (j);

return;
}

static void Fstopcod
(int i, int fm, int pfwds [3], int ffs [3],
int ffs [3], int fbase, bool hit_ignore,
vector <Orf_t> & orf_list)

{
    Orf_t orf;
    int geln, orf_len;

    if (pfwds [fm] == 0)
    {
        Hffs (fm, i - 1, ffs [fm],
            fbase, geln, orf_len,
            Gic && ! hit_ignore);
        ffs [fm] = i - 1;
    }
    else
    {
        geln = i - ffs [fm] - 1;
        orf_len = i - pfwds [fm] - 4;
    }

    if (geln >= Min_Geln)
    {
        orf . Sspos (i - 1);
        orf . Set_Fm (1 + (fm + 1) % 3);
        orf . Set_Geln (geln);
        orf . Set_Orf_Len (orf_len);
        orf_list . push_back (orf);
    }

    ffs [fm] = INT_MAX;
    pfwds [fm] = i - 1;
}

```

APPENDIX B-MAJOR MODULES FROM SOURCE CODE

```

    return;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////// Context model using hash table////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
struct htab *addhash (char *key, char *data) {
    struct htab *newhash;
    struct htab *chh;
    unsigned int i, hvl;

    newhash = (struct htab *) (malloc(sizeof(struct htab)));
    if (newhash == NULL) {
    }

    for(i = 0; i <= strlen(key); i++) {
        newhash->key[i] = key[i];
    }
    for(i = 0; i <= strlen(data); i++) {
        newhash->data[i] = data[i];
    }

    hvl = hash(key);

    if (hht[hvl] == NULL) {
        hht[hvl] = newhash;
        hht[hvl]->parent = NULL;
        hht[hvl]->cld = NULL;
    }
    else {
        chh=hht[hvl];
        while(chh->cld!= NULL) {
            chh=chh->cld;
        }
        chh->cld= newhash;
        newhash->cld= NULL;
        newhash->parent = chh;
    }

    return newhash;
}

struct htab *fhsh(char *key) {
    unsigned int hvl;
    struct htab *chh;

    hvl = 0;

    hvl = hash(key);

    if (hht[hvl] == NULL) {
        return NULL;
    }

    if (!strcmp((hht[hvl]->key), (key))) {
        chh = hht[hvl];
        return chh;
    }
    else {
        if (hht[hvl]->cld== NULL) {
            return NULL;
        }

        chh = hht[hvl]->cld;

        if (!strcmp((chh->key), (key))) {
            return chh;
        }
    }
}

```

APPENDIX B-MAJOR MODULES FROM SOURCE CODE

```

    }

    while (chh->cld!= NULL) {
        if (!strcmp((chh->key), (key))) {
            return chh;
        }
        chh = chh->cld;
    }
    if (!strcmp((chh->key), (key))) {
        return chh;
    }
    else {
        return NULL;
    }
}
}

void hprfile(int profile) {
    int max, total, i, nds, used;
    double avg;
    struct htab *chh;

    max = total = i = nds = used = 0;
    avg = 0;

    if (profile > 0) {
        for (i = 0; i < HELEMEN; i++) {
            if (hht[i] != NULL) {
                used++;
                nds = 0;
                chh = hht[i];
                while(chh != NULL) {
                    nds++;
                    if (profile > 3) {
                        fprintf(stderr, "%lx -> %lx: %s (%s) -> %lx\n", chh->parent, chh,
chh->key, chh->data, chh->cld);
                    }
                    chh = chh->cld;
                }
                if (nds != 0) total += nds;
                if (nds > max) {
                    max = nds;
                }
                if (profile > 1) {
                    fprintf(stderr, "i: %d\n", i);
                }
            }
        }
        avg = (double)(total) / (double)(used);
    }
}
}

```

```

Totalscore
(char * string, int len, int fm, double * tscore)

{
    double result, x;
    int st, stop = mln - 1;
    int i;

    if (pdc == 1)
        fm = 0;
    assert (0 <= fm && fm < pdc);

    if (Verbose > 0)
        printf ("Totalscore len = %d fm = %d\n", len, fm);
}

```

APPENDIX B-MAJOR MODULES FROM SOURCE CODE

```

    result = tscore [0] = 0.0;

    for (i = 0; i < mln - 1; i++)
    {
        x = Pwin (i, string, fm);
        if (Verbose > 0)
            printf ("%7d: %8.3f\n", i, x);
        result += x;
        fm = (fm + 1) % pdc;
        tscore [i + 1] = result;
    }

    for (st = 0; stop < len; st++, stop++)
    {
        x = Fwin (string + st, fm);
        if (Verbose > 0)
            printf ("%7d: %8.3f\n", st + mln - 1, x);
        result += x;
        fm = (fm + 1) % pdc;
        tscore [stop + 1] = result;
    }

    return;
}

void Markov :: Total_Score
    (const string & s, vector <double> & score, int fm) const

// Set score [i] to be the score of substring s [0 .. i]
// for each position in s . Use this model with the first base
// in fm fm .

{
    double result, x;
    const char * cstr = s . c_str ();
    int st, stop;
    int i, n;

    if (pdc == 1)
        fm = 0;
    assert (0 <= fm && fm < pdc);

    n = s . length ();
    score . resize (n);

    result = 0.0;

    stop = Min (mln - 1, n);
    for (i = 0; i < stop; i++)
    {
        x = Pwin (i, cstr, fm);
        /**ALD
        //printf ("Total_Score: i = %2d ch = %c prob = %6.4f\n", i, s [i], exp (x));
        result += x;
        if (fm == pdc - 1)
            fm = 0;
        else
            fm++;
        score [i] = result;
        }

    for (st = 0; i < n; st++, i++)
    {
        x = Fwin (cstr + st, fm);
        /**ALD
        //printf ("Total_Score: %-.*s st = %3d i = %3d ch = %c prob = %6.4f\n",
        // mln, cstr + st, st, i, s [i], exp (x));
        result += x;
        if (fm == pdc - 1)
            fm = 0;

```

APPENDIX B-MAJOR MODULES FROM SOURCE CODE

```
        else
            fm ++;
        score [i] = result;
    }
    return;
}
```

BIBLIOGRAPHY

[MM] “A critical review of gene prediction software Mark” McElwain Bioc 218 Final Paper

[LMS] Lior Pachter, Marina Alexandersson, Simon Cawley.
“Applications of Generalized Pair hidden markov models to alignment and gene finding problems.”
Journal of Computational Biology. 2002, 9(2): 389-399.
doi:10.1089/10665270252935520.

[BMI] Biostatics and Medical Informatics <http://www.biostat.wisc.edu/bmi576/>

[RM] Rajeev K. Azad and Mark Borodovsky
“Probabilistic methods of identifying genes in prokaryotic genomes:Connections to the HMM theory “
BRIEFINGS IN BIOINFORMATICS. VOL 5. NO 2. 118–130. JUNE 2004

[AEGL2] Access Excellence Graphics Library
<http://www.accessexcellence.org/RC/VL/GG/gene2.html>

[AEGL1] Access Excellence Graphics Library
<http://www.accessexcellence.org/RC/VL/GG/basePair2.html>

[AEGL3]http://www.accessexcellence.org/RC/VL/GG/protein_synthesis.html

[BP] <http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/T/Transcription.html>

[RSP] RNA Structure primer <http://www.rnabase.org/primer/>

[TCGF] Tim Chen Gene Finding University of Southern California

[CW94] Higgins D., Thompson J., Gibson T.Thompson J.D., Higgins D.G., Gibson T.J.(1994).
“CLUSTAL W: improving the sensitivity of progressivemultiple sequence alignment through sequence weighting,position-specific gap penalties and weight matrix choice.”
Nucleic Acids Res. 22:4673-4680.

[FST90]W. R. Pearson (1990)
“Rapid and Sensitive Sequence Comparison with FASTP and FASTA Methods in Enzymology”
183:63 - 98.

[BST97] Altschul, SF, Madden, TL, Schaffer, AA, Zhang, J, Zhang, Z, Miller, W, and DJ Lipman (1997)

BIBLIOGRAPHY

“Gapped BLAST and PSI-BLAST: a new generation of protein database search program.”

Nucleic Acids Res. 25(17):3389-402.

[HAUSS94] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler.

“Hidden Markov models in computational biology: Applications to protein modeling.”
Journal of Molecular Biology , 235:1501--1531, February 1994.

[HEND97] John Henderson Steven Salzberg Kenneth H. Fasman

“Finding Genes in DNA with a Hidden Markov Model”

(1997) Journal of Computational Biology

[GENS97] Burge, C. and Karlin, S. (1997)

“Prediction of complete gene structures in human genomic DNA”

J. Mol. Biol. 268, 78-94.

[GE96] D. Kulp, D. Haussler, M.G. Reese, and F.H. Eeckman (1996).

“ A generalized Hidden Markov Model for the recognition of human genes in DNA.”

ISMB-96, St. Louis, MO, AAAI/MIT Press.

[GM93] Borodovsky M. and McIninch J.

"GeneMark: parallel gene recognition for both DNA strands."

Computers & Chemistry, 1993, Vol. 17, No. 19, pp. 123-133

[ORP98] Frishman, D. , Mironov, A., Mewes, H.-W. , and Gelfand,M. (1998).

“Combining diverse evidence for gene recognition in completelysequenced bacterial genomes”.

Nucl. Acids Res.

[CRIT99] Jonathan H. Badger and Gary J. Olsen.

“CRITICA: Coding Region Identification Tool Invoking Comparative Analysis.”

Journal of Molecular Biology and Evolution, Vol. 16 , pp.512-524 , 1999.

[DSCAN02] Irmtraud M. Meyer and Richard Durbin

“Comparative ab initio prediction of gene structures using pair HMMs”

Bioinformatics Vol. 18 no. 10 (2002) pp. 1309-1318

[SLAM03] Marina Alexandersson, Simon Cawley, and Lior Pachter

“SLAM: Cross-Species Gene Finding and Alignment with a Generalized Pair Hidden Markov Model”

Genome Res., Mar 2003; 13: 496 - 502.

[Hein03] Jakob Skou Pedersen , and Jotun Hein

“Gene finding with a hidden Markov model of genome structure and evolution”

Bioinformatics Vol. 19 no. 2 2003 Pages 219-227

BIBLIOGRAPHY

[ROS00] Batzoglou,S., Pachter,L., Mesirov,J., Berger,B. and Lander,E.S. (2000)
“Human and mouse gene structure: comparative analysis and application to exon
prediction. “
Genome Res., 10, 950–958.

[SGP03] G. Parra, P. Agarwal, J.F. Abril, T. Wiehe, J.W. Fickett and R. Guigó.
"Comparative gene prediction in human and mouse."
Genome Research 13(1):108-117 (2003)

[BW] BIOWEB
http://bioweb.uwlax.edu/GenWeb/Molecular/Seq_Anal/Translation/translation.html