Probabilistic Arithmetic Automata

Applications of a Stochastic Computational Framework in Biological Sequence Analysis

Inke Herms

PhD thesis submitted to the
Faculty of Technology, Bielefeld University, Germany
for the degree of Dr. rer. nat.

Referees:

Prof. Dr. Jens Stoye, Bielefeld University
Prof. Dr. Sven Rahmann, Dortmund University

August 2009

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706
Printed on non-aging paper according to DIN-ISO 9706

Vinum bonum deorum donum.

ABSTRACT

The immense amount of biological sequence data available these days requires efficient and sensitive analysis in order to provide e.g. the identification of unknown proteins, or information about the similarity between DNA sequences. Furthermore, new challenges to computational sequence analysis are posed by short sequence reads resulting from modern high throughput sequencing technologies such as 454 or Solexa/Illumina.

Viewing biological sequences, such as DNA and proteins, as strings allows their investigation under a generative random string model. That is to say, one can define a probabilistic null model that generates random strings as representatives of a class of sequences. From these, one can deduce general statistical properties.

In this thesis, we give a thorough derivation of a probabilistic model, called *probabilistic arithmetic automaton* (PAA). This models sequences of operations associated to operands depending on chance and provides the computational framework to calculate the exact distribution of the value resulting from those operations. For instance, the PAA framework can be used to compute the expected molecular mass of a peptide resulting from the cleavage reaction of a protease.

Moreover, we show that the framework is sufficiently general to cover completely different applications arising in the computational analysis of biological sequences. To this end, we consider three distinct levels of biosequences, namely 1) amino acid sequences, 2) long DNA sequences and genomes, and 3) short nucleotide sequence reads.

In the first application, protein identification by means of mass spectrometry and database search, we compute characteristical statistics of so-called peptide mass fingerprints to obtain a reasonable, database-independent significance value for the identification of an unknown protein. Going one step further than recent approaches, we additionally incorporate post-translational modifications and incomplete enzymatic digestion that alter the measured molecular masses and, hence, may influence the search results.

The second application arises from the context of DNA similarity search. We use the PAA framework to investigate the quality of filtration criteria employed to select candidate sequences from a comprehensive nucleotide sequence database. The PAA we propose comprises recent models and provides additional statistics. This allows us to investigate different definitions of optimality not discussed formerly.

Searching for similar DNA sequences, which provides the basis for comparative genomics in general, was enabled by the growing amount of nucleotide sequences stored in sequence databases. This development was accelerated by high throughput sequencing strategies such as 454 sequencing, that allow for faster sequencing at reduced price. However, these technologies yield relatively short reads of sequenced nucleotides, which poses new challenges to genome assembly tools. By means of the PAA approach, we compute the length distribution of sequence reads resulting from 454 sequencing. Moreover, we discuss how to adjust the machine settings to obtain on average the longest reads possible. The designed PAA is used for evaluation.

Besides the PAA framework and its applications, we present a biologically motivated random string model adjusted to protein sequences, referred to as SSE model. It captures properties of local segments forming protein secondary structures. In order to evaluate the model's capability, we compare four random string models by means of penalized model selection criteria. We show that among these models, the SSE model yields the most plausible description of considered protein sequences, outperforming the widely used i.i.d. and first-order Markov model.

ACKNOWLEDGMENTS

There was a time when I thought that I would never reach this point and write the last parts of this thesis. But, as you see, I do so and I am really glad that I succeeded (at least) in this respect. It is an encouraging feeling to fill these pages. After all, I really enjoyed writing up the work I did during the last three years. Before I finally close, I want to thank all the lovely people that made those years unforgettable and that supported me in any respect.

First and foremost, I want to thank Jens Stoye for his encouragement, for his aid and advisable discussions both from the academic and the personal side. Thanks also to Sven Rahmann who initialized this project and who pointed out valuable directions.

With Epameinondas Fritzilas (Nondas), José Augusto Quitzau (Zé), and Eyla Willing, there was always an especially good mood in our office. Besides concentrated work, we could talk, laugh, and discuss. Further, I will remember many great coffee breaks, cocktail parties, games evenings, and dinner with Nondas Fritzilas, Peter Husemann, Katharina Jahn, Zé Quitzau, Eyla Willing, and Roland Wittler (in alphabetic order). I especially appreciated the amiable gestures and the last-minute support of Eyla and Roland. Thank you all for your friendship and the marvelous time in Bielefeld.

Moreover, I thank Heike Samuel for her advice in all the administrative issues and particularly for being so interested and amiable. Thanks also to Susanne Schneiker-Bekel for providing me with biological data and for making the contact to Rafael Szczepanowski, whom I thank for his endurance in answering my questions about 454 sequencing.

I also acknowledge the financial and academic support of the International NRW Graduate School in Bioinformatics and Genome Research.

Further, the cooperation with Stefan Wolfsheimer was a great experience, and I am pleased that we still intercommunicate.

I want to thank Zsuzsanna Lipták (Zsuzsa), Roland Wittler, Nondas Fritzilas, Katharina Jahn, and Peter Husemann for proofreading parts of the manuscript and giving valuable comments. In particular, I am very grateful to Zsuzsa, who

gave me various advice in academic and personal concerns.
Surely, it is my pleasure to thank my parents, Sigrid and Burkhard, who supported me in my studies, who taught me to be ambitious and to conclude things I began, who encouraged me to carry out my various interests and who give me the feeling to be proud of me. Thank you both so much.
Last but not least, I am very grateful to have a wonderful husband and a lovely daughter, who always knew when to encourage me and when to distract me. Robert, Alicia, I love you.

CONTENTS

CHAPTER 1

INTRODUCTION

In the 1970s, DNA sequencing emerged, and the first sequences were discovered by means of laborious methods. With the advent of shotgun sequencing, the first bacterial genome, *Haemophilus influenzae*, was published in 1995. Since then, large-scale DNA sequencing efforts led to an exponential increase in the amount of sequence data stored in sequence databases. In particular, sequencing projects such as the Human Genome Project (yielding 3.3 billion base pairs of the entire human genome) produced an immense amount of data, which made automatic, computational methods necessary. Today, comprehensive databases contain millions of sequences comprising billions of nucleotides.

Besides genomic sequences, experimental technologies such as microarrays or mass spectrometry additionally yield a host of data that has to be stored, processed, analyzed, visualized and compared. Arising from these needs, the discipline of computational molecular biology mainly relies on quantitative methods in order to interpret the large amount of data.

Many problems faced are essentially statistical due to the stochasticity inherent to biological systems, the random processes responsible for evolution, and errors intrinsic to measurements from large-scale experiments. Hence, most sophisticated methods to analyze biological sequence data are based on *probabilistic modeling*. Generally, a model refers to a system that simulates the object under study. A probabilistic model generates a whole class of objects, assigning each an associated probability. It is always understood as a reference, a so-called *null model*, describing representatives of a considered class of objects. Here, the objects correspond to biological sequences and, hence, a probabilistic model might for instance generate a family of related sequences.

Examples of probabilistic methods employed in computational sequence analysis include the inference of phylogenetic trees using maximum likelihood approaches,

the computation of the significance of a sequence alignment based on probabilistically derived scores, and the classification of proteins or protein domains by means of hidden Markov models to sequence families.

A hidden Markov model (HMM) is a powerful probabilistic method capable of heterogeneities in the considered objects. It generates a sequence of observations determined by a sequence of "hidden" states. That is to say, there is no one-to-one correspondence between states and observations since different states can lead to the same observation. HMMs are for instance used to find CpG islands, i.e. DNA stretches with high abundancy of CG dinucleotides.

In this thesis, we investigate the practicability of a novel probabilistic model extending HMMs, focussed on problems from computational sequence analysis. The method carries along a computational framework that provides the straightforward computation of statistics associated to biological sequences such as DNA and proteins, or alignments thereof. Furthermore, we design a generative model for protein sequences. For the general purpose of probabilistic modeling, the considered sequences are abstracted to strings. We therefore have a brief look at the biological background of DNA and proteins and their string representation. More details can be found in various textbooks, e.g. in [129].

DNA: *Deoxyribonucleic acid* (DNA) is a double-stranded macromolecule (double helix) that carries the genetic information of a cell. This is to say, it contains the instructions necessary to construct other components of cells, such as proteins and RNA molecules. Each strand consists of nucleotides, where the backbone is formed by repeating units of sugar (deoxyribose) and phosphate, joined by ester bonds, and a nucleotide base is attached to each sugar. The DNA bases are adenine (A), cytosine (C), guanine (G), and thymine (T). The two strands are antiparallel and stabilized by complementary base pairings between A and T as well as between C and G. The sequence of bases along the backbone encodes the genetic information.
Thus, a DNA sequence is represented by the order of nucleotide bases on one strand, denoted by the upper-case letters A, C, G, and T.

The information in the DNA is transcribed into RNA and (after processing steps) further translated into proteins. This indirect information transfer is known as the central dogma of molecular genetics.

Proteins: As the name *protein*, deduced from *protos* (greek ($\pi\rho\tilde{\omega}\tau o\varsigma$)= primary), suggests, *proteins* are essential macromolecules of all organisms and play a central role in almost all processes within cells. For instance, they take part in the storage or transport of substances as oxygen or other molecules, they are important in signal transduction and immune response, have structural or mechanical functions, or regulate metabolic reactions in the form of enzymes. Proteins are polymers of amino acids, referred to as *polypeptides*. Amino acids are organic molecules carrying a carboxyl (COOH) and an amino (NH_3) group. The 20 amino acids differ only in the variable side chain. Corresponding chemical groups determine the physical and chemical properties of the amino acids. When the carboxyl

group of one amino acid is arranged next to the amino group of another amino acid, they can be covalently linked by a condensation reaction. This is to say that a water molecule is split of thus forming a so-called *peptide bond*. A polypeptide is read from the N-terminus (the free amino group) to the C-terminus (the free carboxyl group). The repetitive succession of nitrogen and carbon atoms -N-C-C-N-C-C- along the polypeptide chain is called the peptide backbone. Hence, a polypeptide is a polymer consisting of many amino acids linked by peptide bonds. Each polypeptide is uniquely determined by the sequence of amino acid residues (the amino acid molecules without terminal OH and H split during the condensation reaction). In fact, a protein consists of one or more polypeptides that are folded and coiled in a specific way. This conformation determines the structure and the function of the protein. We refer to this in Chapter 4.
The term "protein sequence" refers to the sequence of amino acids. Hence, in a generic way, a protein sequence is represented as sequence of upper-case letters denoting the amino acids.

1.1 Notational Conventions

Throughout this thesis, we use the following notations. Let Σ denote a finite alphabet, i.e. a finite, non-empty set of discriminative characters, and let the set Σ^* include all finite words, called *strings*, over Σ. It also includes ϵ, the empty string. Any subset of Σ^* is called a *formal language* over Σ. The set Σ^l contains all strings of length l, i.e. strings $s = s[1]s[2]\dots s[l]$ with $|s| = l$. Here, $s[i]$ denotes the character at position i, and $s[i,j]$ refers to the substring of s ranging from position i to j (inclusive). For convenience, we define $s[i,j] = \epsilon$ for the cases $i > j$, $i < 1$, and $j > |s|$. Further, $\Sigma^+ := \Sigma^* \backslash \{\epsilon\}$ is the set of all non-empty words over Σ. The concatenation of objects r and s is written as rs, where "object" refers to characters, strings, or sets, respectively. For two sets A and B, the concatenation is defined as $AB = \{ab \,|\, a \in A, b \in B\}$. The cardinality of a set C is denoted by $|C|$.
Moreover, for two strings $r, s \in \Sigma^*$, $r \lhd s$ indicates that r is a prefix of s, while $r \blacktriangleleft s$ says that r is a suffix of s. In particular, we abbreviate the sets of all prefixes and suffixes (including ϵ) of a string s by $\mathrm{Prefixes}(s) = \{s[1,i] \,|\, 0 \leq i \leq |s|\}$ and $\mathrm{Suffixes}(s) = \{s[i,|s|] \,|\, 0 \leq i \leq |s|\}$, respectively.
Further, $\mathcal{L}(X)$ denotes the probability distribution of a random variable X with corresponding generic probability measure $\mathbb{P}$. If X is continuous, $\mathbb{P}$ defines the probability density function (pdf), otherwise if X is a discrete random variable, $\mathbb{P}$ refers to the probability mass function (pmf).
Moreover, $A \subseteq B$ includes the case $A = B$, whereas $A \subset B$ does not. The set $\mathbb{N}$ denotes all positive natural numbers, i.e. $\mathbb{N} = \{1, 2, 3, \dots\}$, while we write $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ for all non-negative integers. For a set $A \subseteq \Sigma$, we write $\bar{A} := \Sigma \backslash A$ to denote the complementary set of A with respect to the superset Σ.

1.2 Organization of the Thesis

This work aims at the thorough treatment of a general, stochastic computational framework called *probabilistic arithmetic automaton* (PAA). This extension of the widely used HMM formalizes a sequence of binary operations where the operands depend on chance. It naturally provides recurrence relations to compute the probability distribution of the value resulting of a given number of operations. We discuss three different applications arising in biological sequence analysis, and, thus, demonstrate the flexibility and applicability of the PAA framework. The problems considered here deal with nucleotide and protein sequences in the course of different genomic and proteomic studies. Connected to modern technologies such as tandem mass spectrometry and high throughput sequencing, the investigated questions pose important tasks in computational biology. Furthermore, analyzing stochastic models for protein sequences inspired us to design a novel, biologically motivated model for amino acid sequences.

The thesis is organized as follows: we dedicate Chapter 2 to the introduction of the probabilistic arithmetic automaton. This probabilistic method to model sequences of operations whose operands depend on chance is derived from the concept of probabilistic automata, augmented with a stochastic process performing arithmetic operations. We recall important characteristics of probabilistic automata, give the formal definition of a PAA following Marschall and Rahmann [84] and an introductory example from gambling. Then, we define recurrence relations related to the PAA framework and state how to efficiently implement the corresponding computations. An applications' overview closes the chapter.

The first application of the PAA framework to computational biology we consider in this thesis relates to peptide mass fingerprinting (PMF), a method for protein identification based on mass spectrometry (MS). Our research continues the thesis of Kaltenbach [54] as we compute characteristics of peptide mass fingerprints derived from random protein sequences. Chapter 3 thus starts with a brief introduction to MS and PMF and a recall of necessary definitions adopted from [54], before we state an appropriate PAA to compute peptide statistics. Furthermore, we set up a PAA modified to incorporate site specific post-translational modifications and account for peptides that are not entirely fragmented. Finally, we compare our results to empirical data, obtained from the Swissprot database, and to the former work.

The application mentioned above motivates the need for a more plausible (string) representation of proteins and peptides. Hence, in Chapter 4, we present the SSE random string model that incorporates information from secondary structure and is thus adjusted to protein sequences. We describe how the model is designed and which parameters are suited for it. To evaluate the quality of the model, we use so-called penalized model selection criteria. We introduce selected criteria, define necessary calculations related to the SSE model, and compare different string models according to the criteria.

Chapter 5 deals with another application of the PAA framework. In local alignment search tools, a filtering criterion, a so-called seed, is applied to select candidate sequences from a database. We use the PAA framework to compute the quality of a given seed. To this end, we give a background in so-called seeded alignment methods and seed sensitivity, the common measure of filtering quality. We present an appropriate PAA, comprising the majority of recent models, and the corresponding recurrence relations. Then, we compare our results to existing methods. Additionally, we discuss alternative optimality criteria and compare them to sensitivity.

In Chapter 6, we consider the third application of the PAA framework. It deals with an up-to-date high throughput sequencing technology, the 454 sequencing, and is the most recent application. After a short introduction to customary and high throughput DNA sequencing in general, we explain 454 sequencing in particular. The latter technology washes deoxynucleotides across a plate with single-stranded DNA probes and measures the amount of nucleotides synthesized by the DNA polymerase at each washing step. We present a PAA that computes, for a given order of repetitive nucleotide washes, the length distribution of sequence reads generated in the course of a given number of washes. Moreover, we discuss a strategy to find the optimal order in which nucleotides should be washed over the plate in order to generate on average the longest reads possible.

Finally, in Chapter 7 we summarize our work and close with an outlook for subsequent research.

Contributions

This thesis yields a thorough treatment of probabilistic arithmetic automata and demonstrates their applicability to various problems in computational biology. The individually designed methods give rise to unifying frameworks and allow for the integration and investigation of issues not treated so far. Besides, we systematically analyzed the settings of the 454 sequencing technology giving hints how to improve the platforms. Furthermore, to the best of our knowledge, we are the first to present a generative random string model for protein sequences incorporating secondary structure information.

Publications

The contents of Chapter 5 have been published in the refereed conference proceedings of the Workshop on Algorithms in Bioinformatics (WABI) 2008 [48]. Further, a poster about the application to peptide statistics was presented at the Summer School of Proteomics Basics in Brixen, Italy, 2006. Moreover, the joint work with Stefan Wolfsheimer on alignment score statistics using arbitrary null models, in particular hybrid models for transmembrane proteins, is under revision in a refereed journal.

Two further papers are in preparation. The first presents the SSE model prepared in Chapter 4. The second is devoted to the contents of Chapter 6.

Software

The algorithms and methods described in this thesis were implemented in Java, Mathematica, and Perl. Source code is available from the author upon request.

CHAPTER 2

PROBABILISTIC ARITHMETIC AUTOMATA

This chapter is dedicated to the introduction of the central object of this thesis, namely the *probabilistic arithmetic automaton* (PAA). This is a stochastic computational framework extending ordinary HMMs by the insertion of a sequence of (arithmetic) operations performed on state emissions. A similar concept has previously been referred to as *weighted HMM* (wHMM) introduced by Kaltenbach [54]. There, the accumulation of emissions is described by a separate formal model, called *Markov additive chain* (MAC) pursuant to Ney and Nummelin [93]. However, the definition using MACs is quite restrictive, since it only allows for adding up the states' emissions. Though this may suffice for many applications from computational biology, we use a more general definition comprising wHMM/MAC following Marschall and Rahmann [84], which shows the flexibility of the model. The application of probabilistic automata and the PAA framework in particular to the context of pattern matching statistics has been investigated by Tobias Marschall in his diploma thesis [83], supervised by the author of this thesis and Sven Rahmann. Here, we give a compressed derivation of *probabilistic pattern matching* in Section 2.1 integrating results of Lladser *et al.* [74]. Next, in Section 2.2, we recall the general definition of the PAA framework. Section 2.3 illustrates the method using a toy example from gambling. In Section 2.4, we present the algorithm PAADIST to compute the value distribution of a PAA and discuss its performance. Finally, we give a foretaste of applications in Section 2.5.

2.1 Introduction to Probabilistic Automata

In applications from molecular biology, one often searches for patterns (motifs) that are "overrepresented" within a given sequence, i.e. patterns that occur more often than expected as a substring of the respective sequence. To this end, one has

to know how many occurrences are expected on average. For even more precise statements, one can consult the statistical significance: Suppose, we have found a pattern n times within a sequence of length t. What is the probability that it occurs at least n times just "by chance"? To specify the notion of "by chance", we consider a (generative) *random text* of length t generated according to an appropriate null model.

Definition 2.1.1 (Random text)**.** *A* random text*, also referred to as* random string*, over a finite alphabet Σ is a stochastic process S with index set $\mathbb{N}$ taking values in Σ. Its finite-dimensional distribution is given by the joint distribution of the constituting characters:*

$$\mathcal{L}(S_I) = \mathcal{L}(S_{i_1}, \ldots, S_{i_t})$$

for all finite index sets $I = \{i_1, \ldots, i_t\} \subset \mathbb{N}$.

Asking for the probability to observe at least n occurrences of the pattern within a random text is referred to as probabilistic pattern matching, since the text to be analyzed is not fixed as in deterministic pattern matching, but rather generated by a random process. Typically, a biological sequence is assumed to be produced by a memoryless i.i.d. source or a Markovian source, or sometimes by a hidden Markov model (HMM), when heterogeneities should be included. We recall the prevalent random string models and refer to the textbooks of Durbin *et al.* [38], Karlin and Taylor [56], Robin *et al.* [105] and the article of Rabiner [101] for a general introduction to Markov chains and HMMs.

Example 2.1.2 (The i.i.d. random string model)**.** *The* i.i.d. model *is a generative random string model, where each character is assumed to be chosen independently of all other characters from the same distribution (independently and identically distributed). The finite-dimensional distributions are then given by the product measure. Hence, for a string S_I with given index set $I = \{1, \ldots, t\}$, the probability is given by the product of the characters' probabilities:*

$$\mathbb{P}(S_I = s[1,t]) = \prod_{i=1}^{t} \mathbb{P}(S_i = s[i])\,.$$

Example 2.1.3 (The k^{th}-order homogeneous Markov model)**.** *A k^{th}-order* homogeneous Markov model *is a generative random string model taking dependencies into account. This is to say, the probability of a character depends on the k previous positions. For the specification of a string's probability, one needs an initial distribution for the first character as well as conditional character distributions for the subsequent positions. The most prominent representative of this class of models is the first-order homogeneous Markov model (M1). The probability of a string S_I for $I = \{1, \ldots, t\}$ under M1 is given by*

$$\mathbb{P}(S_I = s[1,t]) = \mathbb{P}(S_1 = s[1]) \prod_{i=1}^{t-1} \mathbb{P}(S_{i+1} = s[i+1] \,|\, S_i = s[i])\,.$$

For a homogeneous Markov chain, the conditional character distributions are identical for all positions i.

A (simple) *pattern* is a string where each position in the string is exactly specified by one letter from the alphabet. However, there are more involved patterns. For instance, a *compound pattern* refers to a finite set of simple patterns. Further, a *regular pattern* can be described by a regular expression. For the sake of simplicity, from now on we write "pattern" synonymous for "set of patterns". A single pattern then corresponds to a singleton.

Previous work on probabilistic pattern matching dealt with different problems.

i) Counting: what is the probability that a pattern occurs n times within a random text of length t?
ii) Occurrence: what is the probability that a pattern occurs or does not occur in a random text of length t? This is a special case of i).
iii) Occurrence position: what is the probability that a pattern is first observed at a certain position within the text?
iv) Distance between occurrences: what is the expected distance between two successive occurrences of a pattern in a random text of length t?

Additionally, when concerned with counting all occurrences of a pattern or calculating the distance between successive occurrences, one has to specify how to handle overlaps. To this end, we distinguish three different count statistics:

- The *overlapping occurrence count*: all substrings of the text which correspond to a word specified by the pattern are counted.
- The *non-overlapping occurrence count* only non-overlapping substrings which correspond to a word specified by the pattern are counted.
- The *occurrence position count*: all positions in the text where at least one word specified by the pattern ends are counted.

Let $I_k(s, w)$ be the indicator variable defining whether the word w ends at position k in a text s, i.e.

$$I_k(s, w) = \mathbb{1}_{\{s[k-|w|+1,k]=w\}} .$$

Further, $I_k(s, W)$ induces the indicator on a set W of words, i.e.

$$I_k(s, W) = \mathbb{1}_{\{\{w \in W \,|\, s[k-|w|+1,k]=w\} \neq \emptyset\}} .$$

Herewith, we define the different occurrence counts separately:

Definition 2.1.4 (Overlapping occurrence count)**.** *Let $W \in \mathcal{P}(\Sigma^+)$ be a set of patterns and $s \in \Sigma^*$ be a text over Σ. The* overlapping occurrence count *of W in s is defined as*

$$N_o(s, W) = \sum_{i=1}^{|s|} \sum_{w \in W} I_i(s, w) .$$

Note that we use the ending positions to indicate a match. In order to count only non-overlapping occurrences of all patterns in W, all matches of a pattern to the text have to be identified first. To this end, we define the matching set.

Definition 2.1.5 (Matching set)**.** *Let $W \in \mathcal{P}(\Sigma^+)$ be a set of patterns and $s \in \Sigma^*$ be a text over Σ.*

$$M_W(s) = \{(i,j) \in \mathbb{N}_0 \times \mathbb{N}_0 \,|\, \exists\, w \in W \,:\, s[i,j] = w\}$$

contains all substrings of s matching any word from W and is called the matching set *of W in s.*

A set $M \subset M_W(s)$ is *non-overlapping* if all intervals $(i_1, j_1), (i_2, j_2) \in M$ with $(i_1, j_1) \neq (i_2, j_2)$ are disjoint. Further, a non-overlapping set M is called *maximal* if there is no non-overlapping matching set M' with $|M'| > |M|$, i.e. with more non-overlapping matches.

Definition 2.1.6 (Non-overlapping occurrence count)**.** *Let $W \in \mathcal{P}(\Sigma^+)$ be a set of patterns and $s \in \Sigma^*$ be a text over Σ.*

$$N_n(s, W) = |M| \quad \textit{where } M \subset M_W(s) \textit{ is non-overlapping and maximal}$$

is called the non-overlapping occurrence count *of W in s.*

The occurrence position count is defined analogously to the overlapping occurrence count. In particular, it agrees with the overlapping count unless some patterns in the set are suffixes of others.

Definition 2.1.7 (Occurrence position count)**.** *Let $W \in \mathcal{P}(\Sigma^+)$ be a set of patterns and $s \in \Sigma^*$ be a text over Σ.*

$$N_p(s, W) = \sum_{i=1}^{|s|} I_i(s, W)\,.$$

is called the occurrence position count *of W in s.*

We now discuss methods from automata theory instrumental in deriving pattern matching statistics.

2.1.1 Deterministic Finite Automata

The questions i) to iv) mentioned above can be answered by means of *probabilistic automata*. The general idea is to extend the use of deterministic finite automata (DFAs) for classical pattern matching problems such that the resulting probabilistic automata quantify probabilities of occurrences of strings in random texts. Let us therefore give a review on DFAs and classical pattern matching, before we state the probabilistic version.

Definition 2.1.8 (Deterministic Finite Automaton (DFA))**.** *A* deterministic finite automaton *is a 5-tuple of the form $G = (Q, \Sigma, \Delta, q_0, F)$, where*

- *Q is a finite set whose elements are called* states*.*

- Σ *is a finite alphabet.*
- $\Delta : Q \times \Sigma \to Q$ *is a transition function determining the unique next state depending on the current state and the character read.*
- $q_0 \in Q$ *is called* start state*.*
- $F \subset Q$ *is a (possibly empty) subset of states. It is called the set of* final states*.*

G can be represented as a graph with vertex set Q and directed edges according to Δ. In other words, there is a directed edge labeled with character $\sigma \in \Sigma$ connecting vertex u to vertex v if $\Delta(u, \sigma) = v$. In particular, for all $u \in Q$ and $\sigma \in \Sigma$, there exists a unique edge labeled with σ that starts in u. The property that in each state the next state is entirely determined by Σ and Δ, is called determinism. In contrast, in a non-deterministic finite automaton, the transition function is replaced by $\Delta' : Q \times \Sigma \to \mathcal{P}(Q)$ which maps the current state to a set of subsequent states.

Outgoing from the start state q_0, a DFA reads a given string over the alphabet Σ character by character. Based on the current state, the next state is determined by means of the transition function Δ. We extend this function to a transition function operating on strings by $\Delta^* : Q \times \Sigma^* \to Q$ with

$$\Delta^*(q, s) = \begin{cases} q & \text{if } s = \epsilon, \\ \Delta^*\big(\Delta(q_0, s[1]), s[2, |s|]\big) & \text{otherwise.} \end{cases}$$

Consequently, the following recursion holds:

$$\Delta^*(q, st) = \Delta^*\big(\Delta^*(q, s), t\big)\,. \tag{2.1.1}$$

Those strings that correspond to a concatenation of edge labels on a path that starts in q_0 and ends in a final state are said to be *accepted* by G. The set

$$L(G) = \{s \in \Sigma^* \mid \Delta^*(q_0, s) \in F\} \tag{2.1.2}$$

of these accepted strings is called the *language* recognized by G. The following definition connects this to the notion of regular languages.

Definition 2.1.9 (Regular language)**.** *A language $L \subset \Sigma^*$ is called* regular *if and only if there exists a DFA $G = (Q, \Sigma, \Delta, q_0, F)$ such that $L = L(G)$.*

Remark 2.1.10. *Note that a regular language, although induced by a finite automaton, may consist of infinitely many strings. Moreover, each finite set of strings is a regular language.*

To conclude this section about deterministic finite automata, observe that there exists more than one DFA accepting the same language. For instance, Figure 2.1 shows two DFAs accepting $A = \big\{\{a, b\}a^*b\big\}$.

DFAs that recognize the same language are called equivalent. Obviously, among equivalent DFAs, it is preferable to operate on this one with the minimal number of states. According to Myhill [90] and Nerode [92], the number of states in a *minimal* automaton equivalent to $G = (Q, \Sigma, \Delta, q_0, F)$ corresponds to the number of equivalence classes in Q. The according equivalence relation on Q is defined as follows:

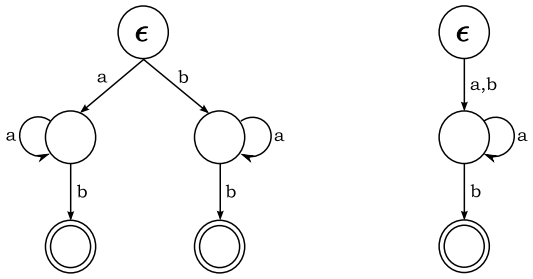


Figure 2.1: Two different DFAs accepting the language $L = \{\{a,b\}a^*b\}$. The initial state is called ϵ, final states are represented by two concentric circles. The automaton to the right is minimal.

Definition 2.1.11 (Equivalence of states)**.** *Let $G = (Q, \Sigma, \Delta, q_0, F)$ be a DFA. Two states $u, v \in Q$ are called* equivalent *if*

$$\Delta^*(u, s) \in F \Leftrightarrow \Delta^*(v, s) \in F \text{ for all } s \in \Sigma^* .$$

A minimal automaton is defined up to isomorphism (refer to Kozen [62]). For a given automaton, the minimal counterpart is obtained merging equivalent states into a single state. The asymptotically fastest minimization algorithm is due to Hopcroft [50]. It runs in $\mathcal{O}(|\Sigma| \cdot n \log n)$ time, where n is the number of states in the given automaton. For a detailed discussion of the algorithm, its implementation and an in-depth analysis, the reader is referred to Knuutila [59].

Here, we give a brief sketch:

1. Separate the final states F from the non-final states $Q \backslash F$.
2. Repeat the following until the partition is not modified anymore:
   Separate any two states which have a transition with the same label to two states which are separated, i.e. that are in different parts of the partition.
3. The remaining partition defines the states of a minimal automaton.

The algorithm starts from an initial partition of the set of states into final states and non-final states, because a final state is not equivalent to any non-final state. In the course of the algorithm, this partition is refined step by step, such that at the end two states are in the same part of the partition if and only if they are equivalent. To this end, all parts are investigated for all characters $\sigma \in \Sigma$ such that a part has to be split if two of its states have an outgoing transition labeled with the same character leading to different parts of the partition.

2.1.2 Pattern Matching

Important work on the subject of pattern matching with finite automata has been presented by Aho and Corasick [2]. They defined the so-called *Aho-Corasick automaton* to detect all occurrences of a finite set W of strings, called keywords,

within a general text. To state the definition, we denote the prefixes and suffixes of a string s by $\mathrm{Prefixes}(s)$ and $\mathrm{Suffixes}(s)$, as defined before. Note that these sets also include the empty prefix and suffix, respectively.

Aho-Corasick Automata

Definition 2.1.12 (Aho-Corasick Automaton (ACA))**.** *An* Aho-Corasick automaton *for a finite set $W \subset \Sigma^+$ of words is a 5-tuple of the form $(Q, \Sigma, \Delta, \Phi, \Theta)$ where*

- $Q = \bigcup_{w \in W} \mathrm{Prefixes}(w)$ *is the set of all prefixes of all keywords.*
- Σ *is a finite alphabet.*
- *The* goto *transition function $\Delta : Q \times \Sigma \to Q \cup \{\star\}$ (with $\star \notin \Sigma$ and hence, $\star \notin Q$) is defined such that*

$$\Delta(q, \sigma) = \begin{cases} q\sigma & \text{if } q\sigma \in Q, \\ \epsilon & \text{if } q = \epsilon \text{ and } \sigma \notin Q, \\ \star & \text{otherwise.} \end{cases}$$

  From the current state q, we follow the transition function to the state $q\sigma$ if it exists, i.e. if any keyword starts with $q\sigma$. Otherwise, the assignment of $\star$ indicates, that we follow a failure link, defined next.
- *The* failure *function $\Phi : Q \to Q$ links a state to its longest suffix which is also a prefix in the ACA. It is defined as*

$$\Phi(q) = \operatorname*{argmax}_{s \in Q \cap \mathrm{Suffixes}(q) \setminus \{q\}} |s| .$$

  This means, whenever a word $q\sigma$ is not a prefix of any keyword and thus would not lead to a match of W, we follow the failure link in order to find the longest suffix of q that, reading σ, may still lead to a hit.
- *The* output *function $\Theta : Q \to \mathcal{P}(W)$ is defined as*

$$\Theta(q) = W \cap \mathrm{Suffixes}(q) .$$

  It displays all keywords ending in state q.

See Figure 2.2 (left) for an example of an ACA for the set of keywords $W = \{\text{bin,bit,in,int,it}\}$. To construct the Aho-Corasick automaton for a finite set of keywords W, denoted ACA(W), we have to build the prefix trie of the given keywords and augment it with the failure links. The failure function guarantees an efficient search for matches. Instead of generally resuming the search when a word $q\sigma$ does not lead to a match, we follow the failure link in order to "remember" the longest already read part that still can be extended to a match. Consider state 6 in the example. If we have read *bin*, then the only character that will lead to another match is t. To this end, the failure link directs us to state 4, which is to

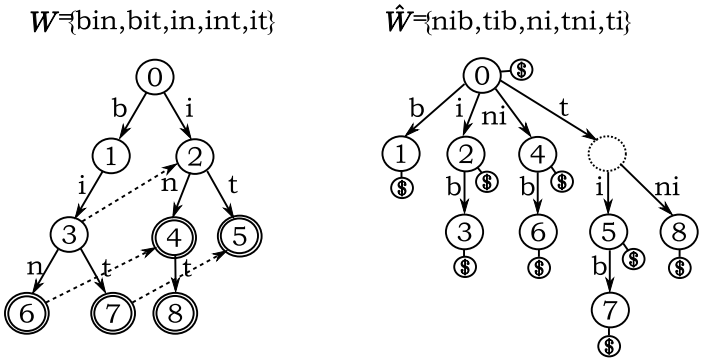


Figure 2.2: To the left, the Aho-Corasick automaton for the set $W = \{\text{bin,bit,in,int,it}\}$ of keywords is shown. It corresponds to the prefix trie of W. Concentric circles indicate $w \in W$. Failure links are displayed as dashed arrows, where we omitted failure links to ϵ, i.e. to vertex 0. The numbers serve as orientation for the connection between failure function and direct descent in the suffix tree of the reversed keywords $\hat{W}$, shown to the right. The longest suffix of a string that is also a prefix of any keyword refers to the longest prefix in the mentioned suffix tree that is also a suffix of any reversed keyword. The dashed node in the suffix tree does not correspond to any node in the ACA.

say that *in* is the longest suffix that can be extended to *int* and hence indicate a further match. If the next character read differs from t, the failure link guides us from state 4 to state 0, i.e. there is no keyword starting with *nt* or *t*, and the longest suffix refers to ϵ.

The construction of ACA(W) takes time proportional to the total length of keywords, i.e. $\mathcal{O}(\sum_{w \in W} |w|)$, for integer alphabets. Such a linear-time construction algorithm can be found in the original paper [2], whereas recently, Dori and Landau [36] presented another nice algorithm with the same running time. They make use of the connection between failure function and direct descent in the suffix tree of the reversed patterns $\hat{W}$, which is demonstrated in Figure 2.2 (right). The failure function assigns a state its longest suffix which is also a prefix of any keyword. In the suffix tree of the reversed keywords this corresponds to the closest ancestor of a state, marked by $\$$, since this is the largest element which is a prefix of the state's label and also itself a suffix of a reversed keyword. We will get back to this relation later in Section 5.3.

Definition 2.1.13. *The DFA $G = (Q, \Sigma, \Delta, \epsilon, W)$ with transition function Δ such that*

$$\Delta(u, \sigma) = \operatorname{argmax}_{x \in \text{Suffixes}(u\sigma) \cap Q} |x| \quad \textit{for all } (u, \sigma) \in Q \times \Sigma \tag{2.1.3}$$

defines the DFA underlying ACA(W).

The longest prefix-suffix property leads us to the following lemma. It states that in an ACA, a path from ϵ to u is labeled with string s if and only if u is the longest suffix of s which is also a prefix of some keyword $w \in W$:

Lemma 2.1.14 (Aho and Corasick [2])**.** *Let $G = (Q, \Sigma, \Delta, \epsilon, W)$ be the DFA underlying ACA(W). For all $s \in \Sigma^*$, $\Delta^*(\epsilon, s) = u$ if and only if $u = \operatorname{argmax}_{x \in \text{Suffixes}(s) \cap Q} |x|$.*

For the proof, we follow Lladser *et al.* [74], Lemma 4.3.

Proof. Induction on the length $|s|$:
The base for $|s| = 0$ is trivial, since for $s = \epsilon$, $\Delta(\epsilon, \epsilon) = \epsilon$ by definition, and $\mathrm{Suffixes}(\epsilon) = \{\epsilon\}$. Thus, $u = \epsilon$. We show that if $|s| = n + 1$ and the lemma holds for all strings of length n, then it also holds for s:
Let u be the longest element in Q such that $u \blacktriangleleft s$. Further, let $\Delta^*(\epsilon, s[1, n]) = v$ and $\Delta^*(\epsilon, s) = w$. Observe that $\Delta^*(v, s[n+1]) = w$ according to Equation (2.1.1). Further, v is the longest element in Q with $v \blacktriangleleft s[1, n]$ according to the induction hypothesis. This implies $w \blacktriangleleft s$, because

$$w = vs[n + 1] \blacktriangleleft s[1, n]s[n] = s\,. \tag{2.1.4}$$

To prove the lemma, it suffices to show that $u = w$. To this end, observe that the following conditions hold:

(i) $|w| \leq |u| \leq |v| + 1$: the first inequality directly follows from the definition of u. We show the second inequality by contradiction: suppose $|u| > |v| + 1$. Since u is the longest element in Q with $u \blacktriangleleft s$ and $vs[n] \blacktriangleleft s$, there would be a non-empty string y such that $u = yvs[n + 1]$. Then, since $u \in Q$, yv must be a prefix of a word in W and hence, $yv \in Q$. This would imply $yv \blacktriangleleft s[1, n]$, which contradicts the defining property of v. Thus, the second inequality follows.

(ii) $|w| \geq |u|$: since $u \blacktriangleleft s$ and $w = vs[n + 1] \blacktriangleleft s$ according to Equation (2.1.4), the second inequality in (i) implies $u \blacktriangleleft vs[n + 1] = w$ and hence, $|u| \leq |w|$.

It follows that $|w| = |u|$. Moreover, $u \blacktriangleleft w$, and thus, $u = w$. □

This lemma shows that ACA(W) recognizes the language Σ^*W, i.e. all words over Σ ending with a pattern from W.

Once constructed, ACA(W) can be used to search a text s for all matches to words specified by W. The appropriate pattern matching algorithm is described in Algorithm 2.1. It reads a given text s character by character, following the edge labeled with the current character in order to determine the next state of the automaton. In each state q, the output function Θ is consulted and if there is any keyword ending in q, $\Theta(q)$ is output. If there is no edge labeled with the currently read character, the algorithm follows the failure link defined by Φ and investigates the respective state.

The pattern matching algorithm ACA-MATCH reports all substrings of the searched text matching any word from the set of given keywords (line 9). Speaking figuratively, it corresponds to a sliding window approach with a window of flexible length. The size of the window is extended (line 7) as long as a potential match can be found. If a character enters the window such that it contradicts a potential match (line 4), the window is shifted (and hence possibly shortened) to the first position that refers to a potential starting position of any keyword. In contrast to the naive approach, here the number of shifted positions may be larger than one.

Algorithm 2.1 ACA-MATCH(K)

Input: ACA $K = (Q, \Sigma, \Delta, \Phi, \Theta)$ corresponding to W, string $s \in \Sigma^+$
Output: Substrings of s matching any word from W

```
1: state ← ϵ
2: for i = 1 to |s| do
3:    σ ← s[i]
4:    while Δ(state, σ) = ⋆ do
5:       state ← Φ(state)        //follow the failure function
6:    end while
7:    state ← Δ(state, σ)
8:    if Θ(state) ≠ ∅ then
9:       report Θ(state)
10:   end if
11: end for
```

ACA-MATCH runs in $\mathcal{O}(n+k)$ time for a scanned text of length n and k occurrences of words from W. Thus, automaton construction and matching takes $\mathcal{O}(m + n + k)$ time with $m = \sum_{w \in W} |w|$. Searching with an ACA is a generalization of the algorithm of Knuth, Morris, and Pratt [58], which is the best known linear-time algorithm ($\mathcal{O}(n+m)$) for the exact pattern matching problem "report all occurrences of a given pattern of length m in a given text of length n" in the case of a single pattern, i.e. $W = \{w\}$ and hence, $m = |w|$.

Remark 2.1.15. *Note that in the case of $|W| = 1$, for all $w \in \Sigma^+$, ACA($\{w\}$) is the automaton with the smallest number of vertices that recognizes the language $\Sigma^*\{w\}$, that is, ACA($\{w\}$) is minimal.*

Counting with an Automaton

In particular, Aho-Corasick automata are able to treat sets of patterns where one pattern is a substring of another pattern. Thus, in contrast to other DFAs, they can indicate if more than one pattern ends at a certain position within a text. However, we are interested in the number of pattern occurrences. To this end, we introduce a *count function*, i.e. a mapping $\mu : Q \to \mathbb{N}_0$, assigning each state the number of patterns ending here. Herewith, we define *Counting Deterministic Finite Automata (CDFA)* as has been done in [83].

Definition 2.1.16 (Counting Deterministic Finite Automaton (CDFA))**.** *A* counting deterministic finite automaton *is a 5-tuple of the form $(Q, \Sigma, \Delta, q_0, \mu)$, where the set of final states in a DFA $(Q, \Sigma, \Delta, q_0, F)$ (according to Definition 2.1.8) is replaced by a count function $\mu : Q \to \mathbb{N}_0$ with $\mu(q) = |\{w \in W \mid w \blacktriangleleft q\}|$.*

Remark 2.1.17. *Note that CDFAs are a generalization of DFAs, since the CDFA $(Q, \Sigma, \Delta, q_0, \mu)$ induces the DFA $(Q, \Sigma, \Delta, q_0, F = \{q \in Q \mid \mu(q) > 0\})$. Hence, the minimization algorithm by Hopcroft [50] can straightforwardly be extended to CDFAs.*

The initial partition has to take the count function into account. Instead of only distinguishing between final and remaining states, two states u and v are in the same initial part if $\mu(u) = \mu(v)$.

In the case of ACA, the respective CDFA is given by $(Q, \Sigma, \Delta', \epsilon, \mu)$ where

$$\Delta'(q,\sigma) = \begin{cases} \Delta(q,\sigma) & \text{if } \Delta(q,\sigma) \neq \star\,, \\ \Delta'\big(\Phi(q),\sigma\big) & \text{otherwise} \end{cases}$$

and $\mu : q \mapsto |\Theta(q)|$ counts the number of output patterns. Hence, via μ, a CDFA can be used to count the number of occurrences of patterns from a finite set W of patterns. We define the occurrence count of W in a given text s with respect to a CDFA as follows.

Definition 2.1.18 (CDFA Occurrence Count)**.** *Given a CDFA $K = (Q, \Sigma, \Delta, q_0, \mu)$ for a finite set W of patterns and a string $s \in \Sigma^*$,*

$$C_K(s) = \sum_{i=1}^{|s|} \mu\big(\Delta^*(q_0, s[1,i])\big) \tag{2.1.5}$$

is called the CDFA occurrence count *of W in s.*

The CDFA occurrence count deduced from the Aho-Corasick counting automaton corresponds to the overlapping occurrence count defined in Section 2.1. This result is stated in the next lemma.

Lemma 2.1.19. *Let W be a finite set of patterns and $(Q, \Delta, \Phi, \Theta)$ the corresponding ACA(W). Further, let $K = (Q, \Sigma, \Delta', \epsilon, \mu)$ be the respective CDFA as defined above. Then it holds for all $s \in \Sigma^+$:*

$$C_K(s, W) = N_o(s, W)$$

with the overlapping occurrence count $N_o(s, W)$ as defined in Definition 2.1.4.

Proof. The equality follows directly from the definitions and the properties of ACA:

$$\begin{aligned} C_K(s,W) &= \sum_{i=1}^{|s|} \mu\big(\Delta^*(q_0, s[1,i])\big) \\ &= \sum_{i=1}^{|s|} |\Theta(\mathrm{argmax}_{x \in \mathrm{Suffixes}(s[1,i]) \cap Q} |x|)| \quad \text{(def. of } \mu \text{ in CDFA)} \\ &= \sum_{i=1}^{|s|} |\Theta(s[1,i])| \quad \text{(construction of ACA)} \\ &= \sum_{i=1}^{|s|} \sum_{w \in W} \mathbb{1}_{\{w \blacktriangleleft s[1,i]\}} \quad \text{(def. of } \Theta \text{ in ACA)} \\ &= N_o(s,W)\,. \end{aligned}$$

□

Moreover, the final states of the ACA satisfy an important property which is useful for our intention to count occurrences of patterns in random strings. This property is described by the following theorem. It is a rephrasing of two lemmata by Aho and Corasick [2]. We cite it and its proof, following Lladser *et al.* [74], Lemma 4.4.

Theorem 2.1.20 (Lladser *et al.* [74])**.** *For $w \in W$ define $B(w) := \{u \in Q \mid w \blacktriangleleft u\}$. For all $w \in W$ and all $s \in \Sigma^*$, w occurs m times as substring of s if and only if the path associated with s in ACA(W) visits the set $B(w)$ exactly m times.*

Proof. Suppose that w occurs m times as substring of s and that the path associated with s in ACA(W) visits $B(w)$ exactly l times. The following properties show that $m = l$.

(i) If for some i, $0 \leq i \leq |s|$, $\Delta^*(\epsilon, s[1,i]) = u \in B(w)$, then, according to Lemma 2.1.14, $w \blacktriangleleft u \blacktriangleleft s[1,i]$. Thus, $m \geq l$.

(ii) If for some $1 \leq i \leq |s|$ it is $w \blacktriangleleft s[1,i]$, and $u = \Delta^*(\epsilon, s[1,i])$, then, according to Lemma 2.1.14, u is the longest string in Q such that $u \blacktriangleleft s[1,i]$. Since $w \in Q$, it follows that $|w| \leq |u|$. In particular, $w \blacktriangleleft u$ and hence, $u \in B(w)$. This shows that $m \leq l$.

Thus, $m = l$, and this completes the proof. □

Theorem 2.1.20 shows that the probability that a word occurs in a random text of a certain length can be determined from the probability that an appropriate automaton visits a specified set of states within a certain number of steps. This transformation of the pattern matching problem into a question about the behavior of a Markov chain is called *Markov chain embedding*. In the next section, we consider the embedding of a random text into the states of a deterministic automaton.

2.1.3 Markov Chain Embedding

Now that we have all necessary notations at hand, we move from deterministic to probabilistic pattern matching. To this end, we describe random walks on automata which is referred to as Markov chain embedding.

Definition 2.1.21 (Markov chain embedding)**.** *Let $X = (X_n)_{n\in\mathbb{N}_0}$ be a random text over Σ and $G = (Q, \Sigma, \Delta, q_0, F)$ be a DFA. Define $Q^G := \Delta^*(q_0, \Sigma^+)$, i.e. $Q^G = \{q \in Q \mid \exists\, s \in \Sigma^+ : \Delta^*(q_0, s) = q\}$. The* Markov chain embedding *of X in G is the sequence of Q^G-valued random variables $X^G = (X_n^G)_{n\in\mathbb{N}_0}$ where (for $n \geq 0$)*

$$X_n^G = \Delta^*(q_0, X_0 \dots X_n)\,.$$

The following theorem shows that this embedded process, is indeed a Markov chain (cf. [74, Theorem 5.1]). In particular, a transition that is triggered by a character $\sigma \in \Sigma$ in the deterministic automaton occurs randomly with probability p_σ in the random walk on the automaton's states. This random walk can be represented by a first-order homogeneous Markov chain, since the transition probability only depends on the current state (and not on preceding states).

Theorem 2.1.22 (Lladser *et al.* [74])**.** *If $X = (X_n)_{n\in\mathbb{N}_0}$ is a sequence of i.i.d. Σ-valued random variables and $G = (Q, \Sigma, \Delta, q_0, F)$ is a DFA, then $X^G = (X_n^G)_{n\in\mathbb{N}_0}$ is a first-order homogeneous Markov chain with initial distribution*

$$\mathbb{P}(X_0^G = u) = \sum_{\sigma\in\Sigma:\Delta(q_0,\sigma)=u} p_\sigma \quad \textit{for all } u \in Q^G\,, \tag{2.1.6}$$

and transition probabilities

$$\mathbb{P}(X_{n+1}^G = v \,|\, X_n^G = u) = \sum_{\sigma\in\Sigma:\Delta(u,\sigma)=v} p_\sigma \quad \textit{for all } u, v \in Q^G\,. \tag{2.1.7}$$

We recall the proof following the original article.

Proof. The proof of (2.1.6) directly follows from $\mathbb{P}(X_0^G = u) = \mathbb{P}(\Delta(q_0, X_0) = u)$. Then, we still have to show two properties:

- Markov property: Observe that $X_{n+1}^G = \Delta(X_n^G, X_{n+1})$ according to Equation (2.1.1). Thus, for all $u_1, \ldots, u_n, v \in Q$ it is

$$\begin{aligned} & \mathbb{P}\big(X_1^G = u_1, \ldots, X_n^G = u_n, X_{n+1}^G = v\big) \\ = \;& \mathbb{P}\big(X_1^G = u_1, \ldots, X_n^G = u_n, \Delta(u_n, X_{n+1}) = v\big) \\ = \;& \mathbb{P}\big(X_1^G = u_1, \ldots, X_n^G = u_n\big) \cdot \mathbb{P}\big(\Delta(u_n, X_{n+1}) = v\big)\,, \end{aligned}$$

  because X_{n+1} is independent of $X_1, \ldots, X_n$. This shows that X^G is a first-order Markov chain.
- Homogeneity: Since the distribution of X_{n+1} does not depend on n, $\mathbb{P}(X_{n+1}^G = v \,|\, X_n^G = u_n, \ldots, X_1^G = u_1) = \mathbb{P}(\Delta(X_n^G, X_{n+1}) = v \,|\, X_n^G = u_n)$ depends only on u_n and v, but not on n. This shows that X^G is homogeneous.

Herewith, (2.1.7) follows immediately. □

This theorem is of major importance. It allows us to calculate statistics linked to patterns in random strings by computing probabilities associated with a first-order homogeneous Markov chain defined on the state space of an appropriate automaton. Thus, we can apply well-known techniques from Markov chain theory and combinatorics in order to solve probabilistic pattern matching problems.

Convergence of Embedded Markov Chains

Before we come to this, let us consider the behavior of the embedded Markov chain after a long period n. In the following, we show in which case the distribution of X^G settles down when time passes ($n \to \infty$). To this end, we recall some important properties of discrete-time homogeneous Markov chains, conforming with Grimmett and Stirzaker [44]. Let $X = (X_n)_{n \in \mathbb{N}_0}$ be a homogeneous Markov chain with state space S and stochastic transition matrix $P = (p_{ij})_{i,j \in S}$. The n-step transition matrix, which is the n^{th} power of P, is denoted as $P^n = \left(p_{ij}^{(n)}\right)_{i,j \in S}$. Moreover, each character occurs with positive probability, which is to say that $p_\sigma > 0$ for all $\sigma \in \Sigma$. Otherwise, if a character has zero probability, it is removed from Σ and all words containing this character are removed from W. Then, these words do not occur in the language recognized by G.

Definition 2.1.23 (Communication)**.** *State i* communicates *with state j ($i \to j$) if there exists an $m \geq 0$ such that $p_{ij}^{(m)} > 0$. Analogously, i and j* intercommunicate *($i \leftrightarrow j$) if $i \to j$ and $j \to i$.*

Definition 2.1.24 (Irreducibility)**.** *A set C of states is called*

- irreducible *if $i \leftrightarrow j$ for all $i, j \in C$,*
- closed *if $p_{ij} = 0$ for all $i \in C$, $j \notin C$.*

In other words, a set of states is irreducible if each state can be reached from each other state. If the whole state space S is irreducible, we also call the Markov chain irreducible. On the other hand, if the chain takes a value in a closed set C, it subsequently never leaves C. If a closed set contains only one state, we call this state *absorbing*.

Definition 2.1.25 (Recurrence)**.** *A state $i \in S$ is called* recurrent *(*persistent*) if*

$$\exists\, n \geq 1 : \mathbb{P}(X_n = i \,|\, X_0 = i) = 1,$$

Otherwise, if this probability is strictly less than 1, i is called transient*.*

To show that the states of an embedded Markov chain are recurrent, we make use of the following lemma (adapted from [44, Section 6.2]).

Lemma 2.1.26. *State i is recurrent if and only if $\sum_{n \geq 0} p_{ii}^{(n)} = \infty$.*

For the proof, the reader is referred to the text book mentioned above. Let us further investigate the time up to the first visit of state i. This is given by

$$T_i = \min\{n \geq 1 \,|\, X_n = i\}\,.$$

For convenience, we define $T_i = \infty$ if the chain never visits state i. We have $\mathbb{P}(T_i = \infty \,|\, X_0 = i) > 0$ if and only if i is transient. Usually, one is interested in the average number of steps to the subsequent visit.

Definition 2.1.27 (Mean recurrence time)**.** *The* mean recurrence time μ_i *of a state* i *is defined as*

$$\mu_i = \mathbb{E}(T_i \mid X_0 = i) = \begin{cases} \sum_{n \geq 1} n \cdot \mathbb{P}(X_1 \neq i, \ldots, X_{n-1} \neq i, X_n = i \mid X_0 = i) & \text{if } i \text{ is recurrent,} \\ \infty & \text{if } i \text{ is transient.} \end{cases}$$

Note that μ_i may be infinite even if i is recurrent. Hence, we classify recurrent states according to the next definition.

Definition 2.1.28 (Positive and null recurrence)**.** *For a recurrent state* i*,*

$$i \text{ is called } \begin{cases} \text{null } \textit{recurrent} & \text{if } \mu_i = \infty, \\ \text{positive } \textit{recurrent} & \text{if } \mu_i < \infty. \end{cases}$$

For the case of a finite state space, this definition leads us to the following property. The lemma and its proof can be found in Grimmett and Stirzaker [44].

Lemma 2.1.29. *If* S *is finite, then at least one state is recurrent and all recurrent states are positive recurrent.*

Proof. See Grimmett and Stirzaker [44], Section 6.3. □

Let us now investigate the existence of a limiting distribution for X as $n \to \infty$. It turns out that this is closely related to the existence of a so-called *stationary distribution*.

Definition 2.1.30 (Stationary distribution)**.** *Let* X *be a Markov chain with state space* S*. The vector* $\pi = (\pi_i)_{i \in S}$ *is called* stationary distribution *for* X *if*

(a) $\pi_i \geq 0$ *for all* $i \in S$ *and* $\sum_{i \in S} \pi_i = 1$*,*

(b) $\pi = \pi P$*, i.e.* $\pi_j = \sum_{i \in S} \pi_i p_{ij}$ *for all* $j \in S$*.*

In particular, this definition comprises $\pi P^n = \pi$ for all $n \geq 0$. This is to say, if X_0 has distribution π, then X_n has distribution π for all n, and in this case, π is the limiting distribution of X as $n \to \infty$. We use the following theorem to investigate the existence of a stationary distribution for X^G.

Theorem 2.1.31 (Grimmett and Stirzaker [44])**.** *An irreducible chain has a stationary distribution* π *if and only if all the states are positive recurrent; in this case,* π *is the unique stationary distribution and is given by* $\pi_i = {}^1/_{\mu_i}$ *for all* $i \in S$*, where* μ_i *is the mean recurrence time of state* i*.*

Proof. See Grimmett and Stirzaker [44], Section 6.4. □

The following corollary, adjusted from the diploma thesis mentioned before, shows that this theorem applies to the Markov chain embedded in a minimal DFA.

Corollary 2.1.32. *Let $X^G = \left(X_n^G\right)_{n\in\mathbb{N}_0}$ be a Markov chain embedded in a minimal DFA $G = (Q, \Sigma, \Delta, q_0, F)$ recognizing $\Sigma^* L$. If there exists an $s \in \Sigma^+$ such that*

$$\Big(\text{Suffixes}(ts) \cap \bigcup_{l\in L} \text{Prefixes}(l)\Big)\backslash\{\epsilon\} = \emptyset \tag{2.1.8}$$

for all $t \in \Sigma^*$, then X^G is irreducible and all states are positive recurrent. Thus, $\pi = (\pi_q)_{q\in Q^G} = \left(1/\mu_q\right)_{q\in Q^G}$ is the unique stationary distribution.

Proof. We have to show that Equation (2.1.8) implies that

(a) X^G is irreducible and

(b) all states $q \in Q^G$ are positive recurrent.

Then, the statement directly results from Theorem 2.1.31. To show (a), observe that for all $u, v \in Q^G$

$$\exists\, m \geq 0 \,:\, p_{uv}^{(m)} > 0 \Leftrightarrow \exists\, r \in \Sigma^+ \,:\, \Delta^*(u, r) = v\,, \tag{2.1.9}$$

because we assume that $p_\sigma > 0$ for all $\sigma \in \Sigma$. We use this equivalence and show that for all pairs of states such a word exists if Equation (2.1.8) holds.

Recall that Q^G is defined as the set of states that are reachable from q_0. We show that also q_0 is reachable from every state $q \in Q^G$, thus every state q' is reachable from every other state q via the path $q \to q_0 \to q'$.

Let q_1 be the current state and $t \in \Sigma^*$ the string read so far, i.e. $q_1 = \Delta^*(q_0, t)$. Reading $s \in \Sigma^+$ leads to another state $q_2 = \Delta^*(q_1, s) = \Delta^*(q_0, ts)$. We show that Equation (2.1.8) implies that $q_2 = q_0$, and hence, q_0 is reachable from every state $q_1 \in Q^G$. To this end, we define

$$L_q = \{r \in \Sigma^+ \,|\, \Delta^*(q, r) \in F\} \text{ for } q \in Q^G\,.$$

In particular, $L_{q_0} = \Sigma^* L$. Moreover, $L_{q_0} \subseteq L_q$ for all $q \in Q^G$ because of the following: since $q \in Q^G$, there exists a word $x \in \Sigma^*$ such that $\Delta^*(q_0, x) = q$. Now let $y \in L_{q_0} = \Sigma^* L$. Then, $xy \in \Sigma^* L$ and thus, $\Delta^*(q_0, xy) = \Delta^*(q, y) \in F$.
We show by contradiction that indeed $L_{q_0} = L_{q_2}$. For this purpose, assume that $L_{q_2}\backslash L_{q_0} \neq \emptyset$ and let $x \in L_{q_2}\backslash L_{q_0}$. Hence, $tsx \in L_{q_0}$, which implies the existence of $r' \in \Sigma^*$ and $l \in L$ such that $r'l = tsx$. But then, we have the following:

- If $|r'| \geq |ts|$, then $l \blacktriangleleft x$ and thus, $x \in \Sigma^* L = L_{q_0}$. But this contradicts the defining property of x.
- If $|r'| < |ts|$, then $x \blacktriangleleft l$ and thus, there exists an $x' \in \Sigma^+$ such that $x'x = l$. However, this implies $x' \in \text{Suffixes}(ts) \cap \text{Prefixes}(l)$, which contradicts (2.1.8).

It follows that $L_{q_2} = L_{q_0}$. Further, since G is minimal, we have $q_2 = q_0$, which completes the proof of irreducibility.

Equation (2.1.9) further implies that all states are recurrent, because of the following observation: let $\Delta^*(q_0, t) = q_1$ and $\Delta^*(q_1, s) = q_0$ as above. But also $\Delta^*(q_1, \sigma s) = \Delta^*(q_1', s) = q_0$ for all $\sigma \in \Sigma$, since the argumentation given above does not depend on q_1. Hence, in general, $\Delta^*(q_1, ys) = q_0$ for all $y \in \Sigma^*$. Thus, $p_{q_1 q_1}^{(|yst|)} > 0$ for all $y \in \Sigma^*$, and $\sum_{n \geq 1} p_{qq}^{(n)} = \infty$ for all $q \in Q^G$. Herewith, part (b) follows by means of Lemma 2.1.29. □

Now, we have provided the theoretical basis for the next section, where we introduce the Probabilistic Arithmetic Automaton and use the evolution of the embedded Markov chain in order to calculate statistics linked to patterns in random texts.

2.2 Formal Definition of Probabilistic Arithmetic Automata

In this section, we introduce *probabilistic arithmetic automata* (PAA) which are used to model sequences of operations whose operands depend on chance. For this purpose, the concept of probabilistic automata as explained in Section 2.1 is extended by calculations on the count function. In particular, a PAA entails the probabilistic variant of the CDFA occurrence count (2.1.5), which is to say that it provides the distribution of accumulated occurrences of patterns in a random string. While previous methods use a Markov additive chain (MAC) defined on the counts output during a random walk on the states of a probabilistic automaton, the PAA framework comprises this process. Besides, it is more flexible in the choice of operations. As the name indicates, MACs only accomplish the accumulation of counts, whereas a PAA permits more complicated operation designs.

In accordance with former definitions, we define the framework of PAAs as introduced in Marschall and Rahmann [84] before we illustrate its principal properties.

Definition 2.2.1 (Probabilistic Arithmetic Automaton (PAA))**.** *A* probabilistic arithmetic automaton *is a* 8*-tuple of the form* $\big(Q, T, q_0, E, e = (e_q)_{q \in Q}, N_g, n_0, \theta = (\theta_q)_{q \in Q}\big)$*, where*

- Q *is a finite set of states.*
- $T = \big(T_{uv}\big)_{u,v \in Q}$ *is a stochastic state transition matrix, i.e.* $\sum_{v \in Q} T_{uv} = 1$ *for all* $u \in Q$*.*
- $q_0 \in Q$ *is called* start state*.*
- E *is a finite set called* emission set*.*

- *Each* $e_q : E \to [0,1]$ *is a* weight distribution *associated with state* q*. In particular,* $\sum_{e \in E} e_q(e) = 1$*.*
- N_g *is a set called* value set*.*
- $n_0 \in N_g$ *is called* start value*.*
- *Each* $\theta_q : N_g \times E \to N_g$ *is an* operation *associated with state* q*.*

If we denote the Dirac measure assigning probability 1 to outcome x with δ_x, the tuple (Q, T, δ_{q_0}) defines a Markov chain on state space Q with initial state q_0 and transitions according to T. We denote its associated state process with $\big(Y_n\big)_{n \in \mathbb{N}_0}$. Thus, $\mathbb{P}(Y_n = q)$ is the probability that the Markov chain visits state q in step n. Analogously, $\big(Z_n\big)_{n \in \mathbb{N}_0}$ represents the sequence of emissions, and $\big(V_n\big)_{n \in \mathbb{N}_0}$ denotes the sequence of values $V_l = \theta_{Y_l}(V_{l-1}, Z_l)$, with $V_0 = n_0$, resulting from the operations performed. In fact, not only the customary arithmetic operations like "$+, -, *, /$" are allowed, but rather any kind of binary operation. Indeed, caution has to be exercised such that emission value 0 is prohibited when the operation is "/".

Remark 2.2.2. *Note that, in general, the set* N_g *of values is not finite. For instance, in the applications we consider later, we have that all states are associated with the operation "+", and* $N_g = \mathbb{N}$*. For our computations, we then use a finite subset* $N \subset N_g$ *by defining an upper bound for the maximal value. In the case that* N_g *is finite,* $N = N_g$*.*

To state it in more practical terms, the PAA starts in state q_0, i.e. $Y_0 = q_0$, with initial value $V_0 = n_0$. Then, according to the transition probabilities, the following state, say q, is determined. A value z_1 depending on the entered state is emitted appropriate to the associated weight distribution e_q. The operation θ_q determines the next value $n_1 = \theta_q(n_0, z_1)$. While traversing the Markov chain, a PAA performs a sequence of operations on a set N of values, starting with the value n_0, and assigns probabilities to those values. Hence, a PAA parses strings yielding not only a boolean result (compare DFAs) but their probability under the respective random string model and the distribution of their emitted values subject to associated operations. Moreover, through the choice of states and emissions, the PAA framework can account for different counting schemes, as discussed in Section 2.1. We make use of this property in Chapter 5.

Remark 2.2.3. *It is noteworthy that the Markov chain* (Q, T, δ_{q_0}) *together with the set of state emissions* $e = (e_q)_{q \in Q}$ *defines an HMM. Thus, PAAs may be treated as an extension of HMMs. However, in the PAA framework the states are usually not unknown like in HMM applications, and we are not interested in the state process but in the value process.*

2.3 A Toy Example from Gambling

Now that we have formally defined the PAA framework, let us consider a toy example (see Figure 2.3).

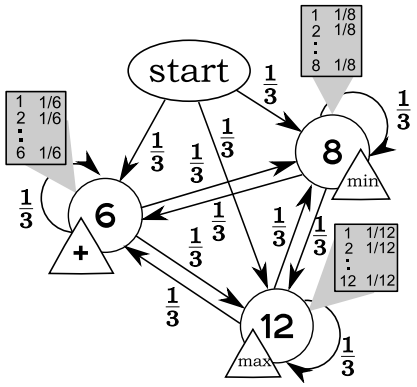


Figure 2.3: Sketch of the PAA modelling the game described in the text: the states correspond to three fair dice with 6, 8 and 12 faces. These are sequentially sampled with replacement. Each die has an associated operation which is performed on the emitted values.

Along with the majority of textbooks about probability theory, we consider an example of gambling. Assume the following game: in a bag there are three fair dice; one 6-faced, one 8-faced and one 12-faced die. The gambler draws a die, rolls it, notes the pips, and puts it back. This procedure is repeated t times. Moreover, there are three operations associated with the dice. The value resulting of rolling the 6-faced die is added to the current total value. Whenever the 8-faced die is drawn, the current total value is replaced by the minimum of itself and the thrown pips, and for the 12-faced die, the maximum of the current total value and the thrown pips is noted. In the end, the gambler wins if the total value exceeds a given threshold. A profit-oriented casino would probably set a threshold above the value expected on average. For instance, a threshold could be chosen such that 40% of the games are expected to be won by the gambler. What is the expected total value, then? And how could one compute the quantiles of the value distribution?

To answer these questions, we use the PAA framework to model the game. Each die is represented by a state. All transition probabilities are set to $1/3$, since each die is drawn with equal probability. The weight distributions associated to the states correspond to uniform distributions over the number of faces of the respective die. In the game described above, the operations refer to "add" (6-faced die), "take the minimum" (8-faced die), and "take the maximum" (12-faced die). The so constructed PAA is shown in Figure 2.3. Note that we do not need to assign a weight distribution nor an operation to the start state. This is superfluous whenever the start state is transient since then $Y_l = q_0$ only for $l = 0$.
Running the Markov chain for t steps yields the distribution of total values. Hence,

we can easily calculate the respective expectation value. Note that, in general, various combinations of arithmetic operations are imaginable.

Table 2.1 displays possible outcomes, running the game for $t = 10$ steps. Y_t denotes the state, i.e. which die was drawn, Z_t shows the thrown pips, i.e. the emitted value, and V_t gives the total value after round t.

Table 2.1: Possible outcomes of ten rounds of the game described in the text. Here, the gambler has obtained a total value of eight after ten rounds.

t	0	1	2	3	4	5	6	7	8	9	10
Y_t	start	8	8	6	12	8	12	12	6	8	6
Z_t	0	6	7	5	10	3	2	8	2	4	4
V_t	0	0	0	5	10	3	3	8	10	4	8

The gambler wins this game if the threshold set by the casino is smaller than 8. The casino would usually calculate the probability to obtain a certain value after t rounds as reference. In the next section, we make the computation of the value distribution $\mathcal{L}(V_t)$ precise.

2.4 Computations and Implementation

The PAA framework yields the probability distribution $\mathcal{L}(V_t)$ of the resulting total value after t steps. We therefore compute the joint state-value distribution $\mathcal{L}(Y_t, V_t)$, from which we determine the sought quantities via marginalization:

$$\mathbb{P}(V_t = v) = \sum_{q \in Q} \mathbb{P}(Y_t = q, V_t = v)\,.$$

For the sake of readability, throughout this thesis we use the short hand notation

$$f_t(q, v) := \mathbb{P}(Y_t = q, V_t = v)\,.$$

The following general recurrence relation gives the joint state-value distribution:

$$f_l(q, v) = \sum_{q' \in Q} \sum_{(v', u) \in \theta_q^{-1}(v)} f_{l-1}(q', v') T_{q'q} e_q(u)\,, \tag{2.4.1}$$

where $\theta_q^{-1}(v)$ denotes the inverse image of v under θ_q, that is the set of all (value, emission) pairs such that $\theta_q(v', u) = v$.

In order to calculate the desired distribution $f_t = \mathcal{L}(Y_t, V_t)$, we start with $f_0 = \mathcal{L}(Y_0, V_0)$ and iteratively apply Equation (2.4.1) to obtain subsequent distributions and finally f_t. We state this algorithmic approach in Algorithm 2.2. In some cases, the recurrence can also be solved analytically by means of a probability generating function (pgf).

Algorithm 2.2 PAADIST

Input: $f_0 = \mathcal{L}(Y_0, V_0)$, $t \in \mathbb{N}$, two tables of size $|Q| \times |N|$
Output: $f_t = \mathcal{L}(Y_t, V_t)$
1: **for** $l = 1$ to t **do**
2: **for all** $q \in Q$ and $v \in N$ **do**
3: compute $f_l(q, v)$ by means of (2.4.1) and table f_{l-1}
4: **end for**
5: store values $f_l(q, v)$ in table f_l
6: rename table f_{l-1} to f_{l+1}
7: **end for**
8: output f_t

As can be seen from Algorithm 2.2, we use a *push-strategy* to implement the recursion. The idea is, starting with f_0, to "push" already computed values forward. Thus, we store the values $f_{l-1}(q, v)$ for all $q \in Q$ and all $v \in N$ in an array of size $|Q| \times |N|$ and iterate over all these entries in order to calculate f_l by means of the recurrence (2.4.1). When all values $f_l(q, v)$ have been found, f_{l-1} can be discarded. Hence, in each step, two arrays have to be stored consuming $\mathcal{O}(|Q| \cdot |N|)$ space. To calculate a table f_l, Equation (2.4.1) has to be evaluated $|Q| \cdot |N|$ times. Further, to derive a single entry $f_l(q, v)$, we have to sum over $|Q| \cdot |E|$ values, because for fixed v, $|\theta_q^{-1}(v)|$ depends only on the emissions in q. Moreover, $\theta_q^{-1}(v_1) \cap \theta_q^{-1}(v_2) = \emptyset$ for $v_1 \neq v_2$, and $\cup_{v \in N} \theta_q^{-1}(v) = N \times E$. Consequently, the update from f_{l-1} to f_l takes $\mathcal{O}(|Q|^2 \cdot |N| \cdot |E|)$ time and hence, the total runtime of Algorithm PAADIST is $\mathcal{O}(t \cdot |Q|^2 \cdot |N| \cdot |E|)$.

For the case of large t, Marschall and Rahmann [84] introduced an alternative algorithm, called *Doubling Technique*, which is based on the Chapman-Kolmogorov Equation for homogeneous Markov chains [39, Chapter XV]. The idea is to reduce the number of update steps (evaluations of Equation (2.4.1)) from t to $\log(t)$. Rather than performing each step f_{l-1} to f_l, we only consider transitions from f_{l-1} to f_{l-1+k} for a step size $k > 1$. Following the original paper, we set

$$U^k(q_1, q_2, v_1, v_2) = \mathbb{P}(Y_{i+k} = q_2, V_{i+k} = v_2 \,|\, Y_i = q_1, V_i = v_1)\,.$$

Note that this definition does not depend on i, because of the homogeneity property. With this, we get the distribution $\mathcal{L}(Y_t, V_t)$ via the correspondence

$$\mathbb{P}(Y_t = q, V_t = v) = U^t(q_0, q, n_0, v)\,.$$

Generalizing the Chapman-Kolmogorov Equation, we have

$$U^{k_1+k_2}(q_1, q_2, v_1, v_2) = \sum_{q \in Q} \sum_{v \in N} U^{k_1}(q_1, q, v_1, v) U^{k_2}(q, q_2, v, v_2)\,. \tag{2.4.2}$$

This update takes $\mathcal{O}(|Q|^3 \cdot |N|^3)$ time. In particular, Equation (2.4.2) holds for $k_1 = k_2 = k$. Then, only the distribution U^k has to be known. All distributions U^{2^j} for $0 \leq j \leq \lceil \log(t) \rceil$ can be computed in $\lceil \log(t) \rceil$ steps, and hence, the total runtime of the Doubling Technique is $\mathcal{O}(\log(t) \cdot |Q|^3 \cdot |N|^3)$.

2.5 Applications Overview

As we have seen in the previous sections, the PAA framework is a flexible probabilistic method to model sequences of operations whose operands depend on chance. It yields recurrence relations for the *exact* probability distribution of the resulting value. Since it refers to a probabilistic automaton extending the concept of DFAs, we can reuse related algorithms. The implementation, as presented in the last section, is straightforward.

The general definition of PAAs comprises different levels of dependence (i.i.d., Markovian) between the operands as well as various operations associated with them. In applications concerning computational biology, in particular biological sequence analysis, addition of emitted values is the most likely operation. This relates to counting patterns within randomly generated biosequences. Usually, this random text is a longer sequence of nucleotides, amino acids, or alignment columns following an appropriate null model. The patterns may be exact or approximate orders of nucleotides, amino acids, or alignment columns. They can be represented by a string, a set of strings, or even more complex formats. To this end, we generally define a pattern in form of a regular expression.

From the computational biology perspective, interesting problems include the following:

- motif statistics in general used for discovery of genes, introns, and regulatory elements
- finding patterns that are statistically over- or underrepresented in given genomes
- computing the significance of protein classifications
- computing the significance of transcription factor binding sites (TFBS)
- determining the sensitivity of seeds used for filtering databases in local alignment search
- computing the length distribution of reads resulting from pyrosequencing
- computing the expected population size/growth under specified evolutionary processes

Furthermore, the capability of the PAA framework definitely exceeds applications from computational biology. We think that this method might be of value for instance for (i) HMM-based speaker recognition, and for (ii) music retrieval and comparison.

Regarding (i), in text-dependent speaker recognition methods [103], the input utterance is represented by a sequence of short-term spectral feature vectors. Variations in such signals are modeled by the emission distributions in the HMM. In order to compare the utterance to a reference model of registered speakers, a similarity score is computed in each state and cumulated over the sequence. Thus, the HMM approach could be replaced by an appropriate PAA model in order to get

the probability distribution that the input utterance corresponds to the registered speakers.

In view of (ii) music retrieval and comparison, projects such as C-BRAHMS at the university of Helsinki consider the task of finding pieces of music in large databases that contain symbolically encoded music [68, 77]. One famous research topic is the "what you hum is what you get" application. The PAA framework could be used in order to compute the distribution of hits of symbolic excerpts to the compositions within the database, and in particular to compute the significance of such a comparison.

In the following chapters, we present selected applications from the field of computational biology which address prevalent research areas.

CHAPTER 3

APPLICATION I: PEPTIDE MASS FINGERPRINTING

The topic of this chapter was motivated by the thesis of Kaltenbach [54], who set up a probabilistic framework to compute the significance of protein identification by means of *peptide mass fingerprinting* (PMF). This is a standard technique based on *mass spectrometry* (MS) and database search to rapidly characterize and quantify proteins. To this end, the protein of interest is biochemically dissociated into smaller peptides, whose masses are determined by MS. The set of peptide masses, the so-called *peptide mass fingerprint*, is then used to query a protein database. We formulate a general PAA to mimic the enzymatic dissociation of random protein sequences and investigate arising peptide fragment statistics. In particular, our model extends the framework mentioned such that protein sequences are modeled by a Markovian rather than an i.i.d. source. Additionally, we state modifications of the model taking inefficiencies during the dissociation and post-translational modifications of proteins into account.

We start with an introduction to mass spectrometry in Section 3.1. For a more detailed discussion on the subject, the reader is referred to one of the textbooks [34, 45, 72, 85] and to the overview article by Aebersold and Mann [1]. Furthermore, we describe corresponding methods for protein identification in Section 3.2. Then, in Section 3.3, we present a PAA appropriate to generate and measure peptide fragments typically resulting in the course of PMF. Moreover, we calculate associated peptide statistics, which allow the computation of a significance value for protein identification by PMF following [54]. Finally, Section 3.4 concludes this chapter with a comparison of our statistics to empirical ones and those derived by Kaltenbach [54].

3.1 Introduction to Mass Spectrometry

One focus in proteomics is to determine the compound of proteins present in a cell and how this compound changes under different conditions. Since most cellular proteomes are very complex and many of the constituting proteins occur with low abundance, related studies require highly sensitive analytical techniques. Mass spectrometry is an efficient tool to characterize and quantify proteins in a high-throughput manner, which improved the accuracy of proteomic studies and even enabled the characterization of entire genomes. In principle, a mass spectrometer is a device that determines the molecular mass, or more precisely the *mass-to-charge ratio* m/z, of the sample molecules present in the compound of interest (the analyte). To this end, it generates gas phase ions of the analyte, separates these ions by their m/z values, and detects them qualitatively and quantitatively. The abundances of respective m/z values are returned in form of a *mass spectrum*. The Molecular masses are typically measured in Dalton (Da), where one Dalton is defined as $1/12$ of the molecular mass of the ^{12}C carbon isotope. This corresponds approximately to the mass of a single proton; i.e. $1\,\mathrm{Da} = 1/N_A\,\mathrm{g} \approx 1.66 \cdot 10^{-24}\,\mathrm{g}$, where N_A denotes Avogadro's number, i.e. the number of atoms in exactly 12 g of ^{12}C carbon.

3.1.1 Instrumentation

While different methods can be used for ionization, separation, and detection, each mass spectrometer consists of the three basic components shown in Figure 3.1.

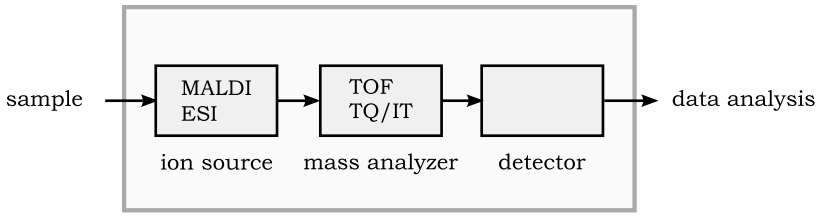


Figure 3.1: General design of a mass spectrometer. The sample of interest is introduced to the ion source which generates gas phase ions. By applying electric/magnetic fields, these ions are transported to the mass analyzer that sorts the ions according to their m/z values. The detector measures an indicator quantity (such as time-of-flight) providing information about the molecular masses and the abundances of the ions present.

The Ionizer

The *ion source* converts gas phase sample molecules into charged particles, i.e. ions, or moves ions that exist in solution into the gas phase. These ions are then transported by magnetic and/or electric fields to the mass analyzer.

Although used in many physical and chemical applications since 1900, earlier ionization methods limited the use of former spectrometers to measure relatively small (light) biological molecules. This could be overcome when *matrix-assisted laser desorption/ionization* (MALDI) [49, 55] and *electrospray ionization* (ESI) [40] were developed in the late 1980s. These techniques allow measurements of up to tens or even hundreds of kilodaltons (kDa) and thus became the methods of choice for MS of proteins and peptides. MALDI-MS is usually used to analyze relatively simple peptide samples, whereas ESI-MS is applied for the analysis of more complex mixtures.

- **Matrix-Assisted Laser Desorption/Ionization (MALDI):** The analyte is mixed with a chemical compound, the *matrix*, which absorbs light at a specific wavelength. The sample is cocrystallized with the matrix and spotted onto a small plate, which is placed into the ion source. A short-pulsed laser beam (with wavelength specific to the matrix) causes the matrix to evaporate and to transfer energy and protons to the analyte. The resulting ions with mainly single positive charge are released into the gas phase and directed into the mass analyzer.
- **Electrospray Ionization (ESI):** In contrast to MALDI, here the peptides to analyze exist as ions in aqueous solution which is passed through a fine needle held at high potential, causing the analyte flow to disperse as a fine spray of multiply charged droplets. The (usually) positively charged droplets are directed to the orifice at the front end of the mass spectrometer, held at lower potential. The orifice is the interface between the ion source (atmospheric pressure) and the mass spectrometer (vacuum). During the passage to the orifice, the droplets shrink by evaporation. The resulting increased charge density creates a Coulomb repulsion force that eventually exceeds the surface tension and causes drop explosion (Coulomb explosion) into smaller drops. This process continues until the drops are small enough to desorb analyte ions into the gas phase.

The Analyzer

After the ions generated by the ion source are transported through an electric/magnetic field, the *mass analyzer* sorts the ions according to their mass-to-charge ratio m/z. Currently, various types of analyzers are used in proteomics research, e.g. time-of-flight (TOF), ion trap (IT), quadrupole (TQ), and Fourier transform ion cyclotron resonance (FT-ICR) analyzers. Since these differ in performance, mass accuracy and cost, different designs are chosen for different tasks. We concentrate on the most common analyzer techniques employed in the context of protein identification by MS: MALDI ionization is usually coupled to *time-of-flight* analyzers that measure the mass of intact peptides, whereas ESI is frequently coupled to *ion trap* or *quadrupole* instruments that generate fragmentation spectra of selected peptide ions.

- **Time-of-flight (TOF):** The ions produced by an ionizer as for instance MALDI are accelerated by an electric field, before they drift through a field-free tube towards the detector. Since the electric field strength is the same for all ions, the speed of a particular ion depends solely on its m/z value. Ions with smaller m/z ratio have higher velocity and hence drift faster through the analyzer tube. The detector records the time between the acceleration and the arrival of the ions, from which the m/z values are computed. A linear TOF analyzer with coupled MALDI ionizer is shown in Figure 3.2.

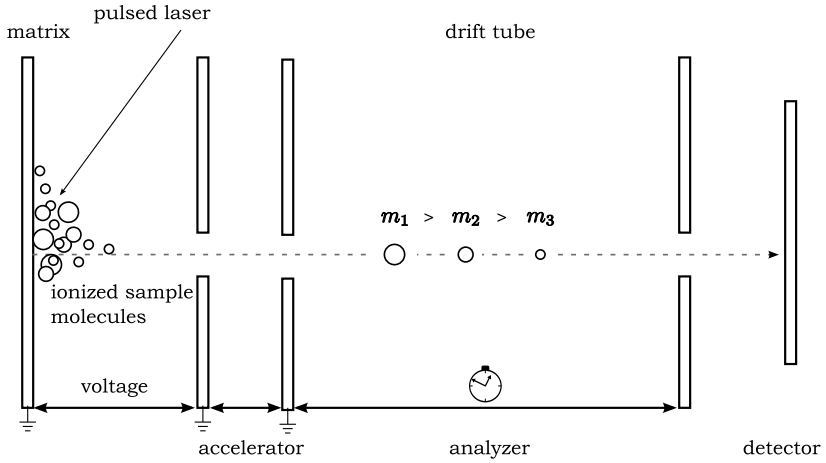


Figure 3.2: Schematic view of a linear MALDI-TOF instrument. During a laser pulse, ions are created from the sample and continuously accelerated by a voltage. Drifting through the field-free tube, the ions are dispersed in time according to their m/z values. Lighter ions reach the detector first.

- **Quadrupole:** A linear quadrupole consists of four metal rods that are arranged in parallel such that each opposing rod pair is electrically connected. Between the rods, an oscillating electromagnetic field is established, forcing the ions to travel through the tube in corkscrew-like trajectories. The quadrupole functions as a mass filter since only ions with a stable trajectory, i.e. with an appropriate m/z value, reach the detector, whereas all other ions are diverted. An entire spectrum is obtained varying the nature of the field.

- **Ion Trap:** Whereas quadrupoles analyze peptide ions "on the fly" during their passage through the analyzer tube, ion traps first collect ions. To this end, the ions are slowed down using helium gas and trapped in a quadrupole field where they are forced into oscillatory trajectories. By applying another field with specific frequency, ions are ejected according to their m/z value. Detecting the ejected ions at different field frequencies produces a spectrum of all peptide ions present in the trap at any given time.

The Detector

The *detector* measures an indicator quantity such as time-of-flight, from which the corresponding m/z values can be deduced. In fact, the detector records the current produced or the charge induced when an ion passes or hits a surface in form of a mass spectrum. For an overview of available techniques, e.g. to transform kinetic energy into an electrical current, the reader is referred to Gross [45]. The strength of the ion current relates to the abundance of ions with that particular m/z value and is thus referred to as "intensity". Note that this does in general not correspond to the number of respective analyte molecules, since the detector is also hit by other particles. This fact complicates a straightforward interpretation of mass spectra [1]. Typically, the whole process is repeated several times because the number of ions leaving the mass analyzer (cf. Figure 3.2) at a particular instance of time is usually quite small. The resulting mass spectra are then averaged to obtain the final spectrum.

3.1.2 Characteristics of Mass Spectra

In nature, elements are usually composed of one or more *isotopes* that occur with different frequencies. Isotopes of an element have the same number of protons in the nucleus, i.e. the same atomic number, but different numbers of neutrons and hence different molecular masses. Therefore, almost all natural molecules occur with different masses according to the *isotope species*, i.e. the isotopes the molecule consists of. If all atom isotopes are those with minimal mass, the isotopic species is called *monoisotopic* and the corresponding molecule mass is termed *monoisotopic mass*. Moreover, the sum of abundance weighted averages of isotopic atom masses yields the *average mass* of the molecule. Table 3.1 displays the natural isotopes of the most common atoms and their frequencies. A list of the 20 standard amino acids with 3- and 1-letter code, molecular composition, monoisotopic and average mass is provided in Table 3.2.

The isotopic distribution of a molecule can be determined from the atoms' isotopic distributions by convolution. The number of distinct isotope species of a molecule $C_{i_C}H_{i_H}N_{i_N}O_{i_O}P_{i_P}S_{i_S}$ consisting of i_C C atoms, i_H H atoms and so on and so far is $(i_C+1)(i_H+1)(i_N+1)\binom{i_O+2}{2}\binom{i_S+3}{3}$, e.g. 72 in the case of glycine and 792 for tryptophan. This follows from the fact that there are $\binom{l+r-1}{r-1}$ possibilities to combine l elements that can choose r types each [24]. The number of isotope species determines the dimension of a molecule's isotopic distribution.

Also in mass spectra, the isotopic distribution is seen. Usually, the monoisotopic peak, referring to the monoisotopic mass of the molecule, is followed by one or more smaller isotopic peaks. Interpreting this information is called isotopic deconvolution. For a closer look at isotopes and isotope patterns in mass spectra, we refer the reader to the article of Böcker *et al.* [24].

Table 3.1: Natural isotopic distribution: Isotopes for the elements CHNOPS, their masses in Dalton and their natural abundances.

Element	Isotope	Mass (Da)	Abundance (%)
Hydrogen	^{1}H	1.007825	99.985
	^{2}H	2.014102	0.015
Carbon	^{12}C	12.000000	98.890
	^{13}C	13.003355	1.110
Nitrogen	^{14}N	14.003074	99.634
	^{15}N	15.000109	0.366
Oxygen	^{16}O	15.994915	99.762
	^{17}O	16.999132	0.038
	^{18}O	17.999161	0.200
Phosphorus	^{31}P	30.973762	100.000
Sulfur	^{32}S	31.972071	95.020
	^{33}S	32.971459	0.750
	^{34}S	33.967867	4.210
	^{36}S	35.967081	0.020

In order to analyze mass spectra, signal processing algorithms are applied to filter noise, normalize the signal level ("baseline correction"), and select mono-isotopic peaks from observed isotopic patterns. Subsequently, peak detection algorithms define ion signals from the processed spectrum. Algorithms for spectrum processing and peak detection are described in [29, 67, 89], to name a few. The identified $^{m}/_{z}$ values together with their measured intensities are recorded in a so-called *peak list*. This is employed for all further tasks such as protein identification explained in the following section. Note that commonly the term "spectrum" is also used in place of "peak list".

Mass accuracy

The mass accuracy of a mass spectrometer indicates the deviation of the instrument's response from the monoisotopic mass of a known entity. It is usually measured in parts per million (ppm). That is to say, a spectrometer with 100 ppm measures a mass of 1000 Da with an error of 0.1 Da. According to [9], current MALDI-TOF instruments have a mass accuracy of better than 50 ppm, and about 10 ppm can be achieved with careful instrument calibration. Modern FT-ICR mass spectrometers operate with an accuracy of approximately 1 ppm. An overview of the resolution, the mass range, the mass accuracy, and the price of leading higher-resolution spectrometers is provided in [10].

Table 3.2: Molecular (monoisotopic and average) masses of the 20 standard amino acids together with their 1- and 3-letter codes and their molecular composition. The molecular formulas are given for residues in a polypeptide chain, i.e. without terminal O- and OH-groups.

Amino acid	Symbols		molecular formula	monoisotopic mass (Da)	average mass (Da)
Alanine	Ala	A	C_3H_5NO	71.037113790	71.079323045
Arginine	Arg	R	$C_6H_{12}N_4O$	156.101111044	156.188746822
Asparagine	Asn	N	$C_4H_6N_2O_2$	114.042927452	114.104467719
Aspartatic Acid	Asp	D	$C_4H_5NO_3$	115.026943030	115.089069711
Cysteine	Cys	C	C_3H_5NOS	103.009184490	103.143711176
Glutamic Acid	Glu	E	$C_5H_7NO_3$	129.042593094	129.116158896
Glutamine	Gln	Q	$C_5H_8N_2O_2$	128.058577516	128.171556905
Glycine	Gly	G	C_2H_3NO	57.021463726	57.052233860
Histidine	His	H	$C_6H_7N_3O$	137.058911874	137.142140206
Isoleucine	Ile	I	$C_6H_{11}NO$	113.084063982	113.160590603
Leucine	Leu	L	$C_6H_{11}NO$	113.084063982	113.160590603
Lysine	Lys	K	$C_6H_{12}N_2O$	128.094963024	128.175293325
Methionine	Met	M	C_5H_9NOS	131.040484618	131.197889547
Phenylalanine	Phe	F	C_9H_9NO	147.068413918	147.178050372
Proline	Pro	P	C_5H_7NO	97.052763854	97.117549470
Serine	Ser	S	C_3H_5NO	87.032028410	87.078627759
Threonine	Thr	T	C_4H_7NO	101.047678474	101.105716944
Tryptophan	Trp	W	$C_{11}H_{10}N_2O$	186.079312960	186.215027571
Tyrosine	Tyr	Y	C_9H_9NO	163.063328538	163.177355085
Valine	Val	V	C_5H_9NO	99.068413918	99.133501417

3.2 Protein Identification by Mass Spectrometry

Mass spectrometry is an emerging technique for the identification of molecular mixtures of proteins, DNA, or metabolites. We concentrate on the identification of protein mixtures, which involves the extraction of a sample from the cell or tissue and protein purification, before separated molecules are measured by MS. In the following, we outline how samples are prepared before they are introduced to the mass spectrometer. Then, we discuss three common methods using MS to identify proteins and peptides, namely peptide mass fingerprinting (PMF), tandem MS, and de novo peptide sequencing. PMF and tandem MS rely on a protein sequence database for the identification, whereas de novo sequencing is a database independent method to deduce the amino acid sequence of a given peptide. However, also the latter is frequently combined with a database search in so-called tag-based approaches.

The continuously growing amount of protein sequences contained in protein sequence databases together with improved MS techniques allow automated protein identification in a high-throughput manner. An overview of the mentioned methods and the corresponding computational identification tools is provided in Matthiesen [85] and in various reviews [1, 7, 35, 87, 95]. More details about experimental techniques are given in the books of Liebler [72], Snyder [114], and Patzkill [96].

3.2.1 Sample Preparation

Before the actual MS analysis, protein extracts are purified. First, proteins are separated from other cell components such as DNA, cellulose, or metabolites. Second, at least in case that one strives for PMF, the proteins in the resulting protein mixture are separated from each other in order to allow reliable MS measurements. There are two common techniques for this *protein separation*, suitable for subsequent analysis by MS: *2D-gel electrophoresis* (2-DE) and *liquid chromatography* (LC). In 2-DE, the proteins are first separated according to their net charge by isoelectric focusing and then, in the second dimension, according to their molecular mass. The separation is usually done by sodium dodecyl sulfate-polyacrylamide gel electrophoresis (2D-SDS-PAGE) which gives a high separation efficiency. In LC, the analyte is dissolved in a liquid (the mobile phase) which moves through the chromatography column (the stationary phase). By interacting with particles of the stationary phase, the analyte is separated. See [96] for a more detailed description of these separation techniques.

For the identification of a single protein, there are generally two approaches referred to as “top-down” and “bottom-up”. In the “top-down” approach, the entire molecule is ionized and introduced to the mass analyzer. On the contrary, in the “bottom-up” approach, the molecule under consideration is first enzymatically cleaved into smaller peptides which are then ionized and introduced to the mass analyzer. This protein digestion is mediated by a *protease*, also called proteolytic enzyme or peptidase, which catalyzes the hydrolysis of peptide bonds. In mass spectrometry applications, so-called endoproteases such as trypsin or pepsin are used that hydrolyze site-specific peptide bonds within the polypeptide chain. For example, trypsin cleaves after arginine and lysine unless the subsequent amino acid is proline. Trypsin is well-suited for MS experiments because the resulting peptide fragments provide at least two protonation sites for efficient ionization [9, 87].

Usually, a “bottom-up” procedure is preferred since the mass of the protein itself is not discriminative enough to identify the protein of interest in a database, whereas the set of peptide masses usually is [123]. Moreover, analyzing an intact protein is difficult because of its large molecular mass, and thus typically requires more expensive MS instruments. We distinguish the following methods, where protein identification can be achieved either by comparing the measured peptide masses to a database, or *de novo*, i.e. without prior knowledge.

3.2.2 Peptide Mass Fingerprinting

Peptide mass fingerprinting (PMF), also referred to as *peptide mass mapping*, was the first available method for protein identification by MS [47] and is still widely used, particularly for high-throughput protein identification [111]. It is a mass-based approach using a reference protein database, where the set of peptide masses is compared to theoretical masses computed for each database entry.

The protein of interest is enzymatically digested into peptides which are measured by the mass spectrometer, and the pattern of the respective peptide m/z values, referred to as *peptide mass fingerprint*, is used for the identification. Considerable amplification yields a reliable mass spectrum which is processed and converted into a peak list as described at the end of Section 3.1. On the other hand, each database sequence is theoretically subjected to the same experimental conditions as the sample protein. First, *in silico* cleavage corresponding to the protease used in the experimental setting yields a set of fragments for each protein sequence. The respective (monoisotopic or average) fragment masses form a theoretical fingerprint which is then compared to the experimental one. A score is assigned to each comparison, and the highest scoring protein is recorded as identification. Ideally, all peptide masses of a mass fingerprint would be shared by the top-ranking protein. However, this is rarely true because of a number of factors such as incomplete digestion with the protease, post-translational modifications, the presence of protein mixtures, or genetic variations [111].

Note that the term "fingerprint" is theoretically not adequate, since the set of peptide masses is not unique. Two peptides with the same multiplicity of amino acids can have the same molecular mass (depending on the isotope species). Furthermore, leucine and isoleucine are isobar, i.e. they have the same atomic composition and hence the same mass distribution. Therefore, peptide mass fingerprints can be stored more precisely taking the intensities into account. Then, the experimental spectrum is compared to theoretical spectra computed *in silico*. See Figure 3.3 for a schematic overview of PMF.

A good measure is sought to assess how well two fingerprints or spectra match. Many strategies have been developed for the comparison of mass spectra (see [87] for a general overview), including the approach of mass spectra alignment by Kaltenbach [54]. The most basic method is the so-called *shared peaks count*. As the name suggests, the number of peaks at the same positions m/z (or within a narrow mass range) in both spectra are counted. The theoretical spectrum with the highest shared peak count gives rise to the best-scoring database sequence. If the corresponding PMF-score exceeds a set threshold, the protein is said to be identified. The definition of this threshold is difficult because it controls the trade-off between false positives (incorrectly identified proteins) and false negatives (protein sequences that are not identified though included in the database). Recent statistical methods determine the optimal threshold such that the false positive rate is below a given percentage [78].

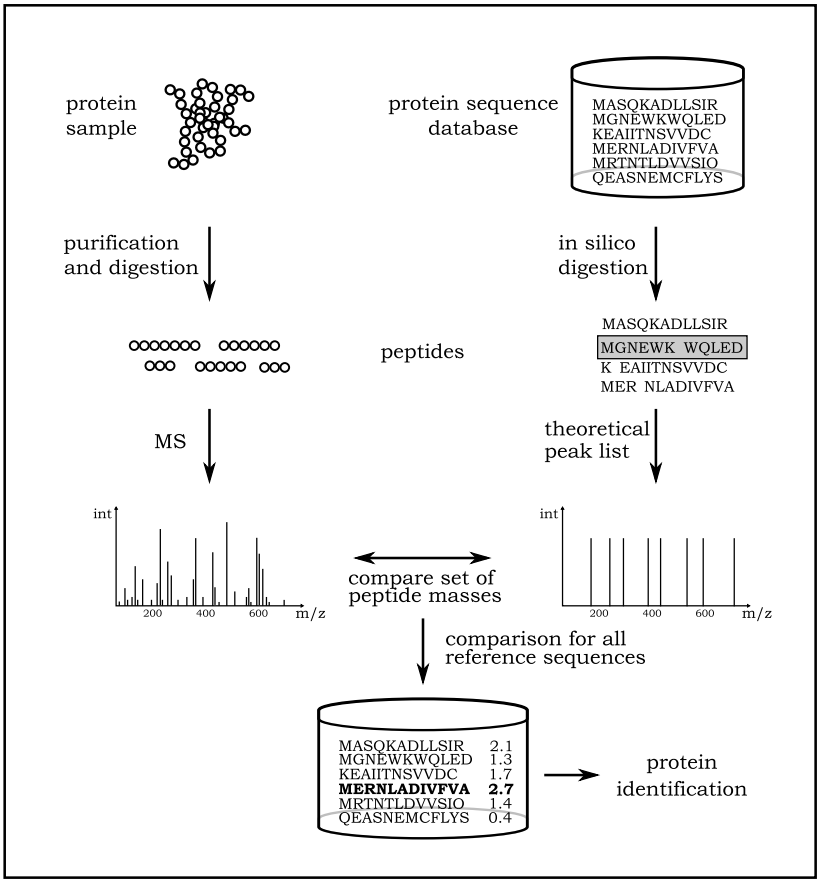


Figure 3.3: Protein identification by peptide mass fingerprinting.

There are many software packages providing protein identification by PMF. A thorough overview of the more popular packages Aldente [75], Mascot [31, 98], MS-Fit as part of ProteinProspector [28], and ProFound [128], as well as comparisons thereof, can be found in [87].

One limitation of PMF is its sensitivity to database size. This is to say that in a larger database the chance of the experimental set of peptide masses to match the masses predicted from database sequences is generally higher. This leads to a decreased confidence of protein identification by PMF in larger databases. Nevertheless, it is reported that PMF scores are quickly computed and in many cases lead to confident identifications. Following McHugh and Arthur [87], there is still a number of recent developments in PMF scoring and applications, which indicates the method's popularity. Today, biologists tend to initially use PMF, and if this fails or remains ambiguous, proceed with more involved but powerful methods such as *tandem mass spectrometry* [128].

3.2.3 Tandem Mass Spectrometry

Tandem mass spectrometry (MS/MS), also called *peptide fragment fingerprinting* (PFF), is a two-stage shotgun proteomics approach to identify and quantify proteins in a large-scale fashion. In contrast to PMF, it requires no protein separation prior to MS-based analysis, but can start directly with a sample of proteins [126]. This is enzymatically digested, resulting in a complex peptide mixture, which is subsequently separated by LC. At the first stage, the peptides are then ionized and subjected to the mass spectrometer, yielding a spectrum as for PMF. At the second stage, selected peptide ions, often referred to as "precursor" ions, are further dissociated into smaller fragment ions. This is achieved in the collision cell of the mass spectrometer, where the peptide collides with a noble gas, and peptide bonds are broken by a process termed *collision-induced dissociation* (CID). These fragments are again measured by MS, yielding the MS/MS spectrum, i.e. the m/z values and intensities of all resulting fragment ions generated from an isolated precursor ion. Since the majority of the peptide ions fragment only once, the fragments measured correspond to either prefixes or suffixes of the original peptide sequence. Depending on the fragmentation site along the peptide backbone, different types of ions with slightly different masses are generated. The most common ion types are the so-called "b-ions" (corresponding to prefixes) and "y-ions" (corresponding to suffixes).

Protein identification again relies on a protein sequence database, similar to the PMF procedure. The experimentally measured peptide MS/MS spectra are compared to theoretical MS/MS spectra, computed for all peptides in the database. A similarity score is assigned to each comparison, and the highest scoring peptide is returned as identification. If several peptides from one database protein have been identified by MS/MS, the corresponding protein sequence is reported as candidate for the protein identification. Usually, a significance value is computed to find the "best" candidate.

Several software tools have been developed to search MS/MS spectra against a protein sequence database, including the two most popular software packages SEQUEST [124, 125] and Mascot [31, 98]. The latter tool allows the interpretation of both PMF and MS/MS data. The textbook of Matthiesen [85] and the review article [87] contain more detailed information on automated protein identification tools and MS/MS database searching.

3.2.4 De novo Peptide Sequencing

Of course, the presented techniques assume that the considered protein or at least a biologically related protein sequence is already contained in the regarded database. In contrast, de novo peptide sequencing uses the information gained in an MS/MS experiment to deduce the primary sequence of a peptide, i.e. the amino acid sequence that gave rise to the MS/MS spectrum, without querying a database. The basic idea is that, for an ideal fragmentation process and an ideal

mass spectrometer, the MS/MS spectrum would contain peaks corresponding to all prefixes and suffixes of the peptide sequence. Thus, the sequence could be determined by converting the mass differences between consecutive ion peaks to amino acids with appropriate molecular masses. Due to the restricted mass accuracy of mass spectrometers and the high complexity of most MS/MS spectra, de novo sequencing tools often return only short, unambiguous sequences called "peptide sequence tags". These tags are then searched against a database, in which a candidate is expected to match multiple tags. Corresponding algorithms are based on finding a path of maximal length through the so-called spectrum graph [33]. An overview of de novo sequencing algorithms and current tag-based approaches is provided in the review of McHugh and Arthur [87].

3.3 Measuring Proteolytic Fragments with a PAA

After this excursion, we now concentrate on protein identification by PMF. As indicated, PMF scores determined by the comparison of peak lists in standard PMF packages are not deterministic, but depend on the size of the considered database. For this reason, it is desirable to define a comparison strategy yielding significance values based on a thorough null model. Kaltenbach [54] introduced p-value scores that are independent of the size of the sequence database and that reduce the influence of sequence length and peak list size. The underlying null model represents protein sequences as i.i.d. random strings, i.e. as sequences of independent random variables that take values from the alphabet of amino acids according to the amino acid composition estimated from the Swissprot database [8, 16].

In this section, we present a PAA that generates random peptide fragments that obey the rules of a related cleavage enzyme and provides statistics as the mass and the joint length-mass distribution of these fragments. This mimics the enzymatic cleavage reaction, e.g. with Trypsin as cleavage agent, and the statistics of peptides usually measured in an MS experiment. In particular, we use the theoretical statistics to calculate the *mass occurrence probability*, i.e. the probability that the peptide mass fingerprint of a given protein contains at least one peptide of a certain mass. This, in turn, provides a significance value for PMF identifications as explored in [54]. However, we formulate the task in the more general PAA framework using a first-order Markovian (M1) source to model random proteolytic fragments. Furthermore, we analyze the influence of incomplete cleavage and post-translational modifications by means of appropriate modifications of the PAA.

Following the work mentioned above, we state the concept of a so-called *cleavage scheme* to formalize the cleavage "rules" of the proteolytic enzyme employed. Moreover, we briefly summarize the characteristics of the resulting peptides and recall the definitions necessary to calculate the mentioned statistics. Then, we present an appropriate PAA as well as the recurrences to compute fragment statistics and mass occurrence probabilities.

3.3.1 In silico Digestion

As described above, in the course of PMF all protein sequences in a (comprehensive) database are digested *in silico*, and the resulting fragments are measured e.g. by summing up the monoisotopic or average amino acid masses listed in Table 3.2. This yields a theoretical fingerprint which is then compared to the empirical one. The more realistic the masses of the theoretical fragments, the better the result from the comparison strategy. This applies in particular to the comprehension of isotopic distributions and to the inclusion of post-translational modifications and missed cleavage sites. Furthermore, for the comparison of mass spectra, a more realistic null model will provide more reasonable significance values.

In order to model the proteolytic digestion of a (random or fixed) protein, we take care of the cleavage rules of the enzyme employed in the experiment. For many digestion agents, cleavage rules depend on one or two adjacent amino acids (see [54, 121] for a list of proteases along with their cleavage rules). For this reason, we adopt the following definition:

Definition 3.3.1 (Cleavage scheme)**.** *Let* $\Sigma = \{A, \ldots, Z\} \backslash \{B, J, O, X, Z\}$ *denote the alphabet of amino acid one-letter codes. A* cleavage scheme (Γ, Π) *is a pair of a set* $\Gamma \subset \Sigma$ *of* cleavage *characters and a possibly empty set* $\Pi \subset \Sigma$ *of* prohibition *characters. If these sets are disjoint, i.e.* $\Gamma \cap \Pi = \emptyset$*, we refer to this as* standard cleavage scheme*. Further, if* $\Pi = \emptyset$*, we call it a* simple cleavage scheme*.*

Strings $P = P_1 P_2 \in \Gamma\bar{\Pi}$ are referred to as *cleavage patterns*. Throughout this chapter, we use the standard cleavage scheme $\Gamma = \{K, R\}$, $\Pi = \{P\}$ corresponding to the widely used enzyme Trypsin, which cleaves after lysine (K) and arginine (R) unless these are followed directly by proline (P). However, the model we will present is capable of all proteases with simple and standard cleavage schemes.

The cleavage scheme determines the *cleavage sites* which describe the *fragmentation* of a protein into peptides, also referred to as (proteolytic) *fragments*. We model amino acid sequences as strings over the alphabet Σ of one-letter codes. To this end, let $\{S_i\}_{i \in \mathbb{N}_0}$ be a stochastic process with index set $\mathbb{N}$ and values in Σ that describes *random* amino acid sequences. Since we analyze protein and peptide sequences, we deal with finite prefixes $S = S_1 S_2 \ldots S_\ell$ that refer to random amino acid sequences of length ℓ. For the treatment of infinite strings and a thorough discussion of the necessary adjustments from infinite to finite strings, the reader is referred to [54].

Occurrences of cleavage patterns within a protein sequence define the cleavage sites where the sequence is split into consecutive, non-overlapping substrings, the fragments.

Definition 3.3.2 (Cleavage site)**.** *Let* S *be a (random or fixed) string of length* ℓ *over* Σ*. Each* $C_i(S)$ *with*

$$C_i(S) := \min\left(\{C_{i-1}(S) < k < \ell \,|\, S_k \in \Gamma, S_{k+1} \notin \Pi\} \cup \{\ell\}\right)$$

and $C_0(S) = 0$ is called a cleavage site *of S and indicates the occurrence of a cleavage pattern. Note that the last character in S always defines a cleavage site, namely for the case that the former set is empty. Further, if S is known from the context, we write C_i instead of $C_i(S)$.*

We denote the index of the last cleavage site by $N_C(S)$. Hence, $C_{N_C(S)} = |S|$. The cleavage sites determine the fragments resulting from proteolytic digestion. A fragment is a substring of S, beginning directly after a cleavage site (if no prohibition character follows) and ending with the subsequent cleavage site.

Definition 3.3.3 (Fragment; fragmentation)**.** *For each i, $1 \leq i \leq N_C(S)$, the substring $F_i(S) := S[C_{i-1} + 1, C_i]$ is called the i^{th}* fragment *of S. The family $(F_i(S))_{i=1}^{N_C(S)}$ of fragments constitutes the* fragmentation *of S.*

In the following we will omit the dependence on S and write F_i and N_C if S is clear from the context. The size of the fragmentation, i.e. the number of fragments resulting from the enzymatic cleavage, is given by the index N_C of the last cleavage site. A sketch of the fragmentation of an amino acid string according to tryptic cleavage is shown in Figure 3.4.

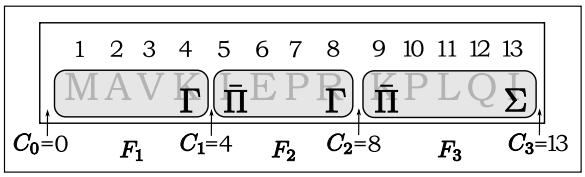


Figure 3.4: Sketch of the fragmentation of a short protein string $S = MAVKIEPRKPLQI$ into $N_C(S) = 3$ fragments according to tryptic cleavage rules.

We distinguish the first fragment F_1 from following fragments F_+ due to their distinct structure. While the first fragment may start with a prohibition character, following fragments do not. Further, the structure of all following fragments but the last is the same, which is to say that the statistics of following fragments are i.i.d. (depending on the remaining string length) [54, 121]. The last fragment deserves particular attendance since it does not necessarily end with a cleavage character because of the finiteness of protein sequences. As mentioned before, the last fragment ends with the last character of the protein sequence irrespective of its type. This is reasonable since the last part cut by the protease is also measured by the mass spectrometer. We will use the abbreviated form $F_\circ$ to denote any fragment.

Remark 3.3.4. *The fragmentation of a string under a cleavage scheme can be seen as renewal process (see e.g. [44, Section 8.3] or [39, Chapters XIII and XIV]) with delay C_1 and the following fragment lengths as inter-renewal sequence. Since the cleavage sites correspond to recurrent events, the fragmentation can indeed be modeled as a regenerative process [121].*

Next, we state an appropriate PAA that models random cleavage fragments and yields their molecular mass distribution according to different isotope species.

3.3.2 Automaton Construction

The desired PAA should model the proteolytic digestion of random protein sequences, yielding statistics as the length and the mass distribution of resulting fragments. As mentioned above, the first fragment has to be distinguished from following fragments. Hence, we will construct two alternative PAAs, one to measure first fragments, one designed for following fragments.

According to Section 2.2, a PAA consists of three components: a Markov chain generating a random sequence of operands, a set of state emissions referred to as weight distributions, and a set of operations associated with the states, which determine the distribution of a sought value. Recall that the state process is denoted $(Y_n)_{n\in\mathbb{N}_0}$, the sequence of emission is represented by $(Z_n)_{n\in\mathbb{N}_0}$, and the sequence of values is identified $(V_n)_{n\in\mathbb{N}_0}$.

In this application, the underlying Markov chain should represent random proteolytic fragments. To this end, we model protein sequences as random i.i.d. or first-order Markovian strings over the alphabet Σ and apply appropriate cleavage rules as described above. Amino acid frequencies (for i.i.d. strings) as well as initial and conditional amino acid distributions (for Markovian strings) are estimated from the Swissprot database. Since our aim is to "measure" the mass of generated fragments, we choose the amino acids as states, and their respective isotopic distributions (cf. Section 3.1.2) as weight distributions. Moreover, since the sought value is the accumulated fragment mass, the operation associated to each state is "+". Thus, in terms of emission and value processes, we have $V_l = V_{l-1} + Z_l$ with $V_0 = 0$, which is to say that we seek a PAA $\big(Q, T, q_0, E, e = (e_q)_{q\in Q}, \mathbb{N}_0, n_0 = 0, (\theta_q(v, e) = v + e)_{q\in Q}\big)$.

We build the Markov chain (Q, T, δ_{q_0}) on the state space

$$Q = \{\epsilon\} \cup \{\sigma \in \Sigma\} \cup \{\zeta\}, \tag{3.3.1}$$

i.e. Q consists of a start state $q_0 = \epsilon$, the 20 standard amino acids, and one final state $\mathcal{F} = \{\zeta\}$. Transitions from the start state to any other state correspond to the initial amino acid distribution $p^0 = (p^0_\sigma)_{\sigma\in\Sigma}$, i.e. the relative frequencies of amino acids at the first position of protein sequences, estimated from the Swissprot database. Transitions T from one amino acid to another are given by the conditional frequency matrix $P = (P_{ij})$ with $P_{ij} = \mathbb{P}(s[k+1] = j \,|\, s[k] = i)$ (for any k) counted from protein sequences s. To include the cleavage rules, transitions outgoing from a cleavage character have to be specified differently: on the one hand, the transition probability $T_{\gamma\pi}$ from a cleavage character $\gamma \in \Gamma$ to a prohibition character $\pi \in \Pi$ is given by the conditional amino acid frequency. On the other hand, the probability of the complementary event, i.e. one minus the sum of all transitions from γ to Π, is assigned to the transition from γ to ζ. Hence, the probability

to transit from γ to any state outside $\Pi \cup \mathcal{F}$ is zero. This intuitively makes sense: either the fragment ends with a cleavage character, or cleavage is suppressed by a prohibition character, which is the only case in which a cleavage character may occur within a fragment. The stochastic matrix of transition probabilities for states $u, v \in Q$ thus reads

$$T_{uv} = \begin{cases} p^0_\sigma & \text{if } u = \epsilon, v = \sigma \in \Sigma, \\ P_{\sigma\sigma'} & \text{if } u = \sigma \in \bar{\Gamma}, v = \sigma' \in \Sigma, \\ P_{\sigma\sigma'} & \text{if } u = \sigma \in \Gamma, v = \sigma' \in \Pi, \\ 1 - \sum_{\pi\in\Pi} P_{\sigma\pi} & \text{if } u = \sigma \in \Gamma, v = \zeta, \\ 0 & \text{otherwise}. \end{cases} \tag{3.3.2}$$

For following fragments, we account for two things: since the initial distribution p^0 for first fragments is estimated from the very first amino acid in protein sequences, we consider the i.i.d. amino acid frequencies $p = (p_\sigma)_{\sigma\in\Sigma}$ instead. Moreover, the probability that a following fragment starts with a prohibition character is zero due to the structure of proteolytic fragments. Hence, for the PAA treating following fragments we modify the transition matrix such that the initial transitions for $u = \epsilon$ are given by

$$T_{uv} = \begin{cases} \frac{p_\sigma}{1-\sum_{\pi\in\Pi} p_\pi} & \text{if } v = \sigma \in \bar{\Pi}, \\ 0 & \text{otherwise}. \end{cases} \tag{3.3.3}$$

As said before, the weight distribution associated to a state corresponds to the respective amino acid's isotopic distribution. In line with [54], we scale the molecular amino acid masses to integers, using a preset mass precision Δ_m. This reflects the accuracy of known molecular masses and the finite mass precision of the spectrometer. For instance, a MALDI-TOF instrument has a mass accuracy of $50\,\mathrm{ppm}$, i.e. $\pm 0.1\,\mathrm{Da}$ for a typical fragment mass of $2000\,\mathrm{Da}$. Thus, it is appropriate to multiply all masses by a factor of 10 and round to the nearest integer, i.e. to consider natural masses up to one decimal. This corresponds to a mass precision of $\Delta_m = 0.1$. Then, $m(\sigma) = \text{round}(m^*(\sigma)/\Delta_m)$ determines the integer mass of the natural mass $m^*(\sigma)$ of amino acid σ (given in Da). The molecule's natural mass $m^*(\sigma)$ depends on the isotope species. We can either use the monoisotopic or average mass or take the entire distribution of natural masses of the distinct isotope species into account.

Denoting the largest amino acid integer mass by $m_{\max}$, we define the emission set $E := \{0, 1, \ldots, m_{\max}\}$ and $|E|$-dimensional weight distributions $e_q : E \to [0, 1]$ for all $q \in Q$. If we design the PAA to use the monoisotopic or average amino acid masses instead of their isotopic distributions, the weight distributions are given by $|E|$-dimensional Dirac measures e_q assigning probability 1 to $m(q)$. For convenience, we set $m(\epsilon) = m(\zeta) = 0$. Note that, as explained in Section 2.2, the weight distribution assigned to ϵ is redundant as it is covered in the initial condition of the recurrence equation.

Corollary 3.3.5. *The PAA* $\big(Q, T, q_0 = \epsilon, E = \{0, 1, \ldots, m_{max}\}, e = (e_q)_{q\in Q}, \mathbb{N}_0, n_0 = 0, (\theta_q(v, e) = v + e)_{q\in Q}\big)$ *with* Q *and* T *given in equations* (3.3.1) *and* (3.3.2) *(with mod-*

ification (3.3.3) *for following fragments), and weight distributions* $e_q : E \to [0, 1]$ *for all* $q \in Q$*, measures the molecular mass distribution of randomly generated proteolytic fragments.*

The construction requires $\mathcal{O}(|Q|^2 + |E|)$ time and space which is constant since $|Q| = 22$ and $|E|$ depends on the mass precision. For mass precision $\Delta_m = 0.1$, we have $|E| = 1892$. Using doubles with 8 byte each, we need 18.56 kb to store the transitions and the weight distributions of the PAA. The constructed PAA is shown in Figure 3.5.

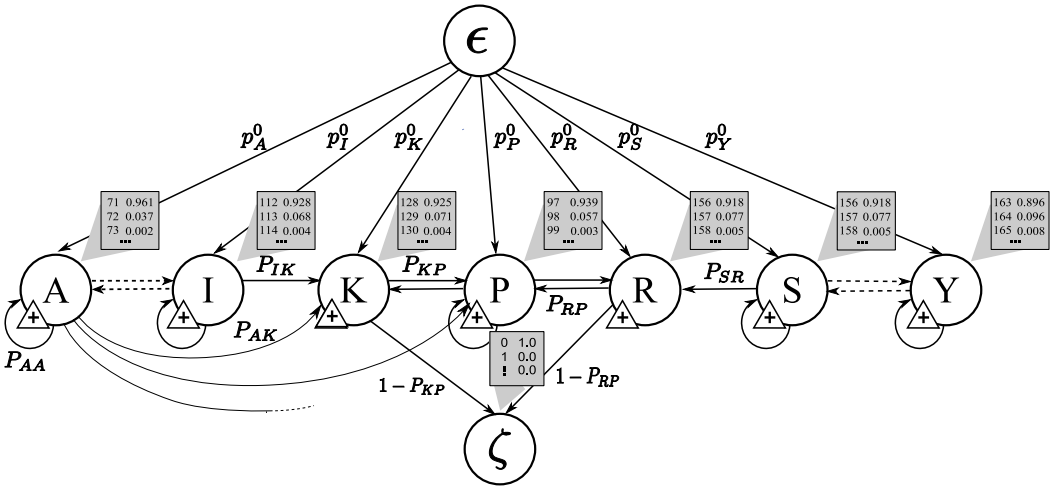


Figure 3.5: Sketch of the PAA measuring the mass of the first fragment resulting from tryptic cleavage of a random polypeptide. Following fragments are handled by a modified PAA where the initial distribution is normalized by $1 - p_P$ and start in P is prohibited since following fragments cannot start with a prohibition character. For the sake of simplicity, not all transitions are shown. The states' weight distributions correspond to the respective isotopic distributions. Here, integer masses for precision $\Delta_m = 1$ are displayed. The operation associated to each state is "+".

Remark 3.3.6. *In the presented model we can easily account for additional masses that are not associated to a specific character. These masses, shared by any proteolytic fragment, are assigned to the start state* ϵ*. In particular, in the course of proteolytic cleavage, the protease hydrolyzes the peptide chain, thus adding an H and an OH group to the N- and C-terminus, respectively. This adds approximately 18 Da to each peptide fragment. Moreover, in the ion source of the mass spectrometer, the fragment is ionized, which results in another mass change. For instance, MALDI ionization adds a single proton of mass approximately 1 Da to a fragment. Hence, in a MALDI-TOF experiment, all fragment masses will be augmented by about 19 Da. We can cover this fact by setting* $m(\epsilon) = n_0 = {}^{19}/_{\Delta_m}$*.*

3.3.3 Fragment Statistics

By means of the constructed PAA, we are now able to compute fragment statistics as the length distribution, the joint length-mass distribution, and in particular the desired mass occurrence probabilities. We denote the length of a fragment by $L(F_\circ)$ and the (integer) mass of a fragment by $M(F_\circ)$. The distribution of fragment length can be deduced directly from the distribution of the number of cleavage sites (see [54]) because $L(F_i) = C_i - C_{i-1}$ for all $1 \leq i \leq N_C$. Furthermore, the length of a random fragment generated by the PAA corresponds to the number of steps until the underlying Markov chain (Q, T, δ_ϵ) arrives in the final state ζ, not counting the last transition to ζ. This is to say that the length distribution of a proteolytic fragment is given by the distribution of the Markov chain's arrival time in the final state ζ. Using the Chapman-Kolmogorov equation, this is given by

$$\lambda_\circ(k) := \mathbb{P}(L(F_\circ) = k) = \mathbb{P}(Y_{k+1} = \zeta) = (\delta_\epsilon T^{k+1})_{[\zeta]}\,,$$

where $[q]$ denotes the index of state $q \in Q$. The choice of the Markov chain parameters accounts for the first or the following fragment, respectively.

The joint state-value distribution $f_t(q, v) = \mathbb{P}(Y_t = q, V_t = v)$ discussed in Section 2.2 yields the joint length-mass distribution $\nu_\circ(k, m) := \mathbb{P}(L(F_\circ) = k, M(F_\circ) = m)$ of a fragment, namely

$$\nu_\circ(k, m) = f_{k+1}(\zeta, m) = \mathbb{P}(Y_{k+1} = \zeta, V_{k+1} = m)\,.$$

In order to state the corresponding recurrences (cf. Equation (2.4.1)), let us first define the *mass support* of a particular state.

Definition 3.3.7 (Mass support)**.** *The set $\mathcal{MS}(q) := \{m \in E \,|\, e_q(m) > 0\}$ of integer masses with positive emission probabilities is called the* mass support *of state $q \in Q$.*

Indeed, as we use the amino acids' isotopic distributions, the mass support of an amino acid state corresponds to the set of integer masses derived from natural masses of possible isotope species. The mass support of ϵ and ζ equals $\{m = 0\}$ (unless additional masses should be taken into account). According to our introduction to the PAA framework, the following recurrence relations hold:

$$f_k(q, m) = \sum_{q' \in Q} \sum_{m' \in \mathcal{MS}(q)} f_{k-1}(q', m - m') T_{q'q} e_q(m') \quad \text{for } q \in Q, k \geq 1 \tag{3.3.4}$$

with initial condition $f_0(\epsilon, 0) = 1$ or $f_0(\epsilon, n_0) = 1$, respectively. The particular parameters account for the kind of fragment.

From the joint length-mass distribution $\nu_\circ(k, m)$, we derive the distributions of fragment length and mass by marginalization. The length distribution $\lambda_\circ(k)$ investigated above can also be determined from $\nu_\circ(k, m)$ by summing over all possible fragment masses, i.e.

$$\lambda_\circ(k) = \sum_{m \in \mathbb{N}_0} \nu_\circ(k, m)\,.$$

Analogously, we determine the fragment mass distribution $\mu_\circ(m) := \mathbb{P}(M(F_\circ) = m)$ by summing over all possible fragment lengths, i.e.

$$\mu_\circ(m) = \sum_{k \in \mathbb{N}} \nu_\circ(k, m)\,.$$

Indeed, we can estimate upper bounds $\mu_{\max}$ and $\lambda_{\max}$ for the maximal integer fragment mass and length detectable by the MS instrument. If $\mu_{\max}$ denotes the maximal integer mass that is detectable by the mass spectrometer, this naturally bounds the maximal fragment mass. Further, if $m_{\min}$ identifies the minimal amino acid mass, i.e. the integer molecular mass of glycine (G), then

$$\lambda_{\max} = \left\lfloor \frac{\mu_{\max}}{m_{\min}} \right\rfloor$$

in order to ensure $\lambda_{\max} \cdot m_{\min} \leq \mu_{\max}$. Furthermore, the joint length-mass distribution together with the length distribution yields the mass avoidance probability, i.e. the probability that a fragment has length k, but not mass m: $\bar{\nu}_\circ(k, m) := \lambda_\circ(k) - \nu_\circ(k, m)$.

The molecular mass of a fragment is determined by the constituting character masses. In particular, for a fixed fragment length l, the mass distribution $\nu_\circ(l, \cdot)$ is given by the convolution of l character weight distributions. Hence, one can identify whether a given mass is decomposable over the set of masses supported by the individual characters. An efficient mass decomposition algorithm has been presented by Böcker and Lipták [15].

We go a step further as we compute the mass occurrence probability of mass m, i.e. the probability that the fragmentation of a random protein sequence contains at least one fragment of mass m. To this end, we introduce the waiting time $W(m) \equiv W(S, m)$ until the first occurrence of a fragment of mass m in the fragmentation of a random finite string S. Or to put it differently, we seek the first index in S where a fragment $F_\circ$ of mass $M(F_\circ) = m$ ends:

$$W(m) := \min\{1 \leq i \leq |S| \,|\, \exists\, k, 1 \leq k \leq N_C(S) : C_k = i, M(F_i) = m\}\,, \tag{3.3.5}$$

where we define $W(m) = \infty$ if the minimum is taken over the empty set, i.e. if the mass m cannot be composed as fragment mass. In terms of the waiting time, the mass occurrence probability is defined as $o^{|S|}(m) := \mathbb{P}(W(m) \leq |S|)$, i.e. the probability that S contains a fragment of mass m. For the computation, we follow [54] as we deduce $o^{|S|}(m)$ from the probability $\bar{o}^{|S|}(m)$ of the complementary event that no fragment of mass m occurs:

$$\bar{o}^{|S|}(m) := \mathbb{P}\big(W(m) > |S|\big) = \mathbb{P}\big(M(F_1) \neq m, \ldots, M(F_{N_C(S)}) \neq m\big)\,.$$

Although fragment masses are not independent, the mass of a fragment becomes conditionally independent of the remaining masses once its length is known. This fact is the basis for the recurrence equation stated in Recurrence (3.1), adapted from [54].

Recurrence 3.1 Recurrence to compute the mass occurrence probability $o^{|S|}(m) = \mathbb{P}(W(m) \leq |S|)$ of mass m in the fragmentation of string S.

$$\bar{o}^{|S|}(m) = \sum_{k=1}^{|S|} \bar{o}_{+}^{|S|-k}(m) \cdot \bar{\nu}_1(k, m), \tag{3.3.6}$$

with initial condition $\bar{o}^0(m) = 1$ and

$$\bar{o}_{+}^{|S|}(m) = \sum_{k=1}^{|S|} \bar{o}_{+}^{|S|-k}(m) \cdot \bar{\nu}_{+}(k, m), \tag{3.3.7}$$

also with initial condition $\bar{o}_{+}^{0}(m) = 1$. Finally, $o^{|S|}(m) = 1 - \bar{o}^{|S|}(m)$.

3.3.4 Missed Cleavages and Post-translational Modifications

Protein identification by MS using a bottom-up approach as explained earlier is complicated by the occurrence of partial enzymatic cleavage which results in peptides with internal missed cleavage sites. Proteases work with different efficiency. They perform best at a particular pH, e.g. Trypsin is a basic protease with a pH optimum of 7 to 8 as it is found in the small intestine. The cleavage of a protein into peptides may be incomplete due to inadequate conditions or the fact that proteases themselves tend to be cleaved by other protease molecules. Moreover, the specificity of the cleavage agent, i.e. the cleavage pattern defined by the enzymatic mechanism, is simplified in most search tools, since actually "it is likely to be influenced by other residues in close proximity of the cleavage site in addition to other factors such as local conformation, tertiary structure and experimental conditions" as stated in [111]. Even for Trypsin, which is reported to have a high cleavage specificity, incomplete digestion is not uncommon, but tryptic peptides usually contain about one missed cleavage site [85, 111].

In order to account for missed cleavages, we modify the presented PAA by adjusting the transition probabilities outgoing from a cleavage character. Assuming that only every second cleavage site results in a fragment, we multiply the transition probabilities from any cleavage character $\gamma \in \Gamma$ to ζ by $1/2$. The remaining half is distributed to the transitions from γ to any other amino acid state $\sigma \in \bar{\Pi}$, corresponding to missed cleavages. Thus, for each $\gamma \in \Gamma$,

$$T_{\gamma v} = \begin{cases} P_{\gamma\pi} & \text{if } v = \pi \in \Pi, \\ (1 - \sum_{\pi \in \Pi} P_{\gamma\pi})/2 & \text{if } v = \zeta, \\ P_{\gamma\sigma}/2 & \text{if } v = \sigma \in \bar{\Pi}, \\ 0 & \text{otherwise}. \end{cases}$$

Corresponding fragment statistics are shown in Section 3.4. In addition, one could even include information about missed cleavage patterns, i.e. amino acid residues favoring missed cleavages. As reported in recent articles [88, 111, 117, 127], it

is more likely to observe one of D, E, G, M, or S following a cleavage character than any other non-prohibition character. Besides, the cleavage characters K and R themselves are very unlikely to occur in direct neighborhood of a cleavage site. These propensities could be taken into account defining a weight function according to empirically observed frequencies. Moreover, any other rate than $1/2$ of missed cleavages can be analyzed.

Another aspect influencing the characterization of a protein is the fact that many proteins are modified after translation. A *post-translational modification* (PTM) is a chemical process that changes the properties of a protein. There exists a whole zoo of PTMs that have different impact on the translated polypeptide [81]. Namely, the modification changes the function of the protein, or it controls the activity state of an enzyme, or it is necessary to transform a precursor molecule into the active protein as for the hormone insulin. Further, it can determine the protein's localization or interactions with other proteins. The covalent modification consists in the addition of functional groups such as acetate or phosphate, in the change of the nature of an amino acid, or in structural changes such as the formation of disulfide bridges. Putative modifications can be discovered by means of MS, comparing the mass measured to the mass expected for the identified protein or peptide [6, 81, 119].

Examples of common modifications include phosphorylation (+80 Da) of tyrosine (Y), serine (S), or threonine (T) residues, acetylation (+42 Da) found on the N-terminus or on lysine (K) residues, and methylation (+14 Da or +28 Da) of arginine (R) or lysine (K). An overview of important PTMs along with the resulting mass shifts in Da is provided in the review of Mann and Jensen [81].

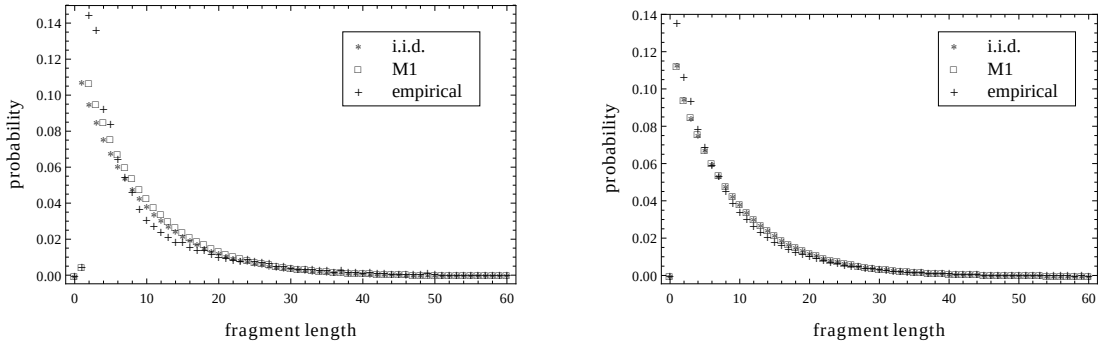
Our aim is to incorporate PTMs in the PAA presented above in order to allow for even more reliable significance values for PMF protein identification, provided that PTMs are considered in the computation of theoretical masses of *in silico* digested database sequences. To this end, we modify the weight distributions associated to the model states. For each amino acid state $q \in Q$ bearing a potential modification, the respective mass shift is included into the weight distribution. That is to say, if a modification introducing a mass shift Δ occurs with frequency $r_{\Delta,q}$ at amino acid q, the masses $m + \Delta$ for all $m \in \mathcal{MS}(q)$ are added to the weight distribution with probability $r_{\Delta,q} \cdot e_q(m)$. These probabilities have to be subtracted from the respective $e_q(m)$ values in order to ensure that the weight distributions remain probability measures. Reasonable frequencies of PTMs associated to particular amino acids are provided in a study of Tsur *et al.* [119].

Example 3.3.8 (Weight distribution of lysine)**.** *Lysine may for instance be methylated once (+14 Da) or twice (+28 Da). Following [119], we assume that single methylation of K occurs with a frequency of* 0.0239 *and dimethylation occurs with probability* 0.0680*. We assume the simplified isotopic weight distribution* $e_K = \{0.925, 0.071, 0.004\}$ *(all zero entries omitted) for mass support* $\{1281, 1291, 1331\}$ *for mass precision* $\Delta_m = 0.1$*. Including the mentioned PTMs, i.e. adding +14 and +28 to each supported mass, the mass support would be extended to*

$\{1281, 1291, 1295, 1305, 1309, 1319, 1331, 1345, 1359\}$*, and the associated weight distribution would read* $\{0.8399925, 0.0644751, 0.0221075, 0.0629, 0.0016969, 0.004828, 0.0036324, 0.0000956, 0.000272\}$*.*

3.4 Results: Comparison of Fragment Statistics

We implemented the presented PAA and computed the statistics described above for a mass precision of $\Delta_m = 0.1$. As mentioned before, we modeled proteolytic fragments by first-order Markovian strings, while Kaltenbach [54] used i.i.d. strings. Our aim is to investigate whether the more complex model influences the quality of fragment statistics. To this end, we also implemented the PAA based on i.i.d. parameters. We compared the length distribution computed for first and following fragments under the Markov model (M1) to those determined under the i.i.d. model and to the empirical frequencies estimated from *in silico* derived fragments of Swissprot sequences, see Figure 3.6. For first fragments, this shows that the M1 model fits the frequencies of short fragments better, but overestimates the frequency for moderate lengths ($8 \leq L(F_\circ) \leq 23$) more than the i.i.d. model. This is to say that also the conditional frequencies do not capture the inhomogeneous nature of the distribution of tryptic cleavage characters within protein sequences.



(a) Length distribution of first fragments.

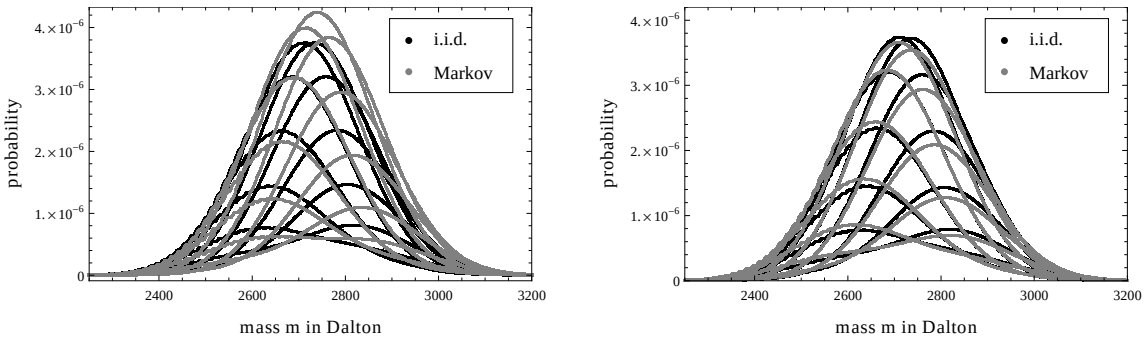
(b) Length distribution of following fragments.

Figure 3.6: Comparison of the theoretical length distributions of first and following fragments resulting from *in silico* digestion of random i.i.d. or Markovian (M1) strings to their empirical counterpart estimated from Swissprot sequences.

For following fragments, the theoretical values are nearly identical. The reason for this is that we use the i.i.d. frequencies instead of p^0, and the Markov conditional frequencies P_{KP} and P_{RP} do not differ much from the Swissprot frequency p_P. Hence, the fragmentation is very similar under both models. As explored in [54], we also observed that the length distribution of following fragments can be approximated by a geometric distribution with parameter $p = 1/\mathbb{E}(L(F_+))$.

The comparison of theoretical length distributions shows that the difference is stronger for first fragments. Here, the difference is mainly introduced by the initial character distribution, which enables the Markov model to better describe the beginning of protein sequences.

Taking the masses into account, we considered the joint length-mass distribution of first and following fragments. The theoretical distributions for fragments of length 25 under i.i.d. and Markov sequences are shown in Figure 3.7.



(a) Length-mass distribution of first fragments of length 25.

(b) Length-mass distribution of following fragments of length 25.

Figure 3.7: Theoretical length-mass distributions of first and following fragments of length 25 resulting from i.i.d. and Markov sequences.

Here, we again observe that the distributions differ more in the case of first fragments. The underlying Markov model shifts the statistics towards heavier fragments.

Using the length distribution and the joint length-mass distribution of first and following fragments as described in Recurrence 3.1, we derived the mass occurrence probability, i.e. the probability $\mathbb{P}(W(m) \leq |S|)$ that mass m occurs in the fragmentation of sequence S. In Figure 3.8, the i.i.d. and M1 theoretical distributions are compared to their empirical counterpart for proteins of length 300. Along with [54], we consider only masses up to 200 Da; larger fragment masses were insufficiently observed in Swissprot sequences in order to estimate their frequencies.

The predicted mass occurrence probabilities shown in Figure 3.8 were computed from PAAs using the monoisotopic amino acid distributions. In addition, isotope species were recognized when using isotopic amino acid frequencies (not shown). Within the considered mass range, the most prominent masses refer to fragments of length one, namely the cleavage characters K and R, which occur with probability of about 0.8. Furthermore, there exist fragments of length two, e.g. GK, that fall into this mass range and occur with relatively high frequency of about 0.2. The agreement between theoretical and empirical frequencies of single character fragments (namely K and R) is better in the case of Markov sequences. Nevertheless,

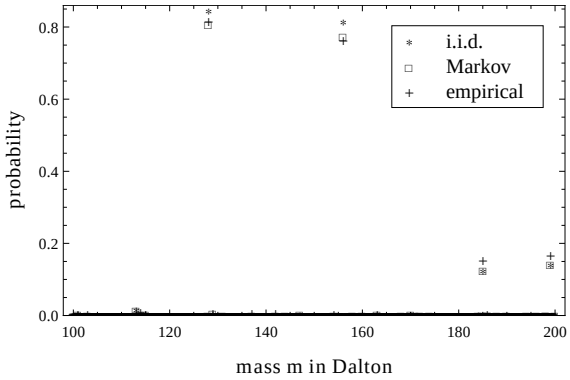


Figure 3.8: Mass occurrence probabilities in the fragmentation of protein sequences of length 300: Comparison of theoretical i.i.d. and M1 values with empirical frequencies.

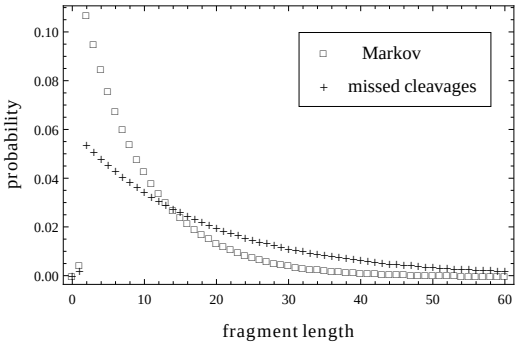
this does not suffice to get a general impression whether the Markov model yields improved statistics.

Besides the comparison of fragment statistics based on i.i.d. and Markov sequences, we wanted to investigate the influence of missed cleavages and post-translational modifications. To this end, we computed the length distribution and the joint length-mass distribution for first and following fragments as well as the mass occurrence probabilities under the respective PAAs described above.

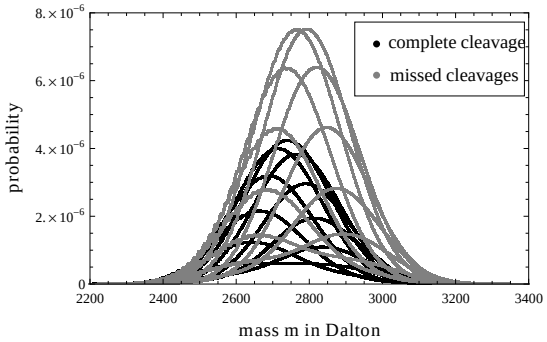
Figure 3.9(a) displays the length distribution of first fragments regarding incomplete cleavage, compared to the theoretical M1 values without integration of missed cleavages. Figure 3.9(b) shows the joint length-mass distribution for fragments of length 25 with and without inclusion of missed cleavages. As expected, the inclusion of missed cleavage sites leads to a decrease in the fragmentation size (not shown) and an increase in resulting peptide lengths and masses. Moreover, internal cleavage patterns tolerated by incomplete cleavage yield different decomposable fragment masses. Hence, also for a fixed length, the distribution of fragment masses is affected by missed cleavages (see Figure 3.9(b)).

According to Siepen *et al.* [111], incomplete digestion plays an important role in protein identification by PMF as it influences not only the score of the top-ranking protein, but also the identified protein. Hence, this should be taken into account for the *in silico* digestion of peptides, and the significance values for peak list alignments should be computed from the adjusted statistics. We leave this as a task for future research.

In the case of PTMs, the length distribution is identical to the M1 distribution (data not shown). This was expected since the considered modifications only change the mass of a fragment, but not its structure. Considering the joint length-mass distribution, inclusion of PTMs leads to a small shift towards higher masses, cf. Figure 3.10. In general, the expected extension in the mass range of fragments as well as in possible mass decompositions was not seen here.



(a) Length distribution of first fragments.



(b) Length-mass distribution of first fragments of length 25.

Figure 3.9: Theoretical length and length-mass distributions of first fragments with and without incorporation of missed cleavages.

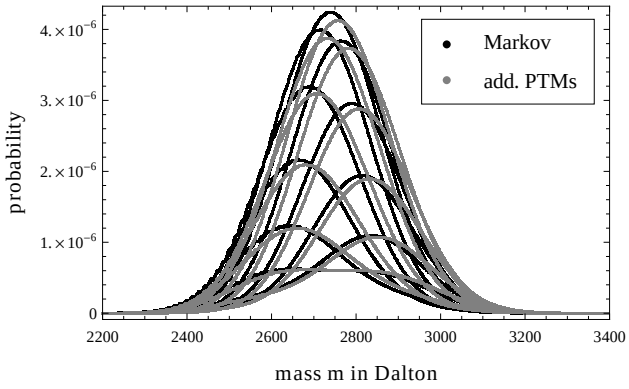
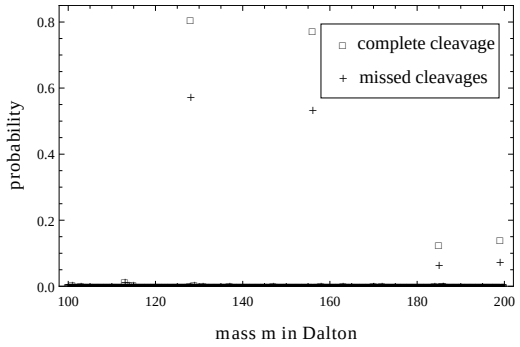


Figure 3.10: Joint length-mass distribution of first fragments of length 25 derived by the PAA under Markovian sequences and additional influence of PTMs.

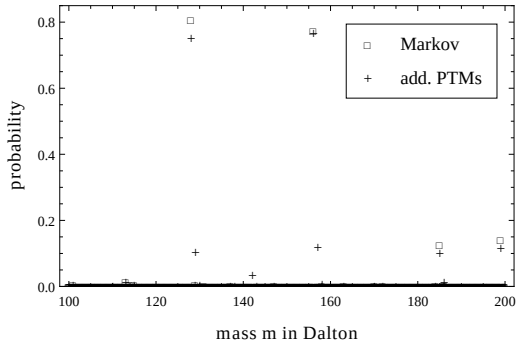
Considering the mass occurrence probabilities, computed for proteins of length 300 and fragment masses up to 1500 Da, we recognized that the frequencies of short fragments are decreased when cleavage sites are missed by the protease, see Figure 3.11(a). This agrees with our former results, cf. Figure 3.9(a). Moreover, the mass occurrence probabilities exhibit more fragment masses occurring in the fragmentation of protein sequences when PTMs are included. This suggests evidence for additional decomposable masses introduced by PTMs. Only the small range of masses between 100 Da and 200 Da as investigated before is shown in Figure 3.11(b).

Directions for future research

As we have seen, the inclusion of missed cleavages or post-translational modifications affects the statistics of proteolytic fragments, especially the mass occurrence



(a) Mass occurrence probabilities in protein sequences of length 300 with and without inclusion of missed cleavages.



(b) Mass occurrence probabilities in protein sequences of length 300 with and without inclusion of PTMs.

Figure 3.11: Influence of incomplete cleavage and PTMs on mass occurrence probabilities.

probabilities. Hence, a next step towards more reliable significance values for protein identification by PMF is the computation of peak list alignments introduced by [54] under incorporation of missed cleavages and PTMs. Our explorations of associated fragment statistics would then yield respective significance values.

In particular, the inclusion of incomplete cleavage modifies the distributional properties of peptide fragments. In prospect of an even more realistic model, it would thus be illuminative to take the propensities of amino acids into account as sketched before.

Another interesting task is the generalization of our framework to the tandem MS context since this more and more becomes the method of choice in the MS society. In this case, proteolytic fragments are further split randomly, e.g. by collision-induced dissociation. This is to say that we could model MS/MS fragments as amino acid strings without internal cleavage patterns, resulting from random cleavage of proteolytic fragments. From an appropriate PAA we could then compute the statistics of MS/MS fragments, analogous to the explorations so far. These should finally be used to come up with a significance value for the comparison of MS/MS spectra.

CHAPTER 4

SSE RANDOM STRING MODEL FOR PROTEIN SEQUENCES

Probabilistic methods in biological sequence analysis mainly concentrate on i.i.d. or low-order Markov sources. Higher-order models as for instance HMMs are used with respect to DNA or RNA sequences, e.g. when the analysis distinguishes between coding and non-coding DNA regions or when it deals with alignments of sequences rather than the sequences themselves. Nevertheless, in the case of protein sequences it lacks a (biologically) meaningful model. In general, the choice of the background model is rarely questioned. Prevalently, the simple i.i.d. model is considered although this ignores dependencies of any kind.

Why is it important do design a proper random string model suited to amino acid sequences? The answer to this is simple: the statistics provided by bioinformatics solutions rely on the quality of the null model. The more reasonable the null model, the more realistic the sought results. Examples of applications using a null model describing protein sequences include

- the identification of unknown proteins,
- the classification of protein domains or protein families, and
- the prediction of secondary structure for a sequence of amino acids.

We discussed the application of protein identification in the previous chapter. There, we modeled proteins and peptides by i.i.d. and first-order Markovian strings. However, proteins (and protein complexes) are described on four different levels, referred to as primary, secondary, tertiary, and quaternary structure [73]. Uniquely determined by the primary structure (for the vast majority of proteins), a protein's biochemical function is due to the three-dimensional (3D) structure, the fold, that brings together amino acids from different parts of the polypeptide

chain(s). As a consequence, chemical groups are positioned in configurations that form a binding site for another protein or a small molecule, or that build the active site of an enzyme affording catalytic activity.

Since the 3D-structure determines the overall function of a protein, structure is much more conserved during evolution than is the sequence of amino acids. Hence, it is desirable to take structural information into account if a proper random string model for protein sequences is sought. To this end, we concentrated on *secondary structure elements* which constitute the three-dimensional shape of local segments of amino acids and thus form the structure most closely related to the amino acid sequence.

This chapter is devoted to the comparison of generative random string models for protein sequences and, in particular, the introduction of a new, biologically motivated string model incorporating information from secondary structure elements. We begin with the methodology of model selection and introduce criteria of multimodel inference in Section 4.1. After summarizing secondary structure elements and their characteristics, we state a new generative random string model dedicated to protein sequences in Section 4.2. As required from a generative random string model, the model we will present enables (i) the generation of random protein sequences of a given length and (ii) the calculation of the probability of a given sequence under the model, i.e. the probability that the sequence was generated by the model. We give the corresponding calculations in Section 4.3. On this basis, we present the conclusive comparison of four random string models and, in particular, the evaluation of the proposed model in Section 4.4.

4.1 Model Selection Criteria

Model selection, as e.g. discussed in [23, 122], refers to the task to find the principle behind a sequence of observations, e.g. outcomes of an experiment. Nevertheless, models are not true in the sense that they do not represent full reality, but rather approximate the reality that underlies a sample of observations. Hence, in model selection theory, the goal is to select the statistical model from a set of (competing) candidate models which explains the observed data “best”. Therefore, a criterion of what is the best model has to be justified. Such a criterion must yield a number for each fitted model, given the data, and it should provide weights which quantify the uncertainty that each model is the best model. Note that there is no true model that generated the data. Rather, there is a concept of truth behind. However, selecting the best model coincides with the task to choose the candidate model that most likely generated the observed data. This relates to the *likelihood function*, often simply referred to as likelihood, which allows us to estimate unknown parameters, based on known outcomes.

Definition 4.1.1 (Likelihood function)**.** *The* likelihood function *of model parameter (vector) θ is defined as the joint probability of a sequence $x = (x_1, x_2, \ldots, x_n)$ of observations (for discrete random variables $X_1, X_2, \ldots, X_n$), considered as a function of θ:*

$$L(\theta) = L(\theta \,|\, x) = \mathbb{P}(x \,|\, \theta) = \mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \,|\, \theta)\,.$$

Note that the likelihood is no probability distribution, i.e. it does not generally sum to one. Rather, it quasi reverses probability. That is to say that $L(\theta \,|\, x)$ lets us reason about parameter θ based on the outcome x, while $\mathbb{P}(x \,|\, \theta)$ gives the probability of outcome x depending on the parameter θ. According to Bayes theorem [13], it holds

$$\mathbb{P}(\theta \,|\, x) = \frac{\mathbb{P}(\theta) L(\theta \,|\, x)}{\mathbb{P}(x)} \propto L(\theta \,|\, x) \tag{4.1.1}$$

for $\mathbb{P}(x) > 0$. Here, $\mathbb{P}(\theta)$ is called the a-priori probability of θ. It expresses the *prior* degree of belief in θ before the experiment was performed. The conditional probability $\mathbb{P}(\theta \,|\, x)$ of θ given that x was observed, is called the *posterior* degree of belief in θ. The likelihood corresponds to the conditional probability $\mathbb{P}(x \,|\, \theta)$ that x occurred if θ were the true parameter.

Example 4.1.2. *For a possible outcome $x = (x_1, x_2, \ldots, x_n)$ of i.i.d. random variables $X_1, X_2, \ldots, X_n$ with probability distribution $f_X(x \,|\, \theta)$, the likelihood is given by*

$$L(\theta) = L(\theta, x) = f_X(x_1 \,|\, \theta) f_X(x_2 \,|\, \theta) \ldots f_X(x_n \,|\, \theta)\,.$$

Note that f_X corresponds to the pdf in the case that X is a continuous random variable, and to the pmf when X is discrete. In view of random strings, which we interpret as random processes taking values in the finite set of amino acids $\Sigma = \{A, \ldots, Z\} \backslash \{B, J, O, U, X, Z\}$, we are concerned with the discrete case. We keep this here, although all concepts we will introduce carry over to sequences of continuous variables.

To decide how likely a certain model M explains the observed data x, we consider the likelihood of model M, i.e. $L(M) = L(M \,|\, x) = \int \mathbb{P}(x \,|\, \theta, M) \mathbb{P}(\theta \,|\, M)\, d\theta$ under parameter vectors θ. Note that this does not depend on the parameters used by the model, but rather takes into account all possible parameter values. Alternatively, instead of integrating over all parameter vectors, it is common practice to use the particular parameter that maximizes the likelihood function, the so-called *maximum likelihood estimator*:

Definition 4.1.3 (Maximum likelihood estimator)**.** *The* maximum likelihood estimator *(MLE) of a model parameter θ is the random variable*

$$\widehat{\theta} = \widehat{\theta}(X_1, \ldots, X_n) = \operatorname{argmax}_\theta L(\theta \,|\, X),$$

maximizing the likelihood function.

The MLE is an asymptotically unbiased point estimator for the parameter θ determining the distribution of X, i.e. $\lim_{n\to\infty} \mathbb{E}(\widehat{\theta} - \theta) = 0$. Further, its variance $\mathbb{V}\mathrm{ar}(\widehat{\theta}) = \mathbb{E}(\widehat{\theta} - \mathbb{E}(\widehat{\theta}))^2$ becomes minimal (with respect to the variance of all other estimators of θ) with increasing sample size. For more information about point estimators and their desired properties, unbiasedness, consistency, and minimal variance, the reader is referred to [91, 115].

In order to determine the MLE of θ, i.e. to find the maximum of the likelihood function, one has to calculate the derivative(s) of $L(\theta)$. As we will see, it is generally convenient to use the logarithm of the likelihood, referred to as *log likelihood*:

Definition 4.1.4 (Log likelihood)**.** *The* log likelihood *of parameter θ, given observations x, is defined as*

$$\ell(\theta) = \ell(\theta \,|\, x) := \ln L(\theta \,|\, x)\,. \tag{4.1.2}$$

Of course, $\ell(\theta \,|\, x) = \sum_{i=1}^{n} \ln f_X(x_i \,|\, \theta)$ for the case of i.i.d. events considered in the example above. Maximizing the log likelihood yields the same result as maximizing the likelihood function since the logarithm is a monotone one-to-one mapping. Yet, maximizing $\ell(\theta, x)$ is much easier since, speaking figuratively, calculating derivatives is easier in a sum than in a product.

Likelihood Ratios

Since the likelihood represents an entire equivalence class of proportional functions (see Equation (4.1.1)), model selection methods usually consider likelihood ratios. In fact, one takes the (natural) logarithm of this ratio. This yields us an additive measure with respect to the outcomes $x = (x_1, x_2, \ldots, x_n)$, referred to as *log likelihood ratio*.

Definition 4.1.5 (Log likelihood ratio)**.** *The* log likelihood ratio *of two competing models M_1 and M_2 is given by*

$$\ln \frac{L(M_1 \,|\, x)}{L(M_2 \,|\, x)} = \ln \frac{\mathbb{P}(x \,|\, M_1)}{\mathbb{P}(x \,|\, M_2)}\,. \tag{4.1.3}$$

A log likelihood ratio of zero indicates that the outcome x is equally likely under both models. If the statistics is positive, the outcome is more likely under M_1; otherwise it is more likely under M_2.

Remark 4.1.6. *The log likelihood ratio yields a prevalent additive scoring system, the so-called* log-odds score *used in biological sequence alignment tools [38]. There, the outcome x refers to a pair of amino acids and the log-odds ratio indicates whether it is more likely that these amino acids are evolutionarily related or rather paired randomly.*

A measure related to the log likelihood ratio is the *Kullback-Leibler divergence* (KL) [66], also known as the *relative entropy* according to Shannon [110], between two probability distributions f and g of a (discrete) random variable X (and analogously for pdfs with the sum replaced by an integral in the continuous case):

$$I(f,g) = \sum_x f_X(x) \ln \frac{f_X(x)}{g_X(x)} = \mathbb{E}_f(\ln f_X(X) - \ln g_X(X)) \,. \tag{4.1.4}$$

While for mathematics the natural logarithm is more convenient, the logarithm to the base two is usually used in information theory. Hence, the units for (relative) entropy are "nats" or "bits", respectively. KL measures the divergence between two distributions, usually between an approximation g and the "true" distribution f. It is non-negative and zero if and only if the two distributions are identical, i.e. if $f = g$. The closer I is to zero, the more similar are the two distributions. Note that it is no metric in the usual sense since it is not symmetric, i.e. $I(f,g) \neq I(g,f)$, and does not satisfy the triangle inequality. Hence, it is called "divergence" rather than "distance". From the information theory perspective [43], the relative entropy measures the expected number of extra bits required to code samples from f when using a code based on g. It thus refers to the information lost about X when g is used to approximate f.

It is important to note that statistics based on the models' likelihoods always favor the most complex model from a set of candidate models. The reason is that a more complex model with more degrees of freedom can fit the data better than any other model from the set, yielding a higher posterior probability. Besides, the likelihood scales with the posterior probability of observations, as we have seen in Equation (4.1.1). Thus, a good model selection method balances "goodness of fit" and "model complexity". In fact, higher order models have to be penalized such that the best suited model is selected that is additionally as simple as possible. Respective model selection methods are called *penalized model selection criteria*.

Penalized Model Selection

In general, *penalized model selection* accounts for the difference in complexity of the considered models. According to the principle of parsimony, the idea is to choose an acceptably good model with as few parameters as possible. This is reasonable in two respects: first, a higher order model provides predictions with less bias but with higher variance, and the aim is to balance these two. Second, a model with more parameters tends to be more sensitive to small changes in the data [122].

Since generally a good model leads to a high maximum likelihood for the data x, penalized model selection criteria refine the log likelihood ratio by introducing a penalty term depending on the sample size n and/or the number k of free parameters in the model. Overall, it has become common practice to choose a model that

maximizes an objective like

$$2 \cdot \ell(\widehat{\theta}, x) - \text{penalty}(\text{number } k \text{ of model parameters}, \text{sample size } n)\,.$$

There exist several criteria, which differ in the penalty adjustment (see [65] for an overview). However, there does not exist the "gold standard" among these criteria. Rather, each penalty proposal has its advantages and disadvantages. One of the first criteria introduced in the statistics literature is the *Information theoretic criterion of Akaike* (AIC). In the following, we give a brief mathematical derivation of this criterion. For a thorough treatment, the reader is referred to the book of Burnham and Anderson [22] or the article of Kuha [65].

The **AIC** defined by Akaike [3] is based on the idea to select the model, among a set $\mathcal{M}$ of candidate models, that minimizes the information loss with respect to the truth. Hence, the AIC criterion chooses the model that minimizes the KL divergence (cf. Equation 4.1.4) between model distributions $g_i(\widehat{\theta}_i)$ $(1 \leq i \leq |\mathcal{M}|)$ and the distribution f considered closest to the truth. We write $\widehat{I}(f, g_i(\widehat{\theta}_i))$ to indicate that the candidate model g_i depends on an estimated parameter, i.e. the respective MLE $\widehat{\theta}_i$. Assuming that all $g_i(\widehat{\theta}_i)$ $(1 \leq i \leq |\mathcal{M}|)$ are independent of f, we have that

$$\widehat{I}(f, g_i) = \mathbb{E}_f(\ln f(X)) - \mathbb{E}_f\big(\ln g_i(X \mid \widehat{\theta}_i)\big)\,.$$

The former expectation is unknown, but since it is the same for all investigated models, it can be seen as a constant not influencing the minimization. That is to say that from the set of candidate models the model is selected that minimizes the *relative KL divergence*

$$\widehat{I}(f, g_i) = C - \mathbb{E}_f\big(\ln g_i(X \mid \widehat{\theta}_i)\big)\,.$$

For two models g_1 and g_2, model g_1 is better if and only if $\widehat{I}(f, g_1) < \widehat{I}(f, g_2)$. The difference

$$\widehat{I}(f, g_1) - \widehat{I}(f, g_2) = \mathbb{E}_f\left(\ln \frac{g_2(X \mid \widehat{\theta}_2)}{g_1(X \mid \widehat{\theta}_1)}\right)$$

quantifies how much better g_1 is, but on a relative scale because without knowing C, we do not know the absolute measure of how good even g_1 is. It refers to the expected log likelihood ratio given in (4.1.3).

Moreover, according to Burnham and Anderson [22], for a rigorous model selection criterion based on the relative KL divergence, one has to estimate $\mathbb{E}(\mathbb{E}\big(\ln g_i(X \mid \widehat{\theta}_i)\big)$. The maximized log likelihood provides a biased estimator, where the bias is approximately the number of free model parameters.

The AIC criterion formalized by Akaike [3] is an asymptotically unbiased estimator [115] of the relative expected KL divergence. The AIC value for a model with MLE $\widehat{\theta}$ and k free parameters, given a sample x, is defined as

$$\text{AIC} := -2 \cdot \ell(\widehat{\theta}, x) + 2k\,. \tag{4.1.5}$$

The first term on the right hand side decreases when more parameters are added to the approximating model, the second term clearly increases. As mentioned before, this presents a trade-off between bias and variance of the estimator.

Remark 4.1.7. *Note that one also finds the definition* $\mathrm{AIC} := -\ell(\widehat{\theta}, x) + k$ *in the literature. As long as both log likelihood and bias correction are multiplied by the same positive constant, this does not change the minimization.*

In applications, one computes the AIC value for each candidate model and finally selects the model with the smallest value. This model is estimated to be the candidate model closest to the unknown reality underlying the observed data. As the criterion chooses the best model on a relative base, usually the AIC differences

$$\Delta_i := \mathrm{AIC}(M_i) - \min_j \mathrm{AIC}(M_j)$$

are computed for all models $M_i \in \mathcal{M}$. These provide a ranking of candidate models with respect to the model estimated to be best, namely the model M_i for which $\Delta_i \equiv 0$. The larger Δ_i is, the less plausible it is that the fitted model M_i is the best model according to the KL criterion. There exist some guideline values indicating the empirical support of a candidate model depending on its AIC difference, see [22].

Note that the AIC criterion may perform poorly when the sample size n is small with respect to the number k of free parameters in the candidate model with highest dimension. For that reason, unless the sample x is much larger than the number of parameters, it is recommended by Burnham and Anderson [22] to use a refinement of the AIC criterion, referred to as *second-order AIC* ($\mathrm{AIC_c}$):

$$\mathrm{AIC_c} = \mathrm{AIC} + \frac{2k(k+1)}{n-k-1} = -2 \cdot \ell(\widehat{\theta}, x) + 2k\frac{n}{n-k-1}\,. \tag{4.1.6}$$

A directive is to use $\mathrm{AIC_c}$ instead of AIC if $n/k < 40$. Otherwise, both criteria are similar (since the correction term is then close to 1) and tend to select the same model.

We apply the discussed criteria later in order to compare different generative string models and, in particular, to evaluate the model we will present in the following.

4.2 Secondary Structure Elements Based Protein Model

In this section, we present the *secondary structure elements* (SSE) based protein model, short SSE model. It is a generative random string model to describe random protein sequences. As mentioned before, statistics based on such a null model afford information as the significance of an alignment score or the average length or mass of a peptide fragment in the application from Chapter 3. There, we introduced a strategy used daily in many areas of life science research: using sequence information to find similar (putatively related) protein sequences in huge databases gathering knowledge which can help to deduce properties such as the structure and the function of an unknown protein. A more refined null model will lead to more reasonable results.

Here, we set up a new, biologically motivated random string model adjusted to protein sequences. It shall incorporate biological information and still depend on a feasible number of parameters. To this end, rather than concentrating on the sequence of amino acids only, we take the secondary structure into account. Hence, we call this background model *secondary structure element based model*, or short *SSE model*.

4.2.1 Secondary Structure Elements

As mentioned before, the secondary structure is the first level of protein folding. Parts of the polypeptide chain fold to form local generic structures that are found in all proteins. Secondary structure, as introduced by Linderstrøm-Lang [73] in 1952, is generally defined by patterns of hydrogen bonds (H-bonds) between polypeptide backbone -NH and -C=O groups (interactions of the kind side chain/side chain or side chain/main chain are irrelevant). The most common secondary structure elements are α-helix and β-sheet.

In nature, many regular structures are helical. A helix results from energetically favorable hydrogen bonding between all the backbone groups. Thus, the helix is a very regular and stable arrangement. In proteins, there are three types of helices, namely the α-helix, the 3_{10}-helix, and the π-helix, which differ in their hydrogen bonding patterns. The α-helix is a particularly stable right-handed helix which is formed in peptide chain stretches with repetitive torsion angles (used to define the conformation of bonds that are free to rotate, see [129]) of approximately $(\phi, \psi) = (-60°, -60°)$. Therefore, the -C=O and -NH groups are hydrogen bonded to the peptide bond four residues away, i.e. there exist H-bonds between the oxygen of amino acid i and the amide group of amino acid $i+4$ and between the nitrogen of residue i and the carboxyl group of residue $i+4$. In the 3_{10} helix, H-bonds exists between residues i and $i+3$, and in the π helix, stabilization results from H-bonds between amino acids i and $i+5$. These patterns are also referred to as $3n$, $4n$, and $5n$. Left-handed helices rarely exist in proteins because of steric hindrance.

Another regular secondary structure is the β-sheet. It is formed by β-strands, i.e. stretches of 5-10 amino acids (whose peptide backbones are almost fully extended), that are connected laterally by three or more hydrogen bonds. Adjacent strands can be in antiparallel, parallel, or mixed arrangement. In antiparallel sheets, the N-terminus of one strand is adjacent to the C-terminus of the next strand, which allows planar hydrogen bonds and is thus very stable. In the slightly less stable parallel arrangement, all strands have the same direction as all N-termini are adjacent.

Moreover, tight turns, isolated β-bridges and loose, flexible loops link the more regular structures helix and sheet. Also single pairs of amino acids can be annotated as turn or β-bridge. In contrast, helices and sheets are required to have a reasonable length, i.e. a certain number of adjacent amino acids in the primary structure have to form the same hydrogen bonding pattern. Otherwise, if this

pattern is too short, the amino acids are annotated as turn or bridge. Metaphorically speaking, repeating turns form helices, repeating bridges are ladders, and connected ladders represent sheets.

4.2.2 Information from Secondary Structure Annotation

In order to derive secondary structure information, we used protein sequences listed in the PDB database [14] that are annotated with secondary structure elements. The annotation is provided via the automated assignment methods DSSP (Dictionary of Protein Secondary Structure) [53] or STRIDE [41]. While DSSP determines secondary structure according to hydrogen-bonding patterns between the amino acid residues, STRIDE additionally uses the backbone torsion angles [129]. We use the DSSP which is the widely accepted standard.

The secondary structure labeling according to DSSP classifies amino acid residues into 8 different structural element classes:

- **B**: Residue in isolated β-bridge; a single β-sheet hydrogen bonding pattern.
- **E**: Extended β-strand; at least two residues in parallel or anti-parallel β-sheet conformation.
- **S**: Bend; a region with high curvature, quantified by the backbone angles.
- **T**: Hydrogen bonded turn ($3n$, $4n$, or $5n$).
- **G**: 3_{10}-helix; at least three residues forming $3n$-turns.
- **H**: α-helix; at least four residues forming $4n$-turns.
- **I**: π-helix; at least five residues forming $5n$-turns.
- **C**: Random coil; in general not B,E,G,H,I,S,T.

The annotation as random coil is somewhat misleading, since it could denote flexible parts of the protein (loops) which do not take part in a local structure as well as residues not classified to any structure element. To get an intuition about the general distribution of structure annotations, we estimated the relative frequencies of amino acids assigned a particular annotation, cf. Figure 4.1. As indicated before, the most prominent structures are α-helix (31.3%) and β-sheet (19.9%).

Next, we investigated the properties of amino acid chains assigned to different structural elements. For this purpose, we cut the annotated protein sequences into substrings subject to the residues' structural correspondence. This is to say that, for a sequence and its annotation like

sequence	`MQHVSAPVFVFECTRLAYVQHK`
annotation	`CCCCCHHHHHHCCCCHHHEEEE`

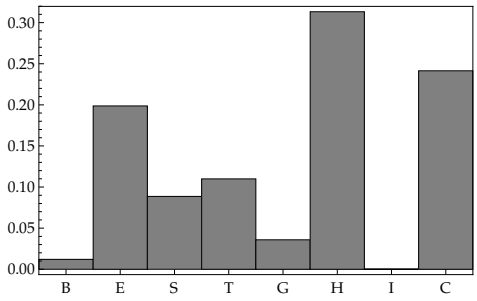


Figure 4.1: Histogram of the relative frequencies of secondary structure annotations according to DSSP for protein sequences in the PDB database.

we produced the following sets of substrings for the structure element classes C, E, and H: $\{MQHVS, ECTR\}$, $\{VQHK\}$, and $\{APVFVF, LAY\}$, respectively. From these strings, we estimated the frequencies of amino acids, the distribution of the initial amino acid, the conditional amino acid frequencies, and the length distribution, separately for each class. A comparison of the distributions of amino acid frequencies within different classes on the basis of relative entropy values is provided in Figure 4.2. Relative entropy, as stated in Section 4.1, measures the divergence between two distributions. It is zero if and only if the distributions are identical. Larger values indicate larger divergence.

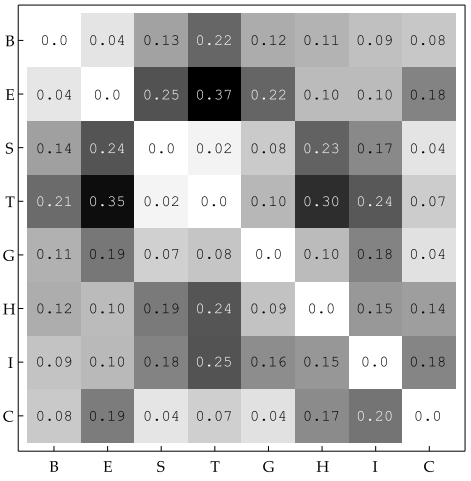


Figure 4.2: Relative entropy between amino acid frequency distributions estimated for different structural elements. Color intensity encodes the strength of deviation on a relative scale. The two distributions are more similar the lighter the corresponding cell color. White indicates zero divergence, i.e. that the two distributions are identical.

Bioinformatics tools such as secondary structure prediction tools [129] or alignment methods using information from SSEs [11] typically collapse DSSP output into three classes associated with helices, sheets, and coils. Overall, there are different strategies to join DSSP classes, see [100]. The effect of various assign-

ments with respect to secondary structure prediction performance was studied in [32]. We observed, however, that the similar distributions within classes S and T differ significantly from those estimated from other elements. Thus, taking into account that only a very small percentage of residues is annotated to B, I, and G, we merged G, H, and I into one class to represent helices, B and E to denote sheets, and S and T to collect bends and turns. C remains unmodified. This is to say, we classified secondary structure elements into the following four classes:

- **C**: coil (random coil),
- **E**: sheet (β-bridges and strands),
- **H**: helix (some kind of helix),
- **T**: turn (bends and turns).

The amino acid frequencies we counted from subsequences assigned to these four elements mainly agree with the propensities of amino acids provided in the literature [129, Chapter 11]. For example, amino acids including methionine (M), alanine (A), leucine (L), glutamic acid (E), and lysine (K) prefer to adopt helical conformations, while other amino acids as proline (P) and glycine (G) are rarely found in helices. These "helix-breakers" interrupt the regular helical structure and are rather seen in turns. Moreover, the large aromatic amino acids as phenylalanine (F), tryptophan (W), and tyrosine (Y), and the β-branched amino acids isoleucine (I), valine (V), and threonine (T) preferably adopt sheet conformations. The amino acid compositions estimated for the four structure classes are shown in Figure 4.3. As can be seen, A, L, E, and K are preferably seen in the helix class, in particular L (11.7%) and A (11.4%). In contrast, G (3.7%) and P (2.4%) occur less frequently and usually at substring ends. Further, our counts agree that β-branched residues V (13.4%) and I (9.7%) prefer sheet conformation, while we do not observe high frequencies for the large aromatic amino acids. However, we recognize that these are rarely found in flexible turns, where G (16.4%) is the most frequent amino acid, as proposed.

4.2.3 Model Architecture and Parameter Estimation

The main idea leading to the SSE random string model was to refine a first-order Markov model by combining individual Markov chains for the four structure classes specified before. Each individual chain generates parts of the final protein sequence, i.e. substrings of the final string. Analogous to the previous chapter, a finite random amino acid sequence of length n is described by the length n prefix of a stochastic process $(S_i)_{i\in\mathbb{N}}$ taking values in $\Sigma = \{A,\ldots,Z\}\backslash\{B,J,O,X,Z\}$, i.e. $S = S_1,\ldots,S_n$ with $S_i \in \Sigma$.

The architecture of the SSE model is shown in Figure 4.4. A first-order Markov chain $Y = (Y_n)_{n\in\mathbb{N}}$ with state space $Q = \{C,E,H,T\}$ defines the order of structural correspondence of amino acids in a protein sequence. In each state (structural element) $v \in Q$, an individual Markov chain $X^v = (X_n^v)_{n\in\mathbb{N}}$ generates parts of the

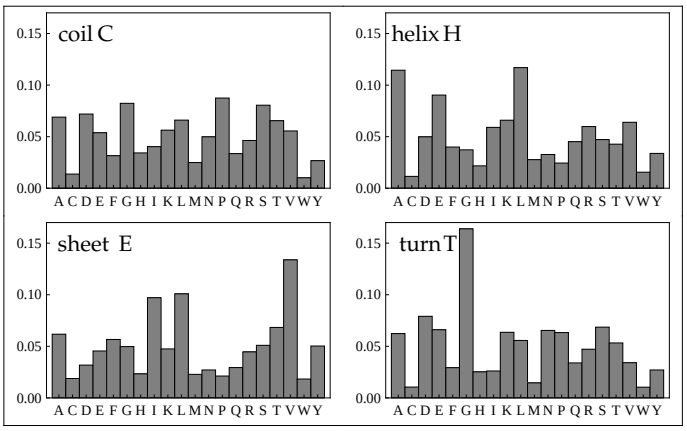


Figure 4.3: Amino acid compositions estimated from PDB annotated sequences after collapsing DSSP elements to the four structure classes C (top left), E (bottom left), H (top right), and T (bottom right).

resulting string. These Markov chains act on the space Σ of amino acid characters. Hence, the corresponding transitions are given by the conditional frequencies of amino acids within the respective structure class. The number of amino acid residues produced at one run in a certain element $v \in Q$ depends on the length distribution associated with the according state.

Let us introduce the following notations:

$$p^0(v, \sigma) := \mathbb{P}(Y_1 = v, S_1 = \sigma)$$

denotes the initial joint probability of amino acids $\sigma \in \Sigma$ and annotations $v \in Q$. Further,

$$T^{uv}_{\sigma\sigma'} := \mathbb{P}(Y_{m+1} = v, S_{m+1} = \sigma' \,|\, Y_m = u, S_m = \sigma) \quad \text{for all } m$$

is the transition probability between two characters while changing from state u to state v, such that $\sum_{v \in Q} \sum_{\sigma' \in \Sigma} T^{uv}_{\sigma\sigma'} = 1$ for all fixed $(u, \sigma) \in Q \times \Sigma$. For each state $v \in Q$, the conditional character frequencies define the transition probability $P^v_{\sigma\sigma'}$ from character σ to σ'. Finally, the random variable $R(v)$ counts the number of residues produced in state v. Its distribution is denoted by $\big(\mathbb{P}(R(v) = k)\big)_{k \in \mathbb{N}}$.

Parameter Estimation

We intend to compare four generative random string models for protein sequences, i.e. the i.i.d. model, the first-order (M1) and second-order (M2) Markov models, and the SSE model. For all of those, we need to estimate appropriate model parameters. As described above, we make use of the respective MLEs. To this end, we first prepared the following sets of PDB sequences: we extracted all $115,267$ sequences from the PDB database that are annotated with secondary structure elements.

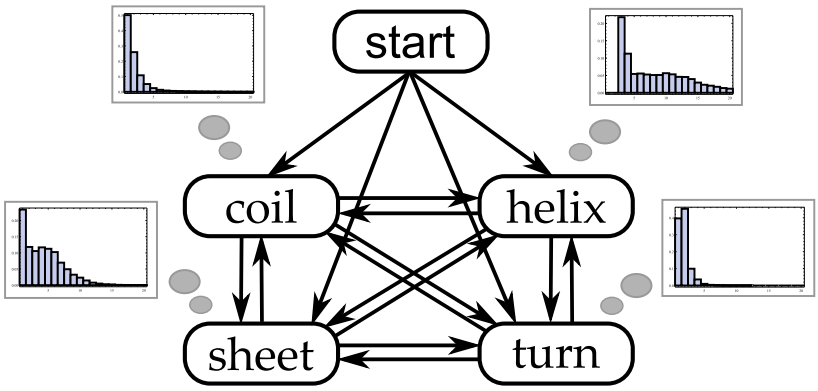


Figure 4.4: Sketch of the secondary structure element based random string model for protein sequences. Four different structure classes $Q = \{C, E, H, T\}$ are taken into account. A first-order Markov chain Y specifies the order of structural correspondence. In each state $v \in Q$, a first-order Markov chain X^v generates substrings of the resulting protein sequence. The length of these substrings is determined by the length distributions associated with each state.

Since it is reported that functional proteins consist of at least 40 residues, we discarded shorter sequences, leaving $109,218$ annotated sequences. Then, we divided these into a training set of 97.083 sequences (88.9%) and a test set of $12,135$ (11.1%) sequences. We used the training sample to estimate necessary model parameters (MLE). That is to say, for the i.i.d. model, we counted the frequencies of amino acids, for a first-order and a second-order Markov model, we counted the initial frequencies as well as the conditional amino acid frequencies (of order one and two, respectively). For the SSE model, we needed to estimate parameters for the superior and the four individual Markov chains. We therefore counted the pairs of initial character and assigned structure class to estimate $(p^0(v,\sigma))_{(v,\sigma)\in Q\times\Sigma}$ as well as the transitions between amino acids with different secondary structure annotation for $(T^{uv}_{\sigma\sigma'})_{u,v\in Q;\sigma,\sigma'\in\Sigma}$. These parameters determine the superior Markov chain Y. Furthermore, after splitting the sequences as explained in Section 4.2.2, we counted the conditional character frequencies $P^v = (P^v_{\sigma\sigma'})_{\sigma,\sigma'\in\Sigma}$ for the individual Markov chains from the corresponding substrings.

The sequences we left out in the training step were then investigated in order to generate a revised test sample. For this purpose, sequences with at least 30% sequence identity with respect to sequences in the training set were removed from the test set. This resulted in 912 (0.8%) revised test sequences.

4.3 Computing Likelihood Values

Our aim is to compare the presented SSE model to customary random string models, i.e. to the i.i.d. model, the M1 model, and the M2 model. To this end, we want

to use the criteria introduced in Section 4.1. Hence, we need to compute the log likelihood $\sum_s \ln \mathbb{P}(s \,|\, M_i)$ of each model M_i, given protein sequences s from PDB or another protein database, e.g. the revised test sequences.

The corresponding calculation of $\ln \mathbb{P}(s \,|\, M_i)$ is easy in the case of i.i.d. and low-order Markov models:

$$\begin{aligned}
\ln \mathbb{P}(s \,|\, \textbf{i.i.d.}) &= \sum_{i=1}^{|s|} \ln \mathbb{P}(S_i = s[i]) = \sum_{i=1}^{|s|} \ln p_{s[i]}, \\
\ln \mathbb{P}(s \,|\, \textbf{M1}) &= \ln p^0_{s[1]} + \sum_{i=1}^{|s|-1} \ln \mathbb{P}(S_{i+1} = s[i+1] \,|\, S_i = s[i]) \\
&= \ln p^0_{s[1]} + \sum_{i=1}^{|s|-1} \ln P_{s[i]s[i+1]}, \\
\ln \mathbb{P}(s \,|\, \textbf{M2}) &= \ln p^0_{s[1]} + \ln P_{s[1]s[2]} + \sum_{i=1}^{|s|-2} \ln \mathbb{P}(S_{i+2} = s[i+2] \,|\, S_i = s[i], S_{i+1} = s[i+1]) \,.
\end{aligned}$$

It is more involved in the case of the SSE model. However, since the model's structure is similar to an HMM, we can apply variants of the forward or backward algorithms [38, Chapter 3]. Hence, we first show how to derive the probability $\mathbb{P}(S = s \,|\, \text{SSE})$ for a sequence s of length n by marginalization, before introducing the logarithm.

$$f(s) := \mathbb{P}(S = s) = \mathbb{P}(S_1, \ldots, S_n = s[1,n]) = \sum_{v \in Q} \mathbb{P}(S_1, \ldots, S_n = s[1,n], Y_n = v) \,. \quad \textbf{(4.3.1)}$$

The following calculation relates to the forward algorithm known from classical HMM theory:

$$\begin{aligned}
f_k(v) &:= \mathbb{P}(S_1, \ldots, S_k = s[1,k], Y_k = v, Y_{k+1} \neq v) \quad \text{for } 1 \leq k < n, \\
f_n(v) &:= \mathbb{P}(S_1, \ldots, S_n = s[1,n], Y_n = v) \,.
\end{aligned}$$

It is the probability to observe the length k prefix of a sequence s, or the entire sequence s, respectively, with the last part generated in element v. Herewith, the probability that sequence s was generated by the SSE model reads

$$f(s) = \sum_{v \in Q} f_{|s|}(v) \,.$$

Moreover, to denote the probability that a substring of s has been generated in state $v \in Q$, we write $p^v(s[j_1, j_2]) := \prod_{i=j_1}^{j_2-1} P^v_{s[i]s[i+1]}$. Herewith, we formulate the system of recurrence relations given in Recurrence 4.1.

Note that, for $n = |s|$, we use $\mathbb{P}(R(v) \geq m) = 1 - \sum_{m'=1}^{m-1} \mathbb{P}(R(v) = m')$ in order to account for the fact that protein sequences are finite. That is to say that for the last

Recurrence 4.1 Computation of the probability that a protein sequence s of length $|s| = n$ was generated by the SSE model.

$$
\begin{aligned}
f_k(v) &= \sum_{u \in Q} \sum_{l=1}^{k-1} f_l(u)\, T^{uv}_{s[l]s[l+1]}\, p^v(s[l+1,k])\, \mathbb{P}(R(v) = k-l) \\
&\quad + P^{start,v}\, \pi^v_{s[1]}\, p^v(s[1,k])\, \mathbb{P}(R(v) = k) \qquad \text{for } 1 \le k < n, \\
f_n(v) &= \sum_{u \in Q} \sum_{l=1}^{n-1} f_l(u)\, T^{uv}_{s[l]s[l+1]}\, p^v(s[l+1,n])\, \mathbb{P}(R(v) \ge n-l) \\
&\quad + P^{start,v}\, \pi^v_{s[1]}\, p^v(s[1,n])\, \mathbb{P}(R(v) \ge n),
\end{aligned}
$$

Output: $f(s) = \sum_{v \in Q} f_n(v)$.

character of s, it is irrelevant whether it is the last character of a substring or not. Nevertheless, (functional) proteins range in length from 40 to several thousand residues. Therefore, multiplying many small probabilities to compute $\mathbb{P}(S = s)$ may lead to underflow errors. To avoid these, we implemented the log likelihood $\ln \mathbb{P}(S = s) = \ln \sum_{v \in Q} f_{|s|}(v)$ as discussed in Section 4.1. For this purpose, however, we had to come up with the logarithms of sums of small values.

To this end, we used the following property: $\ln(x + y) = \ln(x(1 + y/x)) = \ln x + \ln(1 + \exp(\ln y - \ln x))$. Thus, if the logarithmical values $\ln x_i$ of the summands in $\ln \sum_{i=1}^n x_i$ are known, we have

$$
\begin{aligned}
\ln \sum_{i=1}^n x_i &= \ln \left(x_1 \Big(1 + \frac{x_2}{x_1} + \frac{x_3}{x_1} + \cdots + \frac{x_n}{x_1} \Big) \right) \\
&= \ln(x_1) + \ln \Big(1 + \frac{x_2}{x_1} + \frac{x_3}{x_1} + \cdots + \frac{x_n}{x_1} \Big) \\
&= \ln(x_1) + \ln \Big(1 + \exp(\ln x_2 - \ln x_1) + \exp(\ln x_3 - \ln x_1) \\
&\quad + \cdots + \exp(\ln x_n - \ln x_1) \Big),
\end{aligned}
$$

where it is reasonable to choose $x_1 = \max_i x_i$. Then, from the Taylor series expansion

$$
\ln(1 + x) = \sum_{k=1}^{\infty} (-1)^k \frac{x^k}{k} \approx x + \mathcal{O}(x^2)
$$

it follows the approximation $\ln(1 + x) \approx x$ for small values of x. This approximation is performed by the method $\log1\mathrm{p}$ incorporated in the Java.Math library.

For our purpose to compute $\ln f(s) = \ln \sum_{v \in Q} f_n(v)$, we defined

$$
\begin{aligned}
F^0_k(v) &:= p^0(v, s[1])\, p^v(s[1,k])\, \mathbb{P}(R(v) = k) \\
F^l_k(v) &:= \sum_{u \in Q} f_l(u)\, T^{uv}_{s[l]s[l+1]}\, p^v(s[l+1,k])\, \mathbb{P}(R(v) = k-l) \qquad \text{for } 1 \le l < k
\end{aligned} \tag{4.3.2}
$$

and the analogs $F_k^{\geq 0}(v)$ and $F_k^{\geq l}(v)$ ($1 \leq l < n$) for the last part of the sequence, respectively. In words, $F_k^l(v)$ defines the probability of a path through the superior Markov chain Y that generates the length k prefix $s[1,k]$ ending in state $v \in Q$ with the last $(k-l)$ characters produced in v. Thus, the F^l correspond to the probabilities of all paths. Clearly, summing up all path probabilities yields the entire probability:

$$f_k(v) = \sum_{l=0}^{k-1} F_k^l(v)$$

and $f_n(v)$ analogously. As mentioned above, the $F_k^l(v)$ are sorted in descending order such that $F_k^0(v) = \max_l F_k^l(v)$, i.e. F^0 then refers to the probability of the most probable path. We then compute

$$\ln f_k(v) = \ln F_k^0(v) + \mathrm{log1p}\Big(\sum_{l=1}^{k-1} \exp\big(\ln F_k^l(v) - \ln F_k^0(v)\big)\Big). \tag{4.3.3}$$

To determine the logarithmical value of the inner sum in (4.3.2), we apply the same principle again.

Caution has to be exercised with sequences that have probability 0 under the considered model. In that case, all paths through the Markov chain have probability 0, leading to $\ln F_n^l(v) = -\infty$ for all $0 \leq l < n$ and all $v \in Q$ (as well as $F^{\geq l'}$ for $0 \leq l' < n$). A problem arises when we want to compute $\exp(\ln F^l - \ln F^0)$. To prevent this, we exclude those summands from the log1p calculation and obtain $\ln f(s) = \sum_{v \in Q} \ln F_n^0(v) = -\infty$.

Runtime

The runtime to compute $\ln f(s)$, for the probability $f(s)$ of a protein sequence s as defined in Equation (4.3.1), is of order $\mathcal{O}(n^2)$, if we precompute the logarithms of the probabilities $p^v(s[j_1, j_2])$. For this purpose, we define the logarithm of the probability for a suffix by $d[v,k] := \ln p^v(s[k,n])$ for $v \in Q$ and $1 \leq k \leq n$ according to:

$$d[v,k] = \begin{cases} 0 & \text{for } k = n, \\ d[v,k+1] + \ln P^v_{s[k]s[k+1]} & \text{for } k < n. \end{cases} \tag{4.3.4}$$

With this $\mathcal{O}(n)$ preprocessing, we get the logarithm of the probability that a substring $s[k,l]$ was generated within element $v \in Q$ via the following update:

$$\ln p^v(s[k,l]) = \begin{cases} 0 & \text{if } l = k, \\ -\infty & \text{if } d[v,k] = -\infty \text{ and } d[v,l] \neq -\infty, \\ \sum_{i=k}^{l-1} \ln P^v_{s[i]s[i+1]} & \text{if } d[v,l] = -\infty, \\ d[v,k] - d[v,l] & \text{otherwise}. \end{cases} \tag{4.3.5}$$

These values can be stored in $\mathcal{O}(n^2)$ space. For the computation of $\ln f(s)$, $|Q| = 4$ values resulting from the calculation of 4.3.3 have to be added. To this end, n

variables F^l have to be computed which takes $\mathcal{O}(n)$ time each. Thus, the total runtime as well as the space requirements are of order $\mathcal{O}(n^2)$.

4.3.1 Proof of Concept

As motivated, the presented SSE model is a refinement of a first-order Markov model. More precisely, the first-order Markov model M1 is a special case of the SSE model. Hence, in order to check our implementation, we set the parameters such that the SSE model defines the inherent Markov model. This is to say, the transitions $\left(P^v_{\sigma\sigma'}\right)_{\sigma,\sigma'\in\Sigma}$ for the individual Markov chains X^v ($v \in Q$) were all determined by the conditional amino acid frequencies estimated from the set of PDB training sequences. Moreover, the transitions $\left(T^{uv}_{\sigma\sigma'}\right)_{u,v\in Q;\sigma,\sigma'\in\Sigma}$ for $u \neq v$ were set to those frequencies, normalized by $1/(|Q|-1)$.

As expected, the log likelihood values of the PDB test sequences computed under this special SSE model agree with the respective M1 values. The model's log likelihood under the revised test sequences is $-561,060$ in both cases. In Figure 4.5, the statistical properties of the computed sample values are shown. Since all values are identical under both models, also the median and the quartiles are the same.

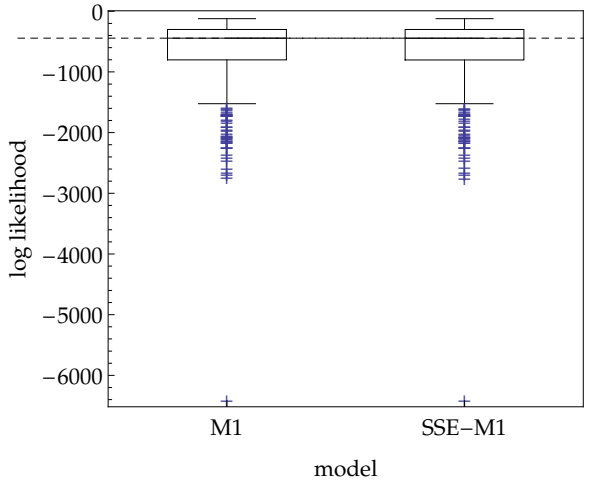


Figure 4.5: Boxplots representing the log likelihood values for the 912 PDB revised test sequences under the M1 model and under the M1 special case inherent in the SSE model.

4.4 Results: Comparison of Random String Models

In order to evaluate the performance of the SSE model, we compared it to the i.i.d. model, the first-order Markov model M1, and the second-order Markov model M2. As mentioned before, the respective model parameters were chosen to be the maximum likelihood parameters, i.e. the relative (conditional) frequencies estimated from the set of PDB training sequences. For instance, the i.i.d. model needs only one distribution of dimension $|\Sigma| = 20$. Since $\sum_{\sigma\in\Sigma} p_\sigma = 1$, the respective degree of freedom equals 19. For the Markov models, additionally conditional probability distributions of dimension $\Sigma \times \Sigma$ and $\Sigma \times \Sigma \times \Sigma$ are considered. Hence, the degree of freedom is $19 + 20 \cdot 19 = 399$ (M1) or $399 + 20 \cdot 20 \cdot 19 = 7999$ (M2), respectively. For the SSE model, the relevant distributions are

- the initial distribution $p^0(v, \sigma)$ of dimension $Q \times \Sigma$,
- the first-order Markov transitions $\left(P^v_{\sigma\sigma'}\right)_{\sigma,\sigma'\in\Sigma}$ within each element $v \in Q$, again each with $|\Sigma|(|\Sigma| - 1)$ degrees of freedom,
- the length distributions $R(v)$ associated with each element $v \in Q$, which we restricted to dimension 100, and
- the transitions $\left(T^{uv}_{\sigma\sigma'}\right)_{u,v\in Q;\sigma,\sigma'\in\Sigma}$ with $|Q|(|Q| - 1)|\Sigma|(|\Sigma| - 1)$ degrees of freedom, because $\sum_{v\in Q}\sum_{\sigma'\in\Sigma} T^{uv}_{\sigma\sigma'} = 1$ and $T^{uu}_{\sigma\sigma'} = 0$ for all character pairs $(\sigma, \sigma') \in \Sigma\times\Sigma$.

In total, the number of free model parameters is thus 6552. Note that we do not need initial character frequencies within the individual elements to specify the according Markov chains, cf. Recurrence 4.1. These are actually covered by the initial distribution and the transitions of the superior Markov chain Y.

The respective numbers of free model parameters are summarized in Table 4.1.

Table 4.1: Numbers of free model parameters for the four models under consideration. The corresponding derivations are given in the text.

i.i.d.	M1	M2	SSE
19	399	7999	6552

Following our explanation above, we first computed the log likelihoods of the four models under the set of test sequences and revised test sequences, respectively. As expected, for both samples, the pairwise log likelihood ratios favor the model with highest dimension since the log likelihood scales with model complexity. This is to say that the ranking according to pairwise log likelihood ratios is

1. M2
2. SSE
3. M1
4. i.i.d.

Further, we computed the average log likelihood values (normalized by sequence lengths) for the revised test sequences under the considered models. The boxplots in Figure 4.6 display the statistics of these values. This yields a visual inspection of the former ranking. Moreover, the boxplots show that the variance increases with the number of parameters, as stated above.

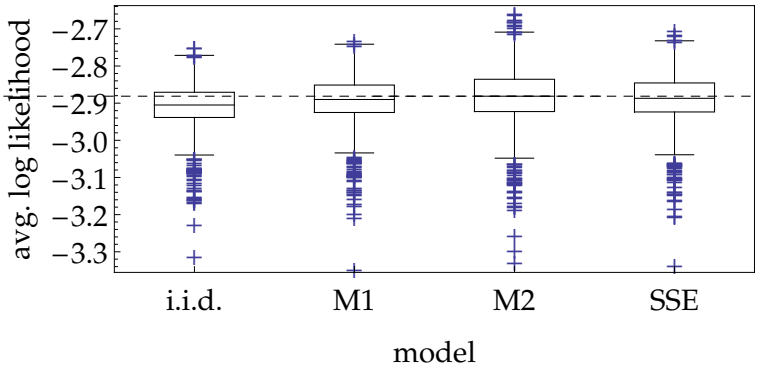


Figure 4.6: Boxplots representing the length-normalized log likelihood values for the 912 PDB test sequences under the four candidate models. As can be seen, the values scale with model complexity.

Next, we used these average log likelihoods in order to determine the models' AIC values and differences. In fact, we used the second-order AIC criterion, $\mathrm{AIC_c}$, because the test samples are quite small with respect to the maximal number of free parameters. Even for the not revised test set of $12,135$ protein sequences, clearly ${}^{12,135}/_{7,999} < 40$. In Table 4.2, we show the ranking of the four candidate models according to $\mathrm{AIC_c}$ under the revised test sequences along with their respective AIC differences $\Delta_i = \mathrm{AIC_c}(M_i) - \min_j \mathrm{AIC_c}(M_j)$.

Table 4.2: Ranking and AIC differences Δ_i (rounded to integers) of the candidate models according to the $\mathrm{AIC_c}$ criterion.

rank	**model**	$\boldsymbol{\Delta_i}$
1	SSE	0
2	M2	63.99
3	i.i.d.	2184.03
4	M1	3536.34

Thus, the AIC criterion estimates the SSE model to be the best candidate model to describe protein sequences. It outperforms the widely used i.i.d. and M1 models as well as M2, which maximizes the goodness of fit part.

Moreover, the number of adjustable parameters for the SSE model can be reduced substantially. For instance, we observed that the first amino acid of all annotated PDB sequences is assigned to random coil. Hence, modeling the initial character annotation such that

$$\mathbb{P}(Y_1 = v, S_1 = \sigma) = \begin{cases} p_\sigma^0 & \text{if } v = C \\ 0 & \text{otherwise,} \end{cases}$$

would reduce the number of free parameters to 6495. Further, approximating the amino acid transitions implying a change of structural annotation rather than using an individual transition probability for each combination would lead to 2007 free parameters. However, this reduction does not affect the resulting ranking. This indicates that the inclusion of more complex (biological) information pays off in order to provide a more plausible description of protein sequences.

Discussion and Directions for future research

The SSE model we presented demonstrates how biological information can be incorporated in a generative random string model. Although it combines Markov chains for four structural elements, it has still less free parameters than the M2 model. Our analysis revealed that, among the considered candidate models, the AIC criterion favors the SSE model in order to describe protein sequences.

Nevertheless, the considered approach has to deal with the fact that it is generally difficult to uniquely link secondary structure to the sequence of amino acids. The problem is that secondary structure annotation is not good enough yet [118]. In particular, long-range interactions affecting secondary structure formation are usually disregarded. Moreover, there are too few annotations available in order to estimate reasonable model parameters. Further, secondary structures are not perfectly regular. Hence, it is difficult to exactly define the ends of those local structures. Moreover, in some cases there exist hydrogen-bonding patterns intermediate between the presented forms.

For these reasons, we think that subsequent research can further improve the performance of the SSE model. It would be interesting to replace the individual Markov chains X^v by more reasonable models taking the characteristics of each structure class into account. For example, since we know that α-helices are built by repetitive H-bonds between amino acids 4 residues apart, we could use a third-order Markov model in the helix state.

Furthermore, the fact that the secondary structure annotation is especially insufficient for initial amino acid residues impairs the SSE results. Thus, an idea for future research is to build a hybrid model leaving out the secondary structure information for the first part of the sequence. In addition, we would like to investigate an application of the SSE model for which the initial amino acid residues are irrelevant and hence the mentioned drawback is circumvented. This application relates to the significance of patterns extracted from protein sequences that are

used as signatures for protein families, domains, or functional sites, contained in the PROSITE database [51]. The SSE model could be used as background model in order to compute significance values for PROSITE patterns [112], i.e. regular expressions used to assign a protein sequence to a specific family.

Another application of the SSE model has been investigated in cooperation with Stefan Wolfsheimer who analyzed the statistics of alignment scores (for the alignment of protein sequences) under arbitrary null models. In particular, our idea was that a more reasonable null model would yield exacter results. However, for efficient computation, a fast update from the null model probability of a protein sequence to the probability of a specified neighbor sequence is desired. The computations for the SSE model were tedious since it had to be recomputed from scratch for each sequence. It would be interesting to investigate how to improve this.

Lastly, the question is how good a null model for the whole universe of protein sequences can be anyway. We think that the SSE model could be more appropriate to describe a family of proteins (e.g. taken from the Pfam database [12]) for which the layout of secondary structures is more similar.

CHAPTER 5

APPLICATION II: COMPUTING ALIGNMENT SEED SENSITIVITY

Now, we look back on the connection between the PAA framework and deterministic automata used to investigate probabilistic pattern matching problems (Section 2.1). In this chapter, we consider a special case of pattern matching statistics arising in computational biology. Here, the pattern is called *seed* and refers to a short initial match between a query and database sequences when searching for homologous sequences by means of heuristic methods referred to as *seeded alignment* techniques.

In this application, the model again shows its universality and flexibility as it comprises the majority of recent models. We generalize the Markov chain approach of Choi and Zhang [26], extending the model to characterize a seed by its entire hit distribution rather than its sensitivity only. The obtained recurrences also allow a symbolic calculation as performed by Mak and Benson [80]. Moreover, the PAA can be designed for *indel seeds* used in the context of gapped alignments [79] as well as for sets of seeds. Our framework also provides a greedy covering algorithm to design a good set of seeds. The contents of this chapter have been published in Herms and Rahmann [48].

In Section 5.1, we describe how seeds are used in heuristic homology search algorithms and summarize related work. Afterwards in Section 5.2, we introduce the underlying string models, namely the *homology model* describing sequence alignments and the *seed model*. We analyze the number of matches of a given seed, and in particular its *sensitivity*, to a randomly generated sequence alignment by means of a properly designed PAA in Section 5.3. Finally, Section 5.4 concludes this chapter with a comparison to other methods and a discussion on alternative criteria to measure the quality of seeds.

5.1 Seeded Alignment and Seed Sensitivity

In large-scale homology search, large databases are searched for sequences that are putatively homologous to a given query sequence. To this end, a local alignment of any pair of query and database sequence has to be computed. Then, those sequences obtaining an optimal alignment whose score exceeds a given threshold, are returned by the particular method. Because of the large size of relevant databases, comparing each position in the query with each position in the database, as in the quadratic-time Smith-Waterman algorithm [113], is too computationally intensive. Since the 1980's, when BLAST [4] was designed, heuristic methods have been used for faster homology search. Most heuristic homology search algorithms [4, 5, 57, 97] (and some repeat detection algorithms [42]) are based on a filtration technique called *seeded alignment*. This is a two step method to quickly detect database sequences similar to a given query sequence. The key idea is to focus on such sequences that share a specified initial pattern with the query. This pattern, the *seed*, is assumed to witness a potential similarity. Then, the optimal alignment has to be computed only for a subset of the database sequences. In the *filtration* step, one identifies candidate sequences that share a common pattern of matching characters with the query. This can be done efficiently by first indexing query and database words of the same length as the seed. In the *confirmation* step, the matches in such candidates, called *hits*, are further investigated by exact local alignment methods, such as the Smith-Waterman algorithm, to see whether they can be extended to high-scoring alignments.

It has been shown by Ma *et al.* [76] that careful seed design is important for both the *sensitivity* and the *specificity* of seeded alignment. The sensitivity is the probability to hit biologically meaningful alignments, the specificity refers to the true negative rate $TN/(TN+FP)$ or, to put it differently, to one minus the false positive rate, i.e. $1 - FP/(FP+TN)$. Increasing the sensitivity usually decreases the specificity, and vice versa. To achieve a good sensitivity/specificity trade-off is the key issue of local alignment tools.

Initially, only contiguous perfect matches (e.g., DNA 11-mers in the initial BLAST implementation) were used as seeds. PatternHunter (PH) by Ma *et al.* [76] was the first tool to systematically advocate and investigate *spaced seeds*, i.e. a discontiguous sequence of required matches: PH looks for 18-mers with at least 11 matching positions distributed as $\pi_{PH} = 111{*}1{*}{*}1{*}1{*}{*}11{*}111$, where 1 denotes a necessary match and $*$ denotes a "don't care" position (match or mismatch).

Over time, various seed models have been proposed in the literature, ranging from consecutive seeds [4, 97] to spaced seeds [17, 20, 27, 76], subset seeds [64], vector seeds [18], and indel seeds [79].

Usually, to qualify a seed, we use a probabilistic model, the so-called *homology model*, to describe all local alignments between the query and the database sequences. The fraction of alignments that satisfy the pattern specified by the seed corresponds to the fraction of selected candidates. The sensitivity of a seed can

then be computed as the probability to match an alignment generated randomly according to the homology model. A good seed exhibits high sensitivity for alignment models that describe evolutionarily related sequences, and low sensitivity values for models that represent unrelated sequences. The latter property ensures that the seed does not detect too many *random hits*. Random hits decrease the efficiency of the filtration phase, since they are checked in vain for a significant alignment. An interesting finding was that the PH approach led to an increase in both sensitivity and filtration efficiency, compared to seeds of contiguous matches. Based on the observations of Ma *et al.* [76], the advantages of spaced seeds over consecutive seeds have been subsequently shown by many authors [20, 26, 71].

Remark 5.1.1. *Since the homology model is used to describe alignments of both evolutionarily related sequences and sequences of biologically unrelated species, a more reasonable name would be* similarity model*. To conform to recent literature, we continue to write* homology model *keeping this differentiation in mind.*

An extension to the previously mentioned seed models dealing with a single seed is the design of a *multiple seed*. This is a set of spaced seeds to be used simultaneously, such that a similarity is detected when it is found by (at least) one of the seeds. The idea to use a family of spaced seeds for BLAST-type DNA local alignment has been suggested by Ma *et al.* [76] and was implemented in PatternHunter II [70]. It has also been applied to local protein alignment [19]. Since finding *optimal* multiple seeds is challenging, most authors concentrate on the design of efficient sets of seeds which lead to a higher sensitivity than optimal single seeds [63, 70, 116]. Kucherov *et al.* [63] characterize a set of seeds only by its selectivity. That is, they compare seeds with 100% sensitivity according to their false positive rate. Recent approaches [52, 60] approximate the sensitivity of multiple spaced seeds by means of correlation functions. Moreover, Kong [60] discussed that sensitivity should rather be measured via a criterion averaging over different target alignment lengths.

When searching optimal seeds, one faces the following problems to evaluate candidate seeds. The second, more general one has not yet been extensively considered in the literature. Note that we give the appropriate definitions and a formal description of different seed models in the subsequent sections.

Problem 1 (Sensitivity computation)**.** *Given a homology model representing the level of sequence similarity, a target length t of alignments, and a set of seeds, what is the probability that an alignment of length t is matched by the seed (at least once)?*

Problem 2 (Hit distribution)**.** *Given a homology model, a target length t of alignments, a set of seeds, and a maximal match number K, what is the probability that an alignment of length t is matched by the seed exactly/at least k times, for $k = 0, \ldots, K$, when counting (a) overlapping matches or (b) non-overlapping matches?*

5.2 Underlying String Models

As prerequisites for the computation of the above-mentioned problems, we need to describe (a) probabilistic models for sequence alignments, (b) seed models defining the initial similarity region, and (c) when a seed hits an alignment, that is to say how to count seed matches by means of an appropriate PAA.

5.2.1 Homology Model

To be able to compute seed sensitivity, we need to describe random alignments with known degree of similarity (e.g., a certain percent identity value). In the context of seed sensitivity computations, we need to model all local alignments. Hence, it is customary *not* to model random sequence pairs and derive alignment properties from those, but to directly model the similarity, i.e. the alignment's *representative string* $\mathcal{A}$ over an alphabet Σ indicating the status of the alignment columns. Table 5.1 shows an example of a representative string for an ungapped alignment. In the simplest and most frequently studied case only substitution

Table 5.1: The representative string $\mathcal{A}$ over $\Sigma = \{0, 1\}$ for an ungapped alignment. Note that in the following, we directly model $\mathcal{A}$ ignoring the particular characters in the aligned sequences.

query	AACGATGTACCTA
target	AAGGACGTTCCAA
$\mathcal{A}$	1101101101101

mutations are considered [17, 20, 26, 27, 76, 116]; i.e. $\Sigma = \{0, 1\}$, referring to matches (1) and mismatches (0). We also deal with the binary alphabet in the case of spaced seeds. On the contrary, indel seeds use the alignment alphabet $\Sigma = \Sigma_\delta = \{0, 1, 2, 3\}$, where additionally 2 denotes an insertion in the database sequence, and 3 indicates an insertion in the query sequence. There are various other alignment alphabets, e.g. the ternary alphabet representing match / transition / transversion in DNA [94], or even larger alphabets to distinguish different pairs of amino acids in the case of proteins [19].

A random alignment can be modeled by a k^{th}-order Markov chain or an HMM. Usually, we use a first-order Markov chain (Σ, P, p^0) with stochastic transition matrix P on the alignment alphabet Σ and initial distribution p^0. Corresponding parameters are referred to as *homology parameters*.

If we consider the binary case $\Sigma = \{0, 1\}$ and an i.i.d. homology model, where the transition probability P_{ij} does not depend on i, then $p^0 = (1 - p, p)$ and

$$P = \begin{pmatrix} 1-p & p \\ 1-p & p \end{pmatrix}$$

with a *match probability* $p \in [0,1]$ which quantifies the average percent identity of such alignments. In this case, p is the only homology parameter. Using indel seeds, a first-order Markov chain is appropriate to model sequence alignments, since this allows to respect that a substitution is more plausible than two consecutive indels (one in each sequence). Hence, the homology model should prohibit the pairs '23' and '32' in a representative string. With the intention to compare our results later on, we work with the transition matrix P proposed by [79]. There,

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} p_0 & p_1 & p_{\mathrm{g}} & p_{\mathrm{g}} \\ p_0 & p_1 & p_{\mathrm{g}} & p_{\mathrm{g}} \\ p_0^* & p_1^* & p_{\mathrm{g}} & 0 \\ p_0^* & p_1^* & 0 & p_{\mathrm{g}} \end{pmatrix} \end{array}, \tag{5.2.1}$$

where p_0 is the probability of a mismatch, p_1 is the probability of a match, and p_{g} refers to the probability of a gap in the alignment. In order to ensure a stochastic transition matrix, $p_i^* = p_i + p_{\mathrm{g}} p_i/(p_0+p_1)$ for $i = 0, 1$. This results from distributing p_g to matches and mismatches. The initial distribution is given by $p^0 = (p_0, p_1, p_{\mathrm{g}}, p_{\mathrm{g}})$. Other transition probabilities are possible, e.g. if alignments with affine gap costs should be modeled. Then, a transition of the form '2'↦'2' or '3'↦'3' is related to gap extension, the other gap probabilities are replaced by the probability corresponding to gap opening.

We conclude this section with a few remarks. From the biological perspective, even sequences with high degree of sequence similarity may be evolutionarily unrelated and other sequences may be homologous although their DNA sequences differ in large parts of the genome. This might indicate that the i.i.d. model over- or underestimates the number of homologous sequences (depending on p). Hence, several groups investigated more involved homology models (often based on HMMs). These were usually designed and trained for a special type of sequences, e.g. for coding and non-coding DNA regions or for protein sequences.

5.2.2 Seed Model

A seed $\pi = \pi[1]\pi[2]\dots\pi[\ell_\pi]$ is a string over an alphabet Ξ of "care" and "don't care" characters (not to be confused with the alignment alphabet Σ). It represents alignment regions that indicate matches at the "care" positions. A seed is classified by (ℓ_π, ω): its *length* $\ell_\pi = |\pi|$ and its *weight* ω, which refers to the number of "care" positions.

A *consecutive seed* is a continuous sequence of "care" positions.

A *spaced seed* is a string over the alphabet $\Xi = \{1, *\}$. The "care" positions are indicated by '1', while '$*$' refers to a match/mismatch wildcard. Reasonable seeds for the purpose of homology search always require $\pi[1] = \pi[\ell_\pi] = 1$.

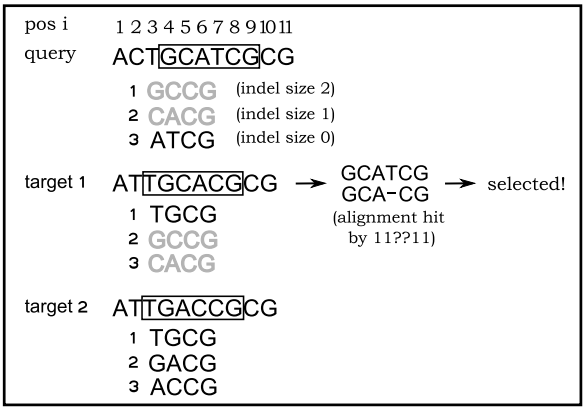


Figure 5.1: Exemplified use of the indel seed $11??11$ to select candidate database sequences. Word extraction (as explained in the text) is shown for relevant positions. Target 1 can be aligned to the query such that the resulting alignment is hit by the seed. Hence, target 1 is selected as candidate, while target 2 is not.

An *indel seed* according to Mak *et al.* [79] is a string over the alphabet $\Xi = \{1, *, ?\}$, where '1' and '$*$' are used as above and '?' stands for zero or one character from the alignment alphabet $\Sigma_\delta = \{0, 1, 2, 3\}$. Two consecutive '?' symbols may not represent '23' or '32', because in an alignment such a case would rather be described by one mismatch (0). This seed model explicitly allows for indels of variable size. We explain how indel seeds are used in the filtration step by the following example. For a more detailed description, the reader is referred to the original article.

Example 5.2.1. *The indel seed $11??11$ permits indels of size zero, one, or two rather than forcing two indels. This implies that "?' can also refer to a match/mismatch position. Query and database sequences are processed as follows. For each position i in a sequence s, we extract three words $s[i-5, i-4]s[i-1, i]$, $s[i-4, i-3]s[i-1, i]$, and $s[i-3, i-2]s[i-1, i]$ ending at this particular position, because the indel size can be zero, one, or two (at the borders we extract less words because of edge effects). Then, extracted words from the query (group A) are compared to extracted words from the database sequence (group B). If any word from A matches any word from B, the database sequence is selected as a candidate sequence. In Figure 5.1, the extraction is shown for position $i = 9$ in the query and position 8 in each of the two target database sequences. The displayed alignment of target 1 and the query is hit by the indel seed and thus, target 1 would be selected as a candidate for further investigation.*

According to Burkhardt and Kärkkäinen [21], a (consecutive or spaced) seed is uniquely determined by its *relative position set* $\mathcal{RP}(\pi) = \{x_1, x_2, \ldots, x_\omega\} = \{y \in \{1, \ldots, \ell_\pi\} \,|\, \pi[y] = 1\}$, where we constrain $x_1 = 1$ and $x_\omega = \ell_\pi$ as mentioned before. Therewith, the *pattern set*, that is the set of all words described by π, is usually defined as $\mathcal{PS}(\pi) = \{T \in \Sigma^{\ell_\pi} \,|\, T[x_i] = 1\,,\, 1 \leq i \leq \omega\}$. Some authors even define a seed as such an ordered list of indices, see [116]. Note that for indel seeds, the

definition of the relative position set is not adequate. Hence, to define a general, unifying model, we introduce *generalized seeds*.

Definition 5.2.2 (Generalized seed)**.** *Let* $\Psi = \{[1], [01], [\epsilon 0123]\}$ *contain the character sets induced by* Ξ*, where we write* $[xy]$ *as shorthand for* $\{x, y\}$*, and let* π *be a seed over* Ξ*. The* generalized seed *that refers to* π *is* $G(\pi) = g(\pi[1]) \dots g(\pi[\ell_\pi])$*, where* $g : \Xi \to \Psi$ *is a mapping such that '*1*'* $\mapsto [1]$*, '*$*$*'* $\mapsto [01]$*, and '*$?$*'* $\mapsto [\epsilon 0123]$*.*

This enables a novel definition of the set of patterns corresponding to a given seed.

Definition 5.2.3 (Pattern set)**.** *Let* $\pi = \pi[1]\pi[2] \dots \pi[\ell_\pi]$ *be a seed of length* ℓ_π *over the alphabet* Ξ*. The* pattern set

$$\mathcal{PS}(\pi) = \left\{ s = s[1] \dots s[\ell_\pi] \mid s[i] \in g(\pi[i]) \right\}$$

contains all words satisfying the seed, i.e. all strings that match $G(\pi)$*.*

Example 5.2.4. *For the spaced seed* π $=1*1*1$ *of length* 5 *and weight* 3 *(class* $(5, 3)$*),* $G(\pi) = [1][01][1][01][1]$ *and* $\mathcal{PS}(\pi) = \{10101, 10111, 11101, 11111\}$*. The patterns of the indel seed* π $=1*1?1$ *with* $G(\pi) = [1][01][1][\epsilon 0123][1]$ *are given by* $\mathcal{PS}(\pi) = \{1011, 1111, 10101, 10111, 10121, 10131, 11101, 11111, 11121, 11131\}$*.*

Definition 5.2.5 (Hit position)**.** *A seed* π *hits the representative string* $\mathcal{A}$ *ending at position* i *if and only if there exists an*

$$M \in \mathcal{PS}(\pi) \text{ such that } \mathcal{A}[i - |M| + 1, i] = M\,.$$

Example 5.2.6. *For the representative string* $\mathcal{A} = \mathcal{A}[1, 16] = 1011111001111101$ *for instance, the consecutive seed* 1111 *hits ending at positions* 6*,* 7*,* 13*, and* 14*, the seed* $\pi = 1 * 11 * 1$ *hits at positions* 6 *and* 16*, and the indel seed* π $=11*1?1$ *hits at positions* 7*,* 14 *and* 16*, respectively.*

Since it has been shown that, considering ungapped alignments, spaced seeds outperform consecutive seeds with respect to filtration efficiency [20, 26, 71], we will from now on concentrate on spaced seeds for ungapped alignments. Why are spaced seeds better than consecutive seeds (of the same weight)? To get an intuition, note first that the *random hit rate* of a seed is mainly controlled by its weight. If we assume uniformly distributed character frequencies (all nucleotides occur with probability $1/4$), the probability that two nucleotides within the aligned sequences match just by chance is $4 * 1/4 * 1/4 = 1/4$. Thus, a seed of weight ω has probability $1/4^\omega$ to hit an alignment of random sequences. This is referred to as random hit rate. In fact, two seeds with the same weight generate approximately the same number of overall hits [76]. On the other hand, if a consecutive seed hits an alignment, it is very likely that it hits again at the next position (cf. Example 5.2.6), since it only requires one more match position. Therefore, a consecutive seed tends to hit an alignment more often than a spaced seed of the same weight.

Consequently, the spaced seed will (usually) detect more similarities and thus, will have better sensitivity.

The idea to use a finite, non-empty set $\Pi = \{\pi_1, \ldots, \pi_m\}$ of spaced seeds, called *multiple spaced seed*, turned out to further improve the quality of filtering [17, 20, 70, 116]. The reason for that is simple: any given seed may fail to detect some homologies. But, different seeds tend to fail at different homologies. Hence, using a combination of seeds whose detection abilities are complementary should significantly improve the overall detection rate. However, increasing the sensitivity generally implies a decrease in specificity. Let us analyze this relation at this point.

In general, there are two alternatives to trade sensitivity for specificity. One possibility to substantially increase sensitivity is to reduce the number of required match positions. This in turn results in an exponential increase (of magnitude 4 or 20 if we assume i.i.d. random DNA or protein sequences with uniformly distributed character frequencies as explained earlier) in the false positive rate and hence decreases specificity and search speed. The other possibility is to add seeds to the search algorithm. As pointed out by Sun and Buhler [116], this increases the false positive rate at most linearly and is thus more attractive than reducing the weight of a single seed.

The patterns of a multiple seed Π are collected in $\mathcal{PS}(\Pi) = \cup_{i=1}^{m} \mathcal{PS}(\pi_i)$. A multiple seed is said to hit $\mathcal{A}$ if at least one of its components does (in contrast to Pevzner and Waterman [99] where all seeds are required to match). From now on, for the sake of generality, we use Π also for single seeds, i.e. $\Pi = \{\pi\}$.

5.3 Computing Seed Sensitivity with a PAA

As we have motivated, sensitivity is a measure for the quality of seeds and hence, for the quality of seeded alignment. It is used in order to compare seeds as well as to design good sets of seeds. We show that the framework of *Probabilistic Arithmetic Automata* (PAA) provides a general and flexible approach to exact seed sensitivity computation. While it covers recent methods [20, 26, 64], the PAA framework can handle both ungapped and gapped alignments. Moreover, it allows the investigation of overlapping and non-overlapping hits of a single or a multiple seed as discussed in Section 2.1. It yields recurrence relations to compute the entire hit distribution and in particular seed sensitivity. From these recurrences, we also derive a polynomial which allows a fast computation for any parameter set.

5.3.1 Automaton Construction

In order to answer the posed questions (Problem 1, Problem 2), the PAA should be designed such that (a) the underlying Markov chain generates a proper random sequence alignment, and (b) the number of seed hits to the alignment are counted.

This is to say that at each position of the alignment, the number of matches ending there is emitted and added to the current count. To this end, we associate to each state the operation "+". Thus, in terms of emission and value processes, $V_l = V_{l-1} + Z_l$ with $V_0 = 0$, reducing the free parameters of the sought PAA to $\big(Q, T, q_0, E, e = (e_q)_{q\in Q}\big)$.

Given the homology model (Σ, P, p^0), we first construct a Markov chain (Q, T, δ_{q_0}) that generates an appropriate random sequence alignment. Since we deal with a probabilistic pattern matching problem for the set of patterns satisfying a seed Π, the set of states Q can be represented by all prefixes of those patterns (following Section 2.1.2). Further, we introduce the (remaining) characters from the alignment alphabet as supplementary states, and a start state $q_0 = \epsilon$ to ensure the initial character distribution. Thus, the state space of a seed Π is given by

$$Q(\Pi) = \{\epsilon\} \cup \{\sigma \in \Sigma\} \cup \bigcup_{M\in\mathcal{PS}(\Pi)} \text{Prefixes}(M)\,. \tag{5.3.1}$$

We mark a finite set $\mathcal{F}$ of final states in order to distinguish states that contribute to the number of seed hits. In contrast to ACA(W), where $\mathcal{F} = W$, here the final states are those hit by Π, that is $\mathcal{F} = \{q \in Q \,|\, \exists\, M \in \mathcal{PS}(\Pi) : M \blacktriangleleft q\}$.

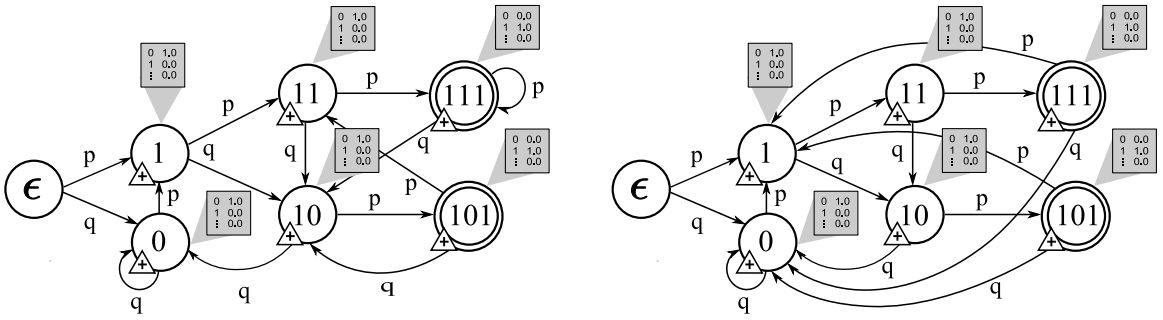
Remark 5.3.1. *In the case that $|\Pi| > 1$, we use Hopcroft's algorithm [50] to minimize the size of the state space as mentioned in Section 2.1.1. Here, the initial partition is induced by grouping states with the same emitted value $C(q)$, the same ingoing and the same outgoing transition probability (depending on the last character in the case of Markov transitions). Certainly, testing for equivalent character transitions is not necessary for the simple case of an i.i.d. alignment model.*

The transitions again remind us of the ACA. Nevertheless, we need to describe (outgoing) transitions for every character $\sigma \in \Sigma$ which implies the following alternative failure function. There is a non-zero outgoing transition from state u to state v if v is the maximal suffix of $u\sigma$, that is if there exists a $\sigma \in \Sigma$ such that $v = \text{argmax}_{x\in Q\cap\text{Suffixes}(u\sigma)} |x|$. The corresponding transition probability is $P_{u[|u|-1]\sigma} = \mathbb{P}(\sigma \,|\, u[|u|-1])$, where the conditional probabilities are given by the homology model. Thus, in the i.i.d. case it is just the character frequency p_σ. The stochastic state transition matrix of the PAA with $u, v \in Q$ is therefore given by

$$T_{uv} = \begin{cases} p^0_\sigma & \text{if } u = \epsilon, v = \sigma \in \Sigma, \\ P_{u[|u|-1]\sigma} & \text{if } u \neq \epsilon,\ \exists\,\sigma \in \Sigma : v = \text{argmax}_{x\in Q\cap\text{Suffixes}(u\sigma)} |x|, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.2}$$

The so constructed Markov chain (Q, T, δ_ϵ) generates a random sequence alignment whose similarity level is specified by the homology parameters. It is adapted to a given seed through the choice of states.

Here, we can easily introduce a variant that shows the flexibility of the PAA approach. Modifying the Markov chain transitions outgoing from a final state, the



(a) PAA counting overlapping hits of $\pi = 1{*}1$.

(b) PAA for non-overlapping hits of $\pi = 1{*}1$.

Figure 5.2: Sketch of the PAA constructed for $\pi = 1{*}1$. Part (a) shows the transitions in order to count all occurrences of π in a random alignment over $\Sigma = \{0, 1\}$ generated according to the i.i.d. model with match probability p. Part (b) is designed to count non-overlapping hits.

model is suited to count non-overlapping seed occurrences. We will use these in Section 5.4.4 in order to define an alternative criterion to rate seed quality. The modification comprises the following: whenever a hit has been found, the Markov chain has to start again in one of the states representing the next character to read. To this end, we set the transitions outgoing from a final state $u \in \mathcal{F}$ to

$$T_{uv} = \begin{cases} P_{1\sigma} & \text{if } v = \sigma \in \Sigma, \\ 0 & \text{if } v \notin \Sigma, \end{cases}$$

since the last character read was a 1 (by definition). Then, the Markov chain becomes reducible, and we remove states from Q that are not reachable from ϵ. Figure 5.2 shows an example with Markov chains adapted to overlapping and non-overlapping hits.

In order to count the number of seed hits, we still need to assign the family $e = (e_q)_{q \in Q}$ of weight output distributions. To this end, let $C(q)$ denote the number of hit counts in state $q \in Q$. Then,

$$C(q) = |\{M \in \mathcal{PS}(\Pi) \,:\, M \blacktriangleleft q\}| \,. \tag{5.3.3}$$

Consequently, we deal with the emission set $E = \{0, 1, \ldots, c_{\max}\}$, $c_{\max} = \max_{q \in Q} C(q)$, and $|E|$-dimensional Dirac measures e_q assigning probability 1 to $C(q)$. Note that $C(q) = 0$ for states $q \notin \mathcal{F}$. Moreover, in the case of non-overlapping hits, $C(q) = 1$ for all $q \in Q$, because of the reduced state space.

Corollary 5.3.2. *The PAA* $\big(Q, T, q_0 = \epsilon, E = \{0, 1, \ldots, c_{max}\}, e = (e_q)_{q \in Q}\big)$ *with* Q *and* T *given in equations* (5.3.1) *and* (5.3.2)*, and* $e_q = \delta_{C(q)}$ *with* $C(q)$ *given in Equation* (5.3.3)*, counts the number of hits of a given seed* Π *to a randomly generated sequence alignment according to the specified homology model. The inherent* Markov Additive Chain $(V_l)_{l \in \mathbb{N}_0}$ *with*

$$V_0 \;\equiv\; 0$$

$$V_l \quad = \quad V_{l-1} + C(Y_l)$$

yields the number V_t of accumulated hits in a random alignment of length t.

Remark 5.3.3. *If we want to explicitly specify the number of overlapping or non-overlapping hits, we write $V_{t;o}$ or $V_{t;n}$, respectively.*

Construction time.

As presented in Section 2.1.2, the ACA can be constructed in linear time, proportional to the total length of keywords (patterns). This includes an elegant algorithm to set up the failure function in linear time. Although already Aho and Corasick [2] formulated a linear-time algorithm in their original work, we point to Dori and Landau [36] who used the suffix tree of the reversed patterns. Following their approach, we can construct the presented PAA in $\mathcal{O}(|\Sigma| \cdot (\ell + |\Sigma|))$ time and space according to the observations hereafter.

Let $\ell = \sum_{M \in \mathcal{PS}(\Pi)} |M|$ denote the total length of seed patterns. Then, the state space Q of the PAA (5.3.1) is of size $\mathcal{O}(\ell + |\Sigma|)$, and the number of transitions is of order $\mathcal{O}(|\Sigma| \cdot (\ell + |\Sigma|))$. To derive the transitions and the number $C(q)$ of hits in each state $q \in Q$, we use the suffix tree of the reversed seed patterns. This is constructed during one scan of the patterns, in time proportional to ℓ [46]. Within a depth-first traversal of the tree, states are labeled with the number of prefixes that represent a reverse pattern, cumulating the numbers of a branch. This can be done in $\mathcal{O}(\ell \cdot |\Sigma|)$ time since we read each reverse pattern in the suffix tree and annotate the corresponding vertex. Transferring these numbers to the corresponding states in the PAA completes the construction and precomputation of C in time and space $\mathcal{O}(|\Sigma| \cdot (\ell + |\Sigma|))$.

Remark 5.3.4. *In our first approach to apply the PAA framework to the context of seed sensitivity computation, we used all ℓ_{max}-mers as states, where ℓ_{max} refers to the maximal length of a seed from Π. In that case, the state space is of size $\mathcal{O}(|\Sigma|^{\ell_{max}})$ with $\mathcal{O}(|\Sigma| \cdot |\Sigma|^{\ell_{max}})$ transitions. The number of ℓ_{max}-mers grows exponentially with the length of the longest seed, whereas in the approach presented in this chapter, the number grows only exponentially with the number of wildcards. Consequently, when the number of wildcards throughout the seeds is smaller than ℓ_{max}, we now have to deal with substantially less states. Indeed, this is the relevant case in multiple seed design in order to ensure high specificity (few random hits).*

5.3.2 Hit Distribution and Seed Sensitivity

In order to compute the sensitivity of a given seed Π (Problem 1) and its entire hit distribution (Problem 2) for a target alignment length t, we seek the distribution $\mathcal{L}(V_t)$ of accumulated seed hits. This is obtained by marginalization from the joint

state-value distribution $f_t(q,k) = \mathbb{P}(Y_t = q, V_t = k)$ as described in Section 2.4. Herewith, we can reformulate the mentioned problems. Seed sensitivity is commonly defined as

$$S(\Pi, t) = \mathbb{P}(V_t \geq 1) = 1 - \mathbb{P}(V_t = 0) = 1 - f_t(0)\,. \tag{5.3.4}$$

It is related to the hit distribution, i.e. the frequency of alignments of length t that contain exactly k hits:

$$\mathbb{P}\big(\{\mathcal{A} \,|\, |\mathcal{A}| = t, V_t = k\}\big) = \mathbb{P}(V_t = k) = f_t(k) \quad \text{for } k \geq 0\,. \tag{5.3.5}$$

The constructed PAA yields the following system of recurrence relations:

$$f_l(q,k) = \sum_{q' \in Q} f_{l-1}(q', k - C(q)) T_{q'q} \quad \text{for } q \in Q, l \geq 1, \tag{5.3.6}$$

with initial condition $f_0(\epsilon, 0) = 1$.

Space and Time Consumptions.

In order to efficiently implement the computation of seed sensitivity, we make use of the vector $F_l(0) = \big(f_l(q,0)\big)_{q \in Q}$ and the update formula $F_l(0) = F_{l-1}(0)T'$. By T', we denote the transition matrix projected to the columns representing q with $C(q) = 0$. This means, all entries within a column corresponding to a state hit by Π are set to 0 in order to ensure $f_l(q,0) = 0$ for q with $C(q) > 0$. Likewise, we use a vector update for the computation of the hit distribution $f_l(q,k)$ for $k \geq 1$. For this purpose we store the matrix $F_l = \big(f_l(q,k)\big)_{q \in Q, 0 \leq k \leq t - \ell_{\min} + 1}$ to compute F_{l+1}, where $\ell_{\min}$ refers to the length of the shortest seed within Π.

For the computation of seed sensitivity (Problem 1), we store the transition matrix as well as the vector C of emission counts for constant-time access. Further, for each update step we store two vectors $F_{l-1}(0)$ and $F_l(0)$ of size $|Q|$. Together, this consumes $\mathcal{O}(|Q|^2)$ space, where $|Q|$ is of order $\mathcal{O}(\ell + |\Sigma|)$. The vector update then takes $\mathcal{O}(|Q|^2)$ time. Hence, we can compute the sensitivity $S(\Pi, t)$ in time $\mathcal{O}(t \cdot |Q|^2)$.

The space consumption of the PAA to compute the entire hit distribution (Problem 2) is $\mathcal{O}(|Q| \cdot (t + |Q|))$, since for each update step we now store two tables F_{l-1} and F_l. The corresponding computation time is of magnitude $\mathcal{O}(t \cdot |Q|^2)$, when C has been stored during the construction. Since $|Q|$ grows exponentially with the number of wildcards (even after minimization), the memory requirements to store the state transitions are the method's bottleneck. For instance, we observed a multiple seed consisting of only three seeds that produces a minimal automaton of size $|Q| = 60,000$. This means, we have to allocate memory for $3,600,000,000$ doubles (8 byte), resulting in approximately 26.8 GB. Even if we scale the probabilities storing integers (4 byte), we need about 13.4 GB memory. In such cases, the approximate methods by Ilie and Ilie [52] or Kong [60] are certainly preferable.

Parameter free calculation.

Another advantage of our approach is the possibility to represent the recurrence relation in a symbolic manner. Such a parameter-free calculation has previously been presented by Mak and Benson [80]. Without setting the homology parameters in advance, a polynomial (in these parameters) can be computed from Equation (5.3.6) by means of a computer algebra system like Mathematica®. The polynomial obtained for the PH seed $111{*}1{**}1{*}1{**}11{*}111$ under an i.i.d. homology model for ungapped alignments of target length 64 is provided in (5.3.7).

$$
\begin{aligned}
&1 - q^7(120\ p^{57} + 60046\ p^{56}q + 6454323\ p^{55}q^2 + 268389309\ p^{54}q^3 + 5829433985\ p^{53}q^4 \\
&+ 78663279218\ p^{52}q^5 + 736493116933\ p^{51}q^6 + 5158005916919\ p^{50}q^7 + 28506821936930\ p^{49}q^8 \\
&+ 129302147880455\ p^{48}q^9 + 495710355475511\ p^{47}q^{10} + 1642746872975316\ p^{46}q^{11} \\
&+ 4788458918512655\ p^{45}q^{12} + 12446316851341899\ p^{44}q^{13} + 29162078256160894\ p^{43}q^{14} \\
&+ 62130106545519883\ p^{42}q^{15} + 121208665847580369\ p^{41}q^{16} + 217759100297917382\ p^{40}q^{17} \\
&+ 361933096117132953\ p^{39}q^{18} + 558610847453807986\ p^{38}q^{19} + 803021552527398866\ p^{37}q^{20} \\
&+ 1077778533431254331\ p^{36}q^{21} + 1353148889713300599\ p^{35}q^{22} + 1591547502223381478\ p^{34}q^{23} \\
&+ 1755650543264156900\ p^{33}q^{24} + 1817824562288653658\ p^{32}q^{25} + 1767643947486681481\ p^{31}q^{26} \\
&+ 1614719381147262338\ p^{30}q^{27} + 1385790635445210792\ p^{29}q^{28} + 1117254821139848731\ p^{28}q^{29} \\
&+ 845940164130258653\ p^{27}q^{30} + 601264312631447468\ p^{26}q^{31} + 400925632502881515\ p^{25}q^{32} \\
&+ 250609569910552071\ p^{24}q^{33} + 146708890435753575\ p^{23}q^{34} + 80343866157172783\ p^{22}q^{35} \\
&+ 41107080446425363\ p^{21}q^{36} + 19619517499141187\ p^{20}q^{37} + 8719836468607201\ p^{19}q^{38} \\
&+ 3601681546299789\ p^{18}q^{39} + 1379369096282095\ p^{17}q^{40} + 488526802204385\ p^{16}q^{41} \\
&+ 159518986099945\ p^{15}q^{42} + 47855698857794\ p^{14}q^{43} + 13136858747458\ p^{13}q^{44} \\
&+ 3284214700565\ p^{12}q^{45} + 743595781777\ p^{11}q^{46} + 151473214816\ p^{10}q^{47} \\
&+ 27540584512\ p^9q^{48} + 4426165368\ p^8q^{49} + 621216192\ p^7q^{50} + 74974368\ p^6q^{51} \\
&+ 7624512\ p^5q^{52} + 635376\ p^4q^{53} + 41664\ p^3q^{54} + 2016\ p^2q^{55} + 64\ pq^{56} + q^{57})
\end{aligned}
\tag{5.3.7}
$$

Herewith, one can quickly assess the sensitivity of a seed under different parameter values. This strategy is of particular interest when we investigate optimal parameter ranges of competing seeds. We will see later (in Section 5.4.3) that different seeds turn out to be optimal for different similarity levels.

5.4 Results: Comparison to Other Methods

In this section, we show that the sensitivity computed by means of the presented PAA agrees with formerly published values. Notice that the PAA approach comprises seed models for ungapped and gapped alignments, and appropriate homology models. Furthermore, we give exact probabilities for the sensitivity of a multiple seed as well as for the entire distribution of the number of seed hits to a random alignment. We present a greedy-covering algorithm to design good multiple seeds and finally, we discuss alternative criteria to rank seeds. Since PH was the first tool that investigated spaced seeds and thus improved the sensitivity/specificity trade-off significantly, subsequently designed seeds were usually compared to the default PH seed $\pi_{PH} = 111{*}1{**}1{*}1{**}11{*}111$. Moreover, the analyses focussed on the PH seed class $(18, 11)$. In order to compare to existing results, we follow the field.

Table 5.2: Sensitivity of the four top-ranking seeds from the PH seed class $(18, 11)$ under the i.i.d. model for different homology parameters and target length 64. For lower homology levels, the PH seed $\pi_{PH} = 111{*}1{**}1{*}1{**}11{*}111$ is optimal. For high similarity levels, the second seed is more sensitive.

p	π_{PH}	$111{**}1{*}11{**}1{*}1{*}111$	$11{*}1{*}1{*}11{**}1{**}1111$	$111{**}11{*}1{**}1{*}1{*}111$
0.1	4.69997×10^{-10}	4.69979×10^{-10}	$\mathbf{4.70020 \times 10^{-10}}$	4.69934×10^{-10}
0.2	$\mathbf{9.62340 \times 10^{-7}}$	9.62336×10^{-7}	9.62332×10^{-7}	9.62334×10^{-7}
0.3	$\mathbf{8.30140 \times 10^{-5}}$	8.30085×10^{-5}	8.30049×10^{-5}	8.30065×10^{-5}
0.4	**0.00193**	**0.00193**	**0.00193**	**0.00193**
0.5	**0.02120**	0.02118	0.02117	0.02117
0.6	**0.13172**	0.13158	0.13140	0.13140
0.7	**0.46712**	0.46698	0.46613	0.46602
0.8	0.88207	**0.88240**	0.88142	0.88121
0.9	0.99843	**0.99848**	0.99842	0.99840

5.4.1 Spaced Seeds

Spaced seeds have previously been intensely studied by many authors. We exemplarily show that our approach yields the same sensitivity values. To this end, we consider the PH seed class $(18, 11)$ and the class $(11, 7)$ with a target alignment length $t = 64$ as it has been done by Choi and Zhang [26]. Regarding the same (top-ranking) four seeds, we computed the sensitivities shown in Tables 5.2 and 5.3.

In general, the computed sensitivities agree with those determined in the original article [26] (data not shown). However, values given there for class $(11, 7)$ and low similarity levels are erroneous. Instead of sensitivity 0.05398 for $p = 0.1$ we computed 5.39821×10^{-6}, and instead of sensitivity 0.1140 for $p = 0.3$ our result is 0.01140. Our values are consistent with the polynomial obtained from the recurrences. Furthermore, our results agree with the following plausibility argument: Let h_i be the probability of the event that the seed from $(11, 7)$ hits the random alignment at position i. If we assume independence of positions, then $h_i = p^7(1-p)^4 + 4p^8(1-p)^3 + \binom{4}{2}p^9(1-p)^2 + \binom{4}{3}p^{10}(1-p) + p^{11}$ for all positions $11 \leq i \leq 64$. For $p = 0.1$ this rough calculation yields a sensitivity of 5.40021×10^{-6}, which shows that our results exhibit the appropriate magnitude.

5.4.2 Indel Seeds

To show that our approach also works for indel seeds and thus for alignments with gaps, we compared our results to sensitivities provided by Mak *et al.* [79]. To

Table 5.3: Sensitivity of the four top-ranking seeds from the seed class $(11, 7)$ under the i.i.d. model for different homology parameters and target length 64.

p	11**1*1*111	11*1**1*111	11*1*1**111	1*111**1*11
0.1	5.39821×10^{-6}	5.39820×10^{-6}	$\mathbf{5.39822 \times 10^{-6}}$	5.39786×10^{-6}
0.2	$\mathbf{6.87015 \times 10^{-4}}$	6.86994×10^{-4}	6.87002×10^{-4}	6.86520×10^{-4}
0.3	**0.01140**	**0.01140**	**0.01140**	0.01137
0.4	**0.07772**	0.07764	0.07763	0.07738
0.5	**0.29376**	0.29302	0.29294	0.29218
0.6	**0.66225**	0.66005	0.65992	0.65954
0.7	**0.93649**	0.93473	0.93468	0.93516
0.8	**0.99805**	0.99786	0.99786	0.99796
0.9	**0.9999992**	0.99999892	0.99999892	0.99999914

Table 5.4: Sensitivity of indel seeds. Homology parameters are given as (t,p_1,p_0,p_g). We compare the sensitivities given by Mak *et al.* [79] to our values.

indel seed	homology parameters	Mak *et al.* [79]	our sensitivity
11*11?1*1111	$(64, 0.7, 0.25, 0.025)$	0.775944	0.777538
11*11?1*1111	$(100, 0.7, 0.25, 0.025)$	0.915175	0.915521
11111*111?1111	$(64, 0.7, 0.2, 0.05)$	0.368235	0.373607
11111*111?1111	$(100, 0.7, 0.2, 0.05)$	0.536510	0.536843
1111111?11111	$(64, 0.75, 0.1, 0.075)$	0.612365	0.613500
1111111?11111	$(100, 0.75, 0.1, 0.075)$	0.792340	0.792590

this end, we investigated different seeds under a Markov homology model with $\Sigma = \{0, 1, 2, 3\}$ and transitions according to (5.2.1). Note that in the original article, the authors use *normalized positions*. This means, only positions in the representative string that refer to a character in the query are counted. Thus, any 2 in $\mathcal{A}$ does not contribute to the target alignment length t. We used the expected number $\bar{t}$ of characters to read up to the t^{th} normalized position and computed $S(\pi, \bar{t})$ as well as $S(\pi, t)$ according to Equation (5.3.4). The results in Table 5.4 and Table 5.5 show that sensitivity $S(\pi, t)$ is in general overestimated when using normalized positions (Mak *et al.* [79], $S(\pi, \bar{t})$). The reason for this is that some alignments are ignored, although sensitivity is defined as the fraction of sequence alignments among all local alignments that are matched by the seed. Comparing spaced and indel seeds of equivalent random hit rates, Mak *et al.* [79] observed that with increasing indel to mismatch ratio, indel seeds outperform spaced seeds, even if the "winning seed" changes with target length. Our method yields the same results (see Table 5.5). Note that the random hit rate for indel seeds depends on its weight as well as on the number of comparisons between any query and database pattern. For

Table 5.5: Sensitivity of pairs of spaced and indel seeds with equivalent random hit rates for different homology parameters (t,p_1,p_0,p_g). We compare the sensitivities given by Mak *et al.* [79] to our values $S(\pi, \bar{t})$ and $S(\pi, t)$. Winning seeds are shown in bold.

seed	homology parameters	Mak *et al.* [79]	$S(\pi, \bar{t})$	$S(\pi, t)$
1111∗111111	$(64, 0.7, 0.2, 0.05)$	0.488697	0.494082	0.470369
1111∗111?1111	$(64, 0.7, 0.2, 0.05)$	0.487703	0.492503	0.468351
1111∗111111	$(100, 0.7, 0.2, 0.05)$	0.669123	0.668821	0.649304
1111∗111?1111	$(100, 0.7, 0.2, 0.05)$	0.670198	0.669884	0.650131
111∗11∗1111	$(64, 0.8, 0.15, 0.025)$	0.943899	0.943609	0.937750
111∗11?11∗111	$(64, 0.8, 0.15, 0.025)$	0.943214	0.942985	0.936985
111∗11∗1111	$(100, 0.8, 0.15, 0.025)$	0.991157	0.990945	0.989497
111∗11?11∗111	$(100, 0.8, 0.15, 0.025)$	0.991239	0.991044	0.989594

instance, the random hit rate of the seed 11??11 is approximately $9 \cdot (1/4)^4$, since for each query position we extract and compare three query patterns with three database patterns (refer to Example 5.2.1) which are required to match at four positions. The accurate probability is actually a little less, because the extracted patterns are not independent.

Remark 5.4.1. *The restriction to one '?' follows the original article [79]. Indel seeds with more '?' wildcards produce a different number of random hits and thus have to be compared to different spaced seeds.*

5.4.3 Multiple Seeds

It is already NP-hard to find a single globally optimal seed [70]. Namely, we have to compute the sensitivity for all seeds of a given class, and the computation takes $\mathcal{O}(t \cdot |Q|^2)$ time for a single seed. In the case of multiple seeds, exhaustive search to find the optimal seed is infeasible because there are two exponential factors. Besides the exponential number of seeds in a given seed class, also the search space, i.e. all combinations of seeds of equal weight, grows exponentially with the number n of seeds. However, "good" multiple seeds are computed by heuristic algorithms presented in [60] and [52]. These approaches use different quality measures correlated with sensitivity. Our exact sensitivity values indicate that their approximations are reasonable (see Table 5.6). As expected, since both methods are exact, our values agree with the dynamic programming algorithm presented by Li *et al.* [70] (data not shown).

Recall that by adding seeds to the search algorithm, we can substantially increase the method's sensitivity while the false positive rate is increased at most linearly (Section 5.2.2). To obtain a good multiple seed, we seek for seeds that produce approximately the same number of overall hits (regulating the specificity), while their

Table 5.6: Sensitivity of formerly approximated multiple seeds. Target alignment length is 64, similarity levels are given in parentheses. Besides, we give the size of the PAA before and after minimization. Considered sets of seeds are $A = \{11{*}1{*}11{**}11{*}1111,\ 111{*}11{**}1{*}1{*}1{**}111\}$, $B = \{111{*}1{*}1{**}1{*}11{*}111,\ 1111{***}1{**}1{***}1{*}1{*}111\}$. We compare our exact sensitivities to approximated values given by Kong [60] (1) and Ilie and Ilie [52] (2).

multiple seed	$\lvert Q\rvert$	$\lvert Q_{\min}\rvert$	approximated sensitivity	our sensitivity
$A \quad (p = 0.75)$	902	498	0.815865 **(1)**, 0.823314 **(2)**	0.820278
$A \quad (p = 0.50)$	902	498	0.038393 **(1)**, 0.038554 **(2)**	0.038855
$B \quad (p = 0.70)$	6276	1563	0.621992 **(2)**	0.581914

detection abilities are complementary (maximizing sensitivity). Since the overall number of hits is mainly determined by the weight, seeds in a multiple seed are limited to the same weight.

We can also apply the PAA framework to design efficient sets of seeds in a greedy manner. The corresponding algorithm is summarized in Algorithm 5.1. Similar to [116], we can successively find seeds (of the same weight and a length at most L) with a complementary set of detectable similarities, i.e. seeds that locally maximize the conditional probability to hit a random alignment, given that the seeds in the set do not match:

$$\mathbb{P}(V_t(\pi) \geq 1 \,|\, V_t(\Pi) = 0) \quad \text{for } \pi \notin \Pi\,. \tag{5.4.1}$$

To this end, we start from the PAA for an initial seed $\Pi = \{\pi_0\}$. Then, for all seeds of the same weight and a length limited by L, we modify a copy of the current PAA such that we can compute the conditional hit probability. The modification includes that states corresponding to $\mathcal{PS}(\Pi)$ are marked as *accepting states* and are removed from the set of final states (line 9). Transitions outgoing an accepting state are set to zero, i.e. once the automaton visits one accepting state, it stays there. Thus, the automaton does no longer count alignments which are also matched by Π. The seed exhibiting the maximal conditional hit probability is chosen as new member of the multiple seed. This procedure is repeated $(n-1)$ times to generate a set of n seeds.

It has been shown by Sun and Buhler [116] that this greedy design yields nearly the same results as the local search implemented in the Mandala software [20], whereas it has a clear advantage in computational cost. The authors identified a seven-fold speedup. This results from the following facts. The local search method starts with n randomly chosen seeds, which are successively modified until no further local improvement in sensitivity is possible. In contrast, greedy covering starts with one seed and has to optimize only one seed in each step. Moreover, most evaluations are performed on sets of $n' < n$ seeds. Clearly, there is no guarantee that the sensitivity of a greedily designed multiple seed is as high as the sensitivity of a set designed simultaneously. To improve the quality of greedy design, one can, for instance, restart the procedure for several initial seeds.

Algorithm 5.1 DESIGNMULTIPLESEED

Input: initial seed π_0 with weight ω, $\Pi = \{\pi_0\}$, PAA(Π) constructed from $\mathcal{PS}(\pi_0)$, number of seeds n
Output: multiple seed Π of size n

$\mathcal{A} \leftarrow \mathcal{PS}(\pi_0)$ //set of accepting states
$\mathcal{F} \leftarrow \emptyset$ //set of final states
while $|\Pi| < n$ **do**
  max $\leftarrow 0$
  candidate $\leftarrow \epsilon$
  for all $\pi \notin \Pi$ with weight ω and $|\pi| \leq L$ **do**
    PAA' $\leftarrow$ copy of PAA(Π)
    add prefixes of π to PAA'
    $F \leftarrow \mathcal{PS}(\pi) \backslash \mathcal{A}$
    set transition probabilities outgoing $\mathcal{A}$ to 0
    hitProb $\leftarrow \mathbb{P}(V_t \geq 1)$ in PAA' //conditional hit probability
    if hitProb $>$ max **then**
      max $\leftarrow$ hitProb
      candidate $\leftarrow \pi$
    end if
  end for
  add candidate to Π, $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{PS}$(candidate)
end while
output Π

5.4.4 Alternative Criteria

In fact, a good seed for homology search should satisfy two competitive criteria:

1. It should maximize the sensitivity (hitting probability) to alignments referring to homologous sequences.
2. At the same time, it should maximize the specificity (probability of no hit) to alignments referring to random, unrelated sequences.

To this end, we investigated optimality criteria which take both conditions into account. Instead of considering only one homology model and maximizing sensitivity, we deal with two alternative homology models; one describing alignments of homologous sequences, referred to as M_{hom}, and one describing alignments of random, unrelated sequences, referred to as M_0. Then, the random variables $V_t(M_{hom})$ and $V_t(M_0)$ denote the number of hits of a given seed up to target length t under the former and the latter model, respectively.

Furthermore, the PAA approach yields the entire distribution of overlapping and non-overlapping seed hits. We also used this fact in order to describe alternative criteria for seed quality, both under a single homology model as well as for the presented competitive criteria. The first optimality criterion we defined intends to rank seeds subject to their probability to produce at least two non-overlapping

hits in a random alignment generated according to a single homology model. This corresponds to the FASTA approach [97]. There, several top-scoring initial hits are combined to an optimal alignment if the penalty for joining them is small enough such that the resulting score exceeds a specified threshold. We refer to this as the *fasta-hits* criterion. Moreover, in analogy to the plausibility argument why spaced seeds are more sensitive than consecutive seeds of the same weight (Section 5.2.2), a seed should produce few (redundant) hits to a single alignment. Note that those hits might overlap. We call the criterion ranking seeds according to the probability to produce few overlapping hits the *few-hits* criterion.

Definition 5.4.2 (Optimality criteria based on seed hits)**.** *A seed from seed class* (ℓ, ω) *is said to be optimal if it maximizes*

fasta-hits*:* $\mathbb{P}(V_{t;n} \geq 2)$*.*
few-hits*:* $\sum_{i=1}^{k} \mathbb{P}(V_t = k)$ *for* $k = 1, \ldots, K$*.*

Sensitivity and Fasta-hits criterion

A good seed should be sensitive to alignments in a wide range of similarity values, since homologous sequences also have diverse similarity. To analyze parameter ranges in which seeds are optimal, we calculate the distributions (5.3.5) of overlapping and non-overlapping hits for all seeds from seed classes $(11, 7)$ and $(18, 11)$ (data not shown). Then, we investigate the partitioning of the parameter space of the i.i.d. homology model according to the fasta-hits criterion and seed sensitivity. Figure 5.3 shows the parameter ranges for best-performing seeds from the PH seed class $(18, 11)$ under both criteria for target alignment lengths 50, 64, and 100. In the course of our studies, we find that the seeds given in Table 5.7 are the best-performing seeds under various homology parameters across different target lengths.

Table 5.7: Seeds from seed class $(18, 11)$ that occur frequently among the best-performing seeds under various homology parameters, target lengths, and different optimality criteria. Seed A is the default PH seed.

A	111∗1∗∗1∗1∗∗11∗111
B	111∗∗1∗11∗∗1∗1∗111
C	111∗1∗∗11∗1∗1∗∗111
D	11∗∗111∗1∗∗1∗111∗1
E	1111∗1∗11∗∗1∗∗∗111

For very low similarity levels $p \leq 0.25$, i.e. in alignments with fewer matches than would be observed by chance (under i.i.d. uniformly distributed characters), the top-ranking seed alters frequently, which is denoted “miscellaneous”. First, we

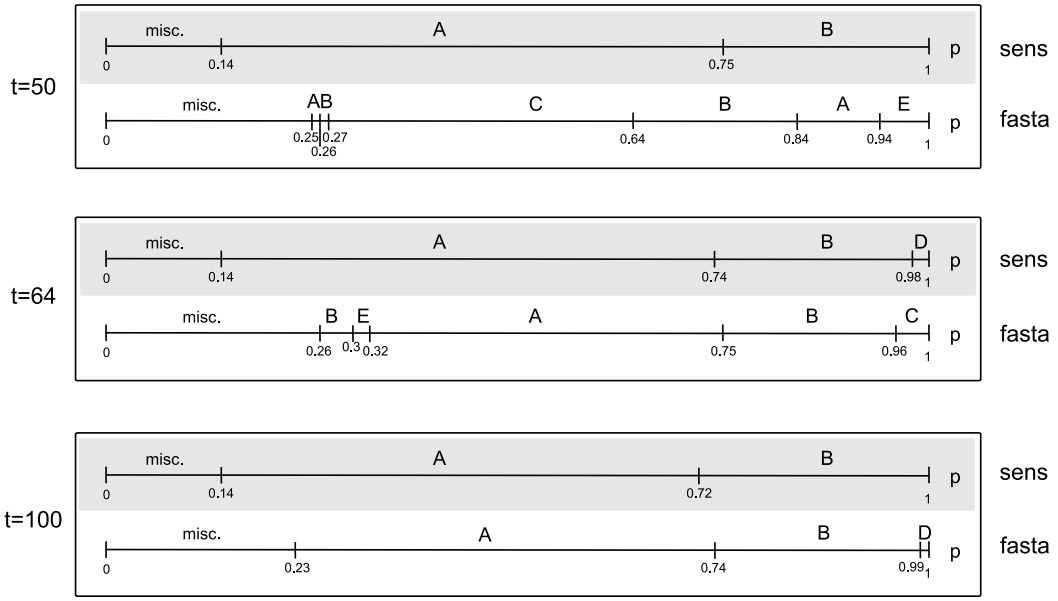


Figure 5.3: Parameter ranges for best-performing seeds from the PH seed class $(18, 11)$ according to sensitivity (top,gray) and the fasta-hits criterion (bottom) for the i.i.d. homology model at target alignment lengths 50, 64, and 100. The top-ranking seeds are given in Table 5.7.

observe that (according to any criterion) different seeds are optimal for different similarity levels. Furthermore, optimality ranges vary as homology region length varies. These properties were also expected from earlier studies [20, 26, 27, 80]. Note, however, that the set of winning seeds according to sensitivity remains relatively stable over a range of lengths, whereas there exist more differences according to the fasta-hits criterion. Moreover, with increasing target alignment length, both criteria agree in the top-ranking seeds for large parts of the parameter space. Nevertheless, they choose different seeds for smaller alignment lengths and for extreme values of p, i.e. to detect closely related and distantly related sequences.

For some parameters, the optimal seed has only a slightly higher sensitivity than its competitors. In such cases, another criterion might help to identify a unique seed. For instance, for $t = 64$ and $p = 0.4$, where the four top-ranking seeds have almost the same sensitivity (cf. Table 5.2), we observed the PH seed A to maximize the fasta-hits criterion clearly.

Few-hits criterion

Next, we investigate the few-hits criterion. It is based on the idea that a seed producing less hits to an alignment, i.e. redundant hits in respect of detecting similarities, finds more putatively homologous sequences and is thus more sensitive. The few-hits criterion appoints a seed to be good if it hits alignments at least

once, but at most k times, for different values of $k = 1, \ldots, K$. We considered non-overlapping hits, because otherwise the criterion would be too restrictive for high similarity levels, where overlapping hits are very likely. The according probabilities were computed up to $K = 10$ for the five seeds from class $(18, 11)$ given in Table 5.7. These were found to be the top-ranking seeds with respect to sensitivity and the fasta-hits criterion, cf. Figure 5.3. For target alignment length $t = 64$, we analyzed the few-hits probabilities under several homology parameter settings. The corresponding rankings for $p = 0.5$, $p = 0.7$, and $p = 0.9$ are presented in Table 5.8. For

Table 5.8: Seed-ranking according to the few-hits criterion for the candidate seeds given in Table 5.7 for target length 64 and homology parameters $p = 0.5$, $p = 0.7$, and $p = 0.9$ for different parameters $k = 1, \ldots, 5$. Rankings that agree with sensitivity are highlighted.

	$p = 0.5$					$p = 0.7$					$p = 0.9$				
k	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**5**	**4**	5	4	1	3	2
2	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**5**	**4**	5	4	1	2	3
3	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**5**	**4**	**2**	**3**	**1**	**4**	**5**
4	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**5**	**4**	**2**	**3**	**1**	**4**	**5**
5	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**5**	**4**	**2**	**3**	**1**	**4**	**5**

$k = 6, \ldots, 10$, the rankings remain the same for all similarity levels. The rankings that agree with the ranking according to sensitivity are highlighted. This shows that in the majority of cases, the few-hits criterion is concordant with sensitivity. Thus, our reasoning holds, but does not provide further information.

Sens-spec criterion

Furthermore, we considered an optimality criterion based on two i.i.d. homology models M_{hom} and M_0 with varying similarity levels p_{hom} and p_0. An intuitive value for p_0 is 0.25, since this refers to the probability to observe a match just by chance. We used the independence of M_{hom} and M_0 and defined the following optimality criterion in order to maximize the conjunction of the competitive conditions mentioned at the beginning of this section.

Definition 5.4.3 (Optimality criterion based on two homology models)**.** *A seed from seed class (ℓ, ω) is said to be* optimal *if it maximizes*

***sens-spec**: $\mathbb{P}(V_t(M_{hom}) \geq k) \cdot \mathbb{P}(V_t(M_0) < k)$ for $k = 1, \ldots, K$.*

This criterion is based on the idea to maximize sensitivity for M_{hom} *and* specificity for M_0, which refers to the case $k = 1$. To put it differently, it aims at maximizing the probability to hit alignments of homologous sequences and at the same time minimizing the probability to hit alignments of unrelated sequences, i.e. computing $\mathbb{P}(V_t(M_{hom}) \geq k) - \mathbb{P}(V_t(M_0) \geq k)$ for $k = 1, \ldots, K$. This is essentially

the same as **sens-spec** since M_{hom} and M_0 are independent, and the probability $\mathbb{P}(V_t(M_{hom}) \geq k \,|\, V_t(M_0) \geq k)$ is close to one. Note that all criteria depending on k are maximized for $k = 1$ since $\mathbb{P}(V_t \geq k)$ decays exponentially with k.

We computed the sens-spec values for $K = 10$, $t = 64$, and various combinations of p_{hom} and p_0 for the candidate seeds given in Table 5.7. A selection of corresponding seed rankings is presented in Table 5.9.

Table 5.9: Seed-ranking for the candidate seeds given in Table 5.7 for target length 64 according to the sens-spec optimality criterion and several homology parameters. Seed rankings for the same value of p_{hom} agree for different p_0.

	$p_{hom} = 0.50$, $p_0 = 0.25$					$p_{hom} = 0.70$, $p_0 = 0.25$					$p_{hom} = 0.90$, $p_0 = 0.25$				
k	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	1	2	3	4	5	1	2	3	5	4	3	1	2	4	5
2	5	4	3	2	1	1	3	2	5	4	2	1	3	4	5
3	5	4	3	2	1	2	4	3	5	1	2	1	3	5	4
4	5	4	3	1	2	5	4	3	1	2	1	2	3	**4**	**5**
5	5	4	3	1	2	5	4	3	1	2	1	2	3	5	4
	$p_{hom} = 0.50$, $p_0 = 0.40$					$p_{hom} = 0.70$, $p_0 = 0.40$					$p_{hom} = 0.90$, $p_0 = 0.40$				
k	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	1	2	3	4	5	1	2	3	5	4	3	1	2	4	5
2	5	4	3	2	1	1	3	2	5	4	2	1	3	4	5
3	5	4	3	2	1	2	4	3	5	1	2	1	3	5	4
4	5	4	3	1	2	5	4	3	1	2	1	2	3	**5**	**4**
5	5	4	3	1	2	5	4	3	1	2	1	2	3	5	4

We observe that, for the same p_{hom}, the rankings mainly agree irrespective of p_0 (see Table 5.9). We found that this holds for the most part of rankings up to $p_0 = 0.5$. Typical alignments of unrelated sequences do not contain more than 50% matches. This means, in general, it suffices to consider one homology model describing alignments of putatively homologous sequences.

Moreover, both tables show the dependence of seed ranking according to the sens-spec criterion on the parameter k. As mentioned earlier, the case $k = 1$ corresponds to the sensitivity/specificity tradeoff. In order to evaluate the quality of different seed rankings, we made the following analysis:

1. We searched the nucleotides database from NCBI with BLASTALL for the DNA sequence of the complete genome of *Corynebacterium glutamicum ATCC 13032* as query.

2. The same search was performed by means of PH with each of the candidate seeds from Table 5.7.

3. We compared the resulting hits from each PH search to the BLAST results with the intention to note additional and missing hits.

This comparison yielded no distinction in the detection ability of the seeds since PH and BLAST results were in agreement. For future research on this topic, a possible approach to biologically judge different seed rankings would be the following: first, one selects sets of sequences that are known to be homologous and that have a certain degree of sequence similarity. Then, one performs a PH search for one of the sequences as query in a nucleotide sequence database. A good seed should yield all sequences in the selected set. Since different seeds have been shown to be optimal for different similarity levels, this property should also be evident from the analysis.

If we can get an intuition which k leads to the most reasonable ranking, it would also be of interest to investigate more plausible homology models M_{hom} and M_0. In particular, training models for coding and non-coding regions should for instance help to identify reasonable criteria and optimal seeds when searching for homologous genes.

CHAPTER 6

APPLICATION III: 454 SEQUENCING READ STATISTICS

In this chapter, we present another application of the PAA framework in the context of DNA sequence analysis. Here, we construct a PAA that computes the length distribution of sequencing reads produced by an up-to-date large-scale sequencing technology.

First, we give a brief introduction to relevant DNA sequencing methods (Section 6.1) and explain the 454 sequencing technology in particular (Section 6.2). Then, we formulate a PAA to calculate the length distribution of sequence reads resulting from the 454 sequencing method in Section 6.3. On this basis, we investigate a potential quality improvement of the sequencing method in Section 6.4.

6.1 Introduction to DNA Sequencing

DNA sequencing refers to the determination of the sequence of the four nucleotide bases within a DNA oligonucleotide. First DNA sequences have been discovered by laborious methods based on two-dimensional chromatography in the early 1970s. Since then, with the development of dye-based labeling techniques and automated analysis, sequencing methods became faster and more convenient. Today, knowledge of DNA sequences (of genes or regulatory elements, for instance) has become indispensable for the exploration of biological processes as well as for applied fields like diagnostics or forensic research.

The advent of DNA sequencing has significantly accelerated biological research and discovery. In 1990, the Human Genome Project was founded with the objective of sequencing an entire human genome. An elaborate genome map would provide the

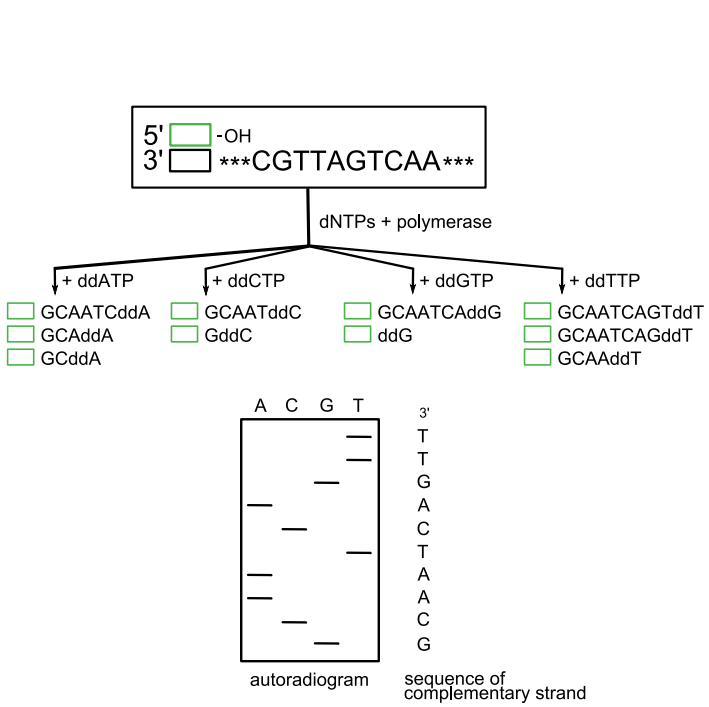
basis for a better understanding of the molecular processes of complex diseases like cancer, diabetes or Alzheimer's disease. Related projects have revealed the DNA sequences of many microbial, plant and animal genomes. In recent years, these projects have driven the development of high throughput sequencing technologies that can parallelize the sequencing process, and hence yield thousands or millions of sequences at once. Furthermore, these strategies are intended to work at a lower cost, per sequenced nucleotide, than standard techniques. Indeed, while sequencing a human genome first consumed 300 million U.S. dollars, the costs have been reduced to $100,000$ dollars today. This advance makes the 1000 Genomes Project possible. Initiated in 2008, it should provide the entirely sequenced genomes of approximately 1200 people from around the world during the next three years. The intention is to generate a more detailed map of the human genome and hence, get an improved picture of DNA variations, especially so-called single nucleotide polymorphisms (SNPs). These point mutations, which occur with a frequency of about 1%, are of special interest, because they can influence the development of diseases and the response to agents like drugs, chemicals or pathogens. In biomedical research, SNPs are used to compare genomic regions of individuals affected and not affected by a disease.

Craig Venter, whose genome was the first entirely sequenced [69], announced an award of $500,000$ dollars for the group that first provides a human genome for a price of only 1000 dollars. This indicates the direction of future progress. Although higher throughput is achieved at the expense of shorter sequence reads (DNA fragments), which implicates the need for improved genome assembly tools, the subject of cost-efficient massively parallel DNA sequencing plays an important role in today's molecular biology, especially in biomedical research. One famous representative of such second-generation sequencing strategies, allowing for parallelization, is Roche's 454 sequencing. We outline this technology after a brief review of traditional methods.

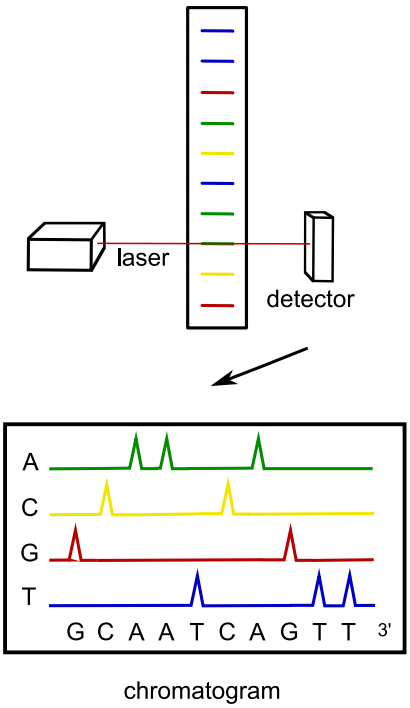
6.1.1 Sanger Sequencing

The possibility to "read" the sequence of nucleotides has induced the era of genome research and the application of bioinformatics tools in sequence analysis. The first sequencing methods were introduced by Maxam and Gilbert [86] and Sanger *et al.* [109] in 1977. Since *Sanger's chain termination* technique uses less toxic chemicals and lower amounts of radioactivity, it rapidly became the method of choice.
For this enzymatic method, the DNA double helix is first heat denatured into single strand templates. Then, starting from a short known primer sequence, the template's complementary strand is elongated by means of the enzyme DNA polymerase. The idea is to generate fragments that end with an identifiable nucleotide and to construct the order of such fragments afterwards. Therefore, the DNA sample is divided into four separate sequencing reactions, all of them containing DNA polymerase and deoxynucleotides (dATP, dCTP, dGTP, dTTP). Moreover, one kind of labeled dideoxynucleotide (ddNTP) is added to each reaction. These molecules

lack the 3'-OH group required to form the phosphodiester bond between two nucleotides. Hence, when a ddNTP is inserted into the synthesized strand, the elongation process is terminated. This results in DNA fragments of various lengths. These fragments are subsequently separated by size by gel electrophoresis, each reaction run on one individual lane of the gel. Since this separation works at a resolution of one nucleotide, the sequence can be read off the visualized bands, as shown in Figure 6.1(a).



(a) Principle of the original chain termination method using four individual reactions and dideoxynucleotides (ddNTPs) labeled with a radioactive tag.



(b) Labeling ddNTPs with fluorescent tags enables sequencing in a single reaction.

Figure 6.1: Overview of the Sanger chain termination method. Dideoxynucleotides (ddNTPs), which terminate the chain elongation by DNA polymerase, are labeled and added to the synthesizing reaction. This results in fragments of varying length which are subsequently separated by size by electrophoresis. The DNA sequence can be read off from the gel bands on the autoradiogram (a) or the chromatogram of fluorescence peaks (b), respectively.

Since the 1990s, dideoxynucleotides are usually labeled with fluorescent dyes with different wavelengths. This allows sequencing in one single reaction rather than separating the sample into four reactions. The resulting fragments are then separated by capillary electrophoresis and stimulated by a laser. Detecting the triggered fluorescence, the sequence of nucleotides can be determined from the chromatogram of fluorescent peaks, see Figure 6.1(b). The development of fluorescently labeled ddNTPs was a prerequisite for automated, high throughput sequencing.

This is of particular importance in large scale sequencing that aims at sequencing large pieces of DNA, such as whole chromosomes. Conventional methods first fractionate the DNA into shorter fragments which are subsequently cloned into a vector and amplified in *E. coli*. Then, the clones are purified from the bacterial colonies and sequenced individually. The resulting sequence reads are typically $800-1000$ nucleotides long. Finally, these reads have to be electronically assembled into one long, contiguous sequence.

6.1.2 Pyrosequencing

Another method offering high throughput automated sequencing is the so-called *pyrosequencing*. This technique was developed by Ronaghi *et al.* [107] in 1996. In contrast to the idea of chain termination, it relies on the detection of pyrophosphate (PPi) released in the course of DNA synthesis by means of the enzyme DNA polymerase. This refers to the sequencing by synthesis principle which involves sequencing a single strand of DNA by synthesizing its complementary strand one base at a time. Therefore, the template strand is immobilized, and unlabeled deoxynucleotides (dNTPs) are added sequentially and removed after the reaction. When a nucleotide is complementary to the first unpaired base of the template, a light signal is produced by means of two more enzymatic reactions. The detection and subsequent analysis of such chemi-luminescent signals finally determines the sequence of nucleotides.
The enzymatic cascade is shown in Figure 6.2. Whenever the polymerase catalyzes the incorporation of nucleotides that are complementary to the template strand, inorganic pyrophosphate (PPi) is released and converted to ATP by ATP sulfurylase. ATP provides energy for the luciferase-mediated oxidation of luciferin, accompanied by a flash of light. The emitted light is recorded by a charge-coupled device (CCD) camera, for instance. The light intensity is proportional to the amount of ATP.

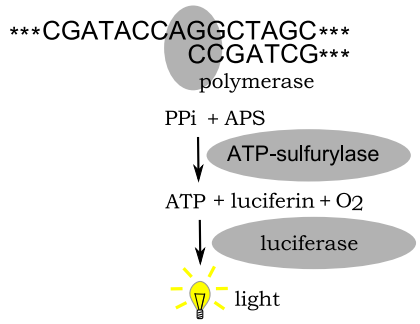


Figure 6.2: Schematic representation of the pyrosequencing method. The four nucleotides are periodically added to the reaction. An incorporation by the polymerase results in an enzymatic cascade producing a detectable flash of light.

Currently, two different pyrosequencing strategies are available: solid-phase py-

rosequencing [107], which utilizes the three-enzyme system described previously, and liquid-phase pyrosequencing [108] using a four-enzyme system. Here, the introduction of the nucleotide-degrading enzyme apyrase allows nucleotides to be added iteratively without any intermediate washing step which is necessary in solid-phase. Apyrase continuously decomposes unincorporated dNTPs and excessive ATP:

$$\begin{aligned} \text{dNTP} &\xrightarrow{\text{apyrase}} \text{dNDP} + \text{dNMP} + \text{phosphate} \\ \text{ATP} &\xrightarrow{\text{apyrase}} \text{ADP} + \text{AMP} + \text{phosphate} \end{aligned}$$

The next deoxynucleotide is added, when degradation is complete. As described in Ronaghi [106], apyrase fulfills the following criteria necessary to allow iterative nucleotide addition: First, all dNTPs are degraded at the same rate. Second, ATP is hydrolyzed and, hence, accumulation of ATP (stimulating luciferase) between cycles is prevented. Third, the time for nucleotide degradation is slower than nucleotide incorporation by the polymerase. Besides, it is important to note that the apyrase ensures primer-directed incorporation to achieve about 100% before the concentration of dNTPs is too low for rapid polymerization.

Pyrosequencing allows fast, high throughput generation of sequenced nucleotide data. It is limited to reads of $300 - 500$ nucleotides length. This can make the process of genome assembly more difficult than it is for reads resulting from chain termination methods, in particular for genomes which contain large amounts of repetitive DNA.

> "Overall, sequence analysis by using high throughput pyrosequencing technology is accurate and reproducible and can be almost fully automated. The technology is also less expensive, time-consuming, and labor-intensive, as well as easier to perform, than conventional Sanger sequencing." [120]

6.2 The 454 Sequencing Technology

454 Life Sciences[1] developed an array-based pyrosequencing technology for massively parallel sequencing, called *454 sequencing* [82]. So far, two platforms have emerged: the Genome Sequencer 20 (GS20) system in 2005 and the Genome Sequencer FLX (GS-FLX) instrument in 2007. They support the analysis of various DNA samples providing a 2 Mb (million base pairs) genome in 4.5 hours, whereas it takes approximately 720 hours with the conventional Sanger sequencing.

In the following, we will explain the 454 sequencing workflow sketched in Figure 6.3. Samples such as genomic DNA are firstly fractionated into shorter double-stranded fragments of 300 to 800 bp. Afterwards, special short adaptors, A and B,

[1] Center of excellence of Roche Applied Science, Roche Diagnostics GmbH, Mannheim

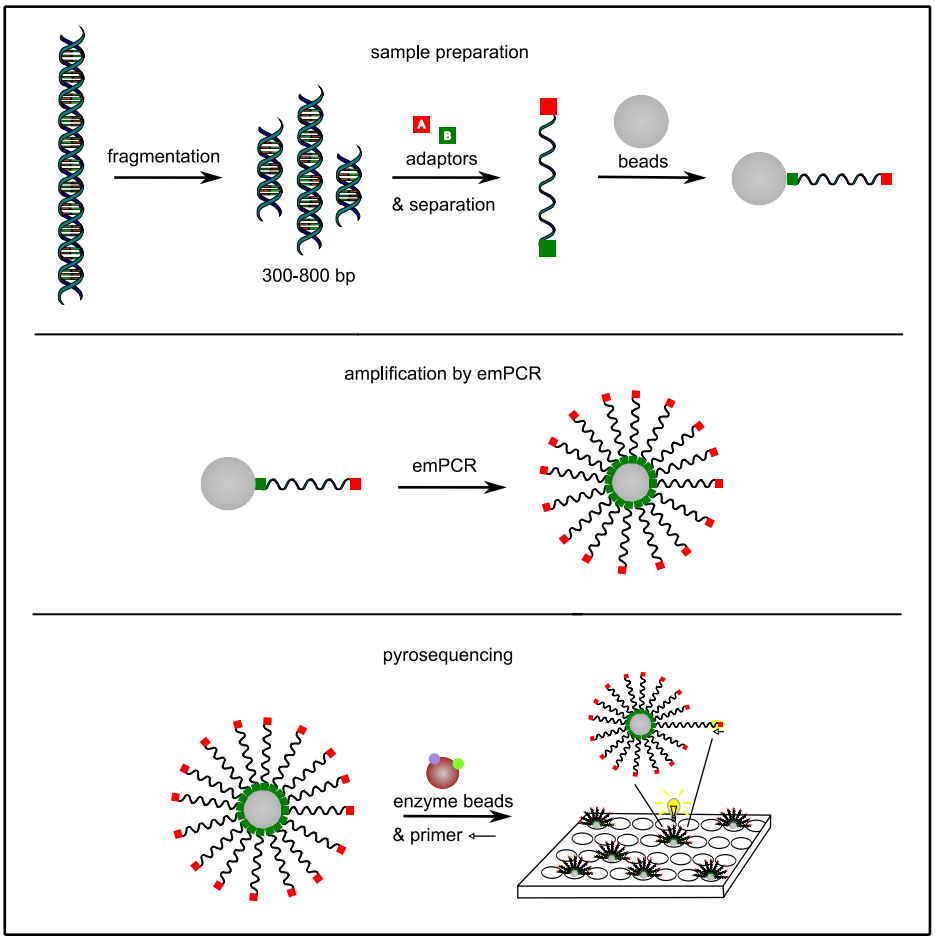


Figure 6.3: Overview of the 454 sequencing technology. DNA fragments are ligated with adaptors, separated into single strands, and immobilized onto DNA capture beads. By means of emulsion PCR the fragments are amplified, resulting in ten million clones bound to each bead. The beads are then deposited into the wells of an array-like optic chip. Additionally, enzymes, primer sequences and nucleotides for the sequencing process are provided. Finally, the unknown sequence is determined by means of pyrosequencing.

are attached to the fragments. These provide primer sequences for both amplification and sequencing. After separation into single strands, those single-stranded DNA (ssDNA) fragments with both A and B adaptors compose the sample library used for subsequent steps.

The library is now mixed with an excess of DNA capture beads that carry oligonucleotides complementary to the B-adaptor sequence. Hence, the ssDNA templates are immobilized onto these beads, usually one library fragment per bead. During the subsequent emulsion PCR (emPCR) procedure, the bead-bound library is emulsified with amplification reagents in a water-in-oil mixture. The beads are captured within droplets of this emulsion, enabling clonal amplification of the

ssDNA fragments within their own microreactors. Subsequently, each bead carries about ten million ssDNA clones.

In the next step, the beads are added to a mixture of reagents containing DNA polymerase. The mixture is deposited into wells of a fibre-optic chip and layered with smaller enzyme carrying beads (containing sulfurylase and luciferase). This layer ensures that the DNA capture beads remain within the wells during the sequencing process. The optical device is a $70mm \times 75mm$ plate containing 1.6 million wells at a diameter of $44\mu m$ each. Usually, 30% to 40% of the wells are loaded with beads, and only one bead (around $30\mu m$) fits into one well.

The loaded device is placed into the sequencing instrument, where sequencing reagents, containing primer sequences and nucleotides, are flowed across the plate. Nucleotides are flowed sequentially in a fixed order. Note that, during nucleotide flows, hundreds of thousands of bead-bound DNA fragments are sequenced in parallel. If a nucleotide complementary to the next unpaired nucleotide on the template strand is flowed into a well, it is synthesized to the template clones, mediated by the polymerase. Incorporation of one or more nucleotides provokes the enzymatic cascade described in Section 6.1.2 and results in a chemiluminescent signal that is recorded by a CCD camera. The detected light intensity is proportional to the number of nucleotides incorporated in one flow. This number mainly refers to homopolymers, that is stretches of one repetitive nucleotide. However, this linearity holds only up to stretches of about eight nucleotides [see 82].

The standard order of nucleotide flows for the two systems mentioned above is T,A,C,G. Each nucleotide is cycled 42 times on the GS20 system, 100 times on the ultra-fast GS-FLX. During the sequencing process, the system's software generates a bar graph of light intensities, called *flowgram*, for each well. These are in turn analyzed, yielding hundreds of thousands of DNA sequences simultaneously.

Actually, the GS-FLX typically generates more than $400,000$ reads per run with an average length of 250 bases. This is a total of 100 megabases of sequence data. In addition, since fall 2008, the company delivers the FLX Titanium reagents and associated software, thereby providing read lengths of 400 bp on average with a throughput of 400 to 600 megabases.

After signal processing, a consensus sequence can be determined from all reads of the sample, or a pool of sample runs, respectively. Therefore, the 454 systems provide *de novo* assembly and whole genome mapping software. After all, the average read length of approximately 400 bases (using the FLX Titanium series) is quite short in comparison to 800 bases provided by the conventional Sanger method. Thus, genome assembly is much more involved here. Nevertheless, a wide range of applications has been addressed by means of 454 sequencing [37], and it is reported that particularly the up-to-date FLX Titanium series efficiently replaces Sanger sequencing. However, both technologies have their drawbacks. While Sanger sequencing is more expensive and time consuming (per base), 454 generates shorter reads with a higher error rate when reading homopolymers.

6.3 Computing 454 Read Length Statistics with a PAA

As introduced above, current high throughput sequencing methods can directly sequence only relatively short reads of 100 to 500 nucleotides. The difficulty in obtaining longer reads consists in the insufficient separation power of longer fragments at a resolution of only one nucleotide. In large scale sequencing applications, however, one aims at sequencing a whole chromosome or even an entire genome. The required assembly is more complicated and error-prone, the shorter the sequenced reads are. Broadly speaking, the average read length has an effect on the overall quality of the assembled DNA sequence.

While the order of flowed nucleotides does not influence the read length distribution when all four nucleotides are equally probable and all nucleotides are flowed once per cycle, this property does no longer hold for arbitrary nucleotide compositions. Although it has been shown that the frequencies of complementary bases (A and T, C and G) are equal in dsDNA, and approximately equal in ssDNA [25], the joint frequencies of (A,T) and (C,G) are generally unbalanced.

To this end, we investigate the exact length distribution of 454 reads and thereby the average read length. Rahmann [102] studied the combinatorics of sequences that can be reliably sequenced by the 454 technology. Recently, Kong [61] presented the length distributions for fixed number of flow cycles and for fixed sequence length. The author gives the probability generating functions and thereby explicit formulas for the respective values for mean and variance. Nevertheless, only flow cycles of length four and exclusively i.i.d. nucleotide probabilities are considered.

Within the PAA framework, we formulate an appropriate model that allows for arbitrarily long flow cycles and different underlying string models, and assigns probabilities to the number of nucleotides incorporated by the polymerase within a 454 sequencing run. In accordance with previous sections, once again we concentrate on first-order Markovian strings. Input to the model is the fixed dispensation order of nucleotide flows in each cycle, defined as follows:

Definition 6.3.1 (Dispensation Order)**.** *A* dispensation order $d = d[1] \dots d[\ell_d]$ *is an ordering of nucleotides, where* $d[i] \in \Sigma = \{A, C, G, T\}$ *for each* $1 \leq i \leq \ell_d$ *and*

i) $\forall \sigma \in \Sigma : |\{1 \leq i \leq \ell_d \,|\, d[i] = \sigma\}| \geq 1$*. This implies* $\ell_d \geq |\Sigma|$*.*

ii) $\forall \sigma \in \Sigma, \forall 1 \leq i < \ell_d : d[i] = \sigma \Rightarrow d[i+1] \neq \sigma$*.*

iii) $d[1] \neq d[\ell_d]$*.*

The systems GS20 and GS-FLX mentioned above use $d = \text{TACG}$. For the moment, we focus on this dispensation order and, hence, deal with the case that each nucleotide is flowed exactly once per cycle, i.e. $\ell_d = |\Sigma|$. Figure 6.4 shows how a read, sequenced from a bead-bound fragment, is determined by the dispensation order.

cycle	nt flow	TACG	GTCA
1	1	-	√
	2	-	√
	3	-	√
	4	√	-
2	5	√	√
	6	-	√
	7	√	-
	8	√	√
3	9	√	-
	10	√	√
	11	-	√ (3×)
	12	-	-
read length:		6	10

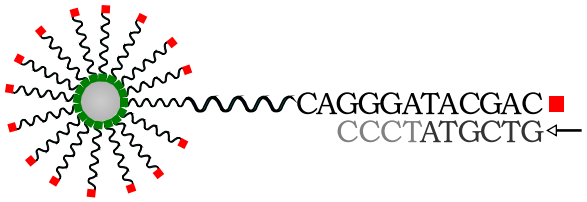


Figure 6.4: Effect of the dispensation order on the read length. Depending on the dispensation order, different numbers of nucleotides complementary to an immobilized fragment can be sequenced during three cycles.

In order to study the influence of the dispensation order on the expected read length in Section 6.4, the PAA has to incorporate information about the order of flowed nucleotides. That means, we cannot only count the steps through the underlying Markov chain, as we did in previous applications, but additionally we have to remember the position within the dispensation order. We do so by introducing weights to the automaton's edges which display the number of flows required to observe a certain transition, i.e. a certain pair of nucleotides within the described read.

Figure 6.5 shows a sketch of an appropriate PAA. We use the set $Q = \{0, 1, \ldots, \ell_d\}$ as states of the Markov chain, where 0 denotes the start state, and state i corresponds to $d[i]$. The transitions $T = (T_{qq'})_{q,q' \in Q}$ relate to the i.i.d. and the conditional nucleotide frequencies (e.g. estimated from the nucleotides and dinucleotides in a nucleotide sequence database), i.e.

$$T_{qq'} = \begin{cases} p_{d[q']} & \text{if } q = 0, q' \neq 0 \\ P_{d[q]d[q']} & \text{if } q, q' \neq 0 \\ 0 & \text{otherwise}\,. \end{cases}$$

In the matrix $S = (S_{qq'})_{q,q' \in Q}$, we store the number of flows inducing a transition from q to q':

$$S_{qq'} = (q' - q) \mod \ell_d\,.$$

Further, the state emission distributions are set to Dirac measures. In all states weight 1 is emitted with probability one. That is to say, whenever a state is visited,

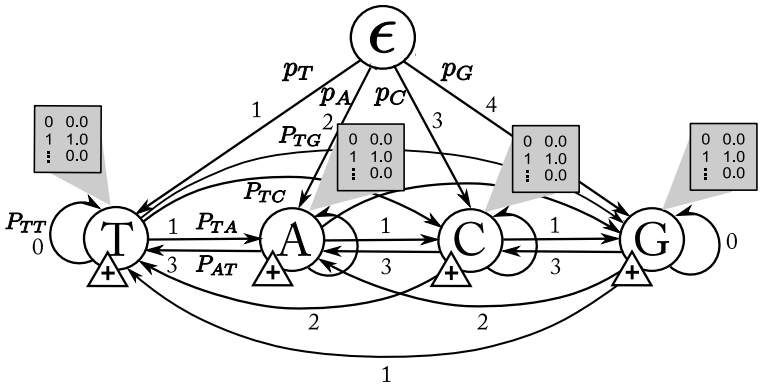


Figure 6.5: Sketch of the PAA counting the number of nucleotides incorporated by the polymerase during the 454 sequencing process when nucleotides are flowed sequentially according to d = TACG. Transitions correspond to $P_{\sigma\sigma'}$ for $\sigma, \sigma' \in \Sigma$ with initial distribution π. Integer weights indicate the number of nucleotide flows inducing the respective transition. Note that we display state i as $d[i]$ (and state 0 as ϵ) for comprehensibility.

one nucleotide is sequenced. The arithmetic operation, namely taking the sum of the emitted value and the so far achieved read length, is the same in all states. Note that homopolymer stretches are modeled by self-transitions which spend 0 steps. This implies exact proportionality of the light signal generated during pyrosequencing.

All in all, this defines a PAA to compute the distribution of 454 read lengths assuming DNA fragments to follow a Markovian string model. As in previous applications, we denote the state process of the underlying Markov chain with $(Y_n)_{n\in\mathbb{N}_0}$ and the value process of incorporated nucleotides with $(V_n)_{n\in\mathbb{N}_0}$. Further prerequisites we need are a function $g : \mathbb{N} \to Q$ that maps the number of nucleotide flows to the state defining the corresponding nucleotide within d, and a function $h : Q \times \mathbb{N} \to \mathbb{N}$ that gives the last preceding flow corresponding to nucleotide q. With the shifted modulo function given by

$$n \bmod_1 \ell_d := (n-1) \bmod \ell_d + 1 \quad \text{for } n \in \mathbb{N}\,,$$

we define

$$\begin{aligned} g(n) &:= (n-1) \bmod_1 \ell_d \quad \text{and} \\ h(q,n) &:= \max\{i : 1 \le i \le n \,|\, g(i) = q\}\,. \end{aligned}$$

Now, we compute the distribution $\mathcal{L}(L(d,t))$ of the random variable $L(d,t)$ denoting the number of sequenced nucleotides during t nucleotide flows according to the order defined by d. We set $f_{d;t}(l) = \mathbb{P}(L(d,t) = l)$ and write $f_t(l)$ when d is clear from the context. It refers to the sum of probabilities for all paths that yield a read of length l after t nucleotide flows. In the PAA framework (as presented in Chapter 2), this is given by $f_t(l) = \sum_{q\in Q} f_t(q,l)$ with joint probability $f_t(q,l) = \mathbb{P}(Y_t = q, V_t = l)$. These probabilities can be computed recursively as given in Recurrence 6.1.

Recurrence 6.1 454 Read Length Distribution

- initial condition for $q = 0$:

$$f_t(0,l) = \begin{cases} 1 & \text{if } t = 0, l = 0 \\ 0 & \text{if } l \geq 1, \\ 0 & \text{if } t \geq \ell_d, \\ 1 - \sum_{q'=1}^{t} T_{0q'} & \text{otherwise}. \end{cases} \tag{6.3.1}$$

  As long as no nucleotide has been synthesized, the Markov chain remains in the start state ϵ.

- for $q \neq 0$, $s < t$:

$$f_s(q,l) = \begin{cases} \sum_{q' \in Q} f_{s-S_{q'q}}(q', l-1) T_{q'q} & \text{if } q = g(s), \\ f_{h(q,s)}(q,l) & \text{otherwise}. \end{cases} \tag{6.3.2}$$

  The probability that after s nucleotide flows the PAA visits state q and l nucleotides have been sequenced computes as follows: when the last flowed nucleotide corresponds to q ($q = g(s)$), then there is one nucleotide sequenced corresponding to q. The recurrence relation reminds us of the general one for the joint state-value distribution as given in Equation (2.4.1). Instead of consulting the preceding step $s-1$, we account for the weights according to the particular transition $q' \to q$. Otherwise, when $q \neq g(s)$, the Markov chain has remained in q since the last corresponding nucleotide flow $h(q,s)$.

- for $q \neq 0$, $s = t$:

$$f_t(q,l) = \begin{cases} \sum_{q' \in Q} f_{t-S_{q'q}}(q', l-1) T_{q'q}(1 - T_{qq}) & \text{if } q = g(t), \\ f_{h(q,t)}(q,l)\Big(1 - \sum_{j=0}^{(g(t)-q) \bmod_1 \ell_d} T_{q\left((q+j) \bmod_1 \ell_d\right)}\Big) & \text{otherwise}. \end{cases} \tag{6.3.3}$$

  For the last flowed nucleotide, we have to account for edge effects. That is to say, correction terms specify forbidden transitions in order to obtain the exact read length distribution.

As can be seen, we make two distinctions. First, we specify whether the currently flowed nucleotide corresponds to the current state of the Markov chain. Thus, we differentiate between $q = g(s)$ and $q \neq g(s)$. The case $q \neq g(s)$ implies that the Markov chain remained in state q, which refers to the last character synthesized, and that, hence, no further transition outgoing q occurred. Second, we distinguish $s < t$ and $s = t$ in order to compute exact read lengths. On the one hand, in the t^{th} nucleotide flow, more than one nucleotide might be sequenced due to homopoly-

mers. On the other hand, if the read length is already reached before the t^{th} flow, the path only contributes if no further flown nucleotide can be incorporated. The correction terms introduced for $s = t$ account for these cases.

In this application, we have extended the conventional PAA framework in the sense that the underlying stochastic process is influenced by deterministic information. Storing the probabilities $f_t(q, l)$ in space $\mathcal{O}(t \cdot |Q|) = \mathcal{O}(t \cdot \ell_d)$, f_t can be computed in time $\mathcal{O}(t^2 \cdot |Q|^2) = \mathcal{O}(t^2 \cdot \ell_d^2)$ according to the push-strategy explained in Section 2.4.

Extension to general dispensation orders

The presented model is also valid for dispensation orders d with $\ell_d > |\Sigma|$. In this case, the transition matrix has to be adjusted properly: A transition from the start state 0 to some state $j \in Q$ happens with positive probability if and only if there is no state $i < j$ with $d[i] = d[j]$. Then, $T_{0i} = p_{d[i]}$ and $T_{0j} = 0$. Similarly, we set the transition from state i to state k to 0 if there exists another state $j \in Q$ with $d[j] = d[k]$ and $S_{ij} < S_{ik}$.

Alternative Model for Homopolymer Stretches

Instead of modeling self-transitions and using Dirac measures as weight distributions, we can delete the self-transitions and use the following state emissions (for $q \neq 0$):

$$e_q(k) = \begin{cases} 0 & \text{if } k = 0, \\ (P_{qq})^{k-1} & \text{if } k \geq 1\,. \end{cases} \tag{6.3.4}$$

In the recurrence equations we then have terms like

$$f_s(q, l) = \begin{cases} \sum_{q' \in Q} \sum_{k=1}^{l} f_{s-S_{q'q}}(q', l-k) T_{q'q} e_q(k) & \text{if } q = g(s), \\ f_{h(q,s)}(q, l) & \text{otherwise}\,. \end{cases} \tag{6.3.5}$$

The basic system including the correction terms for the case $s = t$ remain the same as given in Recurrence 6.1.

Irrespective of the representation, the presented PAA assumes that the light intensities detected during pyrosequencing are exactly proportional to the number of incorporated nucleotides. It would be interesting to further investigate how to modify our model in order to adapt to the technological settings better and account for errors in reading homopolymer stretches. To this end, states can be added to the automaton (such that the self-transitions happen at smaller rate), or the weight distributions can be altered. Therefore, one might take the intersections of normal distributions into account, following [104].

6.3.1 Proof of Concept

Before we use the presented model to derive characteristics of the expected read length subject to the dispensation order, we show the verification of the length distribution obtained comparing it to empirical data. For this purpose, the genetic department of Bielefeld University provided 454 sequence reads from two GC-rich bacteria, *R. lupinii* and *S. meliloti*. One sample was produced on the GS20 system, one sample with the GS-FLX instrument. We also have the assembled contigs available. From these, we estimated the (conditional) nucleotide frequencies which were used as model parameters.

For a first evaluation, we concentrated on the GS-FLX data and calculated the theoretical read length distribution for $t = 400$ nucleotide flows (100 cycles) by means of the recurrences (6.3.1), (6.3.2), and (6.3.3). The comparison to its empirical counterpart is shown in Figure 6.6.

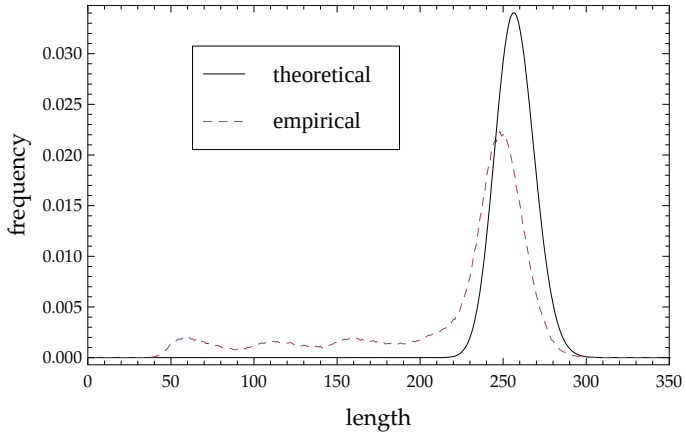


Figure 6.6: Comparison of the read length distribution calculated by the PAA to its empirical counterpart. Nucleotide frequencies as well as the empirical read lengths were counted from an actual sequencing project of a GC-rich bacterium.

One observes that the empirical distribution is shifted towards shorter reads. Even read lengths shorter than 100 nucleotides occur. Theoretically, we expect read lengths of at least 100 nucleotides since each cycle TACG should lead to at least one incorporation. This indicates the influence of the system's signal processing steps. In order to meet respective heuristic adaptations, we also implemented a naive read simulator. It first generates $100,000$ random DNA fragments with a length of 600 nucleotides according to the estimated conditional nucleotide frequencies. Then, it creates a flowgram for each fragment, where the emitted light intensity for a homopolymer of length l is modeled by a normal distribution $\mathcal{N}(l, 0.15\sqrt{l})$ following Richter *et al.* [104]. Noise signals, resulting from asynchronous incorporation at the template clones within one well, are introduced with a rate of 1.9% in agreement with Margulies *et al.* [82]. From the flowgram, the final read length can be

derived either with or without further processing.

The quantile-quantile plot in Figure 6.7 shows the accordance of simulation and theoretical length distribution. Application of the so-called TrimBackValleyFilter is reported to result in an apparent decrease in read lengths of 10% to 20%. From our analysis, we observed that the simulated reads were 11.3% shorter on average (from 257 to 228 nt) when we accounted for the TrimBack filter. The thereby shifted distribution reflects the layout of the empirical distribution, as indicated in Figure 6.8.

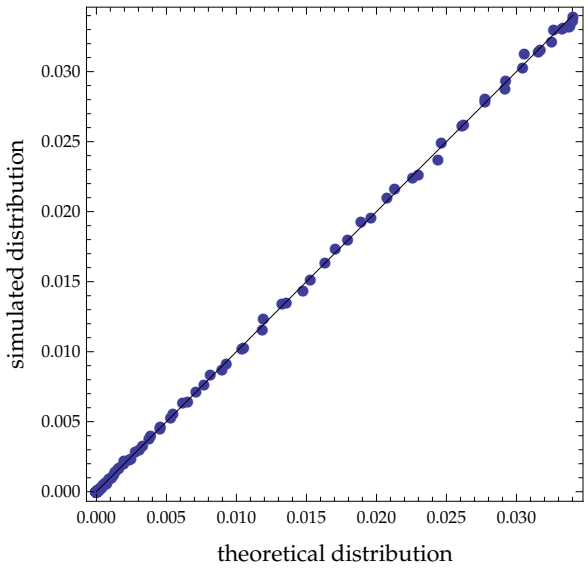


Figure 6.7: Quantile-Quantile Plot of the theoretical length distribution and the simulated values. The quantiles of the simulated distribution fit the quantiles of the theoretical distribution quite well.

6.4 Finding the Optimal Dispensation Order

As discussed before, the length of sequenced reads and the sequence coverage influence the power of customary assembly tools. At least for the dimensions related to shotgun sequencing, it is easier to assemble reads to contigs (sets of overlapping DNA segments), which are then disposed to the complete DNA sequence, the longer the reads are and the higher the sequence coverage is. Although research centers employing the up-to-date 454 FLX Titanium sequencing kits state that the 454 sequencing technology efficiently replaces the conventional Sanger sequencing, there are still endeavors for improvement. In general, the average length of produced reads depends on the dispensation order. Particularly if the composition

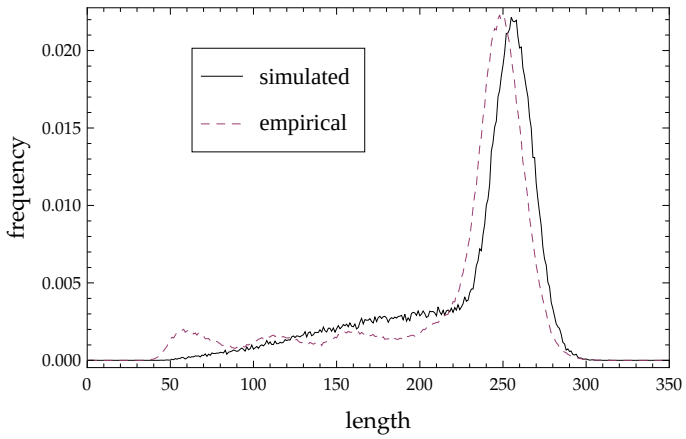


Figure 6.8: Comparison of the simulated data and the empirical read length distribution. Application of the TrimBack filter yields 11.3% shorter reads on average.

of nucleotides is heterogeneous, such as in GC-rich genomes, the average read length can be enhanced significantly.

As a starting point, we exhaustively enumerated all valid dispensation orders d of a fixed (short) length ℓ_d ($4 \leq \ell_d \leq 10$) and computed the expected read length $\mathbb{E}(L(d,t))$ after t nucleotide flows by means of the PAA presented above. The nucleotide composition for the studied cases is $p = (p_A, p_C, p_G, p_T) \approx (0.21, 0.29, 0.29, 0.21)$. In Table 6.1, we display the ranking of dispensation orders of length $\ell_d = |\Sigma| = 4$ according to the expected read length after $t = 400$ flows. Here, the standard order TACG does not perform well. In particular, the winning dispensation orders CGAT, ATCG, TCGA, and GATC yield on average about 10% longer reads.

Moreover, we observed that dispensation orders of length four provide the highest expected read length (among those lengths we considered), cf. Figure 6.9.

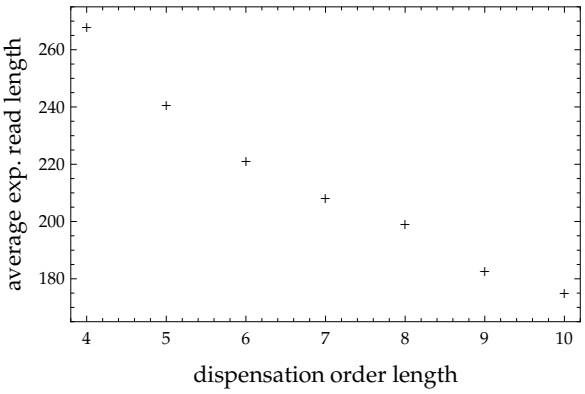


Figure 6.9: Average expected length of reads generated in 400 flows by dispensation orders of length four to ten.

Furthermore, it can be seen from Table 6.1 that always ℓ_d dispensation orders yield almost the same expected read length. These orders are circular shifts, consider e.g. the winning orders CGAT, TCGA, ATCG, and GATC. According to this observation, we suggest the following criterion in order to find the optimal dispensation order among all orders of length ℓ_d:

Definition 6.4.1 (Optimality criterion based on dinucleotide frequencies)**.** *A dispensation order $d = d[1, \ell_d]$ is said to be* optimal *among orders of length ℓ_d if and only if it maximizes*

$$\sum_{i=1}^{\ell_d} P_{\tilde{d}[i]\tilde{d}[i+1]}\,,$$

where $\tilde{d} = dd[1]$ represents the cyclic order of nucleotides.

This is to say, we seek the order d that maximizes the sum of dinucleotide frequencies within $dd[1] = d[1] \dots d[\ell_d]d[1]$.

We computed the ranking of dispensation orders of a given length ℓ_d ($4 \leq \ell_d \leq 10$) according to the mentioned criteria and compared the results to the rankings with respect to the expected read length. This showed that the optimality criterion suggested perfectly agrees with the expected read length.

A corresponding algorithm to find a dispensation order satisfying this criterion could be derived by means of an appropriate graph-theoretic approach [30, Chapter 24 and 26]. The translation into a graph-based problem reads as follows:

Problem 3. *Given a directed, weighted graph $G = (V, E)$ with vertices $V = \Sigma$ and edges (u, v) for $(u, v) \in V \times V$ weighted by the respective dinucleotide frequencies, i.e. $w(u, v) = P_{uv}$, find a Hamilton circuit of maximal weight, i.e. a directed cycle maximizing the sum of edge weights that visits each vertex exactly once.*

Note that it is reasonable to specify the length of the dispensation order since otherwise repeated orders are favored with respect to their constituting components, i.e. TACGTACG would be favored against TACG although both yield the same expected read length and the shorter order would be preferred from the experimental point of view.

Our current interest consists in the formulation of the modified task and the implementation of appropriate algorithms in order to find a dispensation order optimal across different lengths. For instance, if there is an upper bound for ℓ_d, the task would be the following:

Problem 4. *Given a directed, weighted graph $G = (V, E)$ with vertices $V = \Sigma$ and edges (u, v) for $(u, v) \in V \times V$ weighted by the respective dinucleotide frequencies, i.e. $w(u, v) = P_{uv}$, find a directed cycle of length at most ℓ_d maximizing the sum of edge weights that visits each vertex at least once.*

The solutions to the posed problems and the preparation of appropriate algorithms represent the overall aims of future research.

Discussion and Directions for future research

We have presented a PAA that yields the length distribution of 454 sequencing reads. Comparison to experimental data has shown that the computed values are reasonable, though further technological properties could be taken into account. In particular, integration of errors resulting from reading long homopolymer stretches, would improve the accordance between theoretical and empirical length distribution. Furthermore, we have shown that the average length of reads produced by 454 pyrosequencing depends on the nucleotide distribution, and the performance may be significantly better compared to the standard order TACG. Furthermore, we discussed a criterion to deduce the optimal dispensation order from the nucleotide frequencies. A corresponding algorithm based on a graph-theoretical formulation still needs to be prepared.

However, several questions remain for future research. First of all, it would be interesting to theoretically investigate what is the highest expected read length possible at all and whether there exist longer dispensation orders ($\ell_d > |\Sigma|$) that lead to a higher expected read length. If so, we have to regard that the number of valid dispensation orders of length l grows exponentially. The ultimate goal is thus to directly design the overall optimal dispensation order, generally for an arbitrary length. Another strategy could be to determine which orders can never be optimal, thereby reducing the search space.

Moreover, it is relevant to decide how heterogeneous (conditional) nucleotide frequencies have to be such that another dispensation order performs better than TACG.

Finally, in practice, one has no (explicit) knowledge of the nucleotide composition in the fragments under consideration. One question is in how many cases does the biologist have a priori information. The other question is whether it would be possible to deduce the dinucleotide frequencies from an initial number of cycles (with the standard dispensation order), proceeding with the optimal order.

Table 6.1: Ranking of dispensation orders d of length $\ell_d = |\Sigma|$ according to the expected read length $\mathbb{E}(L(d, 400))$ (rounded to integers).

rank	d	$\mathbb{E}(L(d, 400))$
1	**CGAT**	282
1	**ATCG**	282
1	**TCGA**	282
1	**GATC**	282
2	CATG	280
2	GCAT	280
2	ATGC	280
2	TGCA	280
3	CAGT	268
3	TCAG	268
3	AGTC	268
3	GTCA	268
4	ACTG	268
4	CTGA	268
4	GACT	268
4	TGAC	268
5	ACGT	257
5	CGTA	257
5	**TACG**	257
5	GTAC	257
6	CTAG	255
6	AGCT	255
6	GCTA	255
6	TAGC	255

CHAPTER 7

CONCLUSION AND OUTLOOK

In this thesis, we gave a thorough derivation of the *probabilistic arithmetic automaton* (PAA), a stochastic computational framework extending the well-known HMM. Additional to HMMs, a PAA performs binary operations on the sequence of emissions thus yielding the exact distribution of the value resulting from a sequence of operations whose operands depend on chance.

We demonstrated the framework's practicability by means of three distinct applications arising from different aspects of biological sequence analysis.

The first application concerns the computation of peptide statistics to assess the significance of protein identification by PMF. We constructed a PAA that generates random (i.i.d. or Markov) proteolytic fragments, e.g. resulting of tryptic digestion of a protein, and measures the corresponding molecular mass distribution. By means of the associated recurrences, we calculated the exact distributions of the length and the mass of proteolytic fragments. Using these, we derived the probabilities that a certain mass occurs as fragment mass after the digestion of a protein sequence of given length, which can be used to determine the significance of protein identification by PMF. We showed that the computed statistics differ only slightly under i.i.d. and Markov strings; in particular, the Markov model is capable of modeling the N-terminal methionine. Furthermore, we compared the statistics to their respective empirical counterparts estimated from the Swissprot database, obtaining a reasonable agreement. Additionally, we studied the influence of missed cleavages and post-translational modifications. We demonstrated that especially incomplete digestion affects peptide statistics and, thus, the significance of protein identification.

For the second application, we built a PAA that computes the sensitivity of seeds used to select candidate sequences from a comprehensive database that share

high sequence similarity with a given query sequence. The presented PAA is general enough to cope with different kinds of seeds and different models used to quantify the similarity of DNA sequences. In addition to former approaches, the framework yields the entire distribution of matchings between a target sequence and the query. We used this to investigate novel criteria to evaluate seeds. In particular, we contrasted two models for the similarity of DNA sequences. A good seed should select sequences that are highly similar to the query, but reject other sequences. Our analysis revealed that this combination of models yield the same results as a single model for putatively related sequences.

The third application emerges from the field of next-generation high throughput sequencing and is, hence, of great interest currently. Related technologies such as 454 sequencing provide relatively short DNA sequence reads, which provides challenges for the computational assembly. We presented a PAA that yields recurrences to compute the exact length distribution of 454 sequence reads. By comparison to experimental data, we depicted the influence of technological issues and illustrated that the theoretical distribution describes the unprocessed data reasonably. Nevertheless, it needs more insight into the technology, especially the applied filter, to get a better agreement of theoretical and empirical distributions. Moreover, the constructed PAA was used to evaluate the expected read length under different dispensation orders. We demonstrated that the optimal order depends on the nucleotide distribution and that the average read length can be enhanced significantly. To this end, we derived a criterion, correlating with the sought expectation, that can be assessed via a graph-theoretical approach.

Furthermore, motivated by the application to protein identification, we presented a generative random string model for protein sequences. The so-called SSE model takes information from secondary structure assignments into account and, hence, provides a biologically relevant model. The development gave rise to a comparison of customary random string models with respect of how well they represent protein sequences. The comparison according to a penalized model selection criterion revealed that the SSE model performs best among the considered candidate models. In particular, it outperforms the widely used i.i.d. and first-order Markov model.

Throughout the thesis, we showed that the PAA framework is an elegant and flexible model able to adapt to individual tasks. The model's structure allows to reuse algorithms connected to DFAs and Markov chain theory, and the corresponding implementations are straightforward.

Open Problems

Several directions for future research have been opened up by this thesis. As mentioned before, there are improvements and unsolved tasks related to the individual applications. For instance, in the PMF context, further investigation of the influence of missed cleavages and post-translational modifications to the peak lists alignment scores were interesting. In particular, one could take care of amino

acid propensities to design an even more plausible PAA including missed cleavage sites. Moreover, a challenging extension would be the design of a PAA accomplishing the statistics of peptide fragment fingerprints resulting from tandem MS. To this end, fragments have to be modeled as amino acid strings without internal cleavage pattern, resulting from random cleavage of a proteolytic fragment.

In the context of 454 sequence read statistics, we seek the overall optimal dispensation order. For this purpose, it would be interesting to theoretically investigate what is the highest expected read length possible at all and whether a longer dispensation order could perform better than the winning order of length four. If so, one could reduce the search space filtering out orders that can never be optimal. A partition of the parameter space indicating how heterogeneous the nucleotide distribution must be such that the expected read length can be enhanced significantly would be another meaningful result.

We discussed various ideas to improve and apply the SSE random string model. First, to circumvent the subtleties of secondary structure assignments especially for the N-terminal segment of protein sequences, one could design a hybrid model joining a customary random string model and the SSE model. For this purpose, one could use the start state to model the leading segment of the sequence. Second, to improve the influence of individual elements, one could replace the individual M1 models by more involved models taking into account dependencies within secondary structure elements that go beyond directly subsequent residues. A further perspective is to analyze the potential of family-specific SSE models. Since the order of secondary structures is more similar in proteins from a protein family, SSE information should yield a discriminative null model with respect to family representatives.
Moreover, the SSE model should be applied as a biologically relevant null model. Two potential applications are the computation of the significance of PROSITE patterns used as signatures for protein families, and the computation of alignment score statistics.

For our needs, we implemented the PAAs designed for the three applications individually. It would be convenient to have a general library capable of adapting to individual settings. Furthermore, it might be a challenging task for software engineers to make the implementations of the recurrences more efficient.
Another idea, proposed by Robert Giegerich, is to formulate the PAA as a stochastic regular grammar and implement this in an algebraic dynamic programming fashion.

On a more general level, one could investigate the expandability of the PAA framework. The definition given in this thesis comprises only time-homogeneous Markov processes. The framework could be generalized to heterogeneous transitions depending on the emissions or the current value, respectively. Moreover, we used operations associated with the automaton's states. It would be challenging to investigate applications where the operations depend on the emissions or the current value, respectively.

Apart from the mentioned applications, we indicated that there are several tasks that could be formulated in the PAA framework; both from the field of computational biology and from other areas. We also outlined various improvements and extensions related to the presented topics. We thus showed that PAAs provide a challenging area of research with a multitude of applications.

References

[1] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003.

[2] A.V. Aho and M.J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, 1975.

[3] H. Akaike. A new look at the statistical identification model. *AutomCont*, 19: 716–723, 1974.

[4] S.F. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *J. Mol. Biol*, 215:403–410, 1990.

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

[6] R. S. Annan and S. A. Carr. The essential role of mass spectrometry in characterizing protein structure: Mapping posttranslational modifications. *J. Protein Chem.*, 16(5):391–402, 1997.

[7] A.E. Ashcroft. Protein and peptide identification: The rôle of mass spectrometry in proteomics. *Nat. Prod. Rep.*, 20(2):202–215, 2003.

[8] A. Bairoch and B. Boeckmann. The SWISS-PROT protein sequence data bank. *Nucleic Acids Res*, 19 Suppl:2247–2249, 1991.

[9] M.A. Baldwin. Protein identification by mass spectrometry: Issues to be considered. *Mol. Cell Proteomics*, 3(1):1–9, 2004.

[10] M.P. Balogh. Debating resolution and mass accuracy in mass spectrometry. *Spectroscopy*, 19(10):34–40, 2004.

[11] C. Bannert and J. Stoye. Protein annotation by secondary structure based alignments (PASSTA). In *Proc. of First International Symposium on Computational Life Sciences (CompLife)*, volume 3695 of *LNBI*, pp. 79–90, 2005.

[12] A. Bateman, L. Coin, R. Durbin, R.D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M.Marshall, S. Moxon, E.L.L. Sonnhammer, D.J. Studholme, C. Yeats, and S.R. Eddy. The Pfam protein families database. *Nucleic Acids Res*, 32(Database issue):D138–D141, 2004.

[13] T. Bayes. *An Essay Toward Solving a Problem in the Doctrine of Chances*. 1763. Letter.

[14] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Res*, 28(1):235–242, 2000.

[15] S. Böcker and Z. Lipták. Efficient mass decomposition. In *Proc. of the 2005 ACM Symposium on Applied Computing (SAC)*, pp. 151–157. ACM, 2005.

[16] B. Boeckmann, A. Bairoch, R. Apweiler, M. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res*, 31(1):365–370, 2003.

[17] B. Brejová, D.G. Brown, and T. Vinar. Optimal spaced seeds for homologous coding regions. *J. Bioinform. Comput. Biol.*, 1(4):595–610, 2004.

[18] B. Brejová, D.G. Brown, and T. Vinar. Vector seeds: an extension to spaced seeds. *J. Computer System Sci.*, 70(3):364–380, 2005.

[19] D.G. Brown. Optimizing multiple seeds for protein homology search. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2(1):29–38, 2005.

[20] J. Buhler, U. Keich, and Y. Sun. Designing seeds for similarity search in genomic dna. In *Proc. of the 7th annual international conference on Research in computational molecular biology (RECOMB)*, pp. 67–75, 2003.

[21] S. Burkhardt and J. Kärkkäinen. Better filtering with gapped q-grams. In *Proc. of the 12th annual symposium on Combinatorial Pattern Matching (CPM)*, pp., 2001.

[22] K.P. Burnham and D.R. Anderson. *Model Selection and Multi-Model Inference*. Springer, 2002.

[23] K.P. Burnham and D.R. Anderson. Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research*, 33(2): 261–304.

[24] S. Böcker, M.C. Letzel, Z. Lipták, and A. Pervukhin. SIRIUS: decomposing isotope patterns for metabolite identification. *Bioinformatics*, 25(2):218–224, 2009.

[25] E. Chargaff. Structure and function of nucleic acids as cell constituents. *Fed. Proc.*, 10(3):654–659, 1951.

[26] K.P. Choi and L. Zhang. Sensitivity analysis and efficient method for identifying optimal spaced seeds. *J. Computer System Sci.*, 68:22–40, 2004.

[27] K.P. Choi, F. Zeng, and L. Zhang. Good spaced seeds for homology search. *Bioinformatics*, 20(7):1053–1059, 2004.

[28] K.R. Clauser, P. Baker, and A.L. Burlingame. Role of accurate mass measurement (+/- 10 ppm) in protein identification strategies employing MS or MS/MS and database searching. *Anal. Chem.*, 71(14):2871–2882, 1999.

[29] K.R. Coombes, S. Tsavachidis, J.S. Morris, K.A. Baggerly, M. Hung, and H.M. Kuerer. Improved peak detection and quantification of mass spectrometry data acquired from surface-enhanced laser desorption and ionization by denoising spectra with the undecimated discrete wavelet transform. *Proteomics*, 5(16):4107–4117, 2005.

[30] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition, 2001.

[31] D.M. Creasy and J.S. Cottrell. Error tolerant searching of uninterpreted tandem mass spectrometry data. *Proteomics*, 2(10):1426–1434, 2002.

[32] J.A. Cuff and G.J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, 34(4): 508–519, 1999.

[33] V. Dancik, T.A. Addona, K.R. Clauser, J.E. Vath, and P.A. Pevzner. De novo peptide sequencing via tandem mass spectrometry: A graph-theoretical approach. In *Proc. of the third annual international conference on Computational molecular biology (RECOMB)*, pp. 135–144, New York, NY, USA, 1999. ACM Press.

[34] K. Do, P. Müller, and M. Vannucci. *Bayesian Inference for Gene Expression and Proteomics*. Cambridge University Press, 2006.

[35] B. Domon and R. Aebersold. Mass spectrometry and protein analysis. *Science*, 312(5771):212–217, 2006.

[36] S. Dori and G.M. Landau. Construction of aho corasick automaton in linear time for integer alphabets. *Combinatorial Pattern Matching* 168–177, 2005.

[37] M. Droege and B. Hill. The genome sequencer flx system–longer reads, more applications, straight forward bioinformatics and more complete data sets. *J. Biotechnol.*, 136(1-2):3–10, 2008.

[38] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Oxford University Press, 1998.

[39] W. Feller. *An Introduction to Probability Theory and its Applications*. John Wiley & sons, 1968.

[40] J.B. Fenn, M. Mann, C.K. Meng, S.F. Wong, and C.M. Whitehouse. Electrospray ionization for mass spectrometry of large biomolecules. *Science*, 246 (4926):64–71, 1989.

[41] D. Frishman and P. Argos. Knowledge-based protein secondary structure assignment. *Proteins*, 23(4):566–579, 1995.

[42] Y. Gelfand, A. Rodriguez, and G. Benson. TRDB–the Tandem Repeats Database. *Nucleic Acids Res.*, 35:D80–D87, 2007.

[43] R.M. Gray. *Entropy and Information Theory*. Springer, 1 edition, 1990.

[44] G.R. Grimmett and D.R. Stirzaker. *Probability and Random Processes*. Clarendon Press, Oxford, 1992.

[45] J.H. Gross. *Mass Spectrometry*. Springer, 2004.

[46] D. Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

[47] W.J. Henzel, C. Watanabe, and J.T. Stults. Protein identification: The origins of peptide mass fingerprinting. *J. Am. Soc. Mass Spectrom.*, 14(9):931–942, 2003.

[48] I. Herms and S. Rahmann. Computing alignment seed sensitivity with probabilistic arithmetic automata. In *Proc. of the 8th Workshop on Algorithms in Bioinformatics (WABI)*, volume 5251 of *Lecture Notes in Computer Science*, pp. 318–329. Springer, 2008.

[49] F. Hillenkamp, M. Karas, R.C. Beavis, and B.T. Chait. Matrix-assisted laser desorption/ionization mass spectrometry of biopolymers. *Anal. Chem.*, 63 (24):1193A–1203A, 1991.

[50] J.E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford, CA, USA, 1971.

[51] N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, B.A. Cuche, E. de Castro, C. Lachaize, P.S. Langendijk-Genevaux, and C.J. Sigrist. The 20 years of PROSITE. *Nucleic Acids Res.*, 36(Database issue), 2008.

[52] L. Ilie and S. Ilie. Multiple spaced seeds for homology search. *Bioinformatics*, 23(22):2969–2977, 2007.

[53] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22: 2577–2637, 1983.

[54] H. Kaltenbach. *Statistics and Algorithms for Peptide Mass Fingerprinting*. PhD thesis, Bielefeld University, 2007.

[55] M. Karas and F. Hillenkamp. Laser desorption ionization of proteins with molecular masses exceeding 10,000 daltons. *Anal. Chem.*, 60:2299–2301, 1988.

[56] S. Karlin and H.M. Taylor. *A First Course in Stochastic Processes*. Academic Press, 2 edition, 1975.

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

[58] D.E. Knuth, J.H. Morris, and V.R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6(2):323–350, 1977.

[59] T. Knuutila. Re-describing an algorithm by hopcroft. *Theor. Comput. Sci.*, 250(1-2):333–363, 2001.

[60] Y. Kong. Generalized correlation functions and their applications in selection of optimal multiple spaced seeds for homology search. *J. Comput. Biol.*, 14 (2):238–254, 2007.

[61] Yong Kong. Statistical distributions of pyrosequencing. *J Comput Biol*, 16 (1):31–42, 2009.

[62] D. Kozen. *Automata and Computability*. Springer-Verlag, New York, 1997.

[63] G. Kucherov, L. Noé, and M. Roytberg. Multiseed lossless filtration. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 02(1):51–61, 2005.

[64] G. Kucherov, L. Noé, and M. Roytberg. A unifying framework for seed sensitivity and its application to subset seeds. *J. Bioinform. Comput. Biol.*, 4(2): 553–569, 2006.

[65] J. Kuha. AIC and BIC - comparisons of assumptions and performance. *SMR*, 33:188–229, 2004.

[66] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[67] E. Lange, C. Gröpl, K. Reinert, O. Kohlbacher, and A. Hildebrandt. High-accuracy peak picking of proteomics data using wavelet techniques. In *Proc. of the Pacific Symposium on Biocomputing (PSB)*, pp. 243–254, 2006.

[68] K. Lemstrom and A. Pienimaki. On comparing edit distance and geometric frameworks in content-based retrieval of symbolically encoded polyphonic music. *Musicae Scientiae*, Sp. Iss. SI:135–152, 2007.

[69] S. Levy, G. Sutton, P.C. Ng, L. Feuk, A.L. Halpern, B.P. Walenz, N. Axelrod, J. Huang, E.F. Kirkness, G. Denisov, Y. Lin, J.R. MacDonald, A.W.C. Pang, M. Shago, T.B. Stockwell, A. Tsiamouri, V. Bafna, V. Bansal, S.A. Kravitz, D.A. Busam, K.Y. Beeson, T.C. McIntosh, K.A. Remington, J.F. Abril, J. Gill, J. Borman, Y. Rogers, M.E. Frazier, S.W. Scherer, R.L. Strausberg, and J.C. Venter. The diploid genome sequence of an individual human. *PLoS Biol.*, 5 (10):e254, 2007.

[70] M. Li, B. Ma, D. Kisman, and J. Tromp. PatternHunter II: Highly sensitive and fast homology search. *J. Bioinform. Comput. Biol.*, 2(3):417–439, 2004.

[71] M. Li, B. Ma, and L. Zhang. Superiority and complexity of the spaced seeds. In *Proc. of the 17th annual ACM-SIAM symposium on discrete algorithms (SIAM)*, pp. 444–453, 2006.

[72] D.C. Liebler. *Introduction to Proteomics*. Humana Press, 2002.

[73] K.U. Linderstrøm-Lang. Proteins and enzymes. volume 6 of *Lane Medical Lectures, Stanford University Publications, Medical Sciences*, pp. 1–115. Stanford University Press, 1952.

[74] M.E. Lladser, M.D. Betterton, and R. Knight. Multiple pattern matching: a markov chain approach. *J. of Mathematical Biology*, 56(1-2):51–92, 2008.

[75] I. Coro C. Hoogland P.A. Binz R.D. Appel M. Tuloup, C. Hernandez. Aldente and BioGraph : An improved peptide mass fingerprinting protein identification environment. In *Proc. of the Swiss Proteomics Society 2003 Congress: Understanding Biological Systems through Proteomics*, pp. 174–176, 2003.

[76] B. Ma, J. Tromp, and M. Li. PatternHunter - faster and more sensitive homology search. *Bioinformatics*, 18:440–445, 2002.

[77] M. Goto M. Leman C. Rhodes M.A. Casey, R. Veltkamp and M. Slaney. Content-based music information retrieval: Current directions and future challenges. In *Proc. of the IEEE*, volume 96, pp. 668–696, 2008.

[78] J. Magnin, A. Masselot, C. Menzel, and J. Colinge. OLAV-PMF: A novel scoring scheme for high-throughput peptide mass fingerprinting. *J. Proteome Res.*, 3(1):55–60, 2004.

[79] D. Mak, Y. Gelfand, and G. Benson. Indel seeds for homology search. *Bioinformatics*, 22(14):e341–e349, 2006.

[80] D.Y.F. Mak and G. Benson. All hits all the time: Parameter free calculation of seed sensitivity. In *Proc. of the 5th asia-pacific APBC bioinformatics conference (APBC)*, volume 5 of *Advances in Bioinformatics and Computational Biology*, pp. 327–340. Imperial College Press, 2007.

[81] M. Mann and O.N. Jensen. Proteomic analysis of post-translational modifications. *Nat. Biotechnol.*, 21(3):255–261, 2003.

[82] M. Margulies, M. Egholm, W.E. Altman, S. Attiya, J.S. Bader, L.A. Bemben, J. Berka, M.S. Braverman, Y. Chen, Z. Chen, S.B. Dewell, L. Du, J.M. Fierro, X.V. Gomes, B.C. Godwin, W. He, S. Helgesen, C. Heen Ho, C. He Ho, G.P. Irzyk, S.C. Jando, M.L.I. Alenquer, T.P. Jarvie, K.B. Jirage, J. Kim, J.R.Knight, J.R. Lanza, J.H. Leamon, S.M. Lefkowitz, M. Lei, J. Li, K.L. Lohman, H. Lu, V.B. Makhijani, K.E. McDade, M.P. McKenna, E.W.Myers, E. Nickerson, J.R. Nobile, R. Plant, B.P. Puc, M.T. Ronan, G.T. Roth, G.J. Sarkis, J.F. Simons, J.W. Simpson, M.Srinivasan, K.R. Tartaro, A. Tomasz, K.A. Vogt, G.A. Volkmer, S.H. Wang, Y. Wang, M.P. Weiner, P. Yu, R.F. Begley, and J.M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, 2005.

[83] T. Marschall. Statistics on random texts. an approach based on finite automata. Diploma thesis, Faculty of Technology, Bielefeld University, Germany, 2007.

[84] T. Marschall and S. Rahmann. Probabilistic arithmetic automata and their application to pattern matching statistics. In *Proc. of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp., 2008.

[85] R. Matthiesen. *Mass Spectrometry Data Analysis in Proteomics*. Humana Press, 2007.

[86] A.M. Maxam and W. Gilbert. A new method for sequencing DNA. *Proc Natl Acad Sci U S A*, 74:560–564, 1977.

[87] L. McHugh and J.W. Arthur. Computational methods for protein identification from mass spectrometry data. *PLoS Comput. Biol.*, 4(2):e12, 2008.

[88] F. Monigatti and P. Berndt. Algorithm for accurate similarity measurements of peptide mass fingerprints and its application. *J. Am. Soc. Mass Spectrom.*, 16(1):13–21, 2005.

[89] J.S. Morris, K.R. Coombes, J. Koomen, K.A. Baggerly, and R. Kobayashi. Feature extraction and quantification for mass spectrometry in biomedical applications using the mean spectrum. *Bioinformatics*, 21(9):1764–1775, 2005.

[90] J. Myhill. Finite automata and the representation of events. Technical Report 57-624, WADC, 1957.

[91] N.E. Nahi. *Estimation Theory and Applications*. Wiley, New York, 1969.

[92] A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc*, 9: 541–544, 1958.

[93] P. Ney and E. Nummelin. Markov additive processes i: Eigenvalue properties and limit, 1987.

[94] L. Noé and G. Kucherov. Improved hit criteria for dna local alignment. *BMC Bioinformatics*, 5:149, 2004.

[95] P.M. Palagi, Pa. Hernandez, D. Walther, and R.D. Appel. Proteome informatics i: Bioinformatics tools for processing experimental data. *Proteomics*, 6 (20):5435–5444, 2006.

[96] T. Patzkill. *Proteomics*. Kluwer Academic Publishers, 2002.

[97] W. Pearson and D. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[98] D.N. Perkins, D.J. Pappin, D.M. Creasy, and J.S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567, 1999.

[99] P.A. Pevzner and M.S. Waterman. Multiple filtration and approximate pattern matching. *Algorithmica*, 13(1/2):135–154, 1995.

[100] G. Pollastri, D. Przybylski, B. Rost, and P.Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins*, 47(2):228–235, 2002.

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

[102] S. Rahmann. Subsequence combinatorics and applications to microarray production, DNA sequencing and chaining algorithms. In *Combinatorial Pattern Matching (CPM)*, volume 4009 of *LNCS*, pp. 153–164, 2006.

[103] D. Reynolds. An overview of automatic speaker recognition technology. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICA SSP)*, volume IV, pp. 4072–4075, 2002.

[104] D.C. Richter, F. Ott, A.F. Auch, R. Schmid, and D.H. Huson. MetaSim: A sequencing simulator for genomics and metagenomics. *PLoS ONE*, 3(10): e3373, 2008.

[105] S. Robin, F. Rodolphe, and S. Schbath. *DNA, Words and Models: Statistics of Exceptional Words*. Cambridge University Press, 2005.

[106] M. Ronaghi. Pyrosequencing sheds light on dna sequencing. *Genome Res.*, 11(1):3–11, 2001.

[107] M. Ronaghi, S. Karamohamed, B. Pettersson, M. Uhlén, and P. Nyrén. Real-time DNA sequencing using detection of pyrophosphate release. *Anal. Biochem.*, 242(1):84–89, 1996.

[108] M. Ronaghi, M. Uhlén, and P. Nyrén. A sequencing method based on real-time pyrophosphate. *Science*, 281(5375):363–365, 1998.

[109] F. Sanger, S. Nicklen, and A.R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, 74(12):5463–5467, 1977.

[110] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, 1948.

[111] J.A. Siepen, E. Keevil, D. Knight, and S.J. Hubbard. Prediction of missed cleavage sites in tryptic peptides aids protein identification in proteomics. *J. Proteome Res.*, 6(1):399–408, 2007.

[112] C.J. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. PROSITE: A documented database using patterns and profiles as motif descriptors. *Brief. Bioinform.*, 3(3):265–274, 2002.

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

[114] A.P. Snyder. *Interpreting Protein Mass Spectra*. Oxford University Press, 2000.

[115] H. Stark and J.W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[116] Y. Sun and J. Buhler. Designing multiple simultaneous seeds for DNA similarity search. *J. Comput. Biol.*, 12(6):847–861, 2005.

[117] B. Thiede, S. Lamer, J. Mattow, F. Siejak, C. Dimmler, T. Rudel, and P.R. Jungblut. Analysis of missed cleavage sites, tryptophan oxidation and n-terminal pyroglutamylation after in-gel tryptic digestion. *Rapid Commun. Mass Spectrom.*, 14(6):496–502, 2000.

[118] C. Tsai and R. Nussinov. The implications of higher (or lower) success in secondary structure prediction of chain fragments. *Protein Sci.*, 14(8):1943–1944, 2005.

[119] D. Tsur, S. Tanner, E. Zandi, V. Bafna, and P.A. Pevzner. Identification of post-translational modifications via blind search of mass-spectra. In *Proc. of the IEEE Computer Society Bioinformatics Conference (CSB)*, pp. 157–166, 2005.

[120] M. Unemo, P. Olcén, J. Jonasson, and H. Fredlund. Molecular typing of neisseria gonorrhoeae isolates by pyrosequencing of highly polymorphic segments of the porb gene. *J. Clin. Microbiol.*, 42(7):2926–2934, 2004.

[121] I-J. Wang, C.P. Diehl, and F.J. Pineda. A statistical model of proteolytic digestion. In *Proc. of the IEEE Computer Society Bioinformatics Conference (CSB)*, pp. 506–508, 2003.

[122] D. Williams. *Weighing the Odds: A Course in Probability and Statistics*. Cambridge University Press, 2001.

[123] J.R. Yates. Mass spectrometry. from genomics to proteomics. *Trends Genet.*, 16(1):5–8, 2000.

[124] J.R. Yates, J.K. Eng, and A.L. McCormack. Mining genomes: Correlating tandem mass spectra of modified and unmodified peptides to sequences in nucleotide databases. *Anal. Chem.*, 67(18):3202–3210, 1995.

[125] J.R. Yates, J.K. Eng, A.L. McCormack, and D. Schieltz. Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Anal. Chem.*, 67(8):1426–1436, 1995.

[126] J.R. Yates, A.L. McCormack, D. Schieltz, E. Carmack, and A. Link. Direct analysis of protein mixtures by tandem mass spectrometry. *J. Protein Chem.*, 16(5):495–497, 1997.

[127] C. Yen, S. Russell, A.M. Mendoza, K. Meyer-Arendt, S. Sun, K.J. Cios, N.G. Ahn, and K.A. Resing. Improving sensitivity in shotgun proteomics using a peptide-centric database with reduced complexity: Protease cleavage and SCX elution rules from data mining of ms/ms spectra. *Anal. Chem.*, 78(4): 1071–1084, 2006.

[128] W. Zhang and B.T. Chait. ProFound: An expert system for protein identification using mass spectrometric peptide mapping information. *Anal. Chem.*, 72(11):2482–2489, 2000.

[129] M. Zvelebil and J.O. Baum. *Understanding Bioinformatics*. Garland Science, 2008.