Hidden Markov Models for the Detection of Motif Repeats in Protein Sequences

Raluca Uricaru
raluca.uricaru@ens-lyon.fr
ENS Lyon, France

16 June 2006

Abstract

We deal with the problem of retrieving the repeated apparitions of given motifs in protein sequences. In particular we are interested in the apparitions of the PPR related motifs in the *Arabidopsis thaliana* PPR proteins. As input, we dispose of the Profile Hidden Markov Models (HMMs) models built for each given motif. The profile models were constructed using HMMER, from sequence fragments alignments. Given the amino acid sequence of a protein, we need to identify the correspondent motif sequence.

We propose a **Cyclic Profile HMM structure** and an efficient implementation for this type of structure. As a result, we prove the feasibility and the correctness of this novel method and we obtain an automatic annotation tool that we use for annotating the PPR proteins from the *Arabidopsis* and from the *rice*.

Keywords: **motif repeats**, **profile HMMs**, **PPR-motifs**, **Viterbi algorithm**, **motif annotation tool**.

Contents

1 Introduction **3**
- 1.1 Protein Sequence. Motifs and Motif Repeats. Sequence Alignment . . . 3
- 1.2 Definition of the Problem . . . 4
  - 1.2.1 *Arabidopsis Thaliana* Genome . . . 4
  - 1.2.2 PPR Proteins and PPR-Related Motifs . . . 4
- 1.3 Previous Approaches in Retrieving the motif Structure of a Protein Sequence for Given motifs . . . 5
- 1.4 Short Overview for our Proposed Solution . . . 6

2 Hidden Markov Models (HMMs) **7**
- 2.1 Definition and General Description . . . 7
- 2.2 Basic Algorithms . . . 8
  - 2.2.1 Forward-Backward Algorithm . . . 9
  - 2.2.2 Viterbi Algorithm . . . 9

3 Profile HMMs **11**
- 3.1 General Description . . . 11
- 3.2 Solving Different Types of Alignments to a Profile Model in HMMER . . . 12
- 3.3 HMMER Model File . . . 13
  - 3.3.1 The Null Model . . . 13
- 3.4 Profile HMM Construction . . . 14
- 3.5 Scoring Methods and Statistical Significance of Structural Alignment Scores . . . 15
  - 3.5.1 The Negative Log Likelihood (NLL) - Score . . . 15
  - 3.5.2 The Bit Score . . . 15
  - 3.5.3 E-value . . . 15

4 Cyclic Profile HMMs **17**
- 4.1 Structure . . . 17
- 4.2 Transition Probabilities . . . 19
  - 4.2.1 Profile Models Internal Transitions Probabilities . . . 20
  - 4.2.2 Profile Models Marginal State Transitions Probabilities . . . 20
- 4.3 Viterbi Algorithm . . . 21
- 4.4 Calibration of the Model . . . 22
- 4.5 Types of Output . . . 22

5 Experiments **23**
- 5.1 Transition Probabilities Experiments . . . 24
- 5.2 Structure Experiments . . . 25
- 5.3 Types of Errors . . . 26
- 5.4 Summary of the Results for the *Arabidopsis Thaliana* Proteins . . . 27
- 5.5 Results for the *Rice* Proteins . . . 27

6 Conclusions and Future Work **28**

1 Introduction

1.1 Protein Sequence. Motifs and Motif Repeats. Sequence Alignment

A *protein* is a complex organic compound, essential to the structure and function of all living cells and viruses. Different proteins perform a wide variety of biological functions: some are enzymes, which catalyze chemical reactions, others play structural or mechanical roles. Their primary structure is a specific sequence of amino acids. The details of this sequence are stored in the code of a gene which is being read by a cell (through the processes of transcription and translation) and used to construct the protein. Commonly, proteins work together to achieve a particular function, and often physically associate with one another to form a complex.

Proteins are among the most actively-studied molecules in biochemistry. Evolution modifies and recombines existing building blocks instead of inventing everything from scratch; in the protein world, these building blocks have been termed *motifs*. Many protein sequences contain motifs that can be traced back to a single ancestral gene and are therefore repeats of a basic genic motif. The identification and characterisation of new motifs is a major goal of protein science. When the same motif is present several times in one sequence, we say that it is a motif repeat. In proteins we usually deal with approximate repeats as, due to deletions, insertions and mutations, repeat units have diverged and are usually separated by large gaps.

The most efficient manner to identify motif repeats in a protein sequence is using dynamic programming to align a sequence to itself and analyse the collection of suboptimal alignments [HH00]. When we talk about *sequence alignment* we understand an arrangement of two or more sequences (subsequences), highlighting their similarity. The sequences are padded with gaps (usually denoted by dashes) so that wherever possible, columns contain identical or similar characters from the sequences involved (see figure 1).

```
tcctctgcctctgccatcat---caaccccaaagt
||||  |||  |||||  |||||   ||||||||||||
tcctgtgcatctgcaatcatgggcaaccccaaagt
```

Figure 1: Sequence Alignment

Mismatches in the alignment correspond to mutations, and gaps correspond to insertions or deletions. A sequence can be aligned to another one using the Smith-Waterman algorithm [SW81], the slightly less accurate but a lot faster BLAST(Basic Local Alignment Search Tool) algorithm [AGM$^+$90] or the FASTA algorithm [LP85]. Sequence alignment can also be used to study the evolution of languages or the similarity between texts.

There exists a large collection of protein multiple sequence alignments and profile Hidden Markov Models (see section 3), called *Pfam* [BCD$^+$04]. The alignments represent some evolutionary conserved structure which has implications for the protein's function. Profile Hidden Markov Models(HMMs) built from the Pfam alignments are very useful for automatically recognizing that a new protein belongs to an existing protein family, even if the homology is weak. Pfam is mostly formed from accurate human crafted multiple alignments. Its latest version contains 3071 families of proteins and is accessible on the world wide web [1].

[1] http://pfam.wustl.edu/

1.2 Definition of the Problem

We deal with a particular instance of the problem of retrieving the motif repeats in protein sequences; it is a slightly less difficult situation, as we already know the motifs that we are searching for. In our case we are interested in identifying the PPR-Related motifs in the sequences of the proteins from *Arabidopsis thaliana* (mouse-ear cress).

The reasons for choosing *Arabidopsis thaliana* and in particular the PPR-Related motifs are multiple: conserved structure of the motifs, their specificity, the presence of the tandem repeated motifs very well suited for our approach, the interest of the plant science community in this species, the access to manual annotation which allowed us to verify the results, the lack of systematic annotation of the PPR-motifs for the other plant species.

1.2.1 *Arabidopsis Thaliana* Genome

The *Arabidopsis thaliana* genome sequence uncovered many previously undescribed and often unsuspected genes. The initial analysis of the genome sequence estimated that 31% of Arabidopsis genes were too dissimilar to genes of known function. Many of the genes in this class fall into families that have greatly expanded in plants or are entirely plant specific. A major challenge is to discover the functions of these genes; amongst these families, the largest ($\approx$ 450 members) and perhaps the most mysterious is defined by the so-called **pentatricopeptide repeat**(PPR) [SP00], [RBTNL06].

Although some individual PPR genes were already described, the existence of a large family of similar proteins only became apparent with the sequencing of the Arabidopsis genome. PPR proteins make up a significant proportion of the proteins with unknown function in Arabidopsis. Only a few very recent articles describe the functional analysis of individual Arabidopsis PPR genes: [ABKL00], [LAea04].

1.2.2 PPR Proteins and PPR-Related Motifs

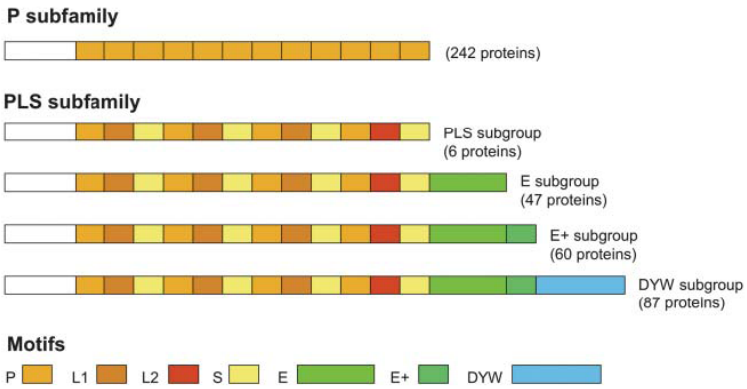

Figure 2: Motif Structure of Arabidopsis PPR Proteins

The PPR motifs are usually present in proteins as tandem arrays of about a dozen repeats; there are 3 such motifs, named **P**, **L**, **S** and one variant of L, named **L2**.

Roughly half of the PPR proteins form the PCMP subfamily, which is land plant specific; the other half is represented by the PPRP subfamily (the proteins from this subfamily usually have only P motif repeats).

PCMPs exhibit a large and variable tandem repeat of a standard pattern of the 3 PPR variant motifs. The association or not of this repeat with 3 non-PPR motifs: **E**, **E+**, **DYW** (see figure 2 for the motif representation) defines 4 distinct classes of PCMPs: A, E, F and H.

The PPRPs can be defined using the following regular expression: $(P^* - S^*)^*$; the more complex structure of the PCMPs can also be summarized using a regular expression: $(P - L - S^*)^* - [E - [E^+ - [Dyw]]]$. For a more detailed description of the motif structure of the Arabidopsis PPR proteins for the PCMP subfamily and its 4 classes, see table 1.

Subfamily	Sequence of motifs	Class
PCMP	$PL(S)^k$-$[PL(S)^m]^n$-PL_2S_2-E-E+-DYW	H
	$[PL(S)^m]^n$-PL_2S_2-E-E+	F
	$[PL(S)^m]^n$-PL_2S_2-E	E
	$[PL(S)^m]^n$-PL_2S_2	A

Table 1: PPR Family

To identify the PPR motifs, the interesting fragments of the protein sequences were extracted and aligned as to obtain separate PPR motifs, and afterwards, the HMMER package [Edd](see section 3) was used on these alignments in order to define the correspondent models [2].

The motif occurrences are adjacent in the amino acid sequences; thus, we define the motif sequence of a protein as the succession of motifs read from the N- towards the C-terminus. For example, to the protein At3g61170 corresponds the motif sequence $S - S - S - S - P - L - S - S - P - L - S - P - L - S - P - L2 - S - E - E + -Dyw$ (where hyphens separate the motifs).

1.3 Previous Approaches in Retrieving the motif Structure of a Protein Sequence for Given motifs

Until the present time, as far as we know, no methods have been developed for automatically retrieving the repeats of given motifs in protein sequences.

All the previous approaches for this problem use profile consensus models built for each of the given motifs, models that can be found in the *Pfam database* (introduced in section 1.1). Other types of models, like *regular expressions* (see PROSITE [3]) do not work for the PPR motifs because of their little conservation (the motifs are too divergent).

The idea is to find the repeated apparitions of each motif, using the HMMER package, and to build the best possible motif sequence, by covering each region with one of the given motifs; in case of overlapping, the solution used is choosing the best scored motif for that region. In fact, all annotations of the repeated apparitions of given motifs in protein sequences of which we dispose, were manual made. As this method is not at all efficient, when speaking in terms of time and of quantity of human labor, an automatic one is needed.

The problem with this type of approach is that we do not have a complete model that can be applied to the sequence, but multiple models that have to be aligned one after the other and then the results put together in order to obtain the most probable sequence of motifs. HMMER, which is the state-of-art in this field, is incapable to find the apparitions of more than one motif in a protein sequence; it was built to model a single motif, therefore it has the capacity of iterating on this motif only.

[2] http://www.evry.inra.fr/public/projects/ppr/ppr.html
[3] http://www.expasy.org/prosite/

Not having a complete model which could be directly applied to the sequence, implies not having a global score for the retrieved motif sequence and so, not being able to compute the significance of the results obtained on different sequences.

We propose a cyclic structure that connects the profile HMMs conceived for each motif, giving the possibility of iterating on each chosen motif.

1.4 Short Overview for our Proposed Solution

We propose using a Profile HMM Cyclic structure. The first question that we need to clarify is: *why choosing an HMM for modeling the motif repetitions.* The arguments in favour of this choice are multiple:

- the consensus models for the motifs are given as profile HMMs,
- it is an automaton and so it can model the different transitions from a model to another,
- it is a probabilistic automaton and so it can offer us the most probable motif sequence,
- it has a probability distribution over the symbol emission for each state and so it is able to recognize even motifs that have evolved and that are far from the original ones,
- it gives us a score for the state sequence that we obtain, score for which we are able to compute its statistical significance. But the strongest of all reasons is that simpler models (like regular expressions) are not powerful enough to model the individual PPR motifs; only HMMs manage to recognize distant similarities (homologies).

We intended to verify the feasibility and the efficiency of the Profile HMM Cyclic structure for modeling the problem of identifying the repeated apparitions of specific motifs in protein sequences. The novelty of our work is that we generalise the usual profile HMM and propose a Cyclic HMM structure. For this structure we adapted the Viterbi algorithm (see section 2.2.2) and we provided an efficient implementation.

My work has been developed in two distinct phases, the research phase and the implementation phase. Multiple experiments have been done along my entire work, at the beginning of the research phase and during the implementation phase.

Firstly, I had to understand the biological problem and to become familiar with the PPR family of proteins and with the existing solutions. For this, I tried to redo the manual annotation of 30 randomly chosen proteins of the *Arabidopsis thaliana*; after retrieving the apparitions of the PPR motifs and the 3 non PPR motifs E, E+, DYW by using HMMER on the 30 proteins, I have computed the most probable motif sequences and I have compared the results to the expert manual made annotation. As the results mostly matched, we came to the conclusion that an automatic method should be well suited.

Secondly, I had to study the HMMs' theory (shortly described in section 2) and in particular the profile HMMs' theory, to acquaint myself with the HMMER implementations of the basic algorithms for the profile HMMs (a succinct overview for the profile HMMs and their particular implementation for HMMER is given in section 3), and finally, to design and to develop my own solution, detailed in section 4. The obtained results have been compared to the existing manual annotation; having an expert manual annotation offered us the possibility of estimating the relevance of our results. The different experiments that have been done are described in section 5.

We obtained an automatic annotation tool which made possible the first annotation of the *rice* proteins. The significance of our work also consists in the fact that it offers the basis for

more complex approaches. Passing, from the retrieval problem, to the more difficult problem of recognition of motifs, is a logical and most useful sequel of our study. Section 6 gives more details in this direction.

2 Hidden Markov Models (HMMs)

2.1 Definition and General Description

HMMs were introduced in the scientific literature by Baum and his team, in the mid 60's. The model in discussion is close to probabilistic automata. A probabilistic automaton (PA) is defined by a set of states and transitions between states, and a probability distribution on the transitions. Every transition has a correspondent symbol which is generated every time that this transition is used. An HMM is defined similarly: its structure is also composed by a set of states, transitions between states and a distribution over the transitions. The biggest difference between the two models is that, for the HMM, the emission of symbols is done on the states and not on the transitions. In addition, every state has not only one symbol associated, but a probability distribution over all the symbols of the alphabet. The most important proof of the connection between the two types of models, is the fact that every HMM can be simulated by a PA with the same number of states.

HMMs are used for generating sequences which can be both discrete and continuous. They were first employed in the field of speech processing; next, HMMs were successfully used in the analysis of biological sequences. Lately, they were discovered as an important tool in information extraction. See [BG00] or [Rab89] for a more detailed description.

An HMM can be defined as a quadruple (S, Σ, T, G), where

- S is a set of N states; it has 2 special states: *start* and *end* which do not emit any symbols and which are used only for initiating and finishing a sequence.
- Σ is an alphabet composed of M symbols.
- $T = S - \{end\} \times S - \{start\} \rightarrow [0, 1]$ is a matrix giving the transition probabilities from one state to another; we call $P(s \rightarrow s')$ the transition probability from the state s to the state s', where $\sum_{s' \in S} P(s \rightarrow s') = 1$.
- $G = S - \{start, end\} \times \Sigma \rightarrow [0, 1]$ is a matrix giving the symbol generation probabilities, associated to each state; we call $P(o \mid s)$ the probability of emitting the symbol o, in the state s, where $\sum_{o \in \Sigma} P(o \mid s) = 1$.

The structure of an HMM is defined by the set of states and by the transitions with non-null probabilities. Figure 3 gives an example of a simple HMM with 7 states and 11 transitions. The procedure of generating a symbol sequence using an HMM consists of: start with the state *start*, move from one state to another following the transition probabilities, generate a symbol on every state you pass by, using the generation probability distribution for that particular state, arrive in *end* state. For example, the sequence *abccb* can be generated by the HMM in figure 3 by starting in *start* state, then passing by the states $1, 3, 5, 5, 2$, and arriving in *end* state. Note that this symbol sequence can be also generated by following the state sequences $start - 2 - 4 - 5 - 5 - 2 - end$ or $start - 1 - 4 - 5 - 5 - 2 - end$.

The probability of generating the symbol sequence *abccb*, following the first described state sequence, is the probability of moving from *start* to 1 (0.5), multiplied by the probability that the state 1 emits symbol *a* (1), multiplied by the probability of moving from 1 to 3 (0.7), multiplied by $\ldots = 6.6 \cdot 10^{-3}$. Following the second and the third state sequence, this probability is respectively

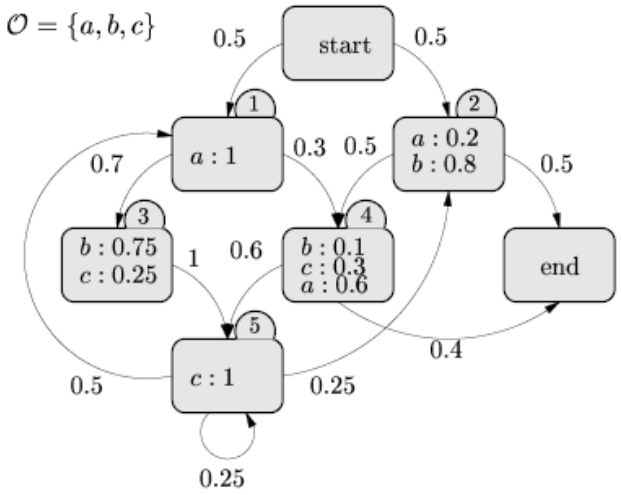

Figure 3: HMM Example

$2.3 \cdot 10^{-4}$ and $7.5 \cdot 10^{-5}$. As there are not any other possible state sequences that could generate this symbol sequence, the probability that the HMM generates the symbol sequence *abccb* is $6.6 \cdot 10^{-3} + 2.3 \cdot 10^{-4} + 7.5 \cdot 10^{-5} = 6.9 \cdot 10^{-3}$ and the most probable state sequence is the first one: $start - 1 - 3 - 5 - 5 - 2 - end$.

In conclusion, HMMs define a stochastic, non deterministic model: a single sequence can be generated in several different manners.

2.2 Basic Algorithms

In order to completely understand the way an HMM works, we have to answer to 3 questions:

- What is the probability of generating the symbol sequence $O = o_1 \dots o_T$?

  We have to compute the generation probability for each possible state sequence and then, sum these probabilities. The probability of generating O when following the state sequence $S = s_0 \dots s_{T+1}$ with $s_0 = start$ and $s_{T+1} = end$ is

  $$P(O \mid H) = \sum_{<s_0 \dots s_{T+1}>} P(s_0 \rightarrow s_1) P(o_1 \mid s_1) \dots P(s_{T-1} \rightarrow s_T) P(o_T \mid s_T) P(s_T \rightarrow s_{T+1})$$

  The direct computation of this probability can be done in $O(TN^T)$ which is not at all practical. Fortunately, there is an efficient method: the forward-backward algorithm that we explain in section 2.2.1.

- Being given the symbol sequence $O = o_1 \dots o_T$, what is the state sequence, that generates this symbol sequence, with the highest probability?

  In this case we do not need the maximal probability, but the state sequence that gives the maximal probability. This state sequence is called the "Viterbi sequence". As for the previous question, we could compute the probabilities for all the state sequences and choose the best sequence. For the same reasons, this direct approach is not suited. Viterbi's algorithm, explained in section 2.2.2, is a dynamic programming algorithm that solves this problem efficiently.

- Let $\mathcal{O} = \left\{O^1, \ldots, O^K\right\}$ be the set of training sequences. How can we adjust the model parameters $\lambda =< T, G >$, in order to well fit these data?

  In the previous 2 cases, we have supposed that we have an HMM, constructed and parameterised in order to suit best the sequences that we need to model. For this question, the problem is to estimate the model's parameters, as we assume that the structure is known. The estimation of parameters is reduced to a training problem. Following the maximum likelihood principle, the goal is to find the parameters $\lambda =< T, G >$ that maximize: $P(\mathcal{O} \mid H) = \prod_{k=1}^{K} P(O^k \mid H)$. This is realized by the *Baum-Welch* EM (Expectation Maximization) algorithm.

  The algorithm starts with an initial parametrisation of the model. It is an iterative procedure, that reestimates the model's parameters at each iteration, till obtaining a stabilisation of the parameters. Unfortunately, the result depends a lot on the initial chosen model, so the algorithm converges towards a local optimum.

We will detail only the first 2 questions, as in our case we were interested in discovering the state sequence that generated the symbol sequence, for a given model and a given symbol sequence.

2.2.1 Forward-Backward Algorithm

We consider the forward variable: $\alpha_t(s) = P(o_1 \ldots o_t, s_t = s \mid H)$ that represents the probability of generating the symbol sequence $O = o_1 \ldots o_t$, starting from the *start* state and arriving in the state s, for the time point t. There is an inductive method of computing this variable:

1. *Initialisation* For $t = 1$, we have

$$\alpha_1(s) = P(start \rightarrow s)P(o_1 \mid s)$$

2. *Induction* For $t = 2, \ldots, T$ we have

$$\alpha_t(s) = \left(\sum_{s' \in S} \alpha_{t-1}(s')P(s' \rightarrow s)\right) P(o_t \mid s) \tag{2.1}$$

If we know $\alpha_T(s)$ - the probability of generating the symbol sequence O and arriving in state s - the computation of $P(O \mid H)$ is immediate:

$$P(O \mid H) = \sum_{s \in S} \alpha_T(s)P(s \rightarrow end)$$

The algorithm is quite natural: it starts by computing the probability of generating the first symbol of the sequence and it continues by adding a new symbol at each step, till it arrives at computing the probability of the entire sequence. Another approach, less natural, would be to execute this computation in the opposite direction. For both algorithms, the complexity is $O(N^2T)$.

2.2.2 Viterbi Algorithm

From all the state sequences that generate the symbol sequence O, we need the one with the highest probability. Viterbi's algorithm gives exactly this state sequence. We start by defining the variable $\delta_t(s)$:

$$\delta_t(s) = \max_{s_0 \ldots s_{t-1}} P(s_0 \ldots s_t = s, o_1 \ldots o_t \mid H)$$

which gives the maximal probability of generating the symbol sequence $O = o_1 \dots o_t$, following an unique state sequence, starting from the *start* state and arriving in state s at the moment of time t.

We compute the value for this variable inductively, similarly as for the forward variable:

1. *Initialisation* For $t = 1$, we have

$$\delta_1(s) = P(start \rightarrow s_1)P(o_1 \mid s_1)$$

2. *Induction* For $t = 2, \dots, T$ we have

$$\delta_t(s) = \left(\max_{s' \in S} \left(\delta_{t-1}(s')P(s' \rightarrow s) \right) \right) P(o_t \mid s) \tag{2.2}$$

Knowing $\delta_T(s)$ for every s, we can compute the maximal probability of generating O for the model H, following an unique state sequence:

$$P(O \mid H, \mathcal{V}) = \max_{s \in S} \left(\delta_T(s)P(s \rightarrow end) \right)$$

Unfortunately, this is not what we are looking for. In this case we need the exact state sequence and for this, we need to memorize at each step, the state that maximizes equation 2.2. We introduce a new variable for designating this state: $\psi_t(s)$. The procedure that computes this variable is similar to the anterior procedure, with the difference that this time we need $argmax$ in place of max:

1. *Initialisation* For $t = 1$, we have

$$\psi_1(s) = start$$

2. *Induction* For $t = 2, \dots, T$ we have

$$\psi_t(s) = argmax_{s' \in S} \left(\delta_{t-1}(s')P(s' \rightarrow s) \right)$$

Once the two variables have been computed for each state s, it rests only to apply a backtracking method in order to obtain the Viterbi state sequence: $start, s_1^*, \dots s_T^*, end$, starting from *end* state.

1. *Initialisation* For $t = T$, we have:

$$s_T^* = argmax_{s \in S} \left(\delta_t(s)P(s \rightarrow end) \right)$$

2. *Traceback* For $t = T - 1, T - 2, \dots, 0$ we have:

$$s_t^* = \psi_t(s_{t+1}^*)$$

It should be noted that the Viterbi algorithm is similar (except for the backtracking step) in implementation to the forward algorithm. The major difference is the maximization in equation 2.2 over previous states which is used in place of the summing procedure in equation 2.1. As for the previous 2 algorithms, the complexity is $O(N^2T)$.

3 Profile HMMs

3.1 General Description

For modeling protein sequence multiple alignments, a particular type of HMM has been developed. This new and particular type of HMM is called a *Profile HMM* [Kro98]. Given a correct multiple alignment from which we build the model, the purpose would be to use this profile HMM to find and score potential matches to new sequences. For complete description of how profile HMMs and probabilistic models are used in computational biology, see [Kro98], [DEKM98] or [BB98].

One difference between the original HMM model and a profile HMM is that the former can have a general structure, but the latter not. The heart of a profile HMM is in fact a linear set of match (M) states, one for each consensus column in the multiple alignment. Each M state emits (aligns to) a single residue, with a probability score that is determined by the frequency that residues have been observed in the corresponding column of the multiple alignment. Each match state therefore carries a vector of 20 probabilities, for scoring the 20 amino acids.

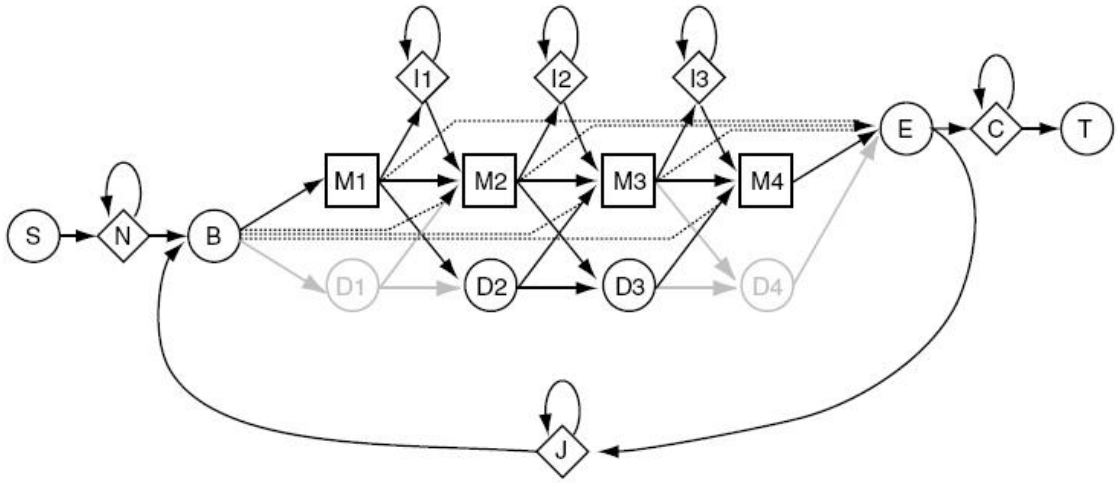

Figure 4: Profile HMM Architecture - HMMER

For our implementation we use a profile HMM structure similar to the structure that **HMMER** implements (see figure 4). *HMMER* is an implementation of profile HMMs for biological sequence analysis [Edd], that offers multiple applications in this field. Once a multiple sequence alignment obtained, using **HMMER** one can build a model, calibrate it (see 3.5) and then align different sequences to this model or search a sequences database for matches of the constructed HMM model.

The most important characteristic of a profile HMM is that it is capable of modeling gapped alignments, e.g. including insertions and deletions, which lets the software describe a complete conserved domain (rather than just a small ungapped motif); moreover the way of building such a model is described in section 3.4. Insertions and deletions are modeled using insertion (I) states and deletion (D) states. Each match state has an I and a D state associated with it. A group of such three states, M/D/I can be named a node.

The core section of the model in figure 4 is composed by M/D/I nodes and flanked by dummy non-emitting states, B and E. The transitions are arranged so that at each node, either the M state is used (and a residue is aligned and scored) or the D state is used (and no residue is aligned, resulting in a deletion-gap character, -). Insertions occur between nodes, and I states have a self-transition, allowing one or more inserted residues to occur between consensus columns. All the probability

abbrev.	name	index	description	symbol emission
M_x	Match	x		K emissions
D_x	Delete	x		non-emitter
I_x	Insert	x		K emissions
S	Start	-		non-emitter
N	N	-	unaligned sequence state	K emissions
B	Begin	-	enter main model	non-emitter
E	End	-	exit main model	non-emitter
C	C	-	unaligned sequence state	K emissions
J	J	-	unaligned sequence state	K emissions

Table 2: State Abbreviations

parameters in the main model are generally estimated from observed frequencies of residues and transitions in a multiple sequence alignment. The way of structuring the model parameters in the profile HMM model files is explained in section 3.3

The other states (S,N,C,T,J) are "special states", which allow different types of alignments. They control algorithm dependent features of the model: e.g. how likely the model is to generate various sorts of local or multi hit alignments (see section 3.2). The algorithm dependent parameters are typically not learned from data, but rather set externally by choosing a desired alignment style.

3.2 Solving Different Types of Alignments to a Profile Model in HMMER

The alignment of a sequence to a profile model can be local/global with respect to the sequence or the model, depending on whether the complete sequence/model is aligned or only a segment of the sequence/model is aligned.

There are several types of possible alignments of a sequence to a profile model. The "global" alignment refers to the fact that the complete target sequence is aligned to a complete path through the model from a start state (S) to the terminal state (T). However, some (possibly most) of the sequence may be assigned to the states N,C,J that generate non homologous, unaligned, random sequence that is not aligned to the main model. Thus, the algorithm dependent parts of the model control the apparent locality of the alignments.

Local alignments with respect to the sequence (i.e., allowing a match to the main model anywhere internal to a longer sequence) are controlled by the N and C states. If the $N \rightarrow N$ transition is set to 0, alignments are constrained to start in the main model at the first node. Similarly, if the $C \rightarrow C$ transition is set to 0, alignments are constrained to end from the very last node.

Local alignments with respect to the model (i.e., allowing fragments of the model to match the sequence) are controlled by $B \rightarrow M$ "entry" transitions and $M \rightarrow E$ "exit" transitions, shown as dotted lines in figure 4. Setting all entries but the $B \rightarrow M_1$ transition to 0 forces a partially "global" alignment in which all alignments to the main model must start at the first match or delete state. Setting all exits to 0 but the final $M \rightarrow E$ transition (which is always 1.0) forces a partially global alignment in which all alignments to the main model must end at the final match or delete state.

The profile HMM structure implemented for HMMER, allowing local alignments, is significantly improved compared to the original Krogh/Haussler profile, as in real alignments of real biological sequences we usually use models of conserved domains, and rarely of a complete target sequence. A conserved domain may be buried in a longer target sequence; there might be more than one

conserved domain in the target sequence; and maybe the target sequence only contains a smaller fragment of the conserved domain. Moreover, sequences may not be complete for technical reasons.

3.3 HMMER Model File

The HMMER format, displayed in figure 5, provides all the necessary parameters to compare a protein sequence to a HMM, including the search mode of the HMM (local vs. global, and more), the null (background) model, and the statistics to evaluate the match on the basis of a previously fitted extreme value distribution.

3.3.1 The Null Model

Usually the *null model* is a simple one-state HMM that says that random sequences are sequences with a specific residue composition. The null model used in HMMER and in our implementation states the expected background occurrence frequencies of the 20 amino acids and also has a parameter, called p_1, which is the G→G transition probability, where G is the unique state of the null model.

For protein models, by default, the 20 residue frequencies are set to the amino acid composition of SWISS-PROT 34 [4], and p1 is set to 350/351, which comes from the fact that the mean length of a protein is about 350 residues).

```
HMMER2.0  [2.2g]
NAME  ic_pprs_S_motifs
LENG  31
ALPH  Amino
RF    no
CS    no
MAP   yes
COM   /Users/Shared/hmmer2.2g/binaries/hmmbuild s.hmm /Users/ian/Projects/PPRs/
PPRdbInterface/build/ic_pprs_S_motifs.aln
COM   /Users/Shared/hmmer2.2g/binaries/hmmcalibrate s.hmm
NSEQ  995
DATE  Fri Aug 23 16:10:32 2002
CKSUM 1818
XT      -8455     -4  -1000  -1000  -8455     -4  -8455     -4
NULT      -4  -8455
NULE     595  -1558     85    338   -294    453  -1158    197    249    902  -1085
-142    -21   -313     45    531    201    384  -1998   -644
EVD   -24.998384   0.299079
HMM         A      C      D      E      F      G      H      I      K      L
M      N      P      Q      R      S      T      V      W      Y
         m->m   m->i   m->d   i->m   i->i   d->m   d->d   b->m   m->e
           -2      *  -9757
     1  -2134   -304   2791    116  -2035  -1175    441  -2634  -1512  -2791  -2726
2342  -2478  -2022  -1133    225   -713  -3038  -3594  -1081     1
```

Figure 5: HMMER Model File

The format of the HMMER profile files consists of one or more HMMs. The format for an HMM is divided into two regions. The first region contains text information in a tag-value schemes. This section is ended by a line beginning with the keyword *HMM*. The second region is of a more fixed whitespace-limited format that contains the main model parameters. It is ended by the // that ends the entire definition for a single profile HMM (this allows including multiple HMM models in the same file).

[4]SWISS-PROT is a protein sequence database distributed by European Molecular Biology Laboratory (EMBL).

In the header section, the important pieces of information for us are: the length of the model (the number of match states), 8 "special" transitions for controlling the algorithm-specific parts of the model (there are four transition probability distributions for N,E,C,J; each distribution involves 2 probabilities, the first one for "moving" and the second one for "looping"), the transition probability distribution for the null model (single G state), the symbol emission probability distribution for the null model.

The main model section starts with an atypical line; it contains 3 fields, for transitions from the B state into the first node of the model. The remainder of the model has 3 lines per node, for M nodes (where M is the number of match states, as given by the model length line). These 3 lines are: the **match emission** line (K numbers for match emission scores, 1 per symbol, in alphabetic order), the **insert emission line** (similar to the match emission line), the **state transition line** (9 fields that represent state transition scores).

The insert emission and state transition lines for the final node M are special, as it has no insert state, and no next node; the $M \rightarrow E$ transition score for the last node is always 0, because this probability has to be 1.0. The way of computing probabilities from the HMMER scores is explained in section 4.2.

3.4 Profile HMM Construction

In the construction of a Profile HMM, we need to focus on 2 problems.

1. The choice of the length of the model: it corresponds in fact to a decision on which multiple alignment columns to assign to match states, and which to assign to insert states. We could use a simple heuristic rule to decide this, like modeling by inserts the columns that are more than half gap characters. Another solution would be to use a MAP[5] algorithm. This MAP algorithm, called the Match-Insert Assignment algorithm recursively calculates the log probability of the optimal model for the alignment up to and including column j, assuming that column j represents a match state (it is marked).

   The probability parameters are estimated from the counts that are implied by marking columns i and j, while leaving unmarked the intervening columns (if any). Further details over the MAP algorithm can be found in [DEKM98].

2. Once the structure is chosen, the second problem that needs to be solved is how to assign the probability parameters. The best way of doing this is to start with a "good" configuration and to continue with a training phase, in order to refine the parameters (see Baum-Welch algorithm in section 2.2).

   Since the alignments are given, we can use the *maximum likelihood estimators* as a starting point. We just need to count up the number of times each transition or emission is used, and assign probabilities according to this. If we dispose of a very large number of sequences in our training alignments, this method gives an accurate and consistent estimate of the probabilities. However, it has problems when there are only a few sequences. The major difficulty is that some transitions or emissions may not be seen in the training alignment, and so would acquire zero probability, which would mean they would never be allowed in the future. As a minimal approach to avoid zero probabilities, we can add pseudocounts to the observed frequencies. The problem with the pseudocount approach is that only the most rudimentary prior knowledge can be contained in a pseudocount vector. For this reason, we need a lot of example data in the alignment, to get good estimates of the parameters.

[5] **M**aximum **A**posteriori **P**robability **A**lgorithm

Other proposed approaches that include better prior information are: a mixture of Dirichlet distributions or substitution matrix mixtures.

3.5 Scoring Methods and Statistical Significance of Structural Alignment Scores

3.5.1 The Negative Log Likelihood (NLL) - Score

Instead of computing the probability $P(O \mid H)$, the forward part of the forward-backward procedure, described in section 2.2.1 can be used efficiently to compute $-\log P(O \mid H)$, the negative logarithm of the probability of a symbol sequence, given the model. We call this the *negative log likelihood (NLL)-score* of the sequence. The average NLL-score of a training sequence is inversely related to the likelihood of the model, and hence serves as a numerical measure of progress for each iteration of the training procedure. The NLL-score can also be used to evaluate how well the model fits a novel "test" sequence. In the case of the Viterbi's algorithm, instead of calculating the NLL-score for a sequence, which involves all the possible paths for that sequence through the model, we compute the negative logarithm of the probability of the single most likely path for the sequence.

Using logarithms allows seeing the probabilities in terms of "gap penalties": $-\log P(x_{l(i)} \mid s_i)$ represents a penalty for aligning the amino acid $x_{l(i)}$ to the position represented by state s_i in the model. The term $-\log P(s_i \mid s_{i-1})$ corresponds to a penalty for using the transition from s_{i-1} to s_i in the model. If this is a transition from a match state to a delete state, then this represents a gap-initiation penalty; if it is from a delete state to a delete state, it represents a gap-extension penalty; if it is from a match state to an insert state, it represents an insertion-initiation penalty; and if it is a transition from an insertion state to itself (a self-loop), then it represents an insertion extension penalty. This means that profile HMMs model affine gap penalties [DEKM98].

3.5.2 The Bit Score

The HMMER bit scores reflects whether the sequence is a better match to the profile model (positive score) or to the null model of non homologous sequences (negative score).

The *bit score* is a log-odds score in log base two (thus, in units of *bits*). Specifically, it is:

$$S = \log_2 \frac{P(O \mid H)}{P(O \mid null)}$$

where $P(O \mid null)$ is the probability of the target sequence given a "null hypothesis" model of the statistics of random sequence (the null model defined in section 3.3.1).

Thus, a positive score means the HMM is a better model of the target sequence than the null model is.

3.5.3 E-value

Definition and General Description

In database search applications in computational biology, we generally wish to determine the statistical significance of a score. Regardless of the means used to arrive at the score, we will wish to know what the probability to see a given score, by chance, in a database of negative (non homologous) sequences, is.

This is the purpose of the *E-value*. The E-value is calculated from the bit score. It tells you how many false positives you would have expected to see at or above this bit score and therefore it is the expected number of false positives with scores at least as high as your hit. Therefore, the lower

E-value, the better it is; an E-value of 0.1, for instance, means that there is only a 10% chance that you would've seen a hit this good in a search of non homologous sequences. The most common thresholds are 0.01 and 0.05. More stringent values are usually used as it is considered that the obtained results are not very accurate. The E-value is dependent on the size of the database you search: the larger the database you search, the greater the number of expected false positives.

While the bit score is telling you how well the sequence matches your HMM, the E-value measures how statistically significant the bit score is. Generally, true homologs have both a good bit score and a good E-value.

E-value Calculation Using Maximum Likelihood Fitting of *Extreme Value Distributions* (EVD)

We generally define the $\mathcal{P}$-value as the probability of observing an event as strong or stronger than you observed, given the null hypothesis; i.e., "How likely is this event to occur by chance?". Particularly for our case, a $\mathcal{P}$-value is the probability of seeing at least one score S greater than or equal to some score x in a database search of n sequences: $P(S \geq x \mid null)$. In the same manner we can define the E-value, as the expected number of non homologous sequences with scores greater than or equal to a score x in a database of n sequences: $nP(S \geq x \mid null)$.

We therefore need a solution for $P(S \geq x)$, the distribution of the score statistic x. Empirically, if you obtain a score of 75, how should the $P(S \geq 75)$ probability be computed? First, we would have to compute many scores using random queries and a random database, next summarize those scores into a distribution and finally compute the area of the distribution to the right of the observed score.

The problem that intervenes is finding the type of distribution correspondent to the set of sequence similarity scores. Analytical solutions for the distribution $P(S \geq x)$ only exist for ungapped (no insertions and no deletions admitted) local alignments, where the distribution is an *extreme value distribution* (EVD) (it looks similar to a normal distribution, but it has a larger tail on the right), see figure 6.

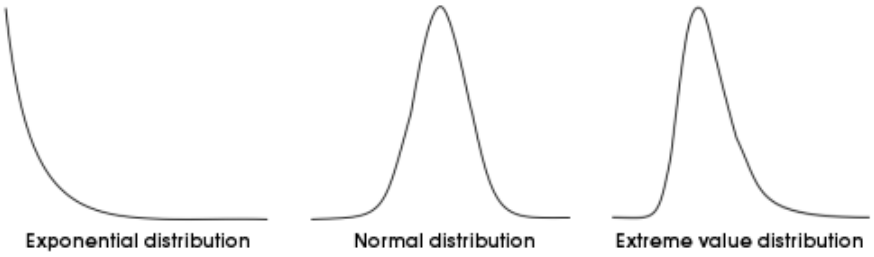

Figure 6: Different Distribution Types

For the EVD we have the following distribution function:

$$P(S \geq x) = 1 - exp[-e^{-x}], \tag{3.1}$$

and for the case of a particular type of score

$$P(S \geq x) = 1 - exp[-e^{-\lambda(x-\mu)}], \tag{3.2}$$

where μ is the location and λ is the scale factor.

For the case of gapped local alignments, simulations indicate that these scores also roughly obey an extreme value distribution, but the relevant parameters can only be determined by simulation

and empirical curve-fitting. Programs such as FASTA, BLASTA and in our case HMMER use such techniques to fit observed score distributions to an EVD.

The simplest fitting method to implement is the *linear regression* method. However, the linear regression method is significatively inferior to a second possible approach *maximum likelihood*. Maximum likelihood (ML) fitting to the EVD has been described in the literature in [Law82] and [Mot92] and has been implemented in HMMER, [Edd97]. Experiments showed that ML fit for 100 data points is better than the linear regression fit for 10,000 data points.

We may wish to fit only the right tail of the histogram, rather than the whole histogram; for example, the left (low scoring) tail may be contaminated with very poor-scoring sequences that do not conform to the EVD, so we do not wish to include them in the fit. We call this a fitting of "censored" histograms.

HMMER, through his calibration program, attempts to do a robust EVD fit to an observed histogram. It censors the data below the peak of the histogram, so it only fits the right side of the curve; it removes outliers to the right first, by cutting above a given score and then iteratively lowers the cutoff and reestimate μ and λ until the high cutoff is an estimated E-value of 1. Finally, a certain loss of precision is introduced because it fits to a binned histogram rather than fitting to a list of observed scores. On simulated EVD data with noise injected on the high side, the implementation is moderately robust and 10,000 samples seem to be sufficient for reasonable fits.

4 Cyclic Profile HMMs

We deal with the problem of retrieving the repeated apparitions of given motifs in protein sequences. In particular we are interested in the apparitions of the PPR related motifs in the *Arabidopsis thaliana* PPR proteins. As input, we dispose of the profile models built for each given motif: the 4 PPR motifs P, L, S, L2 and the 3 non-PPR motifs E, E+, DYW. The profile models were constructed using HMMER, from sequence fragments alignments, as described in section 3. Given the amino acid sequence of a protein, we need to identify the succession of the 7 motifs.

Like it was already mentioned, we propose a **Cyclic Profile HMM structure**.

4.1 Structure

In order to retrieve the motif sequence for the PPR proteins, we have to take into consideration the information that we already have about the motif structure of this group of proteins. We know that the PPR proteins' sequences are generally divided into 2 structurally different parts [RBTNL06]:

- the initial part which is composed from repeated apparitions of the PPR motifs (as described in figure 2);
- the final part (which can be missing) is composed from the 2 final PPR motifs and the 3 non-PPR motifs (depending on the class, as explained in table 1); this part is usually characterized by single apparitions of each motif.

Therefore, the structure that we designed for this particular problem has 2 different parts (see figure 7), in order to model the protein sequence in the best possible way:

- the cyclic profile HMM part; in the final version of the program this part of the structure is composed of 3 profile models that are completely connected to each other: the profile model built for P, the one built for L and the one built for S. It ends with the (T) state which has transitions towards any of the final profile models and towards the final state of the structure,

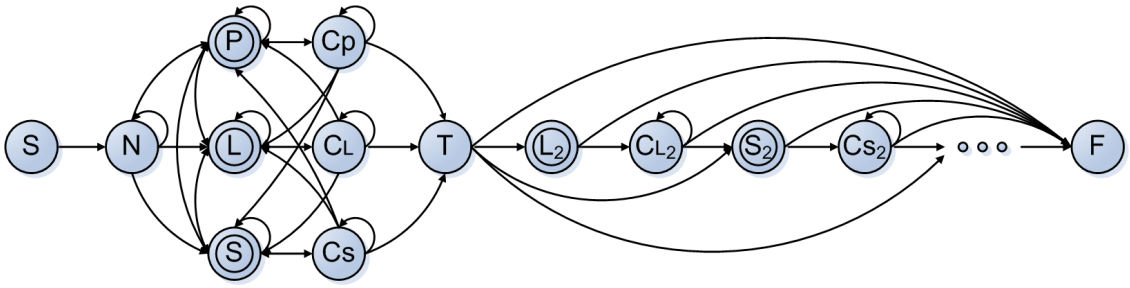

Figure 7: Profile HMM Cyclic Structure

(F) (in order to model inclusively the proteins that have no final linear part, which belong to the A class, see table 1). As the figure explains, we allow transitions between any 2 profile models. This part of the structure is the most important, as it is the true innovation of our approach.

- the linear profile HMM part; composed of 5 profile models that come one after the other, in a fixed linear order. As to fit all 4 classes of proteins, it is possible to arrive in the final state of the structure (F) from any of the final motifs (in order to model the 4 classes of PCMP proteins, see table 1).

Figure 7 represents the schema of the implemented structure. The states that appear in the figure have the following signification:

- *the 2-circle states* are in fact complete profile HMMs (the names of these states correspond to the names of the motifs);
- *S state* has the same role as for the HMMER profile structure (see section 3.1); it is the initial state of the model, non-emitter;
- *N state* has the same role as in HMMER; it is a random sequence emitting state (the probability distribution over all the symbols of the alphabet corresponds to the composition of SWISS-PROT 34, like for the null model emissions, described in section 3.3.1). N is unaligned to the sequence; it covers the initial section of the sequence that is not part of any motif;
- *T state* is the final state of the cyclic part of the structure. It is a non-emitter state that connects the two parts: the cyclic and the linear one;
- *F state* is the non-emitter final state of the complete structure;
- *C states* play the same role as the J states in figure 4. As N, Cs are unaligned, random sequence emitter states. Each C state models insertions between two consecutive motifs. The final C state of the structure, that follows after the last of the profiles in the linear part of the structure, has the same role as the C state from figure 4: modeling the final section of the protein sequence, not part of any motif.

As mentioned, the 2-circle states are complete profile models whose internal structure is similar to the structure of the HMMER profile models, see figure 4. Still, we brought some changes to this structure, which are detailed in the 3 paragraphs below.

In the original structure, the transition from the B state (the begin state which enters the main model) to the first match state of the model and also the transition from the last delete state to the

E state (the end state which exits the main model), were eliminated (in the figure they were greyed out). Internally in HMMER, these delete states exist in the "probability form" of the model (when the model is being worked with in every way except alignments) but they are carefully removed in the "search form" of the model (when the model is used for alignments). The problem appears because of the J state. If a continuous non-emitting "mute cycle" were allowed (J, B, D states, E, and back to J), dynamic programming recursions would fail. The probabilities for these paths are being absorbed by the entry and exit probabilities.

For us the problem is slightly different and so, we allowed the 2 mentioned transitions (the transition from the B state to the first match state and the transition from the last delete state to the E state). In the case that such a continuous non-emitting "mute cycle" would appear, the penalty for following the sequence of delete states (penalty in the sense that for entering a delete state we have a very low score) would be so big, that we would obtain a much better score from another possible state sequence; if not, if we do not obtain a better state sequence, it signifies that the motifs were so alienated from the original ones, that we were not able to recognize any of them.

As we allowed the begin-delete and the delete-end transitions, we were able to suppress the begin-match transitions, except the one to the first match state, and the match-end transitions, except the one of the final match state towards the end (the probabilities for the begin-match transitions were added up in order to obtain the single begin-match transition probability and the probability for the single match-end transition becomes 1.0). We had another reason to eliminate these match and towards match transitions: because we are doing global alignment to the model, and so, we want to penalize any deletion of the initial or the final part of the profile model.

The cyclic structure is completely parameterised: the number of profile models that make up this part of the structure can be modified, the transition probabilities between the models are given as input and they can be modified as needed, transitions between any of the models are allowed, but if we do not want to admit one of the transitions, we simply have to set that specific transition probability to 0.

The structure, when referring to the implementation, was designed in order to benefit from the particularities of this special type of HMM: most of the structure is linear, it is partitioned in fully independent regions (there are no transitions that cross 2 regions, e.g. the match states of a profile model are not connected to the match states of another profile model), for each state we have a small limited number of entering transitions, e.g. from a match state we can only pass to the next match state, to the next delete state, to the correspondent insert state, or, in the case of the last match state, to the final state of the profile model. For combinatorial reasons (see section 4.3 for the number of states of the model) we can not create a general structure that could store the probability transitions between any pair of states, but we need to particularize the structure for every type of state. This, of course, changes completely the implementation of the Viterbi algorithm (see section 4.3).

4.2 Transition Probabilities

We separate the transitions in 2 categories:

- *the profile models internal transitions*; the probabilities for these transitions are retrieved from the HMMER profile model files;
- *the profile models marginal states transitions*, which refer to the transitions from the E and the C states to the C and the B states, from N to the B states of the models from the cyclic part of the structure and to the T state, from T to the B states of the models from the linear part and to the F state; they are given in a separate file or computed evenly (the probability

of a transition exiting from a state is 1.0 divided to the total number of transitions exiting from that specific state).

4.2.1 Profile Models Internal Transitions Probabilities

The probabilities for the *internal transitions* are therefore computed from the scores given for these transitions in the HMMER profile model files (the format of HMMER files was detailed in section 3.3).

As mentioned, HMMER files deal with scores and not with probabilities. Still, we need to make the score conversion to probabilities in order to renormalizes the probability distributions to sum to 1.0 and so, to eliminate minor rounding/conversion/numerical imprecision errors. Next, we pass once again to scores. We need to work with scores instead of probabilities because scores have a better range (probabilities can quickly converge to 0), and because the numbers are more meaningful to a human reader to a certain extent: positive values mean a better than expected probability, and negative values a worse than expected probability.

The score that we obtain is in fact the bit score from section 3.5.2. The calculation of scores from probabilities is:

$$score = \lfloor 0.5 + INTSCALE * log_2(prob/null - prob) \rfloor ,$$

so conversely to get a probability from the scores in an HMM save file:

$$prob = null - prob * 2^{score/INTSCALE},$$

where INTSCALE represents an integer value used to scale the integer scores (the scores that we obtain are too big to work with).

Notice that you must know the null probability to convert scores back to HMM probabilities. For the match and the insert emissions, the null probability used to convert them is the relevant null model emission probability. The match, insert, delete states transitions have 1.0 as null probability, as like as B, and E state transitions.

4.2.2 Profile Models Marginal State Transitions Probabilities

The probabilities for the *profile models marginal states transitions* are given in a separate file or computed evenly and afterwards, transformed in scores, as above. For N and C, the loop transitions have the null model loop transition as null probability, and the move transitions have the null model move transition as null probability; T and F, as they are non-emitter states, have 1.0 as null probability.

We have tried to estimate as well as possible the probabilities of transition between the marginal states of the profile models, employing the maximum likelihood estimation type. Using the existing manual annotation, we quantified the number of times a motif came after another, the number of times a certain motif started or ended the motif sequences, the number of times a PPR motif ended the motif cyclic part of the motif sequence. Afterwards, we used the obtained counts to estimate the transition probabilities, in the following manner:

1. the $E_i \rightarrow C_i$ and the $C_i \rightarrow C_i$ transition probabilities could not be estimated from the manual annotation, as we lack information about the gap lengths between two consecutive motifs. They were, therefore, estimated using the personal annotated sequences and then eliminated from the probability distribution in order to compute the other E and C transition probabilities;

2. the $E_i \rightarrow B_j$ and the $C_i \rightarrow B_j$ transition probabilities (where i and j denote 2 motifs which can be different or not) were estimated by dividing the number of times motif j follows motif i in the annotated sequences by the total number of times the i motif appears;

3. the $N \rightarrow B_i$ transition probabilities (where i denotes one of the PPR motifs) were estimated by dividing the number of times the i motif is the first motif of the motif sequence, over the total number of amino acids that appear at the beginning of the protein sequence without being assigned to a motif, added to the number of times each motif is the first one in the sequence;

4. the $N \rightarrow N$ transition probability was estimated by dividing the total number of amino acids that appear at the beginning of the protein sequence without being assigned to a motif, over the same sum as for the $N \rightarrow B_i$ transitions;

5. the $T \rightarrow F$ transition probability was estimated by dividing the number of times one of the cyclic motifs is the last one of the sequence (i.e., the times we do not have any linear part at the end of the sequence) over the number of proteins;

6. the $T \rightarrow B_i$ transition probabilities (where i denotes one of the final non-cyclic motifs) were estimated by dividing the number of times the motif i is the first of the linear motif sequence, over the number of proteins;

A simple improvement to this counting technique was used: add 1 to each estimated count in order to avoid 0 probabilities, and so, give a chance to every existing transition. Thanks to this improvement, we were able to recognize even the exception cases. More about the results obtained using the counting and the improved counting technique, in section 5.

4.3 Viterbi Algorithm

From all the state sequences that can generate a given amino acid sequence, we need the one with the highest probability. Viterbi's algorithm gives exactly this state sequence, see section 2.2.2 for details.

The induction phase of the Viterbi algorithm consists of maximizing over all the possible previous states (of the current state) scores. Because of the huge number of states that compose the HMM model: $< 1 \times S + 1 \times N + 8 \times C + 8 \times E + 8 \times B + 1 \times T + 1 \times F + 34 \times 3$ (match, insert and delete states for P profile model) + 35×3 (L model) + 31×3 (S model) + 36×3 (L2 model) + 31×3 (S2 model) + 91×3 (E model) + 33×3 (E+ model) + 106×3 (DYW model) = $1219 states >$, we could not possible try any 2 state combinations as in the original implementation of the algorithm.

The structure that we designed for this particular type of HMM gives us the opportunity to implement a more efficient version of Viterbi's algorithm (see the arguments in favour of this affirmation in section 4.1). As already explained, every state of the model has a small limited number of previous possible states; that is why the implementation of the algorithm must be fully dependent on the typology of the structure, e.g. for a B state we need to verify, like previous states, only the C and the E states (all the C and E states of the models in the cyclic part, in case B is the begin state of a cyclic model, only the previous profile model C and E states, in case B is the begin state of one of the linear final models), the N state (in case B is the begin state of one of the cyclic models) and the T state (in case B is the begin state of one of the models from the final linear part).

As a result, the score for the final state is the score of the best motif sequence for the given protein sequence. During the algorithm we also build a traceback matrix that keeps, for every state, the

previous chosen state. The traceback algorithm retrieves the corresponding state sequence, starting from the final state and going backwards towards the initial state of the model. Once again, the number of states asks for a lot of attention in the implementation phase; the dynamic programming structures, used for computing the score and for retrieving the state sequence, are allocated and resized using the reallocation procedures in place of the deallocation-allocation strategy.

In section 2.2.2 we mentioned that the Viterbi's algorithm complexity is N^2T, where N is the number of states and T is the length of the symbol sequence. Taking advantage from the particular structure of the profile models, HMMER manages to obtain a complexity linear in N, NT. In our case, as we apply the same type of optimizations, we obtain a linear complexity in the number of states, $NT + N\_Mo^2T$, where N_Mo is the number of profile models from the cyclic part. Even though we obtain a square complexity in the number of motifs participating in the cyclic part, as the number of such motifs is usually small (in our case 3), we still get a linear complexity.

4.4 Calibration of the Model

When searching a protein sequence with an HMM model, by using the Viterbi algorithm, we obtain the best possible motif sequence, as explained in section 4.3. The problem with these scores is that we are not able to measure their significance. Obtaining the statistical significance of a given score, means computing the expectation value (E-value) of this score.

We have previously defined the E-value, as the expected number of non homologous sequences with scores greater than or equal to a score x in a database of n sequences: $nP(S \geq x \mid null)$. Analytical solutions for the distribution $P(S \geq x)$ only exist for ungapped local alignments, where the distribution is an *extreme value distribution* (EVD). For the distribution function of the EVD, see equation 3.2. Maximum likelihood (ML) fitting to the EVD is the best method of estimating the μ and λ parameters (details over the EVD distribution parameters computation in section 3.5.3).

For example, for the computed EVD parameters $\mu = -26.876400$ and $\lambda = 0.423303$, for a score of -29.076000, we obtain an E-value of 0.92 and for a score of 784.775024, we obtain an E-value of $2.3e(-236)$.

The number of random generated sequences, for which we compute the histogram that is used to do the fitting, is a parameter of the program; like HMMER, we usually use a number of 5000 random sequences that seems to be a pretty good choice; it was empirically chosen as a tradeoff between accuracy and computation time.

4.5 Types of Output

```
>AT1G18485
LSPLSPLSPLSPLSPLSPL2S2EE+DYW
>AT1G19720
LSPLSPPPLSPPPLSPL2S2EE+DYW
>AT1G31920
SPLSPLSPL2S2EE+DYW
>AT2G33760
SPLSPLSPL2S2EE+DYW
>AT3G61170
SSPLSPLSPLSPLSL2S2EE+DYW
```

Figure 8: FASTA Output

Our program offers 2 types of output:

- *the simple motif sequence output* given in a format similar to the unaligned sequence FASTA format (see figure 8): each sequence is preceded by a line starting with >; the first word on this line is the name of the sequence. The remaining lines contain the motif sequence itself.

  This is a standard format in bioinformatics and it is also the format of the manually annotated sequences and so we use this type of output to make comparisons between the 2 set of results.

- *the complete alignment* of the amino acid protein sequence to the HMM model, with the consensus correspondent for each symbol of the sequence and the motif correspondent to each region of the protein sequence (see figure 9).

  The top line is represented by the profile HMM consensus; the gaps between the motifs are modeled by hyphens(-) and the insertions by ".". The amino acid shown for the consensus is the highest probability amino acid at that position according to the HMM. Capital letters mean "highly conserved" residues: those with a probability larger than 0.5 for protein models.

```
>AT1G31920: score 784.546021, E-value   2.7e-236
*-------------------------------------------------------------[dvvvyna
                                                              ++++ ++
 mikapilqsllasrddlthnpevnnfggkeqeclyllkrchnidefkqvharfiklslfys SSFSASS
 NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN[S_____

LidmYaKcG...GdleeArkvFdeMpe][dvvtyntlIsglckaGrleeAlelfeeMkekGi.i][defT
++  +a +G ++ +   A  +         ++ +nt+I+g+++   +eeAl +++eM+++G +   d+fT
VLAKCAHSGwenSMNYAASIFRGIDDP  CTFDFNTMIRGYVNVMSFEEALCFYNEMMQRGNeP  DNFT
_____________________________S][P_____________________________________P][L___

lasvLkACaslgaLslGkqiHgyviKsGfds][dvvvynaLidmYaKcGdleeArkvFdeMper][dvvt
++++LkAC++l ++++GkqiHg+v K+G++   dv+v+n+Li+mY++cG++e    vF++++ +  +  +
YPCLLKACTRLKSIREGKQIHGQVFKLGLEA  DVFVQNSLINMYGRCGEMELSSAVFEKLESK  TAAS
_______________________________L][S_______________________________S][P___

yntlIsglckaGrleeAlelfeeMkekGiP]--[defTlasvLkACaslgaLslGkqiHgyviKsGfds]
+++++s+ + +G+++e l lf+ M ++        e+ ++s+L ACa+ gaL+lG++iHg+ +++  ++
WSSMVSARAGMGMwSECLLLFRGMCSETNL ka EESGMVSALLACANTGALNLGMSIHGFLLRNISEL
_______________________________P]CC[L___________________________________L]
```

Figure 9: Complete Alignment

The center line shows letters for "exact" matches to the highest probability residue in the HMM, or a "+" when the match has a positive score and is therefore considered to be "conservative" according to the HMMs view of this particular position in the model.

The third line shows the sequence itself; deletions are modeled by hyphens (-).

The last line shows the motif correspondent to each region of the sequence; it also aligns the gap modeling states (N, C) to the amino acid sequence.

This type of output is extremely useful to the biologists, as it helps them retrieve the structure of a particular motif, visualize the mutations that appeared in the motifs (eventually find new motif variants). This global motif sequence alignment was never performed before.

5 Experiments

Along the implementation phase of the research project, we have done several types of experiments. We have been experimenting different solutions for the marginal states transitions probabilities (described in section 4.2) and different types of structures. The results have been estimated using

the manually annotated sequences for the PPR proteins of **Arabidopsis Thaliana** (the annotation of the protein sequences was done in the same format as the first type of output of our program, see section 4.5). We also applied our HMM model on the proteins of the **rice**, for which we obtained the first systematic annotation. The model was also tested on **random chosen non PPR proteins** of Arabidopsis; the obtained results proved that the PPR-motifs are entirely specific to the PPR proteins.

In the result tables the following notations are being used:

- *same length sequences* signifies the proteins for which we obtained a motif sequence with the same length as the one obtained using manual annotation. The motif sequences that we obtain (if they are different in length from the manually annotated ones) include the motif sequences obtained using manual annotation (only the prefix or the suffix differ); we call the proteins that enter in this category *different length sequences*;
- *identical* denotes the number of proteins which have exactly the same motif sequences for both annotations; *identical common part* has the same meaning, only that this time we talk about different length motif sequences aligned to the left or to the right (the common part represents the regions in the 2 sequences that have the same length, equal to the minimum length of the 2 sequences, obtained after the one side alignment);
- $\approx$*identical* includes the proteins with motif sequences that differ only for the motifs for which the manual annotation is not sure of (marked with an interrogation sign);
- *diff.* signifies the proteins for which our program and the manual annotation gave completely different results.

Observation It is important to underline the fact that for the proteins for which we obtained a motif sequence of different length of the one obtained using manual annotation, the two lengths usually differ by 1 or 2 motifs. Also, for the proteins for which we found a different annotation, in the majority of cases the difference comes from 1 or 2 motifs (possibly because of an error in the manual annotation).

5.1 Transition Probabilities Experiments

The transition probabilities experiments were done on the complete structure (that allows transitions between any of the cyclic motifs).

- *probabilities estimated by counts*: as explained in section 4.2.2 we used a maximum likelihood parameter estimation, in a rough approach and in a slightly improved approach (by adding 1 to the obtained counts in order to allow the production of unseen situations).
- *almost evenly distributed probabilities*: equal probabilities for all the transitions that exit a certain state. We call them "almost" even because, as it was empirically observed, the loop-N transition probability should be very close to 1.0, in order to model the initial region of the protein sequence that is not part of any motif.

  We observed that giving a smaller probability to the loop-N transition determines the HMM model to find additional not annotated motifs at the beginning of the protein sequence. This is due to 2 causes: the beginning of the sequence is composed of very alienated motifs that were not taken into consideration by the manual annotation, or, the model simply obtained a better score when assigning several motifs to this part of the sequence than when considering it as a gap.

	same length sequences			different length sequences		
type of prob. estimation	identical	≈identical	diff.	identical common part	≈identical common part	diff. common part
max. likelihood estim.	95	30	6	37	18	10
improved max. likelihood estim.	94	30	7	38	17	10
even distribution	97	33	9	33	13	11

Table 3: Comparative Results for the Transition Probabilities Experiments

The results obtained for the two types of probability distribution, are summarized in table 3. Unexpectedly, the best results were obtained when using the almost even probability distribution approach: we consider that we obtained the best results because we have found the largest number of identical motif sequences, **97 from a total number of 196 proteins tested**. The number of proteins for which we obtained different motif sequences slightly increased. The explanation for both results is that when using this type of distribution we make no initial assumption on the motif structure of the proteins and so we do not favour any of the possible motif sequences; that is why this solution is more suited to correctly annotate the exceptions but can be a little unstable for the "normal" situations. The errors are detailed in the next section.

5.2 Structure Experiments

The different structure experiments were done using an even distribution for the probability transitions between the profile models.

- *the observed structure with a final linear part* which does not have all the possible transitions between the cyclic profile models; we allow only: $P \rightarrow L$, $L \rightarrow S$, $S \rightarrow S$ and $S \rightarrow P$, but not $P \rightarrow S$ (see regular expression of PCMP in section 1.2.2). This type of structure used the observations made on the manually annotated sequences, it is more efficient in terms of execution time but it is not able to correctly annotate the exceptions.

  Observation We obtain the same type of structure if we use a not improved maximum likelihood estimation of the transition probabilities on the complete structure;

- *the complete structure with a final linear part* with all the possible transitions between the cyclic profile models; it is slightly less efficient in terms of execution time but it is able to annotate correctly the exception. It can be a little unstable when applied to the "normal" motif sequences;

- *generalised structure (the complete cyclic structure)*; it has no final linear part (all the motifs are considered to be cyclic motifs and we have transitions between any of the 7 motifs). It is less efficient in terms of execution time, less stable (it can easily confound a motif with another, mostly in the case of the S motif which is very close to the P one)), but it is able to recognize any appearing exceptions.

The observed structure is the structure with which we began our study. Because we thought that using the observations that were made over the motif structure of the PPR proteins could improve our results and also because we assumed that the huge number of states could seriously

	same length sequences			different length sequences		
type of structure	identical	≈identical	diff.	identical common part	≈identical common part	diff. common part
observed	91	37	5	30	20	13
complete	97	33	9	33	13	11
generalised	55	17	51	29	12	32

Table 4: Comparative Results for the Structure Experiments

influence the execution time, the logical way to begin our work was to implement a particular, less complicated structure. The next step was to implement a complete structure, more general and with less previous assumptions. The final experiment was to try a completely general structure which obtained less good results but which is most useful in the case of a generalisation of the problem to any cyclic motifs on any type of proteins.

As it was expected, the best results were obtained for the second type of structure, which represents the best compromise between generalisation, efficiency and correctness. The results obtained for the 3 types of structures are summarized in table 4. The errors that appear are similar to the ones obtained for the probability experiments, and are detailed in the next section.

5.3 Types of Errors

There are several types of errors that are likely to appear in our automatic annotation. Next, we tried to list all possible errors and to give an explanation for of them.

- the program finds additional not annotated motifs at the beginning of the protein sequence; the gaps in the manual annotation that have been filled with motifs in the automatic annotation can be introduced in the same category of errors.

  Example: for the protein At4g14820, the manually annotated motif sequence is $L-S-P-L-S-S-P-L-S-P-L2-S2-E-E+-DYW$; we obtained the following additional motif at the beginning of the sequence: P.

  Explanation: alienated motifs at the beginning of the protein (or between 2 motifs) or just that the model gives a better score for this motif sequence. For the additional motifs at the beginning at the sequence, the solution that we tested was to increase the loop-N transition probability; this generates a reduction of the number of additional found motifs.

- in the manually annotated data there are motifs of which we can not be sure of (probably because they are alienated motifs) and so they are marked with an interrogation sign. For this type of motifs, the automatic annotation can give a different solution; it is a problem of interpretation.

  Example: for the protein At4g35130 the manually annotated motif sequence is $S-P-L-S-P-L-S-\mathbf{S}?-L-S-P-L-S-P-L2-S2-E-E+-DYW$; we obtained the following motif sequence: $S-P-L-S-P-L-S-\mathbf{P}-L-S-P-L-S-P-L2-S2-E-E+-DYW$. In this case, the P motif is more likely to be correct.

- exceptions that the implemented structure is not able to recognize: for example repeated apparitions of the $E+$ motif.

Example: for the protein At3g53360 the manually annotated motif sequence is $P-L-S-P-L-S-P-L-S-P-L-S-P-L-S-P-L-S-P-L2-S2-E-E+-\mathbf{E}+$; we obtained the following motif sequence: $P-L-S-P-L-S-P-L-S-P-L-S-P-L-S-P-L2-S2-E-E+$.

Solution: the structure can be easily transformed in order to allow new types of transitions. In our case, the fact that we deal with an annotation error is most probable.

- single motifs are sometimes contradicted by the automatic annotation.

  Example: for the protein At1g20230 the manually annotated motif sequence is $S-P-L-S-P-P-L-S-\mathbf{S}-P-L-S-P-L2-S2-E-E+-DYW$; we obtained the following motif sequence: $S-P-L-S-P-P-L-S-\mathbf{P}-P-L-S-P-L2-S2-E-E+-DYW$.

  Explanation: there could be errors in the manual annotation, or we obtain a better score for that motif (this is influenced by the transition probabilities).

5.4 Summary of the Results for the *Arabidopsis Thaliana* Proteins

Having access to an expert manual annotation for the proteins of *Arabidopsis thaliana*, we were able to verify our results and to decide the best choice of parameters. In the case of this family of proteins, the best solution is to use the complete structure described in section 5.2 with an even distribution for the transition probabilities. In this case we obtain 97 protein sequences perfectly and 33 almost perfectly annotated between the proteins for which we obtained motif sequences with the same length as the manually annotated sequences (from the total number of 196 proteins tested); between the proteins for which we obtained motif sequences of different lengths, we got 33 identical and 13 almost identical on the common part (as explained in section 5.2). 20 proteins had different annotated motif sequences and the errors that appeared have been explained in the previous section.

We obtained the 4 class distribution of the PCMP proteins, described in table 1. An example of a motif sequence annotated protein for each class is given next:

1. **A** >At3g26630: $S-P-L-S-S-P-L-S-P-L2-S2$
2. **E** >At3g16610: $L-S-P-L-S-P-L-S-P-L-S-P-L-S-P-L2-S2-E$
3. **F** >At2g37320: $L-S-P-L-S-P-L2-S2-E-E+$
4. **H** >At4g15720: $S-P-L-S-P-L-S-P-L2-S2-E-E+-DYW$

The experiments done on the non PPR proteins from *Arabidopsis thaliana* gave us what we have expected: the PPR proteins were not identified in the protein sequence.

5.5 Results for the *Rice* Proteins

The *rice* is one of the most important plant species in economy. The rice genome is the second plant genome that has been completed; 505 PPR proteins were extracted from OryGenesDB, [DRL+06] (GeneFarm gives a different number of PPR proteins but this difference appears only because they changed the threshold used when searching the PPR motifs).

For the first time, a systematic annotation was made for the rice PPR proteins, annotation that needs to be verified with a biology expert. The tests that we were able to make verify the correctness of our annotation. We know that from the 505 PPR proteins, approximately 229 are PPRP proteins and from the rest, which are part of the PCMP family, 88 belong to the H class. Our annotation found 232 PPRP proteins and 88 PCMP proteins belonging to the H class.

The experiments made on the *rice* prove the efficiency, in terms of execution time, of our approach: for annotating the approximately 500 protein sequences, our program needed less than 1 minute.

6 Conclusions and Future Work

We designed a Profile HMM Cyclic structure that models the complex repetitions of multiple motif combinations, we developed an efficient implementation and we finally obtained an annotation tool that was used on the proteins from the PPR family, at the *Arabidopsis thaliana* and at the *rice*. We verified our results by comparing them to the manual annotation.

As a result, our work is able to prove the capacity of the Profile HMM Cyclic structure to model the problem in question. We verified the efficiency and the feasibility of the method in terms of execution time, for a large number of sequences and a large number of motifs. The correctness of the results was proved by comparing them to the manually annotated sequences.

After doing several experiments, we have come to create a general structure completely adaptable to new situations, and so we obtained an automatic annotation tool.

We obtained the first annotation for the *rice* proteins, annotation that needs to be verified with a biology expert. Some preliminary tests that we were able to perform strengthened the confidence in the correctness of our work.

One direction of development would be to try to identify the PPR motifs in proteins from other species, which contain alienated versions of these motifs. We can also apply our annotation tool for the retrieval of other repeated motifs in large protein families: leucine rich repeats, kelch motif repeats (see [RBTNL06] for supplementary details).

An interesting paraphrase for our problem would be: searching for a black cat in a dark room, not knowing whether it is there. The question becomes extremely difficult if you don not even know what a cat is. In bioinformatics terms, a logical sequel of our work would be to pass from the "recognition" problem to the extremely more difficult "inference" problem, that consists of identifying the motifs, and afterwards passing to the recovery of the repeated apparitions of these motifs.

The idea would be to completely predict the motif structure of a given protein sequence. This means also predicting the structure of the model (the transitions between the retrieved profile models). We would have to start from unaligned protein sequences and obtain the motifs that appear and the motif structure for each protein sequence. This problem is very difficult even for the prediction of one repeated motif; the existence of several repeated motifs increases the difficulty level. We should probably make some initial suppositions that could lightly simplify the problem, as for example choosing a fixed motif length.

References

[ABKL00] S. Aubourg, N. Boudet, M. Kreis, and A. Lecharny. In *Arabidopsis thaliana*, 1% of the genome codes for a novel protein family unique to plants. *Plant Molecular Biology*, 42(4):603–613, 2000.

[AGM+90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Molecular Biolology*, 3(215), 1990.

[BB98] P. Baldi and S. Brunak. *Statistical Models and Methods for Lifetime Data*. MIT Press, 1998.

[BCD+04] A Bateman, L Coin, R Durbin, RD Finn, V Hollich, S Griffiths-Jones, A Khanna, M Marshall, S Moxon, EL Sonnhammer, DJ Studholme, C Yeats, and SR Eddy. The Pfam protein families database. *Nucleic Acids Res.*, 32, 2004.

[BG00] L. Bréhélin and O. Gascuel. Modèles de Markov cachés et apprentissage de séquences. *Le temps, l'espace et l'évolutif en science du traitement de l'information*, pages 407–421, 2000.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *Biological Sequence Analysis.* Cambridge University Press, 1998.

[DRL+06] G Droc, M Ruiz, P Larmande, A Pereira, P Piffanelli, J B Morel, A Dievart, B Courtois, E Guiderdoni, and C Prin. OryGenesDB: a database for rice reverse genetics. *Nucleic acids research*, (34), 2006.

[Edd] S.R. Eddy. HMMER. *http://hmmer.wustl.edu/.*

[Edd97] S.R. Eddy. Maximum likelihood fitting of extreme value distributions. 1997.

[HH00] A. Heger and Liisa Holm. Rapid automatic detection and alignment of repeats in protein sequences. *Proteins: Structure, Function and Genetics*, (41):224–237, 2000.

[Kro98] A. Krogh. An introduction to hidden Markov models for biological sequences. *Computational Methods in Molecular Biology*, pages 45–63, 1998.

[LAea04] C. Lurin, C. Andres, and S. Aubourg et al. Genome-Wide Analysis of Arabidopsis Pentatricopeptide Repeat Proteins Reveals Their Essential Role in Organelle Biogenesis. *Plant Cell*, 16(8):2089–2103, 2004.

[Law82] J. F. Lawless. *Statistical Models and Methods for Lifetime Data.* John Wiley and Sons, 1982.

[LP85] DJ Lipman and WR. Pearson. Rapid and sensitive protein similarity search. *Science*, 1985.

[Mot92] R. Mott. Maximum likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores. *Math. Biol.*, pages 59–75, 1992.

[Rab89] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 2(77):257–285, 1989.

[RBTNL06] E. Rivals, C. Bruyère, C. Toffano-Nioche, and A. Lecharny. Formation of the Arabidopsis Pentatricopeptide Repeat Family. *Plant Physiology*, 141:1–15, 2006.

[SP00] Ian D. Small and N. Peeters. The PPR motif - a TPR-related motif prevalent in plant organellar proteins. *Trends in Biochemical Sciences*, 25(2):46–47, 2000.

[SW81] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Molecular Biology*, pages 147–197, 1981.