COMENIUS UNIVERSITY, BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

BIOINFORMATICS OF SEQUENCES WITH REPETITIVE MOTIFS

(Master's Thesis)

2012

Bc. Peter Kováč

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Bioinformatics of Sequences with Repetitive Motifs

Master's Thesis

Study Program: Computer Science
Branch of Study: 2508 Computer Science
Department: Department of Computer Science
Supervisor: Mgr. Tomáš Vinař PhD.

Bratislava, 2012

Bc. Peter Kováč

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Peter Kováč
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický

Názov: Bioinformatika sekvencií s repetitívnymi motívmi

Cieľ: Mnohé rodiny proteínov sú zaujímavé tým, že obsahujú viacero navzájom sa podobojúcich kópií toho istého motívu. Takáto štruktúra proteínu značne komplikuje bioinformatické analýzy, ako napríklad zarovnávanie sekvencií. Úlohou je vyvinúť nové modely a algoritmy, ktoré sa s týmto problémom dokážu vysporiadať.

Vedúci: Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KI - Katedra informatiky

Dátum zadania: 19.10.2010

Dátum schválenia: 27.04.2012 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Peter Kováč
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	9.2.1. Computer Science, Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English

Title: Bioinformatics of Sequences with Repetitive Motifs

Aim: A typical feature of several protein families is that their proteins contain multiple occurrences of a specific motif that are very similar to each other. This feature presents a challenge to classical bioinformatics algorithms, such as algorithms for sequence alignment. The goal of this thesis is to design new models and algorithms that will address this problem.

Supervisor: Mgr. Tomáš Vinař, PhD.
Department: FMFI.KI - Department of Computer Science

Assigned: 19.10.2010

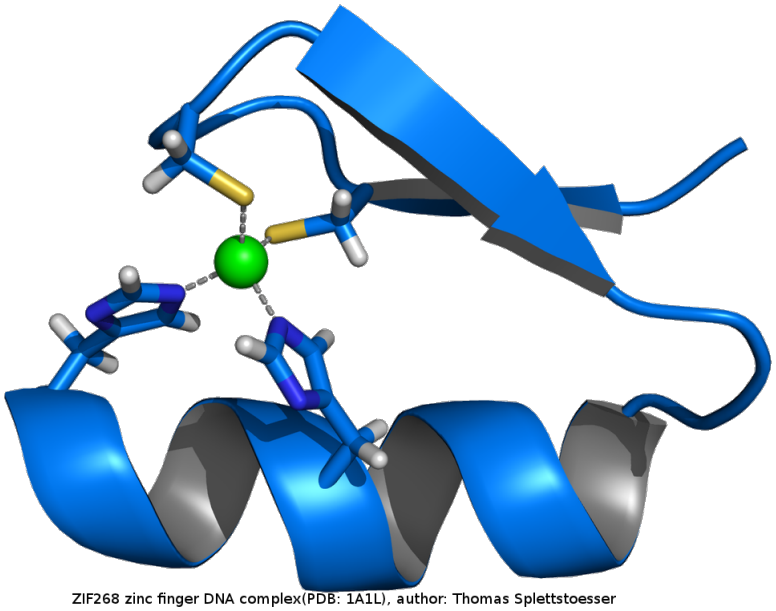
Approved: 27.04.2012 prof. RNDr. Branislav Rovan, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

I wrote this thesis by myself, only with the help of the referenced literature, under the careful supervision of my thesis advisor.

_____



ZIF268 zinc finger DNA complex(PDB: 1A1L), author: Thomas Splettstoesser

I thank my supervisors Tomáš Vinař and Broňa Brejová for their ideas and remarks that helped to shape this work. I thank my family for being supportive and patient.

Abstrakt

Autor:	Bc. Peter Kováč
Názov:	Bioinformatics of Sequences with Repetitive Motifs
Univerzita:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Katedra:	Katedra informatiky
Vedúci:	Mgr. Tomáš Vinař, PhD.
Počet strán:	58
Rok:	2012

Navrhli sme algoritmus na zarovnávanie sekvencií s opakujúcimi sa motívmi. Práca je motivovaná biologickými štúdiami o zinc finger proteínoch, dôležitej rodine regulačných proteínov. Evolučné procesy spôsobili, že tieto sekvencie obsahujú rôzne počty zinc finger motívov (krátych podslov so špecifickými symbolmi na každej pozícii). Náš algoritmus využíva dva typy skrytých markovovských modelov (HMM): párové HMM a profilové HMM. Dynamické programovanie, ktorým počítame zarovnania, je založené na známom Viterbiho algoritme. Model sme vyhodnocovali na reálnych dátach a dosiahli sme lepšie výsledky ako existujúce riešenie.

Kľúčové slová: zarovnanie sekvencií, skrytý markovovský model, dynamické programovanie.

Abstract

Author:	Bc. Peter Kováč
Title:	Bioinformatics of Sequences with Repetitive Motifs
University:	Comenius University in Bratislava
Faculty:	Faculty of Mathematics, Physics and Informatics
Department:	Department of Computer Science
Supervisor:	Mgr. Tomáš Vinař, PhD.
Number of pages:	58
Year:	2012

We present a method for alignment of two sequences containing repetitive motifs. This is motivated by a biological studies of proteins with zinc finger domain, an important group of regulatory proteins. Due to their evolutionary history, sequences of these proteins contain a variable number of different zinc fingers (short subsequences with specific symbols at each position). Our algorithm utilizes two types of hidden Markov models (HMM) for accomplishment of the task: pair HMMs and profile HMMs. The dynamic programming algorithm that computes the motif alignments is based on the well known Viterbi algorithm. We evaluated our model on real world sequences of zinc finger proteins and were able to outperform existing solution.

Keywords: sequence alignment, hidden Markov model, dynamic programming.

Contents

Introduction **1**

1 Alignment of Sequences with Repetitive Motifs **3**
1.1 Sequence Alignment 4
1.1.1 Standard Approach to Alignment 5
1.1.2 Probabilistic Approach to Alignment 11
1.2 Sequence Motifs 16
1.2.1 Zinc-fingers: Structure, Function, Features 16
1.2.2 Sequence Motif Identification 18
1.3 MotifAligner Approach to Motif Alignment 23

2 Profile-Profile-Pair Approach **25**
2.1 Pairwise Alignment of Individual Motifs 26
2.1.1 PPP Dynamic Programming 29
2.1.2 General Case Recurrences 31
2.1.3 Boundary conditions 37
2.1.4 Termination and traceback 40
2.2 Alignment of Whole Motif Arrays 41
2.3 Algorithm Complexity and Implementation Notes 41

3 Experiments **43**
3.1 Dataset Preparation 43
3.2 Estimating Parameters of Our Model 44
3.2.1 Pair HMM Parameters 45
3.2.2 Profile HMM Parameters 47
3.2.3 Parameters of the Needleman-Wunsch Algorithm 47
3.3 The PPP Score Distribution 48
3.4 Alignment Accuracy 50

Conclusion **53**

References **58**

Introduction

> In God we trust; all others must bring data.
>
> W. Edwards Deming

In this thesis we propose a new algorithm for pairwise alignment of sequences containing repetitive motifs. Pairwise sequence alignment is among the most intensively studied problems in computational biology. There are numerous variations of the general alignment problem, each with different biological application, but the main course stays the same: for a symbol in one sequence, find its counterpart in the other sequence.

The specific version of the general problem addressed in this thesis is mainly motivated by proteins with zinc finger domain, a large group of regulatory proteins. Due to their evolutionary history, sequences of these proteins contain variable number of different zinc fingers, short subsequences with specific symbols at each position. Zinc fingers are tandemly repeated at the end of zinc finger proteins, forming the so called zinc finger arrays.

The repeat region is the most interesting feature of zinc finger proteins from computational point of view. There is a great variability in the number of fingers. Some proteins contain just few of them while other may have up to 30 or 50. Some positions in the finger motif are more conserved than other.

Traditional sequence alignment tools generally do not perform well when aligning zinc finger arrays with different number of zinc fingers. For that reason, recent studies of zinc finger evolutionary history excluded zinc finger arrays from comparative analysis, which limits the soundness of their conclusions. Reliable alignment could provide a new insights into processes that have driven the evolution of zinc finger proteins.

A common and popular way of modelling features of biological sequences in computer science terms is via probabilistic models called Hidden Markov models (HMM). HMMs have states and transitions between them, just like determin-

istic finite automata. Additionally, HMMs define probability distributions on transitions and symbol emissions.

Our algorithm utilizes two types of hidden Markov models for accomplishment of the task: pair HMMs and profile HMMs. Pair HMMs assign a probability to alignment of two sequences. Traditional sequence alignment methods can be viewed as a search for highest probability alignment in a pair HMM. Profile HMMs describe properties of sequence motif (such as zinc finger domains), characterizing symbol distributions at individual positions of the motif. In our model, the two HMMs are combined, resulting in a pairwise alignment algorithm with position specific scoring scheme.

The thesis consists of three chapters. In the first chapter, we describe the repetitive sequence alignment problem, sequence motifs, and MotifAligner, the only existing solution to the problem that we are aware of. The second chapter contains an elaborate description of our algorithm. The third chapter shows the practical side of our solution and evaluates its performance on real world data.

Chapter 1

Alignment of Sequences with Repetitive Motifs

This chapter introduces the concepts of a sequence motif, general alignment problem and difficulties that arise in repetitive sequence alignment. At the end of the chapter we present an existing method that served as a role model for some aspects of our own work.

In bioinformatics, a **sequence** is simply a word over some alphabet. The fundamental bioinformatic alphabets are the DNA alphabet, $\Sigma_{\text{DNA}} = \{\text{A}, \text{C}, \text{G}, \text{T}\}$, the RNA alphabet, $\Sigma_{\text{RNA}} = \{\text{A}, \text{C}, \text{G}, \text{U}\}$ (both with symbols called nucleotides, or bases) and the amino acid alphabet (sometimes called the protein alphabet),

$$\Sigma_{\text{AA}} = \{\text{A}, \text{C}, \text{D}, \text{E}, \text{F}, \text{G}, \text{H}, \text{I}, \text{K}, \text{L}, \text{M}, \text{N}, \text{P}, \text{Q}, \text{R}, \text{S}, \text{T}, \text{V}, \text{W}, \text{Y}\}. \tag{1.1}$$

Amino acids form long chains from which the proteins are built. The process of building a protein starts with transcription of genetic information encoded in chromosomes (long DNA molecules in cell nucleus, or simply, words over DNA alphabet). The DNA string is transcribed into a specific type of RNA molecule (a word over RNA alphabet). Later, the RNA molecule is processed, three nucleotides at a time, and each triplet is translated into amino acid chain according to the genetic code. Readers eager for biological details should read a modern molecular biology textbook, for example [Lodish et al., 2007] or [Russell, 2010].

The genetic code assigns one amino acid to each of the 64 possible RNA triplets. From formal language point of view, this complex molecular process is just an application of homomorphisms $h_1 : \Sigma^*_{\text{DNA}} \to \Sigma^*_{\text{RNA}}$ and $h_2 : \Sigma^*_{\text{DNA}} \to \Sigma^*_{\text{AA}}$, which can be composed to $g : \Sigma^*_{\text{DNA}} \to \Sigma^*_{\text{AA}}$.

1.1 Sequence Alignment

Over the time, DNA molecules in cells mix, mutate, and get repaired. Sometimes the alteration causes organism death, sometimes it is advantageous and it is passed onto next generations. Multiple modifications accumulate and can lead to new strains, breeds or, ultimately, species. A lot of effort in molecular biology, genetics, evolutionary biology, and taxonomy is dedicated to trace these changes.

In the world of biological sequences, this means that there are relationships between individual sequences. Biologists can characterize them with their own dictionary, stemming out of the current comprehension of the nature. Mathematicians, statisticians, and computer scientists, on the other hand, need to express these relations in terms they understand. One of the most useful ways of revealing and quantifying these relationships is sequence alignment (chapter 2 in [Durbin et al., 1998]).

Definition 1 (Global pairwise sequence alignment)**.** For a pair of sequences $x = x_1x_2 \dots x_n$, $y = y_1y_2 \dots y_m$ over some alphabet Σ, we define **global pairwise sequence alignment** to be a pair of sequences x' and y' over the alphabet $\Sigma' = \Sigma \cup \{-\}$ iff:

(i) for all $0 \leq i \leq n, 0 \leq j \leq m$ there exists $u_i, v_j \in \{-\}^*$ such that

$$x' = u_0x_1u_1x_2u_2 \dots x_nu_n,\ y' = v_0y_1v_1y_2v_2 \dots y_mv_m, \tag{1.2}$$

(ii) x' and y' have the same length, $|x'| = |y'| = \ell$,

(iii) for all $0 \leq i \leq \ell :\ x'_i \in \Sigma \vee y'_i \in \Sigma$.

Non-empty u_i and v_j are called **gaps**. A **match** in a sequence alignment occurs when two related symbols are aligned (this often means that the symbols are identical). Conversely, a **mismatch** is an aligned pair of unrelated symbols.

In other words, we just insert gap symbols into sequences so that when the resulting sequences are stacked one on to the other, related symbols are in the same column (Fig. 1.1). Notice that (iii) ensures that for every aligned symbol pair, at most one symbol is $-$. We can define **local pairwise alignment** of x and y as a global pairwise sequence alignment of x'' and y'', where x'' is a subsequence (substring) of x and y'' is a subsequence of y. The definitions of **global** and **local multiple sequence alignments** are analogous, the only difference is that it involves more than two sequences (Fig. 1.2).

```
WEAREKNIG---HTSOFTHEROUNDTABLE
WE--DAN--CEWHEN--EVERWEAREABLE
```

Figure 1.1: Example of global pairwise alignment (first two verses of The Monty Python Camelot Song).

```
ZNF626_4799/12    YKC--EECGKAF-NQSSILTTHERIILERN-
ZNF727_4861/2     YKC--EECGKDC--RLSDFTIQKRIHTADRS
ZXDB_644/5        YQCAFSGCKKTF-ITVSALFSHNRAHFREQE
LLNL1236_4814/2   SMC--PECSKTSATDSSCLLMHQRSHTGKRP
ZNF23_141/15      FQC--KECGKAF-HVNAHLIRHQRSHTGEKP
ZNF721_3714/29    YKC--KECGKAF-KSYYSILKHKRTHTRGMS
ZFP2_3745/11      YEC--NECGKAF-SQSAYLIEHQRIHTGEKP
consensus         y C   eCgK f       l  h R ht
```

Figure 1.2: Example of multiple alignment (sequences of human zinc fingers, see Section 1.2.1). Columns with similarity $\geq 50\%$ are highlighted.

In general, sequence alignments define functional, structural, or evolutionary relationships, and the right alignments are not necessarily alignments with the highest number of symbol identities. In order to discriminate between good and bad alignments, we use task specific scoring functions. We can define the general sequence alignment problem as follows.

Definition 2 (General sequence alignment problem)**.** Given input sequences $\vec{x} = x_1, \ldots, x_n$ and a scoring function $f : \Sigma' \times \ldots \times \Sigma' \to \mathbb{R}$, find the optimal alignment $\vec{x}^*$ such that

$$\vec{x}^* = \arg\max_{\vec{x}'} f(\vec{x}'). \tag{1.3}$$

It is clear that enumerating all alignments and keeping only the highest scoring alignment is not the most effective solution to this problem. Therefore, we will limit our interest to scoring functions for which effective algorithms exist.

1.1.1 Standard Approach to Alignment

Scoring functions used in practice are usually based on the substitution matrices. A **substitution matrix** S is a $|\Sigma| \times |\Sigma|$ matrix indexed by symbols of Σ, where for each $a, b \in \Sigma$ the $S[a, b]$ is the score of a being aligned with b. Aligning symbol to a gap yields the gap penalty, usually a negative constant $-g$. The score of an alignment is just the sum of the scores of individual columns in the alignment:

$$f(x', y') = \sum_{i=1}^{l} S'[x'_i, y'_i], \tag{1.4}$$

where $|x'| = |y'| = \ell$ and

$$S'[x'_i, y'_i] = \begin{cases} -g & \text{if } x'_i \text{ or } y'_i \text{ is a gap,} \\ S[x'_i, y'_i] & \text{otherwise.} \end{cases} \tag{1.5}$$

The BLOSUM matrices are among the most popular scoring matrices used today for protein sequence alignment [Henikoff and Henikoff, 1992], [Eddy, 2004]. Each value of $S[a, b]$ in the BLOSUM matrix is normalized and rounded log-odds ratio of the likelihood that a and b are related to the likelihood that they are unrelated. More formally, consider sequences x and y, both of the same length n, aligned without gaps. Assume that they are not related and were generated from some random model R with symbols drawn independently from some background probability distribution q. Then the probability of the two is just the product of the background probabilities:

$$P(x, y \mid R) = \prod_{i=1}^{n} q(x_i) \prod_{j=1}^{n} q(y_j). \tag{1.6}$$

However, if we assume that every aligned pair a, b has been independently derived from some unknown common ancestor c with some probability $p(a, b)$ then the probability of the whole alignment is:

$$P(x, y \mid C) = \prod_{i=1}^{n} p(x_i, y_i). \tag{1.7}$$

Relating these two models we get:

$$\frac{P(x, y \mid C)}{P(x, y \mid R)} = \frac{\prod_{i=1}^{n} p(x_i, y_i)}{\prod_{i=1}^{n} q(x_i) \prod_{j=1}^{n} q(y_j)} = \prod_{i=1}^{n} \frac{p(x_i, y_i)}{q(x_i) q(y_i)}. \tag{1.8}$$

By taking the logarithm of (1.8), we get a sum instead of the multiplication:

$$\log \frac{\prod_{i=1}^{n} p(x_i, y_i)}{\prod_{i=1}^{n} q(x_i) \prod_{j=1}^{n} q(y_j)} = \sum_{i=1}^{n} S[x_i, y_i], \tag{1.9}$$

where S is the resulting substitution matrix (values are usually scaled and

rounded so that the matrix contains only integers),

$$S[a,b] = \frac{1}{\lambda} \log \frac{p(a,b)}{q(a)q(b)}. \tag{1.10}$$

The question remains, what are the right distributions q and p. In case of BLOSUM matrices, these are derived from the BLOCKS database [Henikoff and Henikoff, 1991] of trusted ungapped multiple sequence alignments. Since our expectations about sequence similarity are different when aligning sequences that diverged long time ago, compared to sequences with recent common ancestry, one matrix is not enough. Therefore, BLOSUM matrices come with another parameter, the clustering level ℓ. Two sequences from the BLOCKS database are in the same cluster, if the percentage of identical symbols in their alignment is greater than $\ell\%$. The relative frequencies $f_\ell(a,b)$ of a and b being aligned are counted, with each cluster weighted as a single sequence. Then, the two probability distributions are calculated simply as

$$p_\ell(a,b) = \frac{f_\ell(a,b)}{\sum_a \sum_b f_\ell(a,b)}, \qquad q_\ell(a) = \sum_b p_\ell(a,b). \tag{1.11}$$

The wide variety of BLOSUM matrices uses clustering levels from the range of 30 to 90 percent. Default choice in most sequence alignment programs is BLOSUM62, with clustering level $\ell = 62\%$.

Scoring functions based on substitution matrices, such as f in (1.4), allow an effective dynamic programming algorithm for finding the highest scoring global alignment, called **Needleman-Wunsch** algorithm [Needleman and Wunsch, 1970]. The variant used most frequently in practice – alignment with affine gap scores – uses two parameters for scoring gaps (section 2.4 of [Durbin et al., 1998]), g and e. The score of a gap of length k is $\gamma(k) = -g - (k-1)e$. When $e < g$, the penalty of $\gamma(k)$ is smaller than the linear penalty $-gk$ for equally sized gap from (1.4). As a consequence, this method favours smaller number of longer gaps to higher number of shorter gaps. Biological explanation supporting this is that mutations resulting in gaps are rare and usually affect several positions at a time.

Algorithm 1 (Needleman-Wunsch with affine gap penalties)**.** The dynamic programming algorithm solves the following problem: given sequences $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$, substitution matrix S, and affine gap parameters g and e, find

the global alignment (x', y') maximizing the score defined as

$$s(x', y') = \sum_{i=1}^{|(x',y')|} \begin{cases} S[x'_i, y'_i] & \text{when } x'_i \text{ and } y'_i \text{ are aligned,} \\ -g & \text{when } x'_i = \text{- or } y'_i = \text{- and } x'_{i-1} \text{ and } y'_{i-1} \text{ are aligned,} \\ -e & \text{otherwise.} \end{cases} \tag{1.12}$$

The first sum goes over all maximal continuous ungapped sections (x'_b, y'_b) in the alignment (with $|(x'_b, y'_b)|$ denoting the length of the block) and the second sum goes over maximal continuous gaps in the alignment, where the gap of length k is penalised by $\gamma(k) = -g - (k-1)e$.

The value of an optimal solution can be described in terms of values of partial solutions. Let $s_{i,j}$ be the optimal solution for some prefixes of the input sequences $x_1 \dots x_i$, $y_1 \dots y_j$. Then, assuming that x_{i+1} and y_{j+1} are aligned, the value of the optimal solution for prefixes $x_1 \dots x_{i+1}$, $y_1 \dots y_{j+1}$ is $s_{i,j} + S[x_i, x_j]$. However, if we assume x_{i+1} to be aligned to a gap then the value of the optimal solution for prefixes $x_1 \dots x_{i+1}$, $y_1 \dots y_j$ is either $s_{i,j} + g$ (if x_i is aligned to y_j and a new gap begins) or $s_{i,j} + e$ (if x_i is aligned to gap too and the gap simply continues, note that we don't change the index of y prefix). Analogously, if we assume y_{i+1} to be aligned to a gap, the value of the optimal solution for prefixes $x_1 \dots x_i, y_1 \dots y_{j+1}$ is either $s_{i,j} + g$ (if x_i is aligned to y_j) or $s_{i,j} + e$ (if y_j is aligned to gap).

We can derive the exact pseudocode for the algorithm from the explanation in the previous paragraph. The algorithm fills $3 \times (n+1) \times (m+1)$ matrix NW. For each i, j, $1 \leq i \leq n$, $1 \leq j \leq m$, the values of cells NW$[k, i, j]$ have the following meaning:

1. NW$[1, i, j]$ stores the value of the optimal solution for the prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ given that x_i is aligned with y_j;
2. NW$[2, i, j]$ stores the value of the optimal solution for the prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ given that x_i is aligned to a gap;
3. NW$[3, i, j]$ stores the value of the optimal solution for the prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ given that y_j is aligned to a gap.

Let us skip the base cases (we will define them soon) and assume the general case for prefixes $x_1 \dots x_i$, $y_1 \dots y_j$, where $1 \leq i \leq n$, $1 \leq j \leq m$. Assuming that all cells NW$[k, i', j']$, where $k \in \{1, 2, 3\}$, $i' < i$,$j' < j$, are filled already and by the

explanation above we get that the values of $\mathrm{NW}[\_, i, j]$ are:

$$\mathrm{NW}[1,i,j] = \max \begin{cases} \mathrm{NW}[1,i-1,j-1] + S[x_i, y_j], \\ \mathrm{NW}[2,i-1,j-1] + S[x_i, y_j], \\ \mathrm{NW}[3,i-1,j-1] + S[x_i, y_j]; \end{cases} \tag{1.13}$$

$$\mathrm{NW}[2,i,j] = \max \begin{cases} \mathrm{NW}[1,i-1,j] - g, \\ \mathrm{NW}[2,i-1,j] - e; \end{cases} \tag{1.14}$$

$$\mathrm{NW}[3,i,j] = \max \begin{cases} \mathrm{NW}[1,i,j-1] - g, \\ \mathrm{NW}[3,i,j-1] - e. \end{cases} \tag{1.15}$$

Now consider the base cases. To make the reasoning easier, we can implicitly think of a special symbol \$ at the beginning of both input sequences, which is by definition always aligned only to itself with zero score. Then the initial scores in corners of the submatrices are:

$$\mathrm{NW}[1,0,0] = 0, \tag{1.16}$$

$$\mathrm{NW}[2,0,0] = \mathrm{NW}[3,0,0] = -\infty. \tag{1.17}$$

Equation 1.16 represents the implicit alignment of \$s, with zero score. The other corner cells do not have meaningful interpretations and are set to $-\infty$ to make sure, we will avoid using them in the recurrence. Next, we have to set the rest of the first row and the first column in each submatrix $\mathrm{NW}[k, \_, \_]$ so that the gaps are also possible at the beginning of the alignment:

$$\mathrm{NW}[2,i,0] = -g - (i-1)e, \qquad 1 \le i \le n \tag{1.18}$$

$$\mathrm{NW}[1,i,0] = \mathrm{NW}[3,i,0] = -\infty, \qquad 1 \le i \le n \tag{1.19}$$

$$\mathrm{NW}[3,0,j] = -g - (j-1)e, \qquad 1 \le j \le m \tag{1.20}$$

$$\mathrm{NW}[1,0,j] = \mathrm{NW}[2,0,j] = -\infty. \qquad 1 \le j \le m \tag{1.21}$$

Consider the value of $\mathrm{NW}[2,1,0]$ according to (1.18). The interpretation is that x_1 is aligned to a gap, and since we implicitly assume aligned \$s in the previous column, the score equals to the gap opening penalty $-g$. In general, the value of $\mathrm{NW}[2,i,0]$ is just the value of the whole prefix $x_1 \dots x_i$ being aligned to a gap of length i and since there is just one way how to do it, it is also the value of the optimal solution. On the other hand, it is not allowed to align the implicit \$ in y with neither x_i nor gap, so all $\mathrm{NW}[1,i,0]$ and $\mathrm{NW}[3,i,0]$ have to be $-\infty$ and (1.19)

is valid. Analogous arguments hold for the y sequence and equations (1.20) and (1.21).

Now, we can verify that after the initialisation and applying the recurrences in a nested loop `for` $i := 1$ `to` n; `for` $j := 1$ `to` m; the value of an optimal solution lies in one of the "bottom right" cells and its value is:

$$s(x', y') = \max\{\mathbf{NW}[1, n, m], \mathbf{NW}[2, n, m], \mathbf{NW}[3, n, m]\}. \tag{1.22}$$

It remains to show, how we can reconstruct the optimal solution x', y'. During the recurrent of some cell NW$[k, i, j]$, we store a pointer to the cell from which the value of NW$[k, i, j]$ was derived (in the initialisation we consider NW$[2, i, 0]$ and NW$[3, 0, j]$ to be derived from NW$[2, i-1, 0]$ and NW$[3, 0, j-1]$, respectively, for $i, j > 2$; and both NW$[2, 1, 0]$ and NW$[3, 0, 1]$ to be derived from NW$[1, 0, 0]$). Then, at the end, we pick up the pointer from one of the bottom right cells in (1.22) and follow the pointers back to the initial cell NW$[1, 0, 0]$. The resulting alignment is built back-to-front during this traceback procedure, starting from the empty alignment. If the cell NW$[k, i, j]$ points to a cell NW$[k', i-1, j-1]$, we add aligned pair (x_i, y_j) to the front of the currently obtained alignment; if it points to a cell NW$[k', i-1, j]$, we prefix the current partial alignment with the aligned pair $(x_i, -)$; and finally, if it points to a cell NW$[k', i, j-1]$ we prefix the alignment with $(-, y_j)$.

The time and space complexity of this algorithm is $O(nm)$ because the calculations in (1.16)-(1.22) take only a constant time and space, and there is just $O(nm)$ of cells to compute. □

A local alignment counterpart algorithm to the Needleman-Wunsch algorithm is the Smith-Waterman algorithm [Smith and Waterman, 1981], [Gotoh, 1982]. The main difference lies in the initialisation step, where the cells at the borders of the matrix (equations (1.18) and (1.20)) are initialised to zero (so the alignment might start anywhere in the input sequences); and in the traceback, where the lookup for cell with the maximal value searches the whole matrix and the traceback continues until reaches the top or the left edge.

One of the most widely used tools for local pairwise sequence alignment is the BLAST heuristics [Altschul et al., 1990], [Altschul et al., 1997]. Its popularity stems mostly from its speed: direct implementations of Needleman-Wunsch, Smith-Waterman or any other exact algorithm cannot be used to build alignments on the scale of whole genomes (the human genome is approximately 3 billion bases long).

1.1.2 Probabilistic Approach to Alignment

The sequence alignment problem can be solved in the probabilistic fashion, too. In this section, we will introduce hidden Markov models [Rabiner, 1989], a versatile tools in probabilistic modelling. We will show, that the algorithms for sequence alignment can be viewed as a special kind of hidden Markov models.

Definition 3 (Hidden Markov model (HMM))**.** Finite non-empty set of states K, alphabet Σ and functions $\pi : K \to [0,1]$, $t : K \times K \to [0,1]$, $e : K \times \Sigma \to [0,1]$ are together called **hidden Markov model** iff following conditions hold:

(i) π is a probability distribution over K, $\sum_{k \in K} \pi(k) = 1$,

(ii) t is a conditional probability function $P(\ell \mid k)$, $\forall k \in K : \sum_{\ell \in K} t(k, \ell) = 1$,

(iii) e is a conditional probability function $P(a \mid k)$, $\forall k \in K : \sum_{a \in \Sigma} e(k, a) = 1$.

The function t is called the **transition** function and e is called the **emission** function. Using the graph theory language, states are sometimes referred to as *nodes* and nonzero transitions as *edges*.

Hidden Markov models are generative models. We can illustrate the generative process on the random printer example. Imagine a printer with unlimited supply of paper and ink and one button. The printer has some hidden set of states and is capable of printing symbols from certain alphabet. When you plug the printer into the electricity network, it sets itself to some state k chosen according to the probability distribution π. Every time you press the button, the printer prints a symbol a, with probability $e(k, a)$ and then changes its internal state to some ℓ, with probability $t(k, \ell)$ (k denotes the current state). Pressing the button multiple times prints whole word. As an external viewer, you don't know the exact sequence of states, which is *hidden*, but you can *observe* the sequence of printed symbols. An important feature of hidden Markov models is the *Markov property* that emissions and transitions both depend only on the current state.

A joint probability of emitting a sequence $x = x_1 \dots x_n$ along the state path $s = s_1 \dots s_n$ in a given HMM is

$$P(x, s) = \pi(s_1) e(s_1, x_1) \prod_{i=2}^{n} t(s_{i-1}, s_i) e(s_i, x_i). \tag{1.23}$$

By mathematical induction it can be shown, that for any positive integer n hidden Markov models define a probabilistic distribution over the set of all sequences and state paths of length n:

$$\sum_{x \in \Sigma^n, s \in K^n} P(x, s) = 1. \tag{1.24}$$

Hidden Markov models are especially handy for performing probabilistic inference. One well known inference algorithm is the Viterbi algorithm which for given input sequence x computes the state path s^* which maximizes the joint probability $P(x, s^*)$:

$$s^* = \arg\max_s P(x, s) \tag{1.25}$$

Algorithm 2 (Viterbi algorithm for general HMM)**.** Viterbi algorithm is a dynamic programming algorithm. Let $\mathrm{V}[\ell, i]$ be the probability of the most probable state path of length i for some prefix $x_1 \dots x_i$ ending in state ℓ. By definition, $\mathrm{V}[\ell, i]$ is a product of the emission of x_i in state ℓ and a transition $t(k, \ell)$ for some state k and a joint probability of some state path ending in ℓ and the prefix $x_1 \dots x_{i-1}$. Since $\mathrm{V}[\ell, i]$ is the probability of *the most probable* state path for $x_1 \dots x_i$, we maximize the product over all $k \in K$ to get the value of $\mathrm{V}[\ell, i]$. Furthermore, the path for prefix $x_1 \dots x_{i-1}$ ending in ℓ has to be the most probable path for $x_1 \dots x_{i-1}$ ending in ℓ. Thus, we can state the dynamic programming relation for $\mathrm{V}[\ell, i]$:

$$\mathrm{V}[\ell, i] = e(\ell, x_i) \max_\ell \{t(k, \ell)\mathrm{V}[k, i-1]\}. \tag{1.26}$$

After filling the whole $|K| \times n$ matrix V columnwise, the probability of the most probable state path for sequence x is the maximal value in the last column, $\max_k \{\mathrm{V}[k, n]\}$. □

We skip the detailed description of the general Viterbi algorithm (which can be found in the Chapter 3 of [Durbin et al., 1998]) and focus on the variant on specific type of HMM useful for sequence alignment, the pair HMM (Chapter 4 of [Durbin et al., 1998], Figure 1.3). Pair HMMs emit pairs of symbols – essentially a sequence alignment – and can be used to solve the sequence alignment problem (Definition 2).

Definition 4 (Pair Hidden Markov Model for Global Alignment)**.** The alphabet used by pair HMM is $\Sigma = \{\binom{a}{b}, \binom{a}{-}, \binom{-}{b} \mid a, b \in \Sigma'\}$ where Σ' is the alphabet of sequences we want to align. The set of states of a pair HMM is

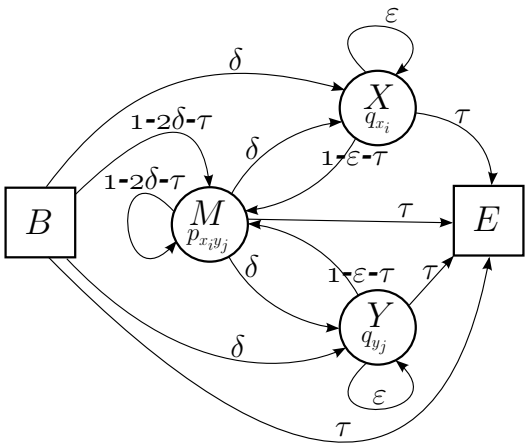


Figure 1.3: A pair HMM for global alignment. Squares indicate silent states. Note that the sum of parameters on the outgoing edges for each state (except E) is one.

$K = \{B, M, X, Y, E\}$. B and E have special meaning – B is the initial (begin) state, i.e. $\pi(B) = 1$ and E is the final (end) state, i.e. $\forall k \in K : \ t(B, k) = 0$. By definition, every complete state path in the pair HMM has to end in the E state. Both B and E are *silent*, which means they do not emit any symbols : $\forall a \in \Sigma : \ e(B, a) = e(E, a) = 0$. State M (match) represents a gapless column: $\forall \binom{a}{\_}, \binom{\_}{b} \in \Sigma : \ e(M, \binom{a}{\_}) = e(M, \binom{\_}{b}) = 0, \forall a, b \in \Sigma' : \ e(M, \binom{a}{b}) = p(a, b)$. Both X and Y, (insert in y, insert in x, respectively) represent a column containing a gap symbol in corresponding input sequence: $\forall \binom{a}{\_}, \binom{\_}{a} \in \Sigma : \ e(X, \binom{a}{\_}) = e(Y, \binom{\_}{a}) = q(a), \forall a, b \in \Sigma' : \ e(M, \binom{a}{b}) = e(X, \binom{\_}{b}) = e(Y, \binom{a}{\_}) = 0$. Transitions are defined in terms of parameters $\delta, \varepsilon, \tau$. δ is the probability of opening a gap, so $t(B, X) = t(B, Y) = t(M, X) = t(M, Y) = \delta$; ε is the probability of extending a gap: $t(X, X) = t(Y, Y) = \varepsilon$ and τ is the probability of ending the alignment, i.e. the transitions to the E state: for each $k \in K \setminus \{E\}$, $t(k, E) = \tau$. The rest of transition probabilities is defined with regard to the condition (ii) from the Definition 3, namely $t(X, M) = t(Y, M) = 1 - \varepsilon - \tau$, $t(B, M) = t(M, M) = 1 - 2\delta - \tau$. The full model is shown in Figure 1.3. Note that transitions and emissions for the X and Y states are the same: we want to model sequence alignments regardless of the sequence input order.

We can solve the sequence alignment problem with the Viterbi algorithm on pair HMMs. The score of an alignment according to a pair HMM is naturally the joint probability of the input sequences and the state path which defines this alignment (there is a 1-to-1 correspondence between alignments and state paths in pair HMM – for each column of the alignment, the absence or presence and the position of a gap symbol unambiguously identifies the state which emitted

the column). Since there are two input sequences in the alignment problem, the Viterbi matrix V must have an extra dimension. Apart from that, the recurrence relations for pair HMM matrix are basically the same like the equation (1.26).

Algorithm 3 (Viterbi algorithm in pair HMM)**.** Once again we solve the problem of the highest scoring alignment, this time in terms of probabilistic model. Given sequences $x = x_1 \dots x_n$, $y = y_1 \dots y_n$ and fully parametrized pair HMM, we want to find the most probable state path in the pair model, which emits x and y:

$$s^* = \arg\max_s P_{\text{pair}}(x, y, s). \tag{1.27}$$

Let $\mathrm{V}[\ell, i, j]$ be the probability of the most probable state path for prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ ending in state ℓ. Just like in the general case (Algorithm 2), it can be written as the product of the emission probability in ℓ and the transition probability from some predecessor k to ℓ and the probability of certain path ending in ℓ. From the alignment point of view, there are again three configurations:

1. $\mathrm{V}[M, i, j]$ is the probability of the most probable state path for the prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ given that x_i is aligned with y_j;
2. $\mathrm{V}[X, i, j]$ is the probability of the most probable state path for the prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ given that x_i is aligned to a gap;
3. $\mathrm{V}[Y, i, j]$ is the probability of the most probable state path for the prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ given that y_j is aligned to a gap.

By definition, we have to start in the B state. However, the transition probabilities are the same for B and M, so we can make an exception and replace the B state by the M state, which simplifies other relations:

$$\mathrm{V}[M, 0, 0] = 1, \tag{1.28}$$

$$\mathrm{V}[X, 0, 0] = \mathrm{V}[Y, 0, 0] = \mathrm{V}[E, 0, 0] = 0. \tag{1.29}$$

To allow gaps at the beginning, relations analogous to (1.18)-(1.21) are used:

$$\mathbf{V}[X,i,0] = \delta\varepsilon^{i-1}\prod_{i'=1}^{i} q(x'_i), \qquad \forall i, 1 \le i \le n, \tag{1.30}$$

$$\mathbf{V}[M,i,0] = \mathbf{V}[Y,i,0](x_i) = \mathbf{V}[E,i,0] = 0, \qquad \forall i, 1 \le i \le n, \tag{1.31}$$

$$\mathbf{V}[Y,0,j] = \delta\varepsilon^{j-1}\prod_{j'=1}^{j} q(y'_j), \qquad \forall j, 1 \le j \le m, \tag{1.32}$$

$$\mathbf{V}[M,0,j] = \mathbf{V}[X,0,j] = \mathbf{V}[E,0,j], \qquad \forall j, 1 \le j \le m. \tag{1.33}$$

Combining ideas from algorithms 1 and 2, we get the following recurrence relations for all $1 \le i \le n$, $1 \le j \le m$:

$$\mathbf{V}[M,i,j] = p(x_i,y_j)\max\begin{cases}(1-2\delta-\tau)\cdot\mathbf{V}[M,i-1,j-1],\\(1-\varepsilon-\tau)\cdot\mathbf{V}[X,i-1,j-1],\\(1-\varepsilon-\tau)\cdot\mathbf{V}[Y,i-1,j-1].\end{cases} \tag{1.34}$$

$$\mathbf{V}[X,i,j] = q(x_i)\max\begin{cases}\delta\cdot\mathbf{V}[M,i-1,j],\\\varepsilon\cdot\mathbf{V}[X,i-1,j].\end{cases} \tag{1.35}$$

$$\mathbf{V}[Y,i,j] = q(y_j)\max\begin{cases}\delta\cdot\mathbf{V}[M,i,j-1],\\\varepsilon\cdot\mathbf{V}[Y,i,j-1].\end{cases} \tag{1.36}$$

We can fill the entire matrix V in a nested loop `for` $i := 1$ `to` n; `for` $j := 1$ `to` m;. By definition, every complete pair HMM state path ends in E, this transition finalizes the computation step of the dynamic programming:

$$\max_s P_{\text{pair}}(x,y,s) = \tau\max\begin{cases}\mathbf{V}[M,n,m],\\\mathbf{V}[X,n,m],\\\mathbf{V}[Y,n,m].\end{cases} \tag{1.37}$$

To reconstruct the optimal alignment, we keep pointers and perform the same traceback procedure like in the Needleman-Wunsch algorithm (Algorithm 1). Traceback starts in the maximal cell in (1.37). Both time and space complexities of this algorithm are $O(nm)$ again. □

Similarity of this algorithm and the Needleman-Wunsch algorithm is evident. It is possible to derive Algorithm 3 in terms of log-odds scores, with the recurrent equations of Algorithm 1 and Algorithm 3 being identical (Chapter 4

of [Durbin et al., 1998]).

Some advantages of probabilistic approach over the standard approach are the ability to answer various quantitative questions about sequences and the ability of plugging the probabilistic model into another probabilistic model directly – we use of this in our model.

1.2 Sequence Motifs

We have already described the problem of sequence alignment, so now is the time to describe sequence motifs. In general, motifs are used to identify certain fragments of biological sequences with specific features stemming from the sequences themselves. `TATAAA` is a an example of short motif called TATA box which serves as a guide for complex molecular machinery and is usually found 25-35 nucleotides ahead of transcription start site of many genes (section 7.3 in [Lodish et al., 2007]).

At first, we will give a brief overview of zinc finger protein domain, which has an easily recognizable motif which usually occurs repeated many times in a row and served as a main motivation for this work. Then we will introduce profile hidden Markov models, a variant of HMMs suitable for motif modelling.

1.2.1 Zinc-fingers: Structure, Function, Features

Zinc finger domain is a relatively small stretch of amino acids contained in certain type of proteins called transcription factors. The term *zinc finger* (coined by [Miller et al., 1985]) is derived from the folding scheme of the domain (Figure 1.4). The purpose of a zinc finger domain is to bind DNA at specific places. We will focus our attention to the Krüppel-type C_2H_2 zinc finger subfamily (abbreviated KRAB-ZNF), which is the largest family of transcriptional regulators in mammals [Urrutia, 2003].

Sequences of C_2H_2 zinc fingers are usually 28 amino acids long. The two *C*s (cysteins) and two *H*s (histidines) which bind to the zinc-ion are the most conserved amino acids in the domain. On the other hand, there are four highly variable positions where virtually any amino acid can occur (Fig. 1.4, see Fig. 1.2 for an example of zinc finger alignment). Different amino acids at these positions cause that the whole domain binds to different DNA regions.

Genes encoding KRAB-ZNF proteins have very interesting domain structure. Usually, there is a region encoding one or more Krüppel-associated box domains

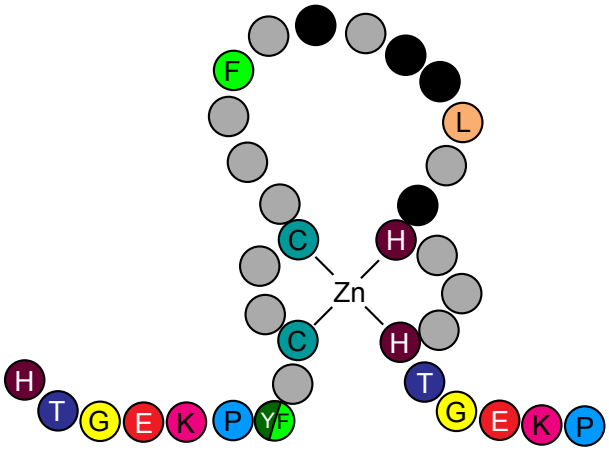


Figure 1.4: The structure of a zinc-finger. Highly variable sites are marked with black color. The most conserved amino acids are the four involved in binding the zinc ion [Schmidt and Durrett, 2004].

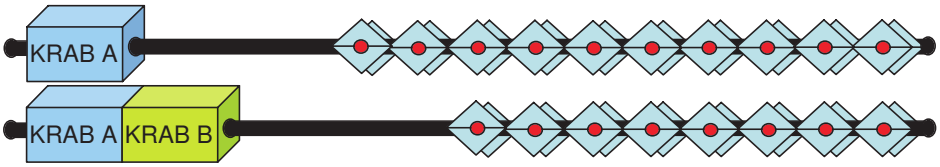


Figure 1.5: Domain organisation of typical KRAB-ZNF genes. Functional domains are encoded in separate exons. The protein contains one or more KRAB domains and an array of 3 to cca. 40 zinc fingers. [Urrutia, 2003]

(KRAB, [Bellefroid et al., 1991]) followed by a zinc finger region. The zinc finger region encodes varying number of zinc finger domains. Sometimes two or more zinc fingers are tandemly repeated, i.e. an exact copy occurs immediately after the end of some zinc finger sequence (Fig. 1.5).

This structure is a result of a dynamic evolutionary history, full of gene duplications [Hamilton et al., 2006], segmental duplications [Nowick et al., 2010] and many mutations which help to gain new functions for duplicated copies. It appears that the more developed the organism is, the more zinc finger proteins it has and furthermore, those proteins have more finger domains [Looman et al., 2002]. However, the exact details are not yet known, mostly because of limited availability of high quality genome assemblies for a high variety of organisms and scientists have to be careful when drawing conclusions about zinc finger evolution [Thomas and Emerson, 2009]. A lot of effort is dedicated to building and maintaining comprehensive catalogues of (mostly human) KZNF genes [Huntley et al., 2006], [Nowick et al., 2011], [Ding et al., 2009].

The repetitive nature of zinc finger protein sequences complicates bioinformatic analyses, such as sequence alignments and computational phylogeny. Con-

sequently, many studies limit their analyses and infer conclusions based only on the more conserved parts of zinc finger genes, for example the KRAB domains or sequences before the zinc finger region (e.g. [Schmidt and Durrett, 2004], [Hamilton et al., 2006]), or dispute the relevance of results of standard methods applied to genes with high variance in the number of fingers [Thomas and Emerson, 2009].

1.2.2 Sequence Motif Identification

The simplest definition of a motif is that it is a certain finite sequence of symbols (with a specific symbol defined at each position). This is too restrictive and not very useful, since a handful of similar sequences can share common properties. Alternatively, we can define motif to be a set of sequences and call each member of this set an motif instance. It might be even better to use regular expressions or finite state machines, instead of simple sets, for denser representation.

We might say that motif occurs in the input sequence if it is a contiguous subsequence of the input sequence. But biological sequences are subject to evolution and change from time to time. So, we might allow some errors, number of which might depend on the motif's length. The subsequence might not be necessarily contiguous or we may omit some symbols from the motif. An important issue is assessing expected number of occurrences of the motif in sequence databases. Another common task is searching for top n best matches in a sequence database. A scoring function can be defined in order to rank the motif occurrences.

Previous paragraphs point out that there is relatively high complexity even in supposedly simple terms like the definition of motif and motif occurrence. However, many aspects of sequence motifs can be successfully handled using profile hidden Markov models (Chapter 5 in [Durbin et al., 1998]). Profile HMMs allow mismatches, insertions and missing symbols to be present in the putative motif sequences even with different probabilities at different positions.

Definition 5 (Profile Hidden Markov Model)**.** Let L be a positive integer. The set of states of a **profile hidden Markov model** is $K = \{M_i, I_i, D_i \mid i \in [1, L]\} \cup \{B, I_0, E\}$ and we say that the profile HMM has length L, or has L *consensus columns* (Fig. 1.6). The begin stae B and end state E are sometimes referred to as M_0, M_{L+1}, respectively. We call the M_i states **match** states, the I_i states **insert** states and D_i states **delete** states.

The only non-zero transition probabilities in the model are the following:

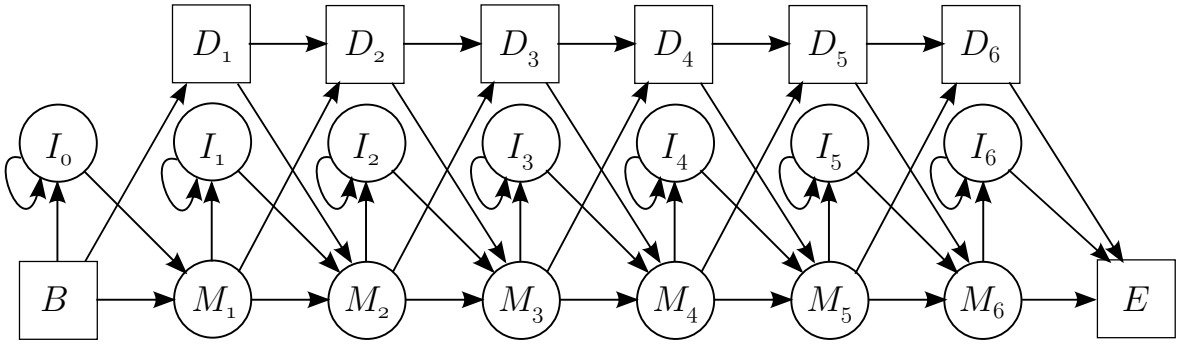


Figure 1.6: Example of a profile HMM. Squares indicate silent states.

```
                          1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
ZNF626_4799/12            Y K C - - E E C G K A F - N Q S S I L T T H E R I I L E R N -
ZNF727_4861/2             Y K C - - E E C G K D C - - R L S D F T I Q K R I H T A D R S
ZXDB_644/5                Y Q C A F S G C K K T F - I T V S A L F S H N R A H F R E Q E
LLNL1236_4814/2           S M C - - P E C S K T S A T D S S C L L M H Q R S H T G K R P
ZNF23_141/15              F Q C - - K E C G K A F - H V N A H L I R H Q R S H T G E K P
ZNF721_3714/29            Y K C - - K E C G K A F - K S Y Y S I L K H K R T H T R G M S
ZFP2_3745/11              Y E C - - N E C G K A F - S Q S A Y L I E H Q R I H T G E K P
consensus                 y C   eCgK f       l   h R ht
```

Figure 1.7: Multiple sequence from Figure 1.2

$t(M_i, M_{i+1})$, $t(M_i, I_i)$, $t(M_i, D_{i+1})$, $t(I_i, I_i)$, $t(I_i, M_{i+1})$, for $i \in [0, L]$ and $t(D_i, D_{i+1})$, $t(D_i, M_{i+1})$, for $i \in [1, L]$. All delete states as well as the begin and end states are silent states and does not emit any symbols. Other states have their own specific emission distributions, that may be different, particularly at different columns.

How can be profile HMMs useful? Consider the multiple sequence alignment of seven zinc finger sequences from Figure 1.2 shown in Figure 1.7 again for convenience. We will build a profile HMM that models this alignment. The total length of the alignment is 31 columns. There are two contrasting column kinds: columns 4, 5 and 13 contain a lot of gaps, while positions 1-3, 6-12 and 13-30 are gapless. In addition, columns 14 and 31 contain just one gap, and we will treat them similarly like the gapless positions. Let's focus to the contiguous almost gapless blocks 1-3, 6-12, 14-31. The total number of columns in the blocks is 28, thus the length of the profile model will be 28. We can easily count the relative frequencies of amino acids in columns of these blocks, for example, in the column 2, the relative frequencies are $f^2_{\mathrm{K}} = 3/7$, $f^2_{\mathrm{Q}} = 2/7$, $f^2_{\mathrm{M}} = f^2_{\mathrm{E}} = 1/7$ and relative frequencies of other amino acids are zero. Those will correspond to the emission probabilities in the respective match states. The transition probabilities for match states will be $t(M_i, M_{i+1}) = 1$, $t(M_i, I_i) = t(M_i, D_{i+1}) = 0$, if not stated differently.

The two gap regions at columns 4-5 and 13 will be modelled by the insert

states. The first gap occurs after the column modelled by the M_3 state (3) and before the column modelled by the M_4 state (6), so it will correspond to the I_3 state. The transition probabilities from the match state will be $t(M_3, M_4) = 6/7$ and $t(M_3, I_3) = 1/7$, following the ratio of gaps vs. non-gaps in the columns 4-5. The probability of k repeated transitions from I_i to I_i itself follows the geometric distribution with parameter $t(I_i, I_i)$. Thus we will set $t(I_i, I_i)$ so that the mean of this distribution is the length of the gap region – 2 in this case ($t(I_i, M_{i+1}) = 1 - t(I_i, I_i)$ to keep the model well defined). The emission probabilities in insert states can be set to some background distributions, or we might use the relative frequency approach again.

Finally, columns with small amount of gaps (14 and 31) will be modelled by the delete states. The lone gap in the column 14 can be viewed as a transition from the match state M_{10} to delete state D_{11}, therefore, $t(M_{10}, D_{11}) = 1/7$, $t(M_{10}, I_{10}) = 1/7$, $t(M_{10}, M_{11}) = 5/7$. Although the second gap in the `ZNF727-_4861/2` sequence has length two, it is modelled as a two gaps of length one by two states I_{10} and D_{11}, because of the different number of gap symbols in columns 13 and 14. And because the delete part of the gap has length 1 only, $t(D_{11}, M_{12}) = 1$, $t(D_{11}, D_{12}) = 0$. Analogous transitions are set to model the gap in the last column.

In general, given a multiple sequence alignment, the process outlined above can be described in the following steps:

1. determine the length of the profile and choose columns to be represented by the match states (consensus columns);
2. this assignment uniquely identifies the state path which has to be taken, for each sequence in the alignment;
3. given the state paths, count how many times each transition is used and set transition probabilities accordingly;
4. set the emission probabilities according to the relative frequencies of symbols in their respective columns.

In fact this is the maximum likelihood estimation, i.e. given data D, it is the assignment of parameters θ that maximizes $P(D \mid \theta)$. As with all ML estimators, this is likely to overfit the data. To overcome this problem, various strategies are used in practice, e.g. adding pseudocounts to each emission and transition probability.

Suppose that we are given a set M of sequences that are known to have some unknown common traits and the task is to find other sequences with similar traits in a large database. We have seen that profile HMMs are a probabilistic representation of multiple sequence alignments. So, if we can acquire a trusted multiple sequence alignment of sequences $m \in M$ (e.g. manually curated by experts), we can build a profile HMM. This model encodes basically the same information as the alignment, but in a more compact form (the size of $|S|$ can be in hundreds in practice). Thanks to the probabilistic foundations of HMMs, we can use this profile model to score other sequences. The score is probability of the given sequence x in the profile model, $P_{\text{profile}}(x)$. This probability can be computed with the Forward algorithm (Chapter 3 in [Durbin et al., 1998]), which is basically the same algorithm as the Viterbi algorithm (Algorithm 2), where instead of computing the maximum of certain set of values, we sum up all the values together. So, we can score all sequences in the database and declare some cutoff score that discriminates the similar sequences.

Apart from finding similar sequences, we can use profile HMM to perform a kind of sequence annotation. In general, sequence annotation is a labelling of particular sequence regions with some specific information. When speaking of profiles, we want to know which positions in the given sequence match which columns in the profile. This is equivalent to finding a state path along which the sequence is emitted. In particular, we want to find the most probable state path,

$$s^* = \arg\max_s P_{\text{profile}}(x, s), \tag{1.38}$$

which can be found by Viterbi algorithm.

Algorithm 4 (Viterbi Algorithm for Profile HMM)**.** For given sequence $x = x_1 \dots x_n$, let $V[M, j, i]$ be the probability of the most probable state path for prefix $x_1 \dots x_i$ ending in state M_j, analogously $V[I, j, i]$ be the probability of the most probable state path for prefix $x_1 \dots x_i$ ending in state I_j and $V[D, j, i]$ be the probability of the most probable state path for prefix $x_1 \dots x_i$ ending in state D_j.

Assuming the begin state to be M_0, we get that

$$V[M, 0, 0] = 1, \tag{1.39}$$

$$V[s, i, 0] = 0, \text{ for all other states } s_i. \tag{1.40}$$

By the similar argument for the structure of the optimal solution like in the pair HMM Viterbi variant (Algorithm 3) and the profile HMM topology, for each $i, 1 \leq i \leq n$, $j, 1 \leq j \leq L$, $k, 2 \leq k \leq L$, the probabilities satisfy following

recurrences:

$$V[M,j,i] = e(M_j, x_i) \max \begin{cases} t(M_{j-1}, M_j)V[M, j-1, i-1], \\ t(I_{j-1}, M_j)V[I, j-1, i-1], \\ t(D_{j-1}, M_j)V[D, j-1, i-1], \end{cases} \tag{1.41}$$

$$V[I,j,i] = e(I_j, x_i) \max \begin{cases} t(M_j, I_j)V[M, j, i-1], \\ t(I_j, I_j)V[I, j, i-1], \end{cases} \tag{1.42}$$

$$V[D,k,i] = \max \begin{cases} t(M_{k-1}, D_j)V[M, k-1, i], \\ t(D_{k-1}, D_j)V[D, k-1, i]. \end{cases} \tag{1.43}$$

Moreover, for each $i, 1 \leq i \leq n$,

$$V[I,0,i] = e(I_0, x_i) \max \begin{cases} t(M_0, I_0)V[M, 0, i-1], \\ t(I_0, I_0)V[I, 0, i-1], \end{cases} \tag{1.44}$$

$$V[D,1,i] = t(M_0, D_1)V[M, 0, i]. \tag{1.45}$$

Since all complete paths has to end in the end state M_{L+1}, it follows that the value of the optimal solution is $V[M, L+1, n]$, which in turn is

$$V[M,L+1,n] = \max \begin{cases} t(M_L, M_{L+1})V[M, L, n], \\ t(I_L, M_{L+1})V[I, L, n], \\ t(D_L, M_{L+1})V[D, L, n], \end{cases} \tag{1.46}$$

The actual computation fills the dynamic programming $3 \times (L+1) \times (n+1)$ matrix first by starting with 1.39 and 1.40, then computes the boudary cases 1.44 and 1.45 in a loop for each $i \leftarrow 1 \ldots n$. Then again loops through all $i, 1 \leq i \leq n$ and through all $j, 1 \leq j \leq L$ and computes 1.41-1.43. Finally the value of the optimal solution is found using 1.46 and the standard traceback procedure constructs the optimal solution itself. The algorithm runs in $O(Ln)$ time and memory. □

The use of profile HMMs in modelling sequence motifs is one of the oldest applications of hidden Markov models in computational biology [Eddy, 1996]. Today, mature tools utilizing these models exists, for example the well known HMMER package [Eddy, 2009]. With HMMER, it is possible to search protein sequences against collections of profiles (to determine what motifs occur in the given sequence) or to search for motif occurrences in a protein sequence database. Other tools allow direct alignment of two profiles that may be dif-

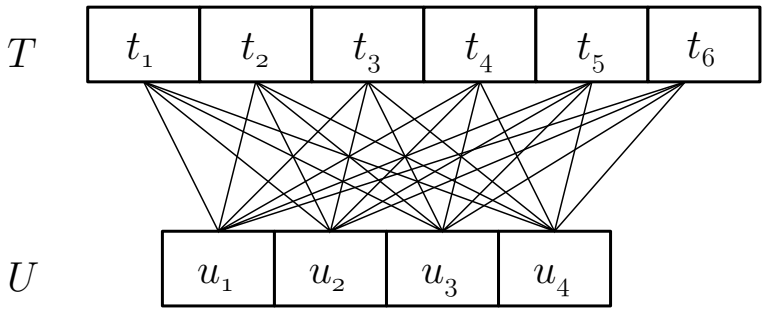


Figure 1.8: MotifAligner approach to motif sequence alignment

ferent (e.g. [Edgar and Sjolander, 2004], [Soding, 2005]), which is useful when comparing motifs instead of plain sequences. Pfam [Punta et al., 2012], a protein family database, uses profile HMMs to categorize protein sequences and to identify common features shared across the members of a family.

1.3 MotifAligner Approach to Motif Alignment

To obtain a high quality alignments even on sequences with highly variable number of zinc finger motifs, [Nowick et al., 2011] developed a pairwise alignment tool called MotifAligner. To our knowledge, it is the only sequence alignment method designed specifically to align sequences with variable number of repetitive motifs. Part of our own work was inspired by this algorithm.

MotifAligner solves the global pairwise sequence alignment problem (Definition 2). Let $x = x_1 \dots x_n$, $y = y_1 \dots y_m$ be the input sequences over an alphabet Σ. MotifAligner uses a profile HMM tool HMMER [Eddy, 2011] and finds all canonical motif occurrences with E-value less than certain threshold c in both input sequences (a canonical motif occurrence for a given profile HMM is a sequence which has the same length as the number of columns of the profile HMM). Let $T = (t_1, \dots, t_a)$ and $U = (u_1, \dots, u_b)$ be the sequences of all motif occurrences found by HMMER in the original input sequences x and y, respectively. Using some substitution matrix S on Σ, MotifAligner computes scores of all gapless pairwise alignments of motifs t_k, u_ℓ (Fig. 1.8), for all $1 \leq k \leq a$, $1 \leq \ell \leq b$:

$$s[t_k, u_\ell] = \sum_{i=1}^{L} S[t_{k_i}, u_{\ell_i}]. \tag{1.47}$$

Then it applies Needleman-Wunsch algorithm (Algorithm 1) to T and U, treating motifs as sequence symbols and using matrix s as the substitution matrix. Let $\tau_0, \dots \tau_a$, $\mu_0, \dots \mu_b$, be sequences from Σ^* (possibly empty) such that

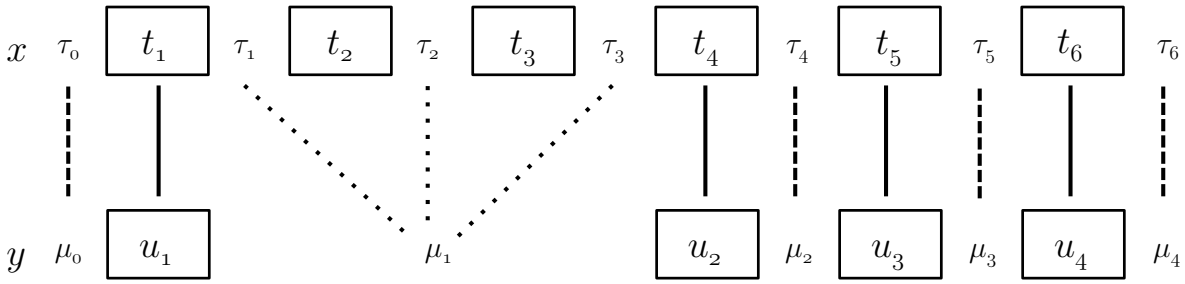


Figure 1.9: An example situation after the alignment of motif sequences T and U from Figure 1.8 is computed. Solid lines indicate that alignment. Dashed lines indicate regular global alignment between the subregions. Dotted lines indicate, that all global alignments of subregions are computed and the one with the maximal score is chosen to the final alignment.

$x = \tau_0 t_1 \tau_1 \dots t_a \tau_a$, $y = \mu_0 u_1 \mu_1 \dots u_b \mu_b$. The result of the Needleman-Wunsch computation is a global alignment of the motifs occurrences, $t_1, \dots, t_a$, $u_1, \dots, u_b$. The only parts of x and y not yet aligned are τ_is and μ_js. The paper [Nowick et al., 2011] does not specify, how to deal with these regions.

Quite reasonable finalisation is to go through the alignment of T and U and whenever there are pairs t_i, t_{i+1} and u_j, u_{j+1} such that t_i is aligned to u_j and t_{i+1} is aligned to u_{j+1}, compute a global alignment of τ_i and μ_j using the scoring matrix S. Align the corner cases τ_0 to μ_0 and τ_a to μ_b as well. The only unaligned regions remaining involve gaps in the T and U alignment. Consider the situation on the figure 1.9. The solid and dashed lines indicate parts that are aligned already. To align the rest, one can compute all three global alignments for $(\tau_1, \mu_1), (\tau_2, \mu_1), (\tau_3, \mu_1)$ and choose the one with the maximal score. Finally, the alignment of the input sequences x and y can be built by concatenating the partial alignments.

Chapter 2

Profile-Profile-Pair Approach

This chapter describes our approach to alignment of sequences with repetitive motifs. We call our method *Profile-Profile-Pair*, because it uses a combination of two copies of a profile HMM and one pair HMM.

The main drawback of the MotifAligner approach is that it does not take advantage of available positional information. For example the zinc-finger motif (Fig. 1.4). There are several high conserved positions, i.e. the four amino-acids binding the zinc ion. On the other hand, the four amino-acids involved in DNA binding are highly variable. Thus the four conserved positions should be used to anchor the sequence alignment while the highly variable positions – when correctly aligned – should indicate whether the two zinc-fingers under comparison are distant or related.

We would like to have a method that takes this positional information into account when scoring the alignments. Since we can identify all motif occurrences in the input sequences using profile HMMs, our prior beliefs about sequence similarity differ from the usual uninformed setting of the sequence alignment. We expect highly conserved positions to be identical in both input sequences, while it is likely that the symbols at highly variable positions are different. Of course, most positions would fall somewhere between the two extremes, with some symbols occurring with high probabilities while others not occurring at all (see e.g. fist position of the zinc finger motif on Fig. 1.4). To sum up, it seems that using position specific scores, instead of uniform scoring function for each column, can lead to better results.

Profile HMMs are designed to handle positional information probabilistically. It is natural to include them in such positional-specific scoring scheme. However, profile HMMs are restricted to processing single sequence at a time, so they are not directly applicable to pairwise sequence alignment.

Here we will describe a hybrid method. The overall scheme is similar to MotifAligner: in order to obtain an alignment of sequences with repetitive motif occurrences, we compute all pairwise individual motif alignments. Then we use a global alignment algorithm operating on motifs to get the resulting alignment. The main difference lies in the individual motif alignment step, where we use a novel approach that includes the positional information extracted from motif sequences.

2.1 Pairwise Alignment of Individual Motifs

The input consists of two sequences $x = x_1 \dots x_{L_x}$ and $y = y_1 \dots y_{L_y}$, instances of the motif, a profile HMM encoding the same motif and a pair HMM. We process both input sequences with the profile HMM (each sequence has its own copy of the profile HMM), but not independently. Rather, we use a pair HMM as a glue. In particular, the pair HMM will guide the state paths and ensure that the final state paths in all three models have meaningful interpretations.

The individual motif alignment is computed via dynamic programming similar to the widely used Viterbi algorithm (Algorithm 2). The algorithm computes the most probable state paths through all three HMMs simultaneously under following two constraints:

Constraint 1 (Profile match states constraint)**.** *If both profile models are in the same match state M_k for some k then the pair model has to be in the match state M.*

Constraint 2 (Pair match state constraint)**.** *If the pair model is in the match state M, then the profile models are in the same match state M_k or in the same insert state I_k for some k.*

In other words, if the pair model is in the state X or Y (which is interpreted as a gap in one of the sequences), the two profile models should not be in the same match state: symbols belonging to the same consensus column should be aligned. However, if both profile models are in the same insert state they can either be evolutionary related, in which case they should be aligned using M state of the pair model, or they could have been inserted in the sequence independently which would correspond to using X and Y states of the pair model. Constraint 2 also implies, that if the profile models are neither in the same match state M_k nor in the same insert state I_k (i.e. either are in completely different columns or in the same column k, but different states M_k and I_k), which means

that the symbols being emitted are not related, then the pair model should not be in the match state.

The ultimate purpose of these constraints is to reduce the space of valid three-state combinations of HMMs involved and keep only combinations that have meaningful interpretation. One might notice that we did not take the delete states of profile HMMs into the consideration. The reason is that there is a preprocessing step before the dynamic programming initialization which removes the delete states from profile HMMs (details are shown in the next subsection).

Our algorithm solves the problem of finding the highest scoring state paths which satisfy Constraints 1 and 2 in the model triple,

$$(s_p^*, s_x^*, s_y^*) = \arg \max_{\substack{\text{valid} \\ s_p, s_x, s_y}} \{s(x, y, s_p, s_x, s_y)\}. \tag{2.1}$$

The score is defined as the product of joint probabilities of the state paths and input sequences (Eq. (1.23)) in the corresponding models,

$$s(x, y, s_p, s_x, s_y) = P_{\text{pair}}(x, y, s_p) P_{\text{profile}}(x, s_x) P_{\text{profile}}(y, s_y). \tag{2.2}$$

We obtain an optimal solution using dynamic programming. Let $s = (s_P, s_X, s_Y)$ be the triple of states of pair and x-profile and y-profile models, respectively. Let the $V[s_P, s_X, s_Y, i, j]$ be the score of the highest scoring state path triple for prefixes $x_1 \dots x_i$, $y_1 \dots y_j$ with the state path of the pair model ending in s_P and state paths of the profile models ending in the s_X and s_Y states respectively, given that s is valid according to Constraints 1 and 2; otherwise, let $V[s_P, s_X, s_Y, i, j]$ be zero.

Assume, that s_P, s_X, s_Y are valid. The value of $V[s_P, s_X, s_Y, i, j]$ is simply a product of joint probabilities of certain state paths S_P, S_X, S_Y ending in s_P, s_X, s_Y respectively and sequence prefixes $x_1 \dots x_i$, $y_1 \dots y_j$:

$$V[s_P, s_X, s_Y, i, j] = P_{\text{pair}}(S_P, x_1 \dots x_i, y_1 \dots y_j) P_{\text{profile}}(S_X, x_1 \dots x_i) P_{\text{profile}}(S_Y, y_1 \dots y_{)}). \tag{2.3}$$

If x_i, y_j are aligned then they had to be emitted in s_P, s_X, s_Y and there exists a state path triple S'_P, S'_X, S'_Y ending in states s'_P, s'_X, s'_Y respectively such that

$S_P = S'_P s_P, S_X = S'_X s_X, S_Y = S'_Y s_Y$ and $V[s_P, s_X, s_Y, i, j]$ can be written as

$$\begin{aligned} V[s_P, s_X, s_Y, i, j] = & \, p(x_i, y_j) e(s_X, x_i) e(s_Y, y_j) t(s'_P, s_P) t(s'_X, s_X) t(s'_Y, s_Y) \cdot \\ & \cdot P_{\text{pair}}(S'_P, x_1 \dots x_{i-1}, y_1 \dots y_{j-1}) P_{\text{profile}}(S'_X, x_1 \dots x_{i-1}) \cdot \\ & \cdot P_{\text{profile}}(S'_Y, y_1 \dots y_{j-1}). \end{aligned} \tag{2.4}$$

Since $V[s_P, s_X, s_Y, i, j]$ is the score of *the highest scoring* state path ending in s_P, s_X, s_Y for $x_1 \dots x_i, y_1 \dots y_j$, we maximize the product on the right hand side of (2.4) over all predecessor state triples and over all paths ending in these predecessors for prefixes $x_1 \dots x_{i-1}, y_1 \dots y_{j-1}$. It turns out, that by definition of V, the score of the highest scoring path ending in s'_P, s'_X, s'_Y for prefixes $x_1 \dots x_{i-1}$, $y_1 \dots y_{j-1}$ is $V[s'_P, s'_X, s'_Y, i-1, j-1]$. Therefore, the value of $V[s_P, s_X, s_Y, i, j]$ can be stated recurrently as

$$\begin{aligned} V[s_P, s_X, s_Y, i, j] = & \, p(x_i, y_j) e(s_X, x_i) e(s_Y, y_j) \cdot \\ & \cdot \max_{(s'_P, s'_X, s'_Y) \in P} \left\{ t(s'_P, s_P) t(s'_X, s_X) t(s'_Y, s_Y) V[s'_P, s'_X, s'_Y, i-1, j-1] \right\}, \end{aligned} \tag{2.5}$$

where P is the set of all state triples that are valid predecessors of s_P, s_X, s_Y given the Constraints 1 and 2 and model topologies. Analogously, if x_i is aligned to a gap, then $V[s_P, s_X, s_Y, i, j]$ is

$$\begin{aligned} V[s_P, s_X, s_Y, i, j] = & \, q(x_i) e(s_X, x_i) \cdot \\ & \cdot \max_{(s'_P, s'_X, s_Y) \in P} \left\{ t(s'_P, s_P) t(s'_X, s_X) V[s'_P, s'_X, s_Y, i-1, j] \right\}, \end{aligned} \tag{2.6}$$

and if y_j is aligned to a gap, then $V[s_P, s_X, s_Y, i, j]$ is

$$\begin{aligned} V[s_P, s_X, s_Y, i, j] = & \, q(y_j) e(s_Y, y_j) \cdot \\ & \cdot \max_{(s'_P, s_X, s'_Y) \in P} \left\{ t(s'_P, s_P) t(s'_Y, s_Y) V[s'_P, s_X, s'_Y, i, j-1] \right\}. \end{aligned} \tag{2.7}$$

Note that equations (2.5)-(2.7) have essentially the same structure as the Viterbi algorithm for pair HMM (Algorithm 3).

Each complete state path in the pair and profile HMMs has to end in the corresponding end state. Thus, in order to find the value of an optimal solution s^*, we go through the set of all state triples s_P, s_X, s_Y and look for the triple which maximizes the product of $V[s_P, s_X, s_Y, L_x, L_y]$ and transition probabilities

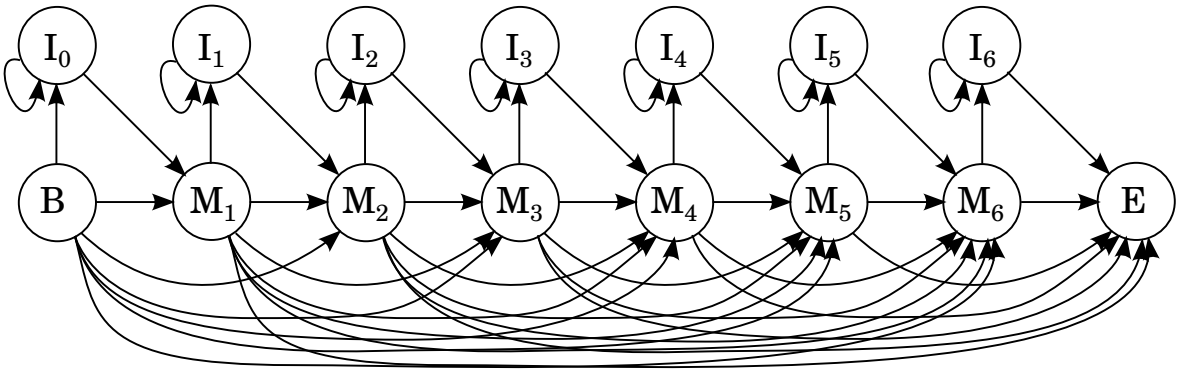


Figure 2.1: Example of a profile HMM after the preprocessing step.

from the triple to the end states in all three models:

$$s(x, y, s_p^*, s_x^*, s_y^*) = \max_{s_P, s_X, s_Y} \{t(s_P, E_{\text{pair}})t(s_Y, E_{\text{profile}})t(s_X, E_{\text{profile}})V[s_P, s_X, s_Y, L_x, L_y]\}. \tag{2.8}$$

We provide a detailed explanation of PPP method in the next sections.

2.1.1 PPP Dynamic Programming

Firstly, we preprocess the profile HMM by removing silent delete states, replacing them with corresponding transition probabilities (Figure 2.1). For every state path $\pi = M_i D_{i+1} D_{i+2} \dots D_{j-1} M_j$, we define the transition from M_i to M_j to be

$$t_{M_i M_j} = t_{M_i D_{i+1}} t_{D_{j-1} M_j} \prod_{k=i+1}^{j-2} t_{D_k D_{k+1}}. \tag{2.9}$$

Such preprocessing does not change the probability distribution defined by the original profile hidden Markov model. The reasons for this procedure will be clearer later when we will explain the general recurrent step of the dynamic programming. In short, when arriving to the state triple (M, M_k, M_k), we want to distinguish between the case when the predecessor profile states were the same match states M_ℓ, M_ℓ and according to the Constraint 1 the symbols they emitted were aligned; and the case when the predecessor profile states were different match states M_ℓ, M_n and according to the Constraint 2 the emitted symbols were not aligned.

The dynamic programming fills $3 \times 2(L+1) \times 2(L+1) \times (L_x+1) \times (L_y+1)$ matrix V, where L is the length of the profile HMM. For every $s_P \in \{M, X, Y\}$, $s_X, s_Y \in \{M_k, I_k \mid 0 \leq k \leq L\}$, $i, 1 \leq i \leq L_x$, $j, 1 \leq j \leq L_y$ the $V[s_P, s_X, s_Y, i, j]$ stores the score of the most probable state path triple for prefixes $x_1 \dots x_i$, $y_1 \dots y_j$

ending in s_P, s_X, s_Y.

In (2.5)-(2.7), we have already shown general structure of the dynamic programming recurrence. Now we have to specify for each state triple the exact form of the set of valid predecessor state triples P. Certain state triples are not allowed according to Constraints 1 and 2. Let us list all valid state combinations. There are just three different pair states (we omit the B and E states for the moment) and, after the profile HMM preprocessing step, only two types of profile states. This yields $3 \times 2L \times 2L$ different combinations in total. We can generalize some of them and describe all valid generalized combinations of states using a shorter list.

An important observation comes from the profile topology (Fig. 2.1). In every profile state path the sequence of column numbers of states is non-decreasing. Furthermore, two consecutive states in the path are allowed to be from the same column only if the first of them is match state M_k and the other is I_k or both are the same I_k.

In general, there are 4 combinations of match/insert states at two profile positions (MM, MI, IM, II) and each of these 4 combinations can be further divided into two cases according to the relative numbers of profile columns. Only two of them are allowed to occur along the pair match state M:

1. (M, M_k, M_k), the same match states in both profile HMMs (the symbols in the two sequences are aligned to each other and they both represent the k-th column of the profile)

2. (M, I_k, I_k), the same insert states in both profile HMMs (the symbols in the two sequences are aligned to each other and they both represent the k-th column of the profile)

If the pair state is X (emitting only the sequence x and leaving a gap in y), the number of possible valid generalized combinations of profile states is higher:

3. (X, M_k, M_ℓ), two different match states, where $\ell < k$, interpreted as a gapped region with sequence x being ahead of y in the profile and both sequences matching certain columns of profile consensus,

4. (X, M_k, M_ℓ), two different match states, where $\ell > k$, interpreted as a gapped region with sequence y being ahead of x in the profile and both sequences matching certain columns of profile consensus,

5. (X, M_k, I_{k-1}), a match state from some profile column k for x and an insert state from the preceding column for y, interpreted as a gapped alignment in which only the x does match the profile consensus,

6. (X, M_k, I_ℓ), a match state from some profile column k for x and an insert state from some column $\ell, \ell \neq k-1$ for y, interpreted as a gapped alignment in which only the x does match the profile consensus,

7. (X, I_k, M_k), an insert state from some profile column k for x and a match state from the same column for y, interpreted as a gapped alignment column in which only the y does match the profile consensus,

8. (X, I_k, M_ℓ), an insert state from some profile column k, $k \neq \ell$ for x and a match state from different column for y, interpreted as a gapped alignment column in which only the y does match the profile consensus,

9. (X, I_k, I_k), an insert state for x and an insert state for y, both from the same profile column, interpreted as a gapped alignment column with independently inserted sequences,

10. (X, I_k, I_ℓ), an insert state for x and an insert state for y, coming from the different profile columns, $k \neq \ell$, interpreted as a gapped alignment with independently inserted sequences.

There are another 8 symmetrical combinations when the pair state is Y, resulting in 18 valid generalised combinations of states in total.

2.1.2 General Case Recurrences

Now, we will describe the dynamic programming recurrence relations for all generalised combinations mentioned in Section 2.1.1. For space saving reasons, we will use symbols $t_{k\ell}$ instead of $t(k, \ell)$ for transition probabilities. For all i, j, k, ℓ, $1 \leq i \leq L_x$, $1 \leq j \leq L_y$, $1 \leq k, \ell \leq L$ following equations hold (all are derived from the general cases (2.5)-(2.7)).

For the pair state M, there are two possible combinations of profile states:

(M, M_k, M_k) and (M, I_k, I_k). The derivation of the first one follows.

$$V[M, M_k, M_k, i, j] = e(x_i, y_j) e_{M_k}(x_i) e_{M_k}(y_j) \cdot$$
$$\cdot \max \begin{cases} t_{MM} \cdot t_{M_\ell M_k} \cdot t_{M_\ell M_k} \cdot V[M, M_\ell, M_\ell, i-1, j-1] & 0 \le \ell < k \\ t_{MM} \cdot t_{I_{k-1} M_k} \cdot t_{I_{k-1} M_k} \cdot V[M, I_{k-1}, I_{k-1}, i-1, j-1] & \\ t_{XM} \cdot t_{M_\ell M_k} \cdot t_{M_n M_k} \cdot V[X, M_\ell, M_n, i-1, j-1] & 0 \le \ell, n < k, n \ne \ell \\ t_{XM} \cdot t_{M_\ell M_k} \cdot t_{I_{k-1} M_k} \cdot V[X, M_\ell, I_{k-1}, i-1, j-1] & 0 \le \ell < k \\ t_{XM} \cdot t_{I_{k-1} M_k} \cdot t_{M_\ell M_k} \cdot V[X, I_{k-1}, M_\ell, i-1, j-1] & 0 \le \ell < k \\ t_{XM} \cdot t_{I_{k-1} M_k} \cdot t_{I_{k-1} M_k} \cdot V[X, I_{k-1}, I_{k-1}, i-1, j-1] & \\ t_{YM} \cdot t_{M_\ell M_k} \cdot t_{M_n M_k} \cdot V[Y, M_\ell, M_n, i-1, j-1] & 0 \le \ell, n < k, n \ne \ell \\ t_{YM} \cdot t_{M_\ell M_k} \cdot t_{I_{k-1} M_k} \cdot V[Y, M_\ell, I_{k-1}, i-1, j-1] & 0 \le \ell < k \\ t_{YM} \cdot t_{I_{k-1} M_k} \cdot t_{M_\ell M_k} \cdot V[Y, I_{k-1}, M_\ell, i-1, j-1] & 0 \le \ell < k \\ t_{YM} \cdot t_{I_{k-1} M_k} \cdot t_{I_{k-1} M_k} \cdot V[Y, I_{k-1}, I_{k-1}, i-1, j-1] & \end{cases} \tag{2.10}$$

Let us take a deeper look at this equation. The value at $V[M, M_k, M_k, i, j]$ has to include the full emission of x_i and y_j in all three models. Then, there is a wide choice of previous cells from which the current cell value could be computed. Both sequence indices are one less in each option and every value considered in the maximum function is composed of the value of the predecessor cell and the product of transition probabilities in all three models for their corresponding source and target states.

One of the options is the cell $V[M, M_\ell, M_\ell, i-1, j-1]$ for some ℓ, $0 \le \ell \le k$, which is an instance of the same equation (2.10).The other option with pair match state M is $V[M, I_{k-1}, I_{k-1}, i-1, j-1]$, an instance of (2.11). Both types of cells represent extension of the matched region in the final alignment, with the only difference related to the profile consensus. The rest of the options with pair states X or Y represent the end of gapped region and beginning of matched region. Valid predecessors include two different profile match states and all combinations, where at least one of the profile states is the insert state I_{k-1}.

Derivation for $V[M, I_k, I_k, i, j]$ is simpler:

$$V[M, I_k, I_k, i, j] = e(x_i, y_j)e_{I_k}(x_i)e_{I_k}(y_j)\cdot \cdot \max \begin{cases} t_{MM}\cdot t_{M_kI_k}\cdot t_{M_kI_k}\cdot V[M,M_k,M_k,i-1,j-1] \\ t_{MM}\cdot t_{I_kI_k}\ \cdot t_{I_kI_k}\ \cdot V[M,I_k,\ I_k,\ i-1,j-1] \\ t_{XM}\cdot t_{M_kI_k}\cdot t_{I_kI_k}\ \cdot V[X,M_k,I_k,\ i-1,j-1] \\ t_{XM}\cdot t_{I_kI_k}\ \cdot t_{M_kI_k}\cdot V[X,I_k,\ M_k,i-1,j-1] \\ t_{XM}\cdot t_{I_kI_k}\ \cdot t_{I_kI_k}\ \cdot V[X,I_k,\ I_k,\ i-1,j-1] \\ t_{YM}\cdot t_{M_kI_k}\cdot t_{I_kI_k}\ \cdot V[Y,\ M_k,I_k,\ i-1,j-1] \\ t_{YM}\cdot t_{I_kI_k}\ \cdot t_{M_kI_k}\cdot V[Y,\ I_k,\ M_k,i-1,j-1] \\ t_{YM}\cdot t_{I_kI_k}\ \cdot t_{I_kI_k}\ \cdot V[Y,\ I_k,\ I_k,\ i-1,j-1] \end{cases} \tag{2.11}$$

Again, there are terms for the full emission of x_i and y_j in all three models. This time, the set of predecessor cells is smaller and covers all valid combinations of pair and profile model states, where both profile states are either I_k insert state or the match state from the same profile column M_k.

When the pair state is X, we have more generalised options. The emission terms are the same in all of them – the emission of gapped symbol $\binom{x_i}{-}$ in the pair model and the emission of x_i in the profile model. Note that we do not emit y_j. For that reason, we neither change the index j nor the state of the profile model for y and hence there are just two transition terms for the pair model and for the x-profile model.

$$V[X, M_k, M_\ell, i, j] = e(x_i, -)e_{M_k}(x_i)\cdot \cdot \max \begin{cases} t_{MX}\cdot t_{M_\ell M_k}\ \cdot V[M,M_\ell,\ M_\ell,i-1,j] & \ell < k \\ t_{XX}\cdot t_{M_nM_k}\ \cdot V[X,M_n,\ M_\ell,i-1,j] & n \neq \ell, n < k \\ t_{XX}\cdot t_{I_{k-1}M_k}\cdot V[X,I_{k-1},M_\ell,i-1,j] & \\ t_{YX}\cdot t_{M_nM_k}\ \cdot V[Y,\ M_n,\ M_\ell,i-1,j] & n \neq \ell, n < k \\ t_{YX}\cdot t_{I_{k-1}M_k}\cdot V[Y,\ I_{k-1},M_\ell,i-1,j] & \end{cases} \quad (\ell < k) \tag{2.12}$$

$$V[X, M_k, M_\ell, i, j] = e(x_i, -)e_{M_k}(x_i)\cdot \cdot \max \begin{cases} t_{XX}\cdot t_{M_nM_k}\ \cdot V[X,M_n,\ M_\ell,i-1,j] & n < k < \ell \\ t_{XX}\cdot t_{I_{k-1}M_k}\cdot V[X,I_{k-1},M_\ell,i-1,j] & \\ t_{YX}\cdot t_{M_nM_k}\ \cdot V[Y,\ M_n,\ M_\ell,i-1,j] & n < k < \ell \\ t_{YX}\cdot t_{I_{k-1}M_k}\cdot V[Y,\ I_{k-1},M_\ell,i-1,j] & \end{cases} \quad (k < \ell) \tag{2.13}$$

With both profile states being match states, M_k for x and M_ℓ for y, we distinguish between cases $\ell < k$ and $\ell > k$. In the former case (eq. (2.12)), there is a possibility of joint transition from $V[M, M_\ell, M_\ell, i-1, j]$ to $V[X, M_k, M_\ell, i, j]$ (i.e. transition from M_ℓ to M_k in the x-profile model), which is not possible in the latter case (eq. (2.13)) because there are no back transitions from states with higher column numbers to states with lower column numbers in the profile model. Other values considered in the maximum function in both equations include transitions from I_{k-1} to M_k and transitions from some M_n to M_k (for $n \neq k$, $n \neq \ell$) in the x-profile model.

$$V[X, M_k, I_{k-1}, i, j] = e(x_i, -)e_{M_k}(x_i)\cdot \\ \cdot \max \begin{cases} t_{MX}\cdot t_{I_{k-1}M_k}\cdot V[M,I_{k-1},I_{k-1},i-1,j] & \\ t_{XX}\cdot t_{M_nM_k}\cdot V[X, M_n, I_{k-1},i-1,j] & n<k \\ t_{XX}\cdot t_{I_{k-1}M_k}\cdot V[X, I_{k-1},I_{k-1},i-1,j] & \\ t_{YX}\cdot t_{M_nM_k}\cdot V[Y, M_n, I_{k-1},i-1,j] & n<k \\ t_{YX}\cdot t_{I_{k-1}M_k}\cdot V[Y, I_{k-1},I_{k-1},i-1,j] & \end{cases} \tag{2.14}$$

$$V[X, M_k, I_\ell, i, j] = e(x_i, -)e_{M_k}(x_i)\cdot \\ \cdot \max \begin{cases} t_{XX}\cdot t_{M_nM_k}\cdot V[X,M_n, I_\ell,i-1,j] & n<k \\ t_{XX}\cdot t_{I_{k-1}M_k}\cdot V[X,I_{k-1},I_\ell,i-1,j] & \\ t_{YX}\cdot t_{M_nM_k}\cdot V[Y, M_n, I_\ell,i-1,j] & n<k \\ t_{YX}\cdot t_{I_{k-1}M_k}\cdot V[Y, I_{k-1},I_\ell,i-1,j] & \end{cases} \quad (\ell \neq k-1) \tag{2.15}$$

There are five generalised possibilities for incoming cells in (2.14), four of them are basically the same in (2.15) and cover all valid combinations of states, where the pair state is X, the y-profile state is I_ℓ and the x-profile state is M_n for some $n < k$ or I_{k-1}. The one extra case in (2.14) occurs when $\ell = k-1$, since it is allowed to be in the pair match state and the same insert states in the profile models at the same time.

$$V[X, I_k, M_k, i, j] = e(x_i, -)e_{I_k}(x_i)\cdot \\ \cdot \max \begin{cases} t_{XX}\cdot t_{M_kI_k}\cdot V[M,M_k,M_k,i-1,j] \\ t_{XX}\cdot t_{I_kI_k}\cdot V[X, I_k, M_k,i-1,j] \\ t_{YX}\cdot t_{I_kI_k}\cdot V[Y, I_k, M_k,i-1,j] \end{cases} \tag{2.16}$$

$$V[X, I_k, M_\ell, i, j] = e(x_i, -)e_{I_k}(x_i)\cdot \\ \cdot \max \begin{cases} t_{XX}\cdot t_{M_k I_k}\cdot V[X,M_k,M_\ell,i-1,j] \\ t_{XX}\cdot t_{I_k I_k}\ \cdot V[X,I_k,\ M_\ell,i-1,j] \\ t_{YX}\cdot t_{M_k I_k}\cdot V[Y,\ M_k,M_\ell,i-1,j] \\ t_{YX}\cdot t_{I_k I_k}\ \cdot V[Y,\ I_k,\ M_\ell,i-1,j] \end{cases} \quad (k \neq \ell) \tag{2.17}$$

Because there are just two incoming edges to any insert state in the profile HMM, configurations with I_k as the x-profile state and a match state M_ℓ as the y-profile state yield simpler equations. Equation (2.16) takes care of the case $\ell = k$, equation (2.16) goes with the negation $\ell \neq k$.

$$V[X, I_k, I_k, i, j] = e(x_i, -)e_{I_k}(x_i)\cdot \\ \cdot \max \begin{cases} t_{MX}\cdot t_{I_k I_k}\ \cdot V[M,I_k,\ I_k,i-1,j] \\ t_{XX}\cdot t_{I_k I_k}\ \cdot V[X,I_k,\ I_k,i-1,j] \\ t_{XX}\cdot t_{M_k I_k}\cdot V[X,M_k,I_k,i-1,j] \\ t_{YX}\cdot t_{I_k I_k}\ \cdot V[Y,\ I_k,\ I_k,i-1,j] \\ t_{YX}\cdot t_{M_k I_k}\cdot V[Y,\ M_k,I_k,i-1,j] \end{cases} \tag{2.18}$$

$$V[X, I_k, I_\ell, i, j] = e(x_i, -)e_{I_k}(x_i)\cdot \\ \cdot \max \begin{cases} t_{XX}\cdot t_{M_k I_k}\cdot V[X,M_k,I_\ell,i-1,j] \\ t_{XX}\cdot t_{I_k I_k}\ \cdot V[X,I_k,\ I_\ell,i-1,j] \\ t_{YX}\cdot t_{M_k I_k}\cdot V[Y,\ M_k,I_\ell,i-1,j] \\ t_{YX}\cdot t_{I_k I_k}\ \cdot V[Y,\ I_k,\ I_\ell,i-1,j] \end{cases} \quad (k \neq \ell) \tag{2.19}$$

We complete the list of equations for allowed configurations with pair state X with both profile models being in insert states I_k and I_ℓ. Just like the previous pair, (2.18) covers the case when $\ell = k$ and (2.19) the opposite, $\ell \neq k$.

It remains to describe all equations for derivation of values for cells with pair state Y. These are analogous to equations (2.12)-(2.19) and are shown here for

completeness.

$$V[Y, M_\ell, M_k, i, j] = e(-, y_j)e_{M_k}(y_j)\cdot \cdot \max \begin{cases} t_{MY}\cdot t_{M_\ell M_k} \cdot V[M,M_\ell,M_\ell,\ i,j-1] & \ell < k \\ t_{XY}\cdot t_{M_n M_k} \cdot V[X, M_\ell,M_n,\ i,j-1] & n \neq \ell, n < k \\ t_{XY}\cdot t_{I_{k-1} M_k}\cdot V[X, M_\ell,I_{k-1},i,j-1] & \\ t_{YY}\cdot t_{M_n M_k} \cdot V[Y,\ M_\ell,M_n,\ i,j-1] & n \neq \ell, n < k \\ t_{YY}\cdot t_{I_{k-1} M_k}\cdot V[Y,\ M_\ell,I_{k-1},i,j-1] & \end{cases} \quad (\ell < k) \tag{2.20}$$

$$V[Y, M_\ell, M_k, i, j] = e(-, y_j)e_{M_k}(y_j)\cdot \cdot \max \begin{cases} t_{XY}\cdot t_{M_n M_k} \cdot V[X,M_\ell,M_n,\ i,j-1] & n < k < \ell \\ t_{XY}\cdot t_{I_{k-1} M_k}\cdot V[X,M_\ell,I_{k-1},i,j-1] & \\ t_{YY}\cdot t_{M_n M_k} \cdot V[Y, M_\ell,M_n,\ i,j-1] & n < k < \ell \\ t_{YY}\cdot t_{I_{k-1} M_k}\cdot V[Y, M_\ell,I_{k-1},i,j-1] & \end{cases} \quad (k < \ell) \tag{2.21}$$

$$V[Y, I_\ell, M_k, i, j] = e(-, y_j)e_{M_k}(y_j)\cdot \cdot \max \begin{cases} t_{XY}\cdot t_{M_n M_k} \cdot V[X,I_\ell,M_n,\ i,j-1] & n < k \\ t_{XY}\cdot t_{I_{k-1} M_k}\cdot V[X,I_\ell,I_{k-1},i,j-1] & \\ t_{YY}\cdot t_{M_n M_k} \cdot V[Y, I_\ell,M_n,\ i,j-1] & n < k \\ t_{YY}\cdot t_{I_{k-1} M_k}\cdot V[Y, I_\ell,I_{k-1},i,j-1] & \end{cases} \quad (\ell \neq k-1) \tag{2.22}$$

$$V[Y, I_{k-1}, M_k, i, j] = e(-, y_j)e_{M_k}(y_j)\cdot \cdot \max \begin{cases} t_{MY}\cdot t_{I_{k-1} M_k}\cdot V[M,I_{k-1},I_{k-1},i,j-1] & \\ t_{XY}\cdot t_{M_n M_k} \cdot V[X, I_{k-1},M_n,\ i,j-1] & n < k \\ t_{XY}\cdot t_{I_{k-1} M_k}\cdot V[X, I_{k-1},I_{k-1},i,j-1] & \\ t_{YY}\cdot t_{M_n M_k} \cdot V[Y,\ I_{k-1},M_n,\ i,j-1] & n < k \\ t_{YY}\cdot t_{I_{k-1} M_k}\cdot V[Y,\ I_{k-1},I_{k-1},i,j-1] & \end{cases} \tag{2.23}$$

$$V[Y, M_\ell, I_k, i, j] = e(-, y_j)e_{I_k}(y_j)\cdot \cdot \max \begin{cases} t_{XY}\cdot t_{M_kI_k}\cdot V[X,M_\ell,M_k,i,j-1] \\ t_{XY}\cdot t_{I_kI_k}\ \cdot V[X,M_\ell,I_k,\ i,j-1] \\ t_{YY}\cdot t_{M_kI_k}\cdot V[Y,\ M_\ell,M_k,i,j-1] \\ t_{YY}\cdot t_{I_kI_k}\ \cdot V[Y,\ M_\ell,I_k,\ i,j-1] \end{cases} \quad (k \neq \ell) \tag{2.24}$$

$$V[Y, M_k, I_k, i, j] = e(-, y_j)e_{I_k}(y_j)\cdot \cdot \max \begin{cases} t_{XY}\cdot t_{M_kI_k}\cdot V[M,M_k,M_k,i,j-1] \\ t_{XY}\cdot t_{I_kI_k}\ \cdot V[X,\ M_k,I_k,\ i,j-1] \\ t_{YY}\cdot t_{I_kI_k}\ \cdot V[Y,\ M_k,I_k,\ i,j-1] \end{cases} \quad (k \neq \ell) \tag{2.25}$$

$$V[Y, I_k, I_k, i, j] = e(-, y_j)e_{I_k}(y_j)\cdot \cdot \max \begin{cases} t_{MY}\cdot t_{I_kI_k}\ \cdot V[M,I_k,I_k,\ i,j-1] \\ t_{XY}\ \cdot t_{I_kI_k}\ \cdot V[X,\ I_k,I_k,\ i,j-1] \\ t_{XY}\ \cdot t_{M_kI_k}\cdot V[X,\ I_k,M_k,i,j-1] \\ t_{YY}\ \cdot t_{I_kI_k}\ \cdot V[Y,\ I_k,I_k,\ i,j-1] \\ t_{YY}\ \cdot t_{M_kI_k}\cdot V[Y,\ I_k,M_k,i,j-1] \end{cases} \tag{2.26}$$

$$V[Y, I_\ell, I_k, i, j] = e(-, y_j)e_{I_k}(y_j)\cdot \cdot \max \begin{cases} t_{XY}\cdot t_{M_kI_k}\cdot V[X,I_\ell,M_k,i,j-1] \\ t_{XY}\cdot t_{I_kI_k}\ \cdot V[X,I_\ell,I_k,\ i,j-1] \\ t_{YY}\cdot t_{M_kI_k}\cdot V[Y,\ I_\ell,M_k,i,j-1] \\ t_{YY}\cdot t_{I_kI_k}\ \cdot V[Y,\ I_\ell,I_k,\ i,j-1] \end{cases} \quad (k \neq \ell) \tag{2.27}$$

2.1.3 Boundary conditions

The matrix V is filled using four nested loops:
`for` $i := 1$ `to` L_x; `for` $j := 1$ `to` L_y; `for` $k := 1$ `to` L; `for` $\ell := 1$ `to` L.
The basic boundary condition that initializes the computation is

$$V[M, M_0, M_0, 0, 0] = 1. \tag{2.28}$$

The start state of pair model is the match state and the start state of the profile model is M_0 and we can assume an implicit \$s to be aligned at the zero sequence

indexes, just like in the Algorithm 1. Since this is the only valid state path that does not emit anything, we do not allow other combinations of states at sequence position $(0, 0)$:

$$V[s_P, s_X, s_Y, 0, 0] = 0, \quad s_P \in \{X, Y\}, s_X, s_Y \in \{M_k, I_k \mid 1 \leq k \leq L\}. \tag{2.29}$$

Next, we want to allow gaps at the beginning of the alignments. This means that the model will process only one of the input sequences and advance it's index while keeping zero index for the other sequence. Some of the combinations of the three model states are not meaningful in this situation and we assign zero score to those cells. Namely,

$$V[M, s_X, s_Y, i, 0] = 0, \quad \forall s_X, s_Y, i \in [1, L_x], \tag{2.30}$$

$$V[M, s_X, s_Y, 0, j] = 0, \quad \forall s_X, s_Y, j \in [1, L_y], \tag{2.31}$$

because there is a conflict between the pair match state and only one of the sequences being emitted. Furthermore,

$$V[Y, s_X, s_Y, i, 0] = 0, \quad \forall s_X, s_Y, i \in [1, L_x], \tag{2.32}$$

$$V[X, s_X, s_Y, 0, j] = 0, \quad \forall s_X, s_Y, j \in [1, L_y], \tag{2.33}$$

because the presence of pair state Y means an insertion in the Y sequence, which is not emitted in (2.32) (analogical argument holds in the (2.33) case).

To finalize the initialisation that enables gaps at the beginning of the alignments, we need to perform similar initialisation step like in the Viterbi algorithm for pair HMMs. We will proceed in one sequence only, leaving the other at the beginning. This time, following from the model topologies, we need to take care of two cases for both sequences, looping through $i \in [1, L_x], j \in [1, L_y], k \in$

$[1, L]$ (in case of (2.35) and (2.37), $k \in [0, L]$):

$$V[X, M_k, M_0, i, 0] = e(x_i, -)e_{M_k}(x_i) \cdot \max \begin{cases} t_{MX} \cdot t_{M_0 M_k} \cdot V[M, M_0, M_0, i-1, 0], \\ t_{XX} \cdot t_{M_n M_k} \cdot V[X, M_n, M_0, i-1, 0], & 1 \leq n < k, \\ t_{XX} \cdot t_{I_{k-1} M_k} \cdot V[X, I_{k-1}, M_0, i-1, 0], \end{cases} \tag{2.34}$$

$$V[X, I_k, M_0, i, 0] = e(x_i, -)e_{I_k}(x_i) \cdot \max \begin{cases} t_{MX} \cdot t_{M_0 I_k} \cdot V[M, M_0, M_0, i-1, 0], \\ t_{XX} \cdot t_{M_k I_k} \cdot V[X, M_k, M_0, i-1, 0], \\ t_{XX} \cdot t_{I_k I_k} \cdot V[X, I_k, M_0, i-1, 0], \end{cases} \tag{2.35}$$

$$V[Y, M_0, M_k, 0, j] = e(-, y_j)e_{M_k}(y_j) \cdot \max \begin{cases} t_{MY} \cdot t_{M_0 M_k} \cdot V[M, M_0, M_0, 0, j-1], \\ t_{YY} \cdot t_{M_n M_k} \cdot V[Y, M_0, M_n, 0, j-1], & 1 \leq n < k, \\ t_{YY} \cdot t_{I_{k-1} M_k} \cdot V[Y, M_0, I_{k-1}, 0, j-1], \end{cases} \tag{2.36}$$

$$V[Y, M_0, I_k, 0, j] = e(-, y_j)e_{I_k}(y_j) \cdot \max \begin{cases} t_{YY} \cdot t_{M_0 I_k} \cdot V[M, M_0, M_0, 0, j-1], \\ t_{YY} \cdot t_{M_k I_k} \cdot V[Y, M_0, M_k, 0, j-1], \\ t_{YY} \cdot t_{I_k I_k} \cdot V[Y, M_0, I_k, 0, j-1]. \end{cases} \tag{2.37}$$

Even though the equations above seem complicated, the process they describe a simple algorithm similar to the Viterbi algorithm for single profile HMM. This is desirable, because when we are proceeding in one sequence only it means that we are actually moving in one profile. In (2.34), we move along the chain of match states, emitting x. For every match state along the chain there are several options how the model could arrive at that state (see fig. 2.1): directly from the start state (the first option in (2.34)), from a previous match state (the second option in (2.34)) and, finally, from the previous column insert state (the last option in (2.34)). We might end up in the insert state too. The first option in (2.35) is applicable to the first insert state of the profile when emitting the first symbol of the sequence, otherwise it's zero. Two other options cover the rest of possible arrivals at the insert state. The difference between the first and the second option in (2.35) is that in the first option, we are opening the gap,

whereas the second option applies when extending the gap, which is indicated by different pair states. The explanations for (2.36) and (2.37) are analogous.

2.1.4 Termination and traceback

Every time we compute a value for any cell, we implicitly keep a pointer to the cell from which the value was derived (this holds in the general recurrence too). We will use those pointers when tracing back the result of the whole dynamic programming computation, just like in the standard Viterbi algorithm.

The termination step completes the calculation of the state path trio. For each submodel it consists of transition from the current state to the end state. This step is simple in the pair model since there are just three states. In the profile models, the end state is treated like the match state, therefore, after the preprocessing step (section 2.1.1) there is a transition from each match state to the end state.

$$end = \max \begin{cases} t_{me} \cdot t_{I_L E} \cdot t_{I_L E} \cdot V[M, I_L,\ I_L,\ L_x, L_y] & \\ t_{me} \cdot t_{M_k E} \cdot t_{M_k E} \cdot V[M, M_k, M_k, L_x, L_y] & 0 < k < L \\ t_{xe} \cdot t_{M_k E} \cdot t_{M_\ell E} \cdot V[X, M_k, M_\ell, L_x, L_y] & 0 < k, \ell < L, k \neq \ell \\ t_{ye} \cdot t_{M_k E} \cdot t_{M_\ell E} \cdot V[Y,\ M_k, M_\ell, L_x, L_y] & 0 < k, \ell < L, k \neq \ell \end{cases} \tag{2.38}$$

As we have mentioned already in Section 2.1.1, for each cell v in V we store a pointer to the cell from which the value of v was derived. This applies to (2.38) as well. We use these pointers to trace back the three resulting state paths π_P, π_X, π_Y (from the pair model, x-profile, and y-profile respectively). The trace back procedure starts with the end pointer and follows the pointers up to the $V[M, M_0, M_0, 0, 0]$ cell, which was initialized at the beginning (since the first step is derived from $V[M, M_0, M_0, 0, 0]$, each traceback will end in this cell).

Of the greatest importance to us is the path in the pair model, since it defines the alignment of x and y. We use the state paths to compute the joint probabilities of sequences and state paths (as defined in (1.23): $P_{\text{pair}}(x, y, \pi_P)$, the probability of generating x and y along the path π_P in the pair model, $P_{\text{profile}}(x, \pi_X)$, and $P_{\text{profile}}(y, \pi_Y)$, the probabilities of generating x and y along the paths π_X and π_Y in the profile model.

2.2 Alignment of Whole Motif Arrays

We use the same procedure as MotifAligner (described in Section 1.3) for alignment of complete motif arrays. We compute all pairwise alignments of individual motifs with the first motif from one motif array and the second from the other. The score of a pairwise motif alignment is based on joint probabilities of motif sequences and state paths in all three models. Since we perform the motif alignment for all pairs of motifs and assign a score to each such alignment, we get a scoring system similar to a scoring matrix. Treating motifs as symbols and using this scoring matrix, we obtain the full alignment of input motif arrays using Needleman-Wunsch algorithm (Algorithm 1).

More formally, for motif arrays $A_x = (x_1, \ldots, x_n)$ and $A_y = (y_1, \ldots y_m)$, we calculate $n \times m$ matrix S, where

$$S(x_i, y_j) = \ln \frac{P_{\text{pair}}(x_i, y_j, \pi_{P_{ij}})}{P_{\text{profile}}(x_i, \pi_{X_i}) P_{\text{profile}}(y_j, \pi_{Y_j})}, \tag{2.39}$$

with $\pi_{P_{ij}}$, π_{X_i} and π_{Y_j} being the state paths as defined in the previous section.

This score compares the hypothesis that the two motif sequences are related (given by probability from the pair HMM) to the hypothesis that these are simply two independent sequences following the same profile (as determined by scores from the two profile HMMs).

2.3 Algorithm Complexity and Implementation Notes

The time complexity of the PPP algorithm on two motif arrays with $O(n)$ motifs, each of length $O(m)$ is $O(n^2 m^6)$. There are $O(n^2)$ individual motif alignments. The time needed to compute one such alignment is $O(L_x L_y L^4)$, where L_x, L_y are the lengths of motifs and L is the number of columns in the profile HMM. This follows from the observation that in the recurrent step of individual motif alignment we fill $3 \times L \times L \times L_x \times L_y$ matrix, and time required to compute each cell is $O(L^2)$, the upper bound on the number of values considered in the maximization in every equation (2.34)-(2.38). Typically, the number of columns in the profile HMM and the length of motifs is almost the same, so we can say that $L_x, L_y, L = O(m)$ and hence the time required to compute the alignment of one motif pair is $O(m^6)$. From the same observation one can easily see that the space complexity is $O(max\{n^2, m^4\})$.

Like other HMM algorithms, PPP multiplies small numbers many times. A straightforward implementation using floating point arithmetics would result in underflow errors. To cope with this we employ a widely used technique and perform all computations in log scale (using natural logarithm). This means that in the initialisation, the 1 in (2.28) becomes 0, the 0s in (2.29)-(2.33) become $-\infty$, every multiplication in (2.34)-(2.38) becomes addition, and the score for an alignment in (2.39) becomes

$$S(x_i, y_j) = P_{\text{pair}}(x_i, y_j, \pi_{P_{ij}}) - P_{\text{profile}}(x_i, \pi_{X_i}) - P_{\text{profile}}(y_j, \pi_{Y_j}). \tag{2.40}$$

Chapter 3

Experiments

This chapter is focused on evaluation of our tool on realistic datasets. Our experiments are focused on evaluation of two aspects of our method, the discrimination power of our score and accuracy of the resulting alignment.

We also describe how to set the parameters of our model. The parameters can be divided to three groups: parameters of the pair model, parameters of the profile model, and parameters of the Needleman-Wunsch algorithm. Most parameters of the Profile-Profile-Pair model are determined by parameters of underlying models. The parameters of Needleman-Wunsch algorithm need to be carefully tuned with regard to a particular dataset to avoid underfitting and overfitting.

3.1 Dataset Preparation

The dataset we used comes from the Human KZNF Catalog [Huntley et al., 2006], hosted at `http://znf.igb.uiuc.edu/human/`. The entries of this database consists of sequence annotations of all known human KRAB zinc finger genes, as well as putative genes (not verified experimentally) and pseudogenes (partial genes that lost their function but retain the structure of a functional gene). Each gene can have multiple alternative transcripts in the database, these are referred to as gene models.

We downloaded the set of all annotations in the catalog. This set labels the NCBI35/hg17 assembly of the human genome released in May, 2004 [IHGSC, 2004], which is not the current human genome assembly at the time of writing – current assembly is GRCh37/hg19, released in February, 2009 [GRC, 2009]. We remapped the annotation set to the hg19 assembly using the *liftOver* tool from the UCSC Genome Browser [Fujita et al., 2011].

Table 3.1: The complete dataset, based on genes from the whole human genome.

	Number of		Finger Motifs		
Genome	Genes	Models	Total	Average	Median
hg19	612	1071	13363	12.48	12
mm9	302	513	5226	10.19	10
canFam2	477	828	9259	11.18	11
rheMac2	578	1010	12143	12.02	12

Table 3.2: The restricted dataset, omitting genes from human chromosome 19 and their orthologs.

	Number of		Finger Motifs		
Genome	Genes	Models	Total	Average	Median
hg19	323	510	5249	10.29	9
mm9	201	314	2818	8.97	8
canFam2	257	406	3710	9.14	8
rheMac2	305	484	4766	9.85	9

Using the whole genome alignments from the UCSC Genome Browser [UCSC, 2009], we identified corresponding locations in the mouse (assembly mm9), dog (canFam2), and rhesus macaque (rheMac2) genomes. These are locations of potential orthologs (genes in different species that evolved from a common ancestral gene by speciation). We translated all genome sequences to amino acid sequences, and worked with these only. We dropped a particular gene from the dataset if it contained fingers shorter than 10 amino acids. We will refer to this set of sequences as the *complete dataset*. Summary statistics for the dataset are shown in Table 3.1. Because of relatively high time complexity of the PPP algorithm, alignment of genes with high number of fingers takes a lot of time. For that reason, we prepared a subset of the complete dataset, omitting genes from human chromosome 19 and their putative ortholgs in other genomes. These genes contain the highest numbers of repeating motifs (30 or more). Summary statistics for this *restricted dataset* are shown in the Table 3.2.

3.2 Estimating Parameters of Our Model

In general, parameters of hidden Markov models are estimated by supervised or unsupervised training. The supervised training requires that correct state path is known for every sequence in the training set, a condition that is rarely satisfied. Hence, the unsupervised training approach is typically used. A stan-

Table 3.3: Background probabilities, $q(a)$, for the BLOSUM85 matrix.

A	R	N	D	C	Q	E	G	H	I
0.072	0.050	0.042	0.054	0.030	0.033	0.054	0.078	0.025	0.067
L	K	M	F	P	S	T	W	Y	V
0.098	0.054	0.024	0.048	0.038	0.058	0.051	0.015	0.035	0.073

dard choice is the Baum-Welch algorithm [Rabiner, 1989] which is a special case of the general Expectation-Maximization algorithm [Dempster et al., 1977] for hidden Markov models. A disadvantage of the unsupervised approach is that EM algorithm on models with many parameters is likely to find local maximum of likelihood of the parameters which leads to worse performance of the model.

Our approach to parameter acquisition is different. Since our model has three sets of parameters and especially the emission and transition probability distributions are defined on large domains, the standard EM approach is likely to fail. Therefore, we have assessed the parameter sets independently.

3.2.1 Pair HMM Parameters

The emission parameters of pair HMM used in our experiments were based on the BLOSUM85 substitution matrix. This particular matrix was chosen because it is suitable for estimating homology of relatively similar sequences and because the authors of MotifAligner used the same matrix in their work [Nowick et al., 2011] and we wanted to get comparable results.

The BLOSUM85 matrix was obtained from the EMBOSS software package [Rice et al., 2000]. We did not use the log-odds matrix itself. Instead, we used the probability distributions p and q from which the matrix was derived (the derivation process is explained in the Section 1.1.1). The emission probability distribution of the pair match state, p, is shown in the Figure 3.1. The emission probability distribution q of the states X and Y is shown in the Table 3.3.

The transition probability parameters $\varepsilon, \delta, \tau$ were set as follows:

- $\tau = 0.0345$ so that the expected length of an alignment is 28, which is the length of a typical human C_2H_2 zinc finger motif;
- $\delta = 0.05185$ so that the expected length of a match region is 13.45, because the most variable region of a zinc finger motif spans positions 12-15;
- $\varepsilon = 0.4769$ so that the expected length of a gap is 1.1.

```
#  BLOSUM Clustered Target Frequencies=qij
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 85
   A      R      N      D      C      Q      E      G      H      I
S 0.0267
R 0.0019 0.0217
N 0.0015 0.0016 0.0172
D 0.0017 0.0012 0.0036 0.0275
C 0.0015 0.0003 0.0004 0.0003 0.0182
Q 0.0016 0.0022 0.0013 0.0013 0.0003 0.0102
E 0.0027 0.0021 0.0018 0.0047 0.0003 0.0034 0.0223
G 0.0051 0.0014 0.0024 0.0022 0.0006 0.0010 0.0016 0.0496
H 0.0009 0.0011 0.0012 0.0008 0.0001 0.0011 0.0011 0.0007 0.0107
I 0.0025 0.0010 0.0007 0.0007 0.0011 0.0007 0.0009 0.0009 0.0004 0.0231
L 0.0035 0.0017 0.0010 0.0010 0.0013 0.0013 0.0014 0.0015 0.0007 0.0108
K 0.0027 0.0058 0.0021 0.0019 0.0004 0.0027 0.0033 0.0019 0.0009 0.0011
M 0.0011 0.0006 0.0004 0.0003 0.0003 0.0007 0.0005 0.0005 0.0002 0.0025
F 0.0013 0.0006 0.0005 0.0005 0.0006 0.0005 0.0006 0.0008 0.0006 0.0026
P 0.0021 0.0008 0.0006 0.0009 0.0003 0.0006 0.0012 0.0010 0.0004 0.0007
S 0.0062 0.0019 0.0028 0.0023 0.0010 0.0016 0.0025 0.0033 0.0009 0.0014
T 0.0036 0.0014 0.0019 0.0015 0.0009 0.0012 0.0018 0.0017 0.0006 0.0023
W 0.0003 0.0002 0.0001 0.0001 0.0001 0.0002 0.0002 0.0003 0.0002 0.0003
Y 0.0010 0.0007 0.0006 0.0004 0.0003 0.0005 0.0006 0.0006 0.0015 0.0012
V 0.0044 0.0012 0.0008 0.0009 0.0014 0.0009 0.0014 0.0013 0.0005 0.0123

   L      K      M      F      P      S      T      W      Y      V
L 0.0457
K 0.0018 0.0200
M 0.0050 0.0007 0.0060
F 0.0051 0.0007 0.0010 0.0224
P 0.0011 0.0012 0.0003 0.0004 0.0231
S 0.0020 0.0024 0.0007 0.0010 0.0014 0.0185
T 0.0027 0.0020 0.0009 0.0010 0.0010 0.0049 0.0174
W 0.0006 0.0002 0.0002 0.0007 0.0001 0.0002 0.0002 0.0094
Y 0.0019 0.0007 0.0005 0.0046 0.0003 0.0009 0.0008 0.0010 0.0160
V 0.0085 0.0014 0.0021 0.0021 0.0010 0.0020 0.0032 0.0003 0.0011 0.0262
```

Figure 3.1: Target probabilities, $p(a, b)$, for the BLOSUM85 matrix.

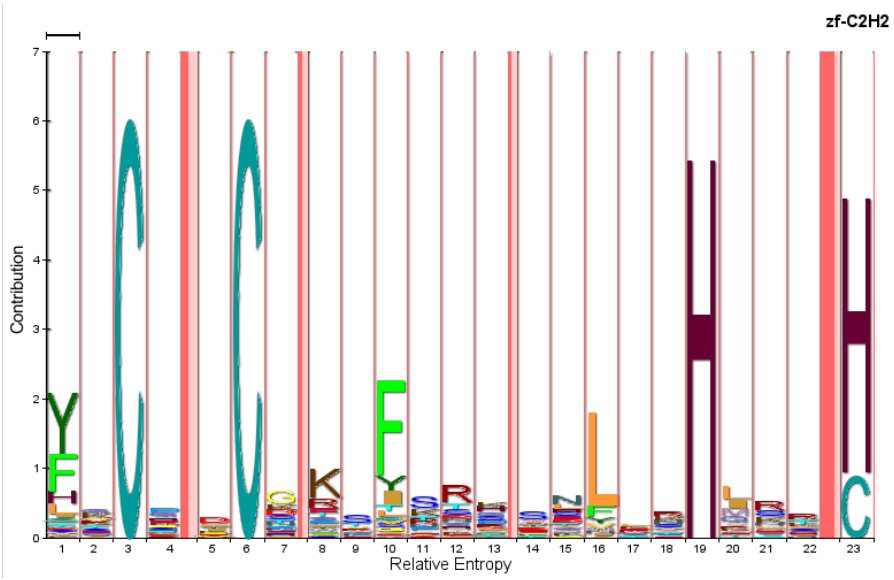


Figure 3.2: The profile HMM of C_2H_2 human zinc fingers from the Pfam database [Bateman et al., 2011], viewed as a HMM logo [Schuster-Bockler et al., 2004].

3.2.2 Profile HMM Parameters

The complete parameter set of the profile model was acquired from the Pfam database entry for the ZNF C_2H_2 family [Bateman et al., 2011]. The length of the profile is 23, which is shorter than the typical human zinc finger motif. The reason is that the model is based on multiple sequence alignment of more diverse sequences from various different species. The model is visualized in Figure 3.2. Another model, based on the alignment of 2000 random finger sequences form the complete dataset, is shown in the Figure 3.3 for comparison. Both figures were generated using the profile HMM visualisation tool LogoMat-M [Schuster-Bockler et al., 2004].

3.2.3 Parameters of the Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm for the whole motif array alignment has three parameters: the gap opening penalty g, the gap extension penalty e, and the substitution matrix s that scores individual motif alignments. In the PPP model, the matrix s is determined by the equation 2.39. In our experiments, the parameters g and e were the only parameters that were not set explicitly and we will state their values when describing the experiments.

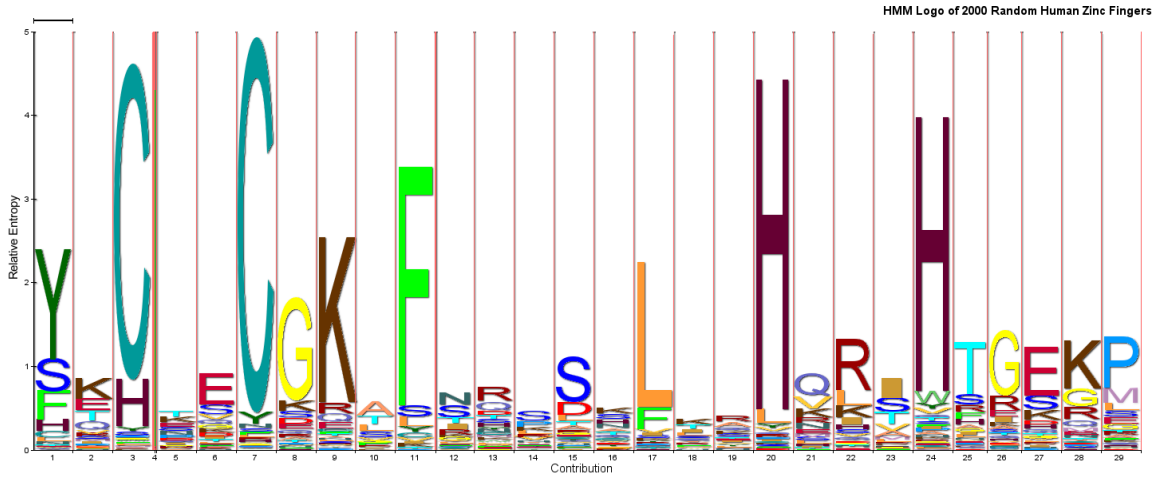


Figure 3.3: The profile HMM of random 2000 human zinc fingers from the complete dataset, viewed as a HMM logo [Schuster-Bockler et al., 2004].

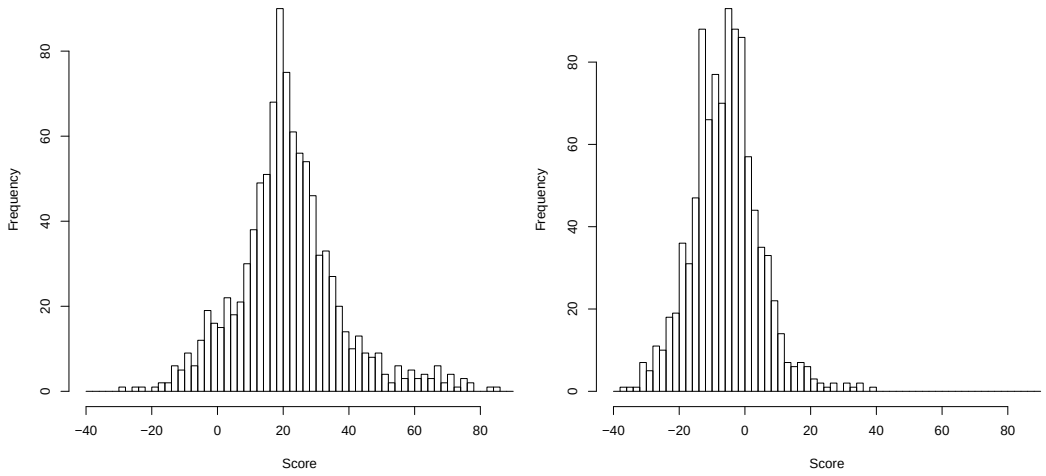
3.3 The PPP Score Distribution

We have explored the distribution of scores of the PPP scoring function (Equation (2.39)). From the complete dataset, we have made two random samples of 1000 sequence pairs. In each pair, one sequence was a human finger and the other was a mammal finger. The first sample contained putative orthologous pairs, i.e. the sequences corresponding to the same protein and to the same motif in that protein, we call this set the *related* set. Sequences in the other sample were chosen independently on each other, we call this set the *random* set.

We have computed an alignment of each sequence pair for both samples. The score distributions are shown in Figure 3.4. Both distributions resemble the normal distribution, with mean of the Related set close to 20 and mean of the Random set at around 5. This is good because we expect the more similar sequences to have higher score.

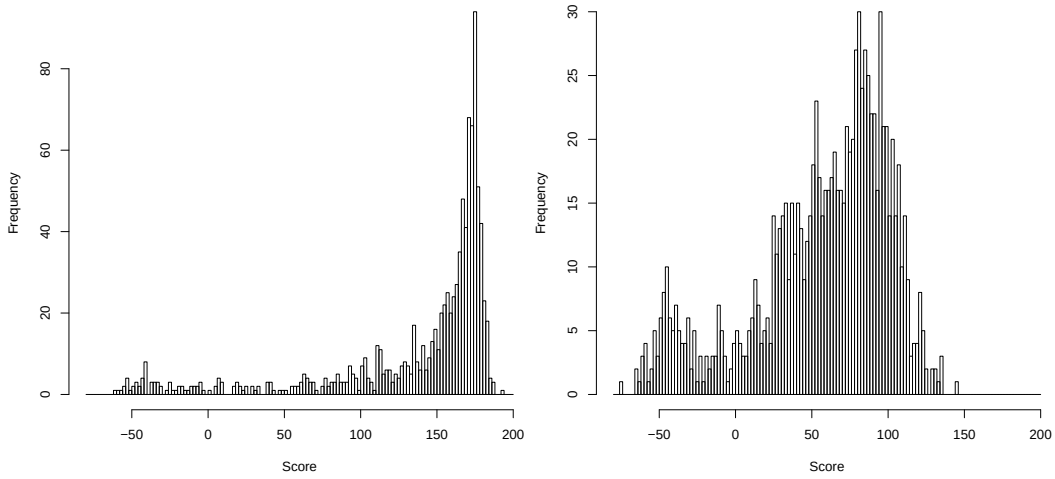
For comparison, in Figure 3.4 we show the score distributions of the MotifAligner approach to alignment of individual motifs, based on the BLOSUM85 substitution matrix (MotifAligner implementation details are in the next section). These distributions does not resemble the normal distribution. There is a relatively heavy tail in the Related set distribution, which is clearly not desirable.

In Figure 3.6, we plotted a ROC curve for better evaluation and comparison. The Related dataset was treated as positive examples and the Random dataset as negative. The curve shows, that PPP model has a better scoring function. However, there is still a room for improvement.



(a) Related (b) Random

Figure 3.4: The score distributions for related and random datasets, PPP model.



(a) Related (b) Random

Figure 3.5: The score distributions for related and random datasets, MotifAligner approach based on the BLOSUM85 matrix.

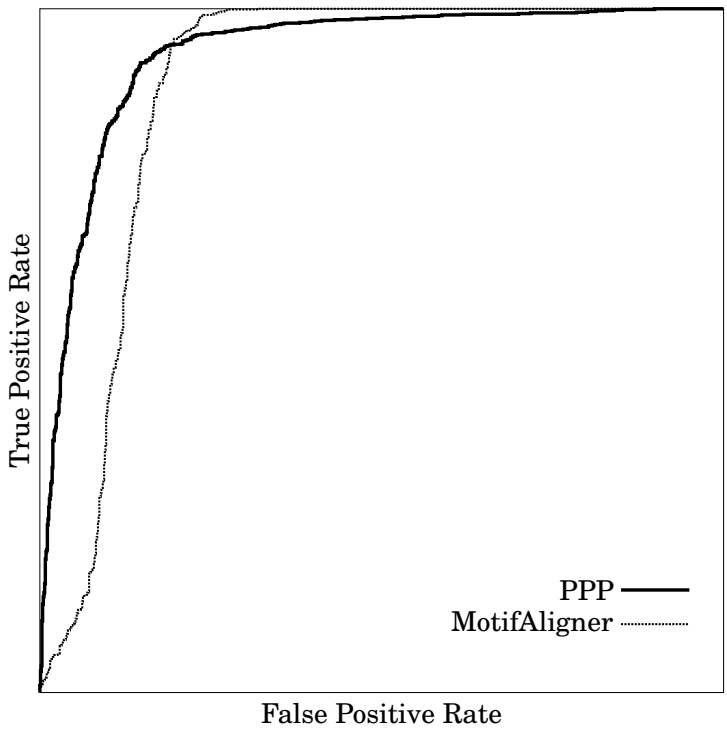


Figure 3.6: ROC curve for Related (positive) and Random (negative) datasets.

3.4 Alignment Accuracy

We have performed an evaluation of PPP alignment accuracy. To be able to compare our algorithm with some other method, we have implemented MotifAligner algorithm (Section 1.3). Since we are not aware of any benchmark for assessing the accuracy of alignment with repetitive motifs that evaluates multiple repeats simultaneously, we have developed our own benchmark (the only benchmark concerning repeats referenced in literature, the BAliBASE reference 6 [Thompson et al., 2005], does not define the correct alignment of motif tuples, it defines correct alignments of individual repeats only).

Our benchmark is based on the *complete dataset* (Section 3.1). We know the exact order of zinc finger motifs in human and mammal protein sequences. We define a rank function on the fingers in a zinc finger array. The rank of a finger in human protein is simply its position in zinc finger array, counted from left to right. The rank of a mammal finger is the rank of the putative orthologous human finger. Suppose that a human zinc finger protein contains three zinc fingers and that an orthologus protein in mouse has two zinc fingers. Moreover, suppose that the first human finger is orthologous with the first mouse finger, the second human finger does not have its ortholog in the mouse protein and that the third human finger is orthologous with the second mouse zinc finger.

Then the rank of the second mouse zinc finger is three.

For alignments of zinc finger arrays of orthologous proteins, we can tell which fingers are correctly aligned and which are not: we consider a pair of motifs to be correctly aligned if and only if the ranks of both sequences in the aligned pair under consideration are the same. We are aware that the true correctness of this method basically depends on the correctness of the whole genome alignments that were used for finding the orthologous sequences.

Because the MotifAligner is not publicly downloadable, we have implemented our own version according to the description given in [Nowick et al., 2011]. All results were obtained using the BLOSUM85 substitution matrix and the gap penalties set to $g = 84$ and $e = 75.6$. In order to be able to align motifs that do not have the same length, we append the necessary number of X symbols to the end of the shorter sequence. The X states for *undetermined* amino acid and is correctly handled by the BLOSUM matrix.

We carried out three tests. In the first one, we aligned all zinc finger arrays of orthologous proteins in the complete dataset. The second and the third experiments simulated a loss of fingers during the evolution – we created two artificial datasets with 1/5 and 1/3 of the total number of fingers removed in each zinc finger array in all four genomes, and we aligned the original human dataset with the four reduced sets.

We tried several different parameter settings for the Needleman-Wunsch step of the PPP algorithm (the values of other parameters are described in the Section 3.2). In particular, we achieved the best results when the gap opening penalty g was set to 30 and the gap extension penalty e to 20. The results of all tests are shown in the Table 3.4.

As we can see, with PPP[1] we were able to outperform the MotifAligner on the *Unchanged* and *1/5 Loss* datasets. On the other hand, our model performed slightly worse when the number of lost fingers was increased. In general, the number of wrongly aligned columns (misaligned plus excess gaps) never exceeded 1.6% of the total number of columns in the alignments for both algorithms. This might point out to weakness in our evaluation benchmark.

Table 3.4: The comparison of MotifAligner and our Profile-Profile-Pair (PPP) model. PPP[1] refers to Needleman-Wunsch gap penalty parameters set to $g = 30$, $e = 20$ and PPP[2] to $g = 20$, $e = 10$. The third column lists the number of different zinc finger array pairs aligned; fourth column contains the sum of the expected length of all alignments (including gaps); the fifth column lists the number of wrongly aligned motifs (as defined in the main text); and the last column contains the difference of measured and expected number of gaps in the alignments.

		Alignment size		Errors	
Dataset	Program	Pairs	Length	Misaligned	Gaps
Complete, Unchanged	MotifAligner	2161	24267	234	0
Complete, Unchanged	PPP[1]	2161	24267	178	2
Complete, Unchanged	PPP[2]	2161	24267	331	28
Restricted, 1/5 Loss	MotifAligner	1609	16096	149	0
Restricted, 1/5 Loss	PPP[1]	1609	16096	139	0
Restricted, 1/5 Loss	PPP[2]	1609	16096	142	6
Restricted, 1/3 Loss	MotifAligner	1651	16227	169	0
Restricted, 1/3 Loss	PPP[1]	1651	16227	254	0
Restricted, 1/3 Loss	PPP[2]	1651	16227	252	2

Conclusion

We have designed and implemented an algorithm for alignment of sequences with repetitive motifs. The algorithm is built on top of two types of hidden Markov models. It utilizes positional information from two copies of a profile HMM and uses a pair HMM to align the motif sequences. We were able to apply our model on real world data, which was one of the goals of this work, and obtained results that outperform the only existing program specifically designed to align sequences with repetitive motifs.

There is still a room for improvement of our work. Apart from obvious upgrades, like a more effective implementation, the underlying model can be enhanced in several ways. For example, an interesting question is whether another scoring function of individual motif alignments would perform better. Such a function might be based on different properties of the underlying models, e.g. the full probability of a sequence, instead of the probability of the Viterbi path.

To alleviate problems caused by the computational complexity of the algorithm, various heuristics could be applied, especially methods avoiding exhausting computations of whole dynamic programming matrices. In order to be able to apply our model to other protein families with repeating motifs, a more robust procedure for parameter estimation should be established. In addition, a method for assessment of statistical significance of alignments may be helpful when computing alignments of large datasets where random homologies are more likely to occur.

The model we have implemented is not the only true way of doing sequence alignment with repetitive motifs. It is very appealing to use a monolithic probabilistic model. We have tried to develop such models on paper, but we were not able to overcome intrinsic difficulties of these models. Authors more skilled in probabilistic modelling might find their way out of the monolithic model problems.

Even though the first studies on the sequence alignment appeared several decades ago, the problem is far from being solved in general. In future, the

pressure for fast, scalable and accurate sequence alignment tools will rise, because of the pertaining exponential growth in the amounts of data produced by biologists. The most serious problem from the practical point of view that we have encountered is the lack of reliable benchmark for assessing the accuracy of alignments with repetitive motifs. A high quality reference is very valuable, because it allows exact evaluation of algorithms and can give a clue where are the weak and the strong parts of a particular program, or how to set parameters to ensure optimal performance. We understand that building such reference requires substantial effort and deep knowledge of all corners of biology. We hope that our work will at least partially serve as a catalyst towards the creation of such resource.

References

[Altschul et al., 1990] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410. [PubMed:2231712].

[Altschul et al., 1997] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–3402. [PubMed:9254694].

[Bateman et al., 2011] Bateman, A., Boehm, S., Sonnhammer, E. L. L., and Gago, F. (2011). Mulitple Sequence Alignment of Zinc Finger C2H2 Type Family. Pfam Family: zf-C2H2 (PF00096). Online. `http://pfam.sanger.ac.uk/family/PF00096`.

[Bellefroid et al., 1991] Bellefroid, E. J., Poncelet, D. A., Lecocq, P. J., Revelant, O., and Martial, J. A. (1991). The evolutionarily conserved Krüppel-associated box domain defines a subfamily of eukaryotic multifingered proteins. *Proc. Natl. Acad. Sci. U.S.A.*, 88(9):3608–3612. [PubMed:2023909].

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38. [JSTOR:2984875].

[Ding et al., 2009] Ding, G., Lorenz, P., Kreutzer, M., Li, Y., and Thiesen, H. J. (2009). SysZNF: the C2H2 zinc finger gene database. *Nucleic Acids Res.*, 37(Database issue):D267–273. [PubMed:18974185].

[Durbin et al., 1998] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis*. 1st edition. Cambridge University Press. 356 p., ISBN: 978-0521629713.

[Eddy, 1996] Eddy, S. R. (1996). Hidden Markov models. *Curr. Opin. Struct. Biol.*, 6(3):361–365. [PubMed:8804822].

[Eddy, 2004] Eddy, S. R. (2004). Where did the BLOSUM62 alignment score matrix come from? *Nat. Biotechnol.*, 22(8):1035–1036. [PubMed:15286655].

[Eddy, 2009] Eddy, S. R. (2009). A new generation of homology search tools based on probabilistic inference. *Genome informatics. International Conference on Genome Informatics*, 23(1):205–211. [PubMed:20180275].

[Eddy, 2011] Eddy, S. R. (2011). Accelerated Profile HMM Searches. *PLoS Comput. Biol.*, 7(10):e1002195. [PubMed:22039361].

[Edgar and Sjolander, 2004] Edgar, R. C. and Sjolander, K. (2004). COACH: profile-profile alignment of protein families using hidden Markov models. *Bioinformatics*, 20(8):1309–1318. [PubMed:14962937].

[Fujita et al., 2011] Fujita, P. A., Rhead, B., Zweig, A. S., Hinrichs, A. S., Karolchik, D., Cline, M. S., Goldman, M., Barber, G. P., Clawson, H., Coelho, A., Diekhans, M., Dreszer, T. R., Giardine, B. M., Harte, R. A., Hillman-Jackson, J., Hsu, F., Kirkup, V., Kuhn, R. M., Learned, K., Li, C. H., Meyer, L. R., Pohl, A., Raney, B. J., Rosenbloom, K. R., Smith, K. E., Haussler, D., and Kent, W. J. (2011). The UCSC Genome Browser database: update 2011. *Nucleic Acids Res.*, 39(Database issue):D876–882. [PubMed:20959295].

[Gotoh, 1982] Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162(3):705–708. [PubMed:7166760].

[GRC, 2009] GRC (2009). Genome Reference Consortium. Online. `http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human/`.

[Hamilton et al., 2006] Hamilton, A. T., Huntley, S., Tran-Gyamfi, M., Baggott, D. M., Gordon, L., and Stubbs, L. (2006). Evolutionary expansion and divergence in the ZNF91 subfamily of primate-specific zinc finger genes. *Genome Res.*, 16(5):584–594. [PubMed:16606703].

[Henikoff and Henikoff, 1991] Henikoff, S. and Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. *Nucleic Acids Res.*, 19(23):6565–6572. [PubMed:1754394].

[Henikoff and Henikoff, 1992] Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.*, 89(22):10915–10919. [PubMed:1438297].

[Huntley et al., 2006] Huntley, S., Baggott, D. M., Hamilton, A. T., Tran-Gyamfi, M., Yang, S., Kim, J., Gordon, L., Branscomb, E., and Stubbs, L. (2006). A comprehensive catalog of human KRAB-associated zinc finger genes: insights into the evolutionary history of a large family of transcriptional repressors. *Genome Res.*, 16(5):669–677. [PubMed:16606702].

[IHGSC, 2004] IHGSC (2004). Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945. [[PubMed:15496913].

[Lodish et al., 2007] Lodish, H., Berk, A., Kaiser, C. A., Krieger, M., Scott, M. P., Bretscher, A., and Ploegh, H. (2007). *Molecular Cell Biology*. 6th edition. New York: W. H. Freeman. 828 p., ISBN: 978-0716776017.

[Looman et al., 2002] Looman, C., Abrink, M., Mark, C., and Hellman, L. (2002). KRAB zinc finger proteins: an analysis of the molecular mechanisms governing their increase in numbers and complexity during evolution. *Mol. Biol. Evol.*, 19(12):2118–2130. [PubMed:12446804].

[Miller et al., 1985] Miller, J., McLachlan, A. D., and Klug, A. (1985). Repetitive zinc-binding domains in the protein transcription factor IIIA from Xenopus oocytes. *EMBO J.*, 4(6):1609–1614. [PubMed:4040853].

[Needleman and Wunsch, 1970] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453. [PubMed:5420325].

[Nowick et al., 2011] Nowick, K., Fields, C., Gernat, T., Caetano-Anolles, D., Kholina, N., and Stubbs, L. (2011). Gain, loss and divergence in primate zinc-finger genes: a rich resource for evolution of gene regulatory differences between species. *PLoS ONE*, 6(6):e21553. [PubMed:21738707].

[Nowick et al., 2010] Nowick, K., Hamilton, A. T., Zhang, H., and Stubbs, L. (2010). Rapid sequence and expression divergence suggest selection for novel function in primate-specific KRAB-ZNF genes. *Molecular Biology and Evolution*, 27(11):2606–2617. [PubMed:20573777].

[Punta et al., 2012] Punta, M., Coggill, P. C., Eberhardt, R. Y., Mistry, J., Tate, J., Boursnell, C., Pang, N., Forslund, K., Ceric, G., Clements, J., Heger, A., Holm, L., Sonnhammer, E. L., Eddy, S. R., Bateman, A., and Finn, R. D. (2012). The Pfam protein families database. *Nucleic Acids Res.*, 40(Database issue):290–301. [PubMed:22127870].

[Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286. [IEEEXPLORE:18626].

[Rice et al., 2000] Rice, P., Longden, I., and Bleasby, A. (2000). EMBOSS: the European Molecular Biology Open Software Suite. *Trends Genet.*, 16(6):276–277. [PubMed:10827456].

[Russell, 2010] Russell, P. J. (2010). *iGenetics: A molecular approach*. Third edition. San Francisco: Pearson Education. 828 p., ISBN: 0-321-61022-9.

[Schmidt and Durrett, 2004] Schmidt, D. and Durrett, R. (2004). Adaptive evolution drives the diversification of zinc-finger binding domains. *Mol. Biol. Evol.*, 21(12):2326–2339. [PubMed:15342798].

[Schuster-Bockler et al., 2004] Schuster-Bockler, B., Schultz, J., and Rahmann, S. (2004). HMM Logos for visualization of protein families. *BMC Bioinformatics*, 5:7. [PubMed:14736340].

[Smith and Waterman, 1981] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, 147(1):195–197. [PubMed:7265238].

[Soding, 2005] Soding, J. (2005). Protein homology detection by HMM-HMM comparison. *Bioinf.*, 21(7):951–960. [PubMed:15531603].

[Thomas and Emerson, 2009] Thomas, J. H. and Emerson, R. O. (2009). Evolution of C2H2-zinc finger genes revisited. *BMC Evol. Biol.*, 9:51. [PubMed:19261184].

[Thompson et al., 2005] Thompson, J. D., Koehl, P., Ripp, R., and Poch, O. (2005). BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61(1):127–136. [DOI:10.1002/prot.20527] [PubMed:16044462].

[UCSC, 2009] UCSC (2009). Human Genome Feb. 2009 (hg19, GRCh37) Pairwise Alignments. Online. `http://hgdownload.cse.ucsc.edu/downloads.html`.

[Urrutia, 2003] Urrutia, R. (2003). KRAB-containing zinc-finger repressor proteins. *Genome Biology*, 4(10):231. [PubMed:14519192].