Universität Bielefeld

Technische Fakultät
Abteilung Informationstechnik
Forschungsberichte

Statistics of Cleavage Fragments in Random Weighted Strings

Hans-Michael Kaltenbach Henner Sudek
Sebastian Böcker Sven Rahmann

Report 2005-06

Impressum: Herausgeber:
Ellen Baake, Robert Giegerich, Ralf Hofestädt, Franz Kummert,
Peter Ladkin, Ralf Möller, Helge Ritter, Gerhard Sagerer,
Jens Stoye, Ipke Wachsmuth

Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31,
33501 Bielefeld, Germany

ISSN 0946-7831

Statistics of Cleavage Fragments in Random Weighted Strings

Hans-Michael Kaltenbach[1,3,*] Henner Sudek[1]
Sebastian Böcker[1,3] Sven Rahmann[2,3]

13.12.2005

[1] Informatics for Mass Spectrometry group, Faculty of Technology, Bielefeld University
[2] Algorithms and Statistics for Systems Biology group, Faculty of Technology, Bielefeld University, D-33594 Bielefeld
[3] Graduate School in Bioinformatics and Genome Research, Bielefeld University
[*] Address correspondence to: michael@cebitec.uni-bielefeld.de

Abstract. Peptide mass fingerprinting is an important technique that allows to identify a protein from its fragment masses obtained by mass spectrometry after enzymatic fragmentation: An experimental mass fingerprint is compared with or aligned to several reference fingerprints obtained from protein databases using in-silico digestion. Recently, much attention has been given to the questions of how to score such an alignment of mass spectra and how to evaluate its significance; results have been developed mostly from a combinatorial perspective. In particular, existing methods generally do not (or only at the price of a combinatorial explosion) capture the fact that the same amino acid can have different masses because of, e.g., isotopic distributions or variable chemical modifications.

We offer several new contributions to the field: We introduce the notions of a *probabilistically weighted alphabet*, where each character can have different masses according to a specified probability distribution, and the notion of a *random weighted string* as a fundamental model for a random protein. We then develop a general computational framework, which we call *weighted HMMs* for various length and mass statistics of cleavage fragments of random proteins. We obtain general formulas for the length distribution of a fragment, the number of fragments, the joint length-mass distribution, and for fragment mass occurrence probabilities, and special results for so-called standard cleavage schemes (e.g., for Trypsin). We also discuss how to efficiently implement the probability computations. Computational results are provided, as well as a comparison to proteins from the SwissProt database.

Contents

1 Introduction

Mass spectrometry (MS) plays a key role in today's proteomics experiments [1]. The main application of mass spectrometry is the identification of proteins either by de-novo sequencing [2] or by peptide-mass fingerprinting together with a database search [11].

In peptide-mass fingerprinting, the protein is biochemically digested into fragments using restriction enzymes. The set of masses of these fragments, the so-called peptide mass fingerprint (PMF), is measured using MS and compared to a set of reference spectra, usually obtained from in-silico digested database sequences. Different comparison tools have been developed, prominent ones being Mascot [13] based on the MOWSE scoring system [12] and ProFound [21]. One major problem in developing spectra comparison methods is to estimate the statistical significance of its results. First of all, a statistical model of the fragmentation and the resulting mass fingerprints is needed.

Presently, there are two major approaches to cope with this problem: (1) One can use a statistical model based on strongly simplifying assumptions, such as the uniform distribution of fragment masses [12], to reduce the problem complexity considerably. However, such models may not fit well to real data [18]. (2) Alternatively, one can derive the model from empirical data, e.g., from large samples of mass spectra [9] where we have to deal with the problem that mass spectra depend on the instruments used, the technical instrument settings such as voltage, laser energy and temperature, and on the biochemical preparation of the samples. Another possibility is to use in-silico digestion of whole protein sequence databases [18]. The problem here is that statistical significance values are data dependent and are thus hard to compare to other results. If, for example, sequences are added to the database, the significance values of other interpretations also change. Moreover, we are frequently only interested in the proteome of one species. Deriving an empirical model is often difficult as most species-specific protein databases are too small to get reliable statistical data for fragments and using a very large non-specific database may result in biased estimates.

In [6], a new approach was proposed to compute significance values. Here, the probability that an i.i.d. random protein sequence contains at least one fragment of certain mass was computed using a dynamic programming approach based on uniform character distributions and for fixed character masses. However, not all amino acids occur with the same frequency in natural proteins, and the same amino acid may have different masses because of isotopic distributions, or, for example, different variable post-translational modifications that may or may not be present. Therefore, more general models and computational approaches are needed.

This report provides both a general model, namely *random weighted strings* that we introduce in Section 2, and a computational framework for several different kinds of fragment statistics, namely *weighted HMMs* (wHMMs) that we define and apply in Section 3. The wHMM framework is quite general; is comprises, for example, random i.i.d. and Markovian proteins of arbitrary order, and many different cleavage rules. For our results, however, we focus on i.i.d. strings and *standard cleavage schemes*, to be defined subsequently. We derive the exact distribution of fragment lengths, the exact distribution of the number of fragments, the exact joint length-mass distribution, and

the mass distribution of fragments. From these results, we derive the probability that a given fragment mass occurs at least once in a string of given length (Section 4).

We have implemented our results and compare the i.i.d. model predictions to empirical data obtained from the SwissProt database, release 48 (September 2005, [3]). In Section 5, we discuss various methods to reduce the memory consumption as well as time efficient implementations.

Related work. Our results can be seen as generalizations of two lines of previous research.

First, our model of probabilistically weighted strings extends the concept of *weighted strings* [8], where the weights of characters are fixed and not probabilistic. Weighted strings have been used in the setting of mass spectrometry to generate peptide candidates [10, 17], to compute possible decompositions of masses into character masses [7] and to find submasses [4]. General combinatorics of weighted strings were investigated in [8].

Second, the waiting times for cleavage points between fragments (i.e., the fragment lengths) are waiting times for specific, possibly overlapping, patterns in strings. For strings without weights, the statistics of such overlapping patterns [14, 19, 15] and sets of patterns [16] have been intensively investigated in bioinformatics and statistics [20], and our results on random weighted strings naturally contain some of these as particular cases.

Notational conventions. We write $\mathcal{L}(X)$ for the distribution of a random variable (r.v.) X; the generic probability measure is denoted by $\mathbb{P}$. Distributions are represented as probability vectors, e.g., we write $x[m] = \mathbb{P}(X = m)$ for some finite range of integers m.

We write $\mathcal{L}(X) \otimes \mathcal{L}(Y)$ for the product measure of $\mathcal{L}(X)$ and $\mathcal{L}(Y)$, i.e., the distribution of the pair (X, Y) if X and Y are independent. Further, $\mathcal{L}(X) \star \mathcal{L}(Y) = \mathcal{L}(X+Y)$ denotes the convolution of the distributions of two independent r.v.s X and Y. These notations generalize to more than two r.v.s. The convolution of two vectors $x[i]$, $y[j]$ is defined as $(x \star y)[k] := \sum_i x[i] \cdot y[k-i]$, where the finite value range of k is derived from the ranges of i and j.

For a string s we denote the substring from index i to index j (inclusive) by $s_{i:j}$, and we write $s^\ell := s_{1:\ell}$.

2 The Random Weighted String Model

In the life sciences, mass spectrometry is traditionally used for proteomics experiments such as de novo sequencing or peptide identification by database search. Similar techniques are in development to measure DNA molecules and metabolites. We restrict ourselves to the modeling of protein and DNA experiments which have a common basis: In both cases, the molecules consist of an unknown sequence of well-known single molecules, amino acids and nucleotides, respectively. In bioinformatics, such molecules

are traditionally modeled as strings over fixed and finite alphabets of characters. In the context of mass spectrometry, we assign a mass to each character and refer to such strings as weighted strings. We extend the weighted string model of [8] to also capture isotopic distributions and post-translational modifications of proteins. In this model, one character may take on different masses with certain probabilities. These probabilities are known up to great precision in the case of isotopes and can be experimentally estimated in the case of post-translational modifications. Moreover, the frequency of occurrence of characters can be estimated from sequence databases or stochastic models of sequence evolution. This leads to stochastic models of weighted strings.

2.1 Weighted Alphabets and Strings

We always assume Σ to be a finite alphabet. The following two definitions restate the concept of weighted strings as used in [8].

Definition 2.1 (Weighted alphabet). Let Σ be a finite alphabet and let $\mu : \Sigma \to \mathbb{Z}$ be a function assigning each character $\sigma \in \Sigma$ its *mass* or *weight* $\mu(\sigma) :\equiv \mu_\sigma$. The pair (Σ, μ) is called a *weighted alphabet* with *character mass function* μ. We write $\mu_{\max} := \max\{\mu(\sigma) : \sigma \in \Sigma\}$ and $\mu_{\min} := \min\{\mu(\sigma) : \sigma \in \Sigma\}$ for the largest and smallest mass in (Σ, μ), respectively.

Remark. We use integer masses for several reasons: Real numbers of arbitrary precision cannot be represented in a computer by standard data types anyway, so it makes sense to restrict weights to numbers in $\mathbb{Q}$ with bounded denominator. By multiplying, these can always be represented as integers. Also, in practice, mass values are only known up to a certain accuracy. While we do not encounter them in real molecules, we do allow negative weights, e.g., to be able to allow modifications that reduce the mass of a molecule.

Definition 2.2 (Weighted string, mass sequence). In general, a *weighted string* is a semi-infinite sequence $(s_i, \mu_i)_{i\in\mathbb{N}}$ over $(\Sigma \times \mathbb{Z})^{\mathbb{N}}$. For a given weighted alphabet (Σ, μ), the sequence $(s_i, \mu(s_i))_{i\in\mathbb{N}}$ is called a *weighted string over* (Σ, μ). We sometimes call the sequence $s = (s_i)_{i\in\mathbb{N}}$ the weighted string and the sequence $(\mu_i)_{i\in\mathbb{N}}$ the *weight sequence*, *mass sequence*, or (deterministic) *mass process* of s. The character mass function is also written with subscripts, i.e., $\mu_{s_i} := \mu(s_i)$. The case of finite (weighted) strings is obtained by considering the length-ℓ prefix of (s, μ) for some $\ell \in \mathbb{N}$.

Definition 2.3 (String mass). The character mass function is extended to finite strings $s^\ell \in \Sigma^\ell$ by setting

$$\mu(s) := \sum_{i=1}^{\ell} \mu(s_i).$$

This definition provides a homomorphism from the non-Abelian semigroup $(\Sigma^*, \circ)$, where $\circ$ denotes concatenation, to the Abelian (semi-)group $(\mathbb{Z}, +)$.

Char.	A	C	D	$\cdots$	Y
Mass (Da)	71.0788	103.1388	115.0886	$\cdots$	163.1760

Table 1: Average isotopic molecular weights in Dalton of some amino acids.

Example 2.4 (Proteins). Proteins are strings over the alphabet of amino acids. There are 20 different amino acids, each having a specific one letter code and a molecular weight, usually given in Daltons (Da), with one Da approximately the mass of a proton. See Table 1 for some molecular weights of amino acids. An extensive table can be found in the appendix.

To obtain integers as character masses, we scale weights by a divisor $\Delta = 10^{-k}$, $k \in \mathbb{N}$ and round to the next integer. Using $\Delta = 0.01$, we get $\mu_A = \text{round}(71.0788/0.01) = 7108$. This discretization of masses is not a problem in practice, as first molecular weights are only known to some precision and second in the setting of mass spectrometry, masses can only be measured up to some precision. Usually, a discretization factor $\Delta = 0.1$ or $\Delta = 0.01$ is sufficient.

2.2 Probabilistically Weighted Alphabets and Strings

In order to capture isotopic distributions and mass modifications of characters, we want to allow multiple masses per character in an alphabet, where each mass is taken with certain probability. As only the mass is probabilistic and no random model for the sequences is (yet) assumed, we call such weighted alphabets *probabilistically weighted alphabets*.

Definition 2.5 (Probabilistically weighted alphabet). Let Σ be a finite alphabet, let $(\Xi, \mathbb{P})$ be an appropriately constructed probability space, and let $\mu : \Sigma \times \Xi \to \mathbb{Z}$ be a *probabilistic character mass function*, assigning to each character $\sigma \in \Sigma$ a random variable $\mu(\sigma, \cdot) = \mu_\sigma(\cdot) : \Xi \to \mathbb{Z}$, so $\mathbb{P}(\mu_\sigma = m)$ denotes the probability that the mass of character σ takes the value m. The pair (Σ, μ) is then called a *probabilistically weighted alphabet*.

Again we denote by $\mu_{\max}$, $\mu_{\min}$ the largest and smallest possible mass in (Σ, μ), with $\mu_{\max} := \max\{m \in \mathbb{Z} : \exists \sigma \in \Sigma \text{ s.t. } \mathbb{P}(\mu_\sigma = m) > 0\}$ and $\mu_{\min} := \min\{m \in \mathbb{Z} : \exists \sigma \in \Sigma \text{ s.t. } \mathbb{P}(\mu_\sigma = m) > 0\}$.

Note that it is sufficient to specify the distribution $\mathcal{L}(\mu_\sigma)$ for each $\sigma \in \Sigma$ and we do not have to explicitly specify the probability space Ξ. Also, if $\mathcal{L}(\mu_\sigma)$ is a Dirac distribution for each $\sigma \in \Sigma$ (in which case μ_σ takes on one value with probability 1), the probabilistically weighted alphabet is the same as a weighted alphabet as we can identify μ_σ with the mass m_σ for which $\mathbb{P}(\mu_\sigma = m_\sigma) = 1$.

Example 2.6 (Isotopes of amino acids). Each amino acid occurs in nature with several masses due to the isotopic masses of the atoms building the amino acid. This fact can be modeled by a probabilistically weighted alphabet, where μ_σ takes on the different isotopic masses of the amino acid σ with probability of occurrence of the corresponding isotopic composition in nature.

Example 2.7 (Post-translational modifications). We can also model so-called post-translational modifications (PTMs) of amino acids with probabilistically weighted alphabets. PTMs are modifications such as phosphor or methyl groups which are attached to certain amino acids after the amino acid sequence has been translated from the gene. PTMs can occur at every amino acid or they can be specific to certain amino acids. Thus, some amino acids of certain type may be modified, others may not be. Probabilistically weighted alphabets can be useful if the frequency of a modification is known or can be estimated for every amino acid. Note that this model can also be combined easily with the above model of isotopic distributions.

Since we would like to consider strings of arbitrary length in what follows, we develop our models from a semi-infinite string $s \in \Sigma^{\mathbb{N}}$ and then use projections to finite length-ℓ prefixes as needed. We also write $s^\ell := s_{1:\ell}$ for this prefix.

In order to define (probabilistically) weighted strings over a probabilistically weighted alphabet, we first have a look at the sequence of masses associated to a fixed sequence of characters; this is now a sequence of random variables, or a stochastic process. We assume that the reader is familiar with the basics on stochastic processes, as described, e.g., in [5]. We assume that the masses of characters at different positions are conditionally independent, given the characters.

Definition 2.8 (Mass process for fixed strings). Let (Σ, μ) be a probabilistically weighted alphabet and let $s \in \Sigma^{\mathbb{N}}$ be a fixed semi-infinite infinite string. Let the masses be chosen independently for each character. Then the *mass process* $(\mu_i)_{i\in\mathbb{N}}$ is defined as a stochastic process having index set $\mathbb{N}$ and taking values in $\mathbb{Z}$, where $\mathcal{L}(\mu_i) := \mathcal{L}(\mu_{s_i})$, and because of independence, the finite dimensional distributions of $(\mu_i)_{i\in I}$ for finite $I \subset \mathbb{N}$ are given by the products

$$\mathcal{L}(\mu_I) := \bigotimes_{i\in I} \mathcal{L}(\mu_i).$$

The double use of μ for both the probabilistic character mass functions (μ_σ) and the mass process (μ_i) of a string should not cause confusion, but rather aid intuition. This definition especially extends the mass sequence of Definition 2.2. It also contains the case of finite strings by restricting I to a subset of $\{1, \dots, \ell\}$.

The mass $\mu(s_I) \equiv \mu_{s_I}$ associated to a fixed finite string is now also a random variable, as it is the sum of the single random masses. Because of independence, its distribution can be computed as the convolution of the single distributions.

Lemma 2.9 (String mass distribution). *For finite $I := \{i_1, i_2, \dots, i_n\} \subset \mathbb{N}$, let $s_I := s_{i_1} s_{i_2} \dots s_{i_n}$. The distribution of the string* string mass *of s_I is given by*

$$\mathcal{L}(\mu(s_I)) = \mathcal{L}(\mu_{i_1}) \star \dots \star \mathcal{L}(\mu_{i_n}).$$

This is the only reasonable way to consistently extend Definition 2.3: For a (non-probabilistically) weighted alphabet, the distribution of the string mass is again a Dirac

distribution, assigning probability 1 to the sum of character masses. It provides a semigroup homomorphism $\mathcal{L}\mu(\circ_i s_i) = \star_i \mathcal{L}\mu(s_i)$ from $(\Sigma^*, \circ)$ to the space of finite-support distributions on $\mathbb{Z}$ with the convolution operation $\star$.

Example 2.10. Let $\Sigma = \{a, b\}$ and (Σ, μ) be a probabilistically weighted alphabet with character mass distributions $\mathbb{P}(\mu_a = 1) = \mathbb{P}(\mu_a = 2) = \frac{1}{2}$ and $\mathbb{P}(\mu_b = 1) = \mathbb{P}(\mu_b = 2) = \mathbb{P}(\mu_b = 3) = \frac{1}{3}$. Let $s = ab$ and $\mu_s = \mu_{ab}$ its string mass. Then the distribution of μ_s is given by

$$\mathbb{P}(\mu_s = m) = \sum_{m_1+m_2=m} \mathbb{P}(\mu_a = m_1, \mu_b = m_2) = \sum_{m' \in \mathbb{Z}} \mathbb{P}(\mu_a = m', \mu_b = m - m').$$

and we obtain:

m	0	1	2	3	4	5	6
$\mathbb{P}(\mu_s = m)$	0	0	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{1}{6}$	0

2.3 Random Weighted Strings

The above definitions did not assume a random model for a string over an alphabet. It is thus not yet possible to capture character frequencies or character dependencies within strings with these models. As a remedy, we first introduce a random string model and then combine it with weighted strings to get a model for random weighted strings.

Definition 2.11 (Random string model). A *random string* over an alphabet Σ is a stochastic process S with index set $\mathbb{N}$ taking values in Σ given by its finite dimensional distributions

$$\mathcal{L}(S_I) = \mathcal{L}(S_{i_1}, \dots, S_{i_n})$$

for all finite $I = \{i_1, \dots, i_n\} \subset \mathbb{N}$.

Example 2.12 (i.i.d. string model). The most simple model for a random string is an *i.i.d. string*. The finite dimensional distributions $\mathcal{L}(S_I)$ then reduce to the product measure $\mathcal{L}(S_I) = \bigotimes_{i \in I} \mathcal{L}(S_i)$. The probability of a string is the product of its characters' probabilities.

We are now able to give a general random string model for weighted strings over both deterministically and probabilistically weighted alphabets. In all cases, we assume that the mass of a character in a string is independent of the mass of all other characters.

Definition 2.13 (Random weighted string). A *random weighted string* is a stochastic process $(S, \mu) = ((S_1, \mu_1), (S_2, \mu_2), \dots)$ with index set $\mathbb{N}$, values in $\Sigma \times \mathbb{Z}$ and finite dimensional distributions

$$\mathcal{L}((S, \mu)_I) = \mathcal{L}(S_I) \otimes \mathcal{L}(\mu_I)$$

where S is a random string and μ is a mass process associated to S. We denote the measure of this distribution by $\mathbb{P}$.

Henceforth, we only discuss the i.i.d. model, in which we assume the characters to be independent and identically distributed. Note however, that all above definitions also capture arbitrary string models, the most prominent one being Markov sequences.

Char.	A	C	D	$\cdots$	Y
Mass (Da)	71.0788	103.1388	115.0886	$\cdots$	163.1760
Freq. (%)	7.85	1.54	5.31	$\cdots$	3.06

Table 2: Average isotopic molecular weights and SwissProt frequencies of amino acids

Example 2.14 (Random proteins). As before, we model a peptide or a protein as a string over the alphabet Σ of amino acids, and every letter in the string has a certain mass dependent only on the character itself, but independent of all other characters within the string, given by μ_σ. The character $L \in \Sigma$, say, at some given position within the sequence may therefore have a different mass as the same character L later in the sequence. A useful random peptide model would take the frequencies of amino acids from a sequence database such as SwissProt [3] as character probabilities (see Table 2). The isotopic character mass distributions can be computed from the isotopic distributions of the atoms by convolution.

If the character probabilities $\mathbb{P}(C = \sigma)$ and the distributions $\mathcal{L}(\mu_C \mid C = \sigma)$ are known for every amino acid σ, we can use the identity $\mathbb{P}(C = \sigma,\, \mu_C = m) = \mathbb{P}(\mu_C = m \mid C = \sigma) \cdot \mathbb{P}(C = \sigma)$ to obtain the joint character-mass-distributions. From these, the mass distribution of the whole string can be computed.

Lemma 2.15. *Let (S, μ) be a random weighted string over (Σ, μ), and let $S^\ell := S_{1:\ell}$ be the finite length-ℓ prefix. Assume that the character probabilities $\mathbb{P}(C = \sigma)$ and the conditional mass distributions $\mathcal{L}(\mu_\sigma)$ are known for every $\sigma \in \Sigma$. Then for $s \in \Sigma^l$ and $m \in \mathbb{Z}$,*

$$\mathbb{P}(S^\ell = s,\, \mu_{S^\ell} = m) = \mathbb{P}(S^\ell = s) \cdot [\star_{i=1}^{\ell}\, \mathcal{L}(\mu_{s_i})](m),$$

where the convolution can be computed inductively in $\mathcal{O}(\ell \cdot (\mu_{\max} - \mu_{\min})^2)$ time using ℓ times the recurrence

$$\mathbb{P}(\mu_{s_{1:\ell}} = m) = \sum_{m'} \mathbb{P}(\mu_{s_1} = m') \cdot \mathbb{P}(\mu_{s_{2:\ell}} = m - m').$$

Under the i.i.d. string model, we can additionally use that $\mathbb{P}(S^\ell) = \prod_{i=1}^{\ell} \mathbb{P}(C = s_1)$, where C denotes a generic random character.

Proof. The proof is straightforward, using independence of characters and of masses for different characters, and the fact that it is known that $S^\ell = s$. For the convolution, we condition on possible values of μ_{s_1} whose number is bounded by $\mu_{\max} - \mu_{\min}$. □

3 Fragmentation of Random Weighted Strings

To identify a biomolecule such as a protein, the mass of the biomolecule itself is of little value. In most mass spectrometry settings, the molecule is therefore cleaved into fragments using a biochemical cleavage reaction. In the case of proteins, these so-called proteases usually cleave right after the occurrence of a specific character. This reaction

can be suppressed if the cleavage character is directly followed by a prohibition character. To formalize this, we introduce the set of *cleavage characters* and the set of *prohibition characters*, which together form a *cleavage scheme*. It is assumed that cleavage takes place after the cleavage character, but some enzymes also cleave in front of the cleavage character. If the prohibition also takes place before the cleavage, all statistics are valid as we can just consider the reverse strings. Appendix C lists some common cases of restriction enzymes. The resulting family of substrings is then called a *fragmentation*.

After formalizing the model (Section 3.1), we examine the length distribution of fragments (Section 3.2), the distribution of their number (Section 3.3), and the joint length-mass distribution (Section 3.4).

Our discussion first focuses on semi-infinite strings $S \in \Sigma^{\mathbb{N}}$ to avoid complications with boundary effects; the necessary adjustments for finite strings are made subsequently. This is reflected in our notation as follows. Whenever we adjust a quantity, e.g., L_i for the length of the i-th fragment, to strings of finite length ℓ, we denote the adjusted random variable by the superscripted string length, e.g., L_i^ℓ. In this section, we write $\bar{\Gamma}$ for the complement of a set $\Gamma \subset \Sigma$ in Σ, i.e., $\bar{\Gamma} := \Sigma \setminus \Gamma$.

3.1 Fragmentation Models: Cleavage Schemes and Weighted HMMs

This report introduces the general *weighted Hidden Markov Model* (wHMM) framework to carry out computations on statistics of proteins and their fragmentations. We start by defining a cleavage scheme, which describes the activity of many peptide-cleaving enzymes, and naturally leads to a wHMM model.

Definition 3.1 (Cleavage scheme (Γ, Π), Quantities γ, π). A *cleavage scheme* is a pair (Γ, Π) of a set of *cleavage characters* $\Gamma \subset \Sigma$, and a set of *prohibition characters* $\Pi \subset \Sigma$.

If the additional constraint $\Gamma \cap \Pi = \emptyset$ (i.e., $\Gamma \subset \bar{\Pi}$) holds, we speak of a *standard cleavage scheme*.

Cleavage schemes with $\Pi = \emptyset$ are called *simple*.

Strings $C = C_1C_2 \in \Gamma \times \bar{\Pi}$ are called *cleavage patterns* as the cleavage reaction takes place within them. For simple cleavage schemes, the cleavage patterns are of length 1, as $\bar{\Pi} = \Sigma$ in this case.

We set $\gamma := \mathbb{P}(S_i \in \Gamma)$, $\pi := \mathbb{P}(S_i \in \Pi)$. For further use, we note that for standard schemes, $\mathbb{P}(S_i \in \Gamma \cap \bar{\Pi}) = \gamma$ and that $\mathbb{P}(S_i \in \bar{\Gamma} \cap \bar{\Pi}) = 1 - (\gamma + \pi)$.

The reason to introduce the special case of *standard* cleavage schemes is that many existing enzymes follow this form, and computations simplify when compared to general cleavage schemes, often allowing us to obtain closed expressions, whereas in the general case, we cannot solve the recursions explicitly.

Example 3.2 (Trypsin). For the frequently used protease Trypsin, we have a cleavage reaction after K or R, if not followed by P, thus $\Gamma = \{K, R\}$ and $\Pi = \{P\}$; this is a standard cleavage scheme. The possible cleavage patterns are of the form $C \in \{K, R\} \times (\Sigma \backslash \{P\})$. Using SwissProt frequencies, we obtain $\gamma = 0.1125$ and $\pi = 0.0483$.

Further examples of enzymes that match our definition of cleavage schemes, and also some exceptions, can be found in Appendix C.

Applying a cleavage scheme on a string results in a fragmentation of this string in consecutive, non-overlapping substrings, the fragments. We make the following definitions.

Definition 3.3 (Cleavage points). Let S be a (random or fixed) semi-infinite string over Σ. Each element of the series of indices $T(S) = (T_i(S))_{i\in\mathbb{N}}$ with $T_0(S) = 0$ and

$$T_i \equiv T_i(S) := \min\{k > T_{i-1}(S) : S_k \in \Gamma, S_{k+1} \in \bar{\Pi}\}$$

is called a *cleavage point* of S. We define $T_i(S) := +\infty$ if the minimum is taken over the empty set. We write T_i short for $T_i(S)$ if S is given from the context.

Definition 3.4 (Cleavage points and cleavage order for finite strings). For finite length prefixes S^ℓ, we define

$$T_i^\ell := \min\{T_i, \ell\},$$

so that eventually all cleavage points lie directly behind the end of the prefix. We also call

$$N^\ell := \min\{k : T_k = \ell\}$$

the *cleavage order* of S^ℓ, as it gives the number of different parts of S^ℓ separated by cleavage patterns.

Remark. For fixed semi-infinite strings, we may also set $N^\ell := \min\{k : T_k = +\infty\}$. For fixed strings, the cleavage points and cleavage order are deterministic quantities; for random strings, they are random variables.

Definition 3.5 (Fragments; fragmentation). For each $i \geq 1$, the substring $F_i := S_{T_{i-1}+1:T_i}$ is called the *i-th fragment* of S. If $T_{i-1} = T_i$, the i-th fragment and the following fragments are empty. We denote the length of fragment F_i by $L_i := T_i - T_{i-1}$. The family $\mathcal{F}_S := (F_i)_{i\geq 1}$ is called the *fragmentation* of S.

For finite strings S^ℓ, we define F_i^ℓ, L_i^ℓ, and $\mathcal{F}_S^\ell$ analogously in terms of T_i^ℓ.

Example 3.6 (Fragmentation of a string). Let $\Sigma := \{A, B, C\}$, $\Gamma := \{B\}$, $\Pi := \{A\}$ and let $s = ABBACCBACBBB$ be a fixed finite string of length $\ell = 12$. Then BB and BC are the cleavage patterns and s is fragmented into $\mathcal{F}_S^\ell = (AB, BACCBACB, B, B)$ with cleavage order $N^\ell = 4$, cleavage points $T_1^\ell = 2$, $T_2^\ell = 10$, $T_3^\ell = 11$, $T_4^\ell = 12$, and fragment lengths $L_1^\ell = 2$, $L_2^\ell = 8$, $L_3^\ell = 1$, $L_4^\ell = 1$. For $i \geq 5$, we have $T_i^\ell = 12$ and $L_i = 0$.

Remark. For semi-infinite strings, the sequence $(T_i)_{i\geq 1}$ defines a renewal sequence with delay T_1 and interrenewal sequence $(L_i)_{i\geq 2}$, since $T_i = T_{i-1} + L_i$ for $i \geq 2$. It is visualized in Figure 1. The cleavage characters not followed by a prohibition character therefore form a regenerative process. Note that the delay corresponds to the length of the first fragment and that all following fragments are i.i.d. and have the same length distribution $\mathcal{L}(L_2)$.

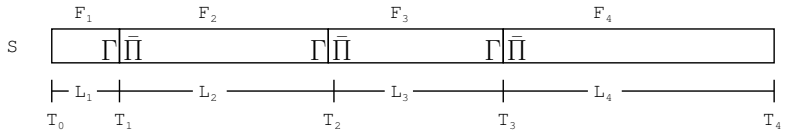

Figure 1: Fragments F_i, cleavage points T_i and fragment length L_i of a string S

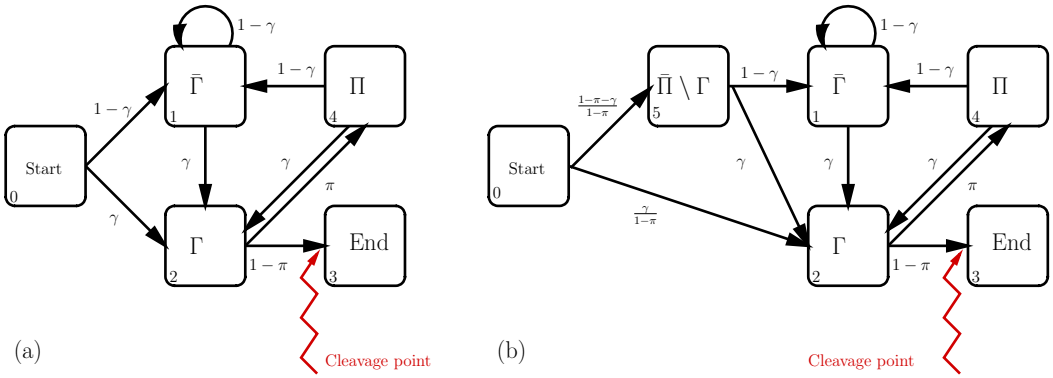

Figure 2: (a) Weighted HMM generating an initial fragment of a random i.i.d. string under a standard cleavage scheme; (b) dito, for an inner fragment. States are labeled with a number i and a character set $A \subset \Sigma$. A wHMM generates a fragment as follows: Starting in the start state numbered 0, it picks a transition according to the probability distribution of the state's outgoing edges and then emits a weighted character from the new state's character set A according to its conditional joint character-mass distribution.

For a simple cleavage scheme ($\Pi = \emptyset$), the renewal sequence can be seen as a non-delayed sequence starting with index 0. In this case, the distribution of the first and second fragment is the same, and the T_i are stopping times.

For finite strings, we have to deal with a stopped renewal sequence. Here, the length distribution of fragment i depends on the remaining string length $\ell - T_{i-1}$.

Weighted Hidden Markov Models (wHMMs). We are interested in several distributional properties of the fragmentation of an i.i.d. random weighted string under a standard cleavage scheme, e.g., the distribution $\mathcal{L}(L_i)$ of fragment length, or the joint length-mass distribution $\mathcal{L}((L_i, \mu_{F_i}))$. To derive such properties, we introduce the framework of weighted HMMs that generate protein fragment sequences together with their mass process.

Definition 3.7 (Weighted HMM). A *weighted Hidden Markov Model (wHMM)* is a 6-tuple $(N, \Sigma, P, p^0, Q, F)$ consisting of a finite set of states N, a finite output alphabet Σ, a (sub-)stochastic state transition matrix $P = (P_{ij})_{i,j \in N}$, a start distribution p^0, a family

$Q = (Q_i)_{i \in N}$ of output distributions of weighted characters $Q_i = (q_{i,(\sigma,m)})_{(\sigma,m) \in \Sigma \times \mathbb{Z}}$, and a set of final states $F \subset N$.

The semantics are as follows: The start state i is picked according to distribution p^0. A transition to a new state, being in state i, is made according to the probability distribution in row i of P. For states $i \in F$, these distributions are defective (i.e., they sum to zero and not to one); the wHMM halts in these states. The sequence of states taken through the wHMM thus forms a *Markov chain* with transition matrix P. When state i is entered after a transition, a random weighted character (C, μ) is output according to the joint character-mass distribution Q_i. We assume that the output characters of the final states do not belong to the fragment sequence (i.e., cleavage occurs before entering F), but we do allow masses to be added in these states to model certain chemical groups at the end of a fragment.

The wHMMs in Figure 2 have a special property, which we do not exploit, but nevertheless note for future reference; they are unambiguous.

Definition 3.8 (Unambiguity). A wHMM is *unambiguous* if for any fixed finite string, there is only one sequence of states that generates this string as output.

Lemma 3.9 (Characterization of unambiguity). *A wHMM is unambiguous if and only if the following property holds for every state i: Fix i, let $J := \{j : P_{ij} > 0\}$ and $\Sigma_j := \{\sigma : \sum_m Q_{j,(\sigma,m)} > 0\}$ for $j \in J$. Then for all $j \in J$, we have $|\Sigma_j| \leq 1$.*

Proof. Necessity is obvious; sufficiency by induction on the length of the string, omitted. □

Example 3.10. The wHMMs in Figure 2 are unambiguous, since from each state, outgoing edges go only into states with disjoint character sets.

As we show in Section 3.4, it is straightforward to construct fragmentation wHMMs for the i.i.d. string model and standard cleavage scheme. However, the framework of wHMMs is much more general: We can construct wHMM models for more complicated cleavage rules, or for Markovian string models, still using the same computational framework, which we present in subsequent sections.

Combinatorial fragment description. While wHMMs provide a general framework, we can make subsequent computations in the special case of standard cleavage schemes more efficient by considering the following more combinatorial description of cleavage fragments: All fragments $F_i \in \mathcal{F}_S$ consist of an inner part with common structure completed by different prefix parts for the first fragment and following fragments. For finite strings, the last character in the last fragment is also of importance.

The inner part of a fragment consists of everything but the last and the first one or two characters of the fragment. For this part, we can by definition of a fragment guarantee that there is no cleavage character not followed by a prohibition character because otherwise the fragment would have been completed before. The fragment then consists of this inner part followed by a cleavage character and preceded by a non-prohibition

character in the case of inner fragments and preceded by an arbitrary character in the case of the first fragment.

Lemma 3.11 (Structure of Fragments). *1. The common inner part of any fragment is an element of the set*

$$\mathcal{I}^l_{\Gamma\Pi} \equiv \mathcal{I}^l := \{s \in \Sigma^l : s_l \notin \Gamma \text{ and for all } i < l : s_i \notin \Gamma \text{ or } (s_i \in \Gamma, s_{i+1} \in \Pi)\}$$

for some inner part length l.

2. The first fragment in the case of a semi-infinite string is described by

$$F_1 \in \mathcal{I}^{L_1-1} \times \Gamma.$$

In the case of a finite string we have

$$\begin{aligned} F_1^\ell &\in \mathcal{I}^{L^1-1} \times \Gamma \qquad \text{if } L_1 < \ell, \\ F_1^\ell &\in \mathcal{I}^{\ell-1} \times \Sigma \qquad \text{if } L_1 \geq \ell. \end{aligned}$$

3. The following fragments must not start with a prohibition character, so we have for $i \geq 2$, noting that $\Gamma \subset \bar{\Pi}$,

$$F_i \in \left[(\bar{\Pi} \setminus \Gamma) \times \mathcal{I}^{L_i-2} \times \Gamma\right] \cup \left[\Gamma \times \Pi \times \mathcal{I}^{L_i-3} \times \Gamma\right].$$

The same characterization holds for F_i^ℓ if $T_i < \ell$. Special care has to be taken for $T_i \geq \ell$ (i.e., $T_i^\ell = \ell$):

$$F_i^\ell \in \left[(\bar{\Pi} \setminus \Gamma) \times \mathcal{I}^{L_i^\ell-2} \times \Sigma\right] \cup \left[\Gamma \times \Pi \times \mathcal{I}^{L_i^\ell-3} \times \Sigma\right].$$

Proof. From the definition of fragments, we know that every cleavage character within a fragment must immediately be followed by a prohibition character, otherwise the fragment would have been finished before.

The first fragment may begin with any character. If the first fragment equals the whole string, it is irrelevant whether the string would have been cut after the last character of the string or not. If the first fragments ends somewhere within the string, its last character must necessarily be a cleavage character.

The same argument is true for all subsequent fragments. Here, we also know that their first character can not be a prohibition character as otherwise the previous fragment would not have been completed. Nevertheless, $\Gamma \subseteq \bar{\Pi}$ and so we have to distinguish whether the fragment immediately starts with a cleavage character or not. □

Connection to wHMM. The description of the internal part of fragments can also easily be seen from the wHMM model in Figure 2: The set $\mathcal{I}^l$ corresponds to all strings of length l that are generated by the sub-graph with states 1,2,4 which is equivalent in both wHMMs. Taking into account the remaining states to get the full fragment yields exactly the combinatorial fragment characterizations.

The previous lemma allows us to split computations concerning fragments into computations on the inner part, which is the same for all fragments and adjusting the prefixes and suffixes. Computation on the inner parts is simple, as the set $\mathcal{I}^l$ can be partitioned into two subsets using only $\mathcal{I}^{l-2}$ and $\mathcal{I}^{l-1}$ which allows a recursive approach.

Lemma 3.12 (Recursive structure of $\mathcal{I}^l$). *The inner parts of fragments $\mathcal{I}^l$ can be recursively written as*

$$\mathcal{I}^l = \left(\Gamma \times \Pi \times \mathcal{I}^{l-2}\right) \cup \left(\bar{\Gamma} \times \mathcal{I}^{l-1}\right),$$

where $\mathcal{I}^1 = \bar{\Gamma}$ and $\mathcal{I}^l = \emptyset$ for $l \leq 0$.

Proof. First note that $\Pi \subseteq \bar{\Gamma}$ and so $\mathcal{I}^2 = (\Gamma \times \Pi) \cup (\bar{\Gamma} \times \bar{\Gamma})$ is indeed correct. Now for $l > 2$, we can decompose $\mathcal{I}^l$ into two disjoint sets $\mathcal{I}^l = \mathcal{I}^l_\Gamma \cup \mathcal{I}^l_{\bar{\Gamma}}$ by assuming $s_1 \in \Gamma$ and $s_1 \notin \Gamma$, respectively:

$$\mathcal{I}^l_\Gamma = \mathcal{I}^l \cap (\Gamma \times \Sigma^{l-1}), \qquad \mathcal{I}^l_{\bar{\Gamma}} = \mathcal{I}^l \cap (\bar{\Gamma} \times \Sigma^{l-1}).$$

As $\Gamma \cap \Pi = \emptyset$, we get

$$\mathcal{I}^l_\Gamma = \{s \in \Gamma \times \Pi \times \Sigma^{l-2} : s_l \notin \Gamma \text{ and for all } 2 < i < l : (s_i \in \Gamma,\, s_{i+1} \in \Pi) \text{ or } s_i \notin \Gamma\},$$

and

$$\mathcal{I}^l_{\bar{\Gamma}} = \{s \in \bar{\Gamma} \times \Sigma^{l-1} : s_l \notin \Gamma \text{ and for all } 1 < i < l : (s_i \in \Gamma, s_{i+1} \in \Pi) \text{ or } s_i \notin \Gamma\},$$

which concludes the proof. □

In the case of simple cleavage schemes, the fragment structure is less complex, and all fragments have the same structure.

Corollary 3.13 (Fragment structure for simple schemes). *For simple cleavage schemes, the structure of the inner part of fragments reduces to*

$$\mathcal{I}^l = \bar{\Gamma}^l,$$

and for semi-infinite strings we get the fragment structure

$$F_i \in \bar{\Gamma}^{L_i - 1} \times \Gamma.$$

For finite strings,

$$\begin{aligned} F_i^\ell &\in \bar{\Gamma}^{L_i - 1} \times \Gamma \qquad \text{if } T_i < \ell, \\ F_i^\ell &\in \bar{\Gamma}^{L_i^\ell - 1} \times \Sigma \qquad \text{if } T_i = \ell. \end{aligned}$$

3.2 Fragment Length Distribution

Let (S, μ) be a semi-infinite i.i.d. random weighted string and let F_i be its i-th fragment of length L_i under cleavage scheme (Γ, Π). Since the first fragment has a different prefix than the following ones, but all following ones are i.i.d., we define

$$\begin{aligned} u_1[l] &:= \mathbb{P}(L_1 = l), \\ u_+[l] &:= \mathbb{P}(L_i = l) \text{ for any } i \geq 2. \end{aligned}$$

The case of finite strings is covered later in this section.

We compute the length distributions by following paths through the wHMMs in Figure 2. Since we are not interested in the weighted characters in the different states at the moment, we can merge states which have the same set of outgoing edges, reducing the wHMMs to 3 states only, and obtain closed formulas for the length distribution.

Theorem 3.14 (Fragment Length distribution). *Given a wHMM $(N, \Sigma, P, p^0, Q, F)$ for either an initial or a subsequent fragment, we have*

$$u_\circ[l] = \sum_{i \in F} [p^0 \cdot P^{l+1}]_i,$$

where $\circ \in \{1, +\}$ depending on whether an initial or subsequent fragment is considered.

Proof. Let p_i^l denote the probability of being in state i after l steps, and define the row vector $p^l := (p_i^l)_{i \in N}$. Then classical Markov chain theory (the Chapman-Kolmogorov equation) states that $p^l = p^0 \cdot P^l$. To achieve fragment length exactly l, we need to be in a final state $i \in F$ after $l + 1$ steps, which leads to the stated formula. □

For the models in Figure 2, we can obtain closed formulas for $u_\circ[l]$.

Lemma 3.15 (Closed Formula for u_1).

$$\begin{aligned} u_1[l] &= \frac{\gamma(1-\pi)}{\alpha}(\lambda_1^l - \lambda_2^l), \quad \textit{where} \\ \alpha &= \sqrt{(1-\gamma)^2 + 4\gamma\pi}, \\ \lambda_1 &= (1 - \gamma + \alpha)/2, \\ \lambda_2 &= (1 - \gamma - \alpha)/2. \end{aligned}$$

Proof. Consider the wHMM in Figure 2a. From the point of view of outgoing transitions, states 0, 1, and 4 are equivalent, so we merge them into state 1, thus obtaining a Markov chain with the following transition matrix $A = (A_{ij})$, where A_{ij} is the conditional probability of moving to state j, being in state i:

$$A = \begin{pmatrix} 1-\gamma & \gamma & 0 \\ \pi & 0 & 1-\pi \\ 0 & 0 & 0 \end{pmatrix}.$$

Let p_i^l denote the probability of being in state i after l steps, and let $p^l := (p_i^l)_{i=1,2,3}$. Then because of the start state now being state 1, $p^0 = (1,0,0)$, and $p^{n+1} = p^l \cdot A$, so $p^l = p^0 \cdot A^n$. Since cleavage occurs *before* entering state 3, $u_1[l] = p_3^{l+1} = (p^0 \cdot A^{l+1})_3 = A_{1,3}^{l+1}$. We obtain an explicit representation of the powers of A by diagonalization: $A = B\Lambda B^{-1}$ with an invertible matrix B and a diagonal matrix Λ so $A^l = (B\Lambda B^{-1})^l = B\Lambda^l B^{-1}$. Using the quantities α, λ_1, λ_2 as stated in the lemma, it is straightforward to verify that

$$A = \begin{pmatrix} \frac{\lambda_1}{\pi} & \frac{\lambda_2}{\pi} & \frac{-(1-\pi)}{\pi} \\ 1 & 1 & \frac{(1-\gamma)(1-\pi)}{\pi\gamma} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{\pi}{\alpha} & \frac{-\lambda_2}{\alpha} & \frac{(1-\pi)\lambda_2^2}{\gamma\pi\alpha} \\ \frac{-\pi}{\alpha} & \frac{\lambda_1}{\alpha} & \frac{-(1-\pi)\lambda_1^2}{\gamma\pi\alpha} \\ 0 & 0 & 1 \end{pmatrix}.$$

The interested reader will find the following relationships helpful: $\lambda_1 + \lambda_2 = 1 - \gamma$, $\lambda_1 - \lambda_2 = \alpha$, $\lambda_1\lambda_2 = -\gamma\pi$, and $\lambda_i^2 = (1-\gamma)\lambda_i + \gamma\pi$ for $i = 1, 2$. From this, we obtain

$$A^l = \frac{1}{\alpha} \cdot \begin{pmatrix} \lambda_1^{l+1} - \lambda_2^{l+1} & \gamma(\lambda_1^l - \lambda_2^l) & (1-\pi)\gamma(\lambda_1^{l-1} - \lambda_2^{l-1}) \\ \pi(\lambda_1^l - \lambda_2^l) & \gamma\pi(\lambda_1^{l-1} - \lambda_2^{l-1}) & (1-\pi)\gamma\pi(\lambda_1^{l-2} - \lambda_2^{l-2}) \\ 0 & 0 & 0 \end{pmatrix},$$

completing the proof. □

Lemma 3.16 (Closed Formula for u_+). *Using the same notation as in the previous lemma,*

$$u_+[l] = \frac{\gamma}{\alpha} \cdot \left[(1-\pi-\gamma)(\lambda_1^{l-1} - \lambda_2^{l-1}) + \gamma\pi(\lambda_1^{l-2} - \lambda_2^{l-2}) \right].$$

Proof. The proof is similar to the previous one, considering the wHMM in Figure 2b, merging states 1, 4, and 5 into state 1, and removing state 0 by noting that $p^1 = ((1-\pi-\gamma)/(1-\pi), \gamma/(1-\pi), 0)$. So

$$\begin{aligned} u_+[l] &= (p^1 \cdot A^l)_3 = (1-\pi-\gamma)/(1-\pi) \cdot A_{1,3}^{l-1} + \gamma/(1-\pi) \cdot A_{2,3}^{l-1} \\ &= \frac{\gamma}{\alpha} \cdot \left[(1-\pi-\gamma)(\lambda_1^{l-1} - \lambda_2^{l-1}) + \gamma\pi(\lambda_1^{l-2} - \lambda_2^{l-2}) \right], \end{aligned}$$

giving the stated result. □

Finite strings. We now give the necessary adjustments for finite strings and make the following definitions.

$$\begin{aligned} u_1^\ell[l] &:= \mathbb{P}(L_1^\ell = l), \\ u_+^\ell[l] &:= \mathbb{P}(L_i^{\ell+k} = l \mid L_{i-1} = k) \text{ for any } i \geq 2 \text{ and any } k \in \mathbb{N}. \end{aligned}$$

The second definition is in fact independent of i and k, and simply defines the conditional distribution of L_i given that there are ℓ characters left in the string.

Lemma 3.17. *For $\circ \in \{1, +\}$, we have $u_\circ^\ell[l] = u_\circ[l]$ if $l < \ell$, and*

$$u_\circ^\ell[\ell] = \sum_{l'=\ell}^{\infty} u_\circ[l'] = 1 - \sum_{l'=1}^{\ell-1} u_\circ[l'].$$

Proof. If $l < \ell$, the boundary condition is irrelevant and we have $u_\circ^\ell[l] = u_\circ[l]$. For $l = \ell$, we have $u_1^\ell[\ell] = \mathbb{P}(L_1^\ell = \ell) = \mathbb{P}(L_1 \geq \ell) = \sum_{l'=\ell}^{\infty} u_1[l'] = 1 - \mathbb{P}(L_1 < \ell) = 1 - \sum_{l'=1}^{\ell-1} u_1[l']$, and a similar argument for u_+^ℓ. □

Lemma 3.18 (Exact values for u_1^ℓ and u_+^ℓ). *Using the same notation as in Lemma 3.15,*

$$\begin{aligned} u_1^\ell[\ell] &= \frac{\gamma(1-\pi)}{\alpha} \cdot \left(\frac{\lambda_1^\ell}{1-\lambda_1} - \frac{\lambda_2^\ell}{1-\lambda_2} \right), \\ u_+^\ell[\ell] &= 1 - \frac{\gamma}{\alpha} \cdot \left[(1-\pi-\gamma) \cdot \left(\frac{\lambda_1^{\ell-1}}{1-\lambda_1} - \frac{\lambda_2^{\ell-1}}{1-\lambda_2} \right) + \gamma\pi \cdot \left(\frac{\lambda_1^{\ell-2}}{1-\lambda_1} - \frac{\lambda_2^{\ell-2}}{1-\lambda_2} \right) \right]. \end{aligned}$$

Proof. The proof is straightforward by combining Lemmas 3.15 and 3.16 with Lemma 3.17, and computing the geometric series. □

Corollary 3.19 (Simple cleavage schemes). *For simple cleavage schemes, where $\Pi = \emptyset$,*

$$\begin{aligned} u_1[l] = u_+[l] &= (1-\gamma)^{l-1} \cdot \gamma, \\ u_1^\ell[\ell] = u_+^\ell[\ell] &= (1-\gamma)^\ell, \end{aligned}$$

i.e., we obtain a (truncated) geometric distribution.

Combinatorial derivation. We can also derive recursions from Lemmas 3.11 and 3.12, first obtaining the inner parts of fragments.

Lemma 3.20 (Structure of inner parts). *Let $u'[l] := \mathbb{P}(X \in \mathcal{I}^l)$ for $X \in \Sigma^l$ be the probability that a random i.i.d. string of length l is an inner part of a fragment. This probability can be computed recursively by*

$$u'[l] = \gamma\pi \cdot u'[l-2] + (1-\gamma) \cdot u'[l-1]$$

with the starting conditions $u'[0] = 1$ and $u'[1] = 1 - \gamma$. It is understood that $u'[l] = 0$ whenever $l < 0$.

Proof. The lemma follows as a direct consequence of Lemma 3.12 and the i.i.d. assumption on S. □

Using the length distribution of internal parts, we can derive the length distribution of complete fragments by using the appropriate prefix/suffix corrections.

Lemma 3.21 (Computations). *The length distribution of complete fragments is given by*

$$\begin{aligned} u_1[l] &= u'[l-1] \cdot \gamma(1-\pi) \\ u_+[l] &= \frac{(1-\gamma-\pi)}{1-\pi} \cdot u'[l-2] \cdot \gamma(1-\pi) + \frac{\gamma\pi}{1-\pi} \cdot u'[l-3] \cdot \gamma(1-\pi) \\ \\ u_1^\ell[\ell] &= u'[\ell] + u'[\ell-1] \cdot \gamma \\ u_+^\ell[\ell] &= \frac{(1-\gamma-\pi)}{1-\pi} \cdot u'[\ell-2] + \frac{\gamma\pi}{1-\pi} \cdot u'[l-3] \end{aligned}$$

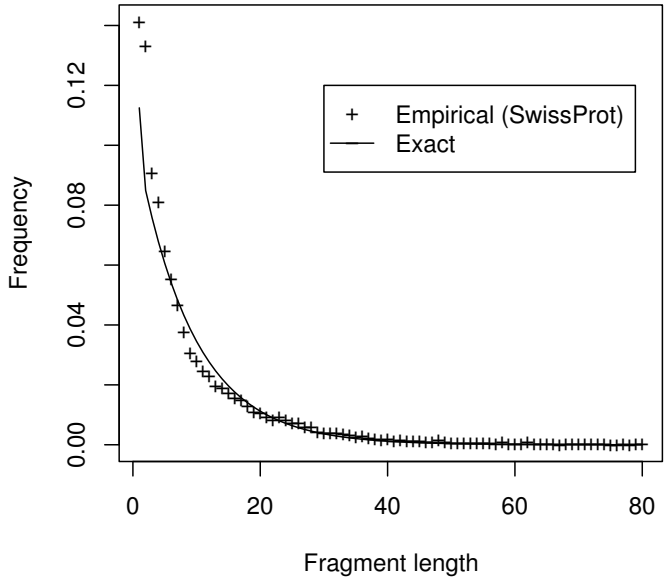

Figure 3: Exact distribution $u_1[l]$ (line) and its empirical counterpart of the SwissProt database (crosses) using Trypsin digestion.

Proof. Simply by multiplying the appropriate prefix and suffix probabilities to the $u'[l]$ formulas. Note that we have to take conditional probabilities for the $u_+[l]$, $u_+^\ell[l]$ tables because we already know that the first characters cannot be a prohibition character simply by the fact the fragment starts there and has a predecessor fragment. □

Lemma 3.22 (Recurrence for u_+). *The length $u_+[l]$ of the second and following fragments in a random weighted string can recursively be computed as*

$$u_+[l] \;=\; (1-\gamma)\cdot u_+[l-1] \;+\; \gamma\pi \cdot u_+[l-2]$$

with $u_+[1] = \gamma$.

Proof. Let $\delta := (1-\gamma-\pi)$. Then

$$u_+[l] \;=\; \delta\gamma \cdot u'[l-2] \;+\; \gamma^2\pi u'[l-3].$$

Using the recurrence equation for $u'[l]$ of Lemma 3.20 for both occurrences of u', we get

$$u_+[l] \;=\; \delta\gamma\left(\gamma\pi \cdot u'[l-4] \;+\; (1-\gamma)\cdot u'[l-3]\right) \;+\; \gamma^2\pi\left(\gamma\pi \cdot u'[l-5] \;+\; (1-\gamma)\cdot u'[l-4]\right)$$

Rearranging terms yields

$$u_+[l] \;=\; (1-\gamma)\cdot\big(\delta\gamma u'[l-3] \;+\; \gamma^2\pi u'[l-4]\big) \;+\; \gamma\pi\cdot\big(\delta\gamma u'[l-4] \;+\; \gamma^2\pi u'[l-5]\big)$$

which we identify as the stated result. □

Connection to wHMM. These combinatorial formulas exactly restate the formulas found by the wHMM approach. By comparing coefficients, we can identify

$$u'[l] \equiv \frac{\lambda_1^{l+1} - \lambda_2^{l+1}}{\alpha}$$

from which the rest follows immediately.

Results Using the Trypsin cleavage scheme, we computed the length distribution $\mathcal{L}(L_1)$ of the first fragment and compared it to the empirical distribution derived from the SwissProt database. As can be seen in Figure 3, the model is in very good agreement to the data. The length distributions of further fragments show similar behavior and are not shown here.

3.3 Number of Fragments

From the distributions for the fragment length we are now able to give the exact distribution for N^ℓ, the number of fragments in a string of length ℓ. As we have already seen, the cleavage points form a renewal sequence on the semi-infinite random weighted string S; therefore results from renewal theory apply. Establishing a connection between $\mathcal{L}(N^\ell)$, which is a quantity of the finite string and the cleavage point distributions $\mathcal{L}(T_k)$ allows us to use these results to get the exact distribution of the cleavage points and the number of fragments.

Lemma 3.23 (Relationship of N^ℓ and T_k). *The cleavage order N^ℓ of a random string of length ℓ is related to the location of cleavage points by*

$$\mathbb{P}(N^\ell \le k) = \mathbb{P}(T_k \ge \ell).$$

Proof. If the k-th cleave point T_k lies at ℓ or beyond, the number of fragments up to position ℓ is at most k, and vice versa. □

To compute $\mathcal{L}(T_k)$ we make use of the renewal equation, in particular the fact that $T_k = \sum_{i=1}^k L_i$ and that the L_i are i.i.d.

Lemma 3.24 (Distribution of cleavage points). *The distribution of cleavage points is given by*

$$\mathcal{L}(T_k) = \mathcal{L}(L_1) \star \mathcal{L}(L_2)^{\star(k-1)} = \mathcal{L}(T_{k-1}) \star \mathcal{L}(L_k).$$

Therefore

$$\mathbb{P}(N^\ell \le k) = \mathbb{P}(T_k \ge l) = \sum_{i=\ell}^{\infty} \mathbb{P}(T_k = i) = 1 - \sum_{i=0}^{l-1} \mathbb{P}(T_k = i).$$

Note that this formula also restates the earlier result $\mathcal{L}(T_1) = \mathcal{L}(L_1)$.

Proof. As already mentioned, we are using the i.i.d. property of the L_i and the fact that $T_k = \sum_{i=1}^{k} L_i = T_{k-1} + L_k$, which also holds for $k = 1$, since $T_0 = 0$ by definition. Recall that the distribution of L_1 is given by the vector $u_1[\cdot]$ and the distribution of L_i for $i \geq 2$ is given by $u_+[\cdot]$ in Section 3.2. □

Results Again, we compared our exact distribution to the empirical SwissProt distribution. Figure 4 shows this comparison together with the often used normal approximation of the cleavage order. Our exact distribution slightly underestimates the tail probabilities but nevertheless agrees much better to the empirical data than the normal approximation. This is also reflected in the comparison of the first two moments: Expected value and standard deviation for the exact distribution are 22.0 ± 6.07, and for the empirical distribution 25.1 ± 7.86.

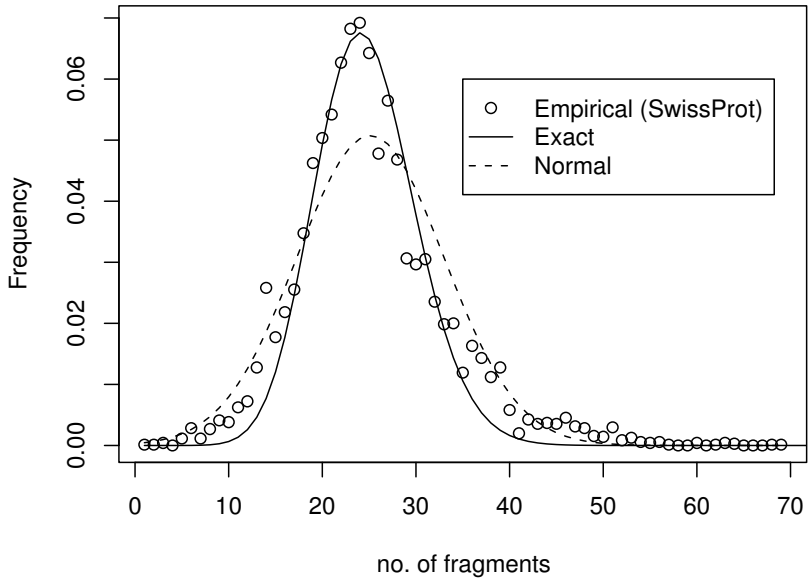

Figure 4: Pooled number of fragments in a protein of length $\ell \in \{200 \ldots 215\}$ with a total of 7050 proteins, using SwissProt frequencies and Trypsin digestion. Points: Empirical distribution derived from SwissProt, Solid line: exact theoretical distribution, Dotted line: normal approximation, both for $\ell = 207$

3.4 Length-Mass Distributions

We now examine the joint distribution of length and mass of fragments of semi-infinite random weighted i.i.d. strings (S, μ) under a standard cleavage scheme (Γ, Π). As in previous sections, the required adjustments for finite strings will be made subsequently.

Let us define

$$\begin{aligned} f_1[l, m] &:= \mathbb{P}(L_1 = l, \mu_{F_1} = m), \\ f_+[l, m] &:= \mathbb{P}(L_i = l, \mu_{F_i} = m) \text{ for any } i \geq 2. \end{aligned}$$

The definition of f_+ is independent of i because all fragments of S except the first are i.i.d.

The distribution can be computed efficiently using the wHMM framework in Figure 2. We first note that, for each state i of the wHMM, we obtain the mass probability distribution function g_i of the character set associated to that state as a mixture of the characters' mass distributions, as stated in the following lemma.

Lemma 3.25 (Mass added in state i). *Let C denote a random character from Σ according to the specified i.i.d. string model. Let A_i be the character set associated to state i in a wHMM as in Figure 2, and let $g_i[m]$ be the probability that a character generated in state i has mass m. Then*

$$g_i[m] = \frac{1}{\mathbb{P}(C \in A_i)} \cdot \sum_{c \in A_i} \mathbb{P}(C = c) \cdot \mathbb{P}(\mu_c = m).$$

In other words, the mass distribution in state i is a mixture of the $|A_i|$ mass distributions $\mathcal{L}(\mu_c)$ with mixture coefficients $\mathbb{P}(C = c)/\mathbb{P}(C \in A_i)$ for $c \in A_i$.

Proof. Given that we are in state i, the probability of generating mass m is the conditional marginal

$$\begin{aligned} g_i[m] &= \mathbb{P}(\mu_C = m \mid C \in A_i) \\ &= \sum_{c \in \Sigma} \mathbb{P}(C = c, \mu_c = m \mid C \in A_i) \\ &= \sum_{c \in \Sigma} \mathbb{P}(C = c \mid C \in A_i) \mathbb{P}(\mu_c = m) \\ &= \sum_{c \in A_i} \mathbb{P}(C = c) / \mathbb{P}(C \in A_i) \mathbb{P}(\mu_c = m), \end{aligned}$$

as claimed. □

Remark. For the wHMM in Figure 2, we have a Dirac distribution at mass zero for the start state 0 and state 3, because the start state does not contribute any mass and cleavage occurs before state 3.

However, if we want to model additional chemical groups that are attached before and after each fragment, we can use these states to model arbitrary mass distributions for these groups.

Theorem 3.26 (Computation of f_1). *Consider the wHMM $(N, \Sigma, P, p^0, Q, F)$ in Figure 2a with state set $N = \{0, \dots, 4\}$, $n := |N| = 5$ and transition matrix P.*

We write the $g_i := (g_i[m])_m$ as column vectors of the same length (padded by zeros if necessary) and set $G := (g_1| \dots |g_n)$.

Let $h_i^l[m]$ be the probability that, after l steps, we are in state i and the cumulative mass of the fragment generated so far is m. Let us set $h_i^l := (h_i^l[m])_m$ as column vectors and defining a matrix $H^l := (h_1^l| \dots |h_n^l)$.

We initially ($l = 0$) have

$$\begin{aligned} h_0^0 &= g_0, \\ h_i^0 &= 0 \quad \text{for any state } i \neq 0, \end{aligned}$$

and for step $l \geq 1$,

$$H^l = (H^{l-1} \cdot P) \star G,$$

where we define convolution in a column-by-column manner (note that the convolution refers to general vectors, not probability distributions):

$$X \star G \equiv (x_1| \dots |x_n) \star (g_1| \dots |g_n) := (x_1 \star g_1| \dots |x_n \star g_n).$$

Using P from Figure 2, this means in detail that

$$\begin{aligned} h_0^l &= 0, \\ h_1^l &= (1-\gamma) \cdot (h_0^{l-1} + h_1^{l-1} + h_4^{l-1}) \star g_1, \\ h_2^l &= \gamma \cdot (h_0^{l-1} + h_1^{l-1} + h_4^{l-1}) \star g_2, \\ h_3^l &= (1-\pi) \cdot h_2^{l-1} \star g_3, \\ h_4^l &= \pi \cdot h_2^{l-1} \star g_4. \end{aligned}$$

Finally,

$$f_1[l, m] = h_3^{l+1}[m].$$

Proof. The initial conditions are obvious (cf. also the above remark). To compute $h_i^l[m]$ for $l \geq 1$, consider the possible states k in step $l-1$, their transition probabilities P_{ki} to state i and the possible masses m' accumulated in step $l-1$. We obtain

$$\begin{aligned} h_i^l[m] &= \sum_{m'} \sum_k \mathbb{P}(\text{in state } k \text{ after } l-1 \text{ steps}, \text{transition to } i, \text{mass added is } m') \\ &= \sum_{m'} \left(\sum_k P_{ki} \cdot h_k^{l-1}[m - m'] \right) \cdot g_i[m'] = [(H^{l-1} \cdot P) \star G]_{m,i}; \end{aligned}$$

thus $H^l = (H^{l-1} \cdot P) \star G$ as claimed. □

Theorem 3.27 (Computation of f_+). *Using the same notation as in the previous lemma, but for the wHMM in Figure 2b with states $N = \{0, \ldots, 5\}$, we have in detail*

$$\begin{aligned} h_0^0 &= g_0, \\ h_i^0 &= 0 \quad \text{for } i \neq 0, \end{aligned}$$

and for step $l \geq 1$,

$$h_0^l = 0, \tag{1}$$
$$h_1^l = (1-\gamma) \cdot (h_1^{l-1} + h_4^{l-1} + h_5^{l-1}) \star g_1, \tag{2}$$
$$h_2^l = \gamma \cdot [1/(1-\pi) \cdot h_0^{l-1} + h_1^{l-1} + h_4^{l-1} + h_5^{l-1}] \star g_2, \tag{3}$$
$$h_3^l = (1-\pi) \cdot h_2^{l-1} \star g_3, \tag{4}$$
$$h_4^l = \pi \cdot h_2^{l-1} \star g_4. \tag{5}$$
$$h_5^l = (1-\pi-\gamma)/(1-\pi) \cdot h_0^{l-1} \star g_5. \tag{6}$$

Finally,

$$f_+[l, m] = h_3^{l+1}[m].$$

Proof. Similar to the proof of the previous lemma. □

Remark. Note that we can remove states 0 and 5 from the wHMM (they are used at most once) by specifying a more complicated initial condition after step 2 in this case. However, the wHMM construction is general and can be applied to more complicated string and cleavage models.

Finite Strings. We use the following notation (cf. Section 3.2):

$$\begin{aligned} f_1^\ell[l, m] &:= \mathbb{P}(L_1^\ell = l, \mu_{F_1^\ell} = m), \\ f_+^\ell[l, m] &:= \mathbb{P}(L_i^{\ell+k} = l, \mu_{F_i^{\ell+k}} = m \mid T_{i-1} = k) \text{ for any } i \geq 2 \text{ and any } k \in \mathbb{N}. \end{aligned}$$

The second definition is in fact independent of $i \geq 2$ and k, and defines the conditional joint distribution of (L_i, μ_{F_i}) given that there are ℓ characters left in the string.

Lemma 3.28 (Computation of f^ℓ). *For $\circ \in \{1, +\}$ and $l < \ell$, we have $f_\circ^\ell[l, m] = f_\circ[l, m]$ for all masses m, and for length ℓ, using the notation of Lemma 3.26,*

$$f_\circ^\ell[\ell, m] = \sum_{i \notin F} h_i^\ell[m].$$

Proof. For $l < \ell$, there is no difference to the semi-infinite string. The fragment ends after position ℓ irrespective of the current state; therefore summing $h_i^\ell[m]$ over all non-final states i leads to the desired marginal. □

Remark (Length distribution as marginal). For $\circ \in \{1, +\}$, we obtain the length distribution $u_\circ$ as a marginal of $f_\circ$:

$$u_\circ[l] = \sum_{m \in \mathbb{Z}} f_\circ[l, m].$$

To conclude this derivation, we state the obvious fact that we can now also compute the probability that a fragment has length l and *not* mass m.

Definition 3.29 (Mass avoidance probabilities). For $\circ \in \{1, +\}$, define

$$\begin{aligned} \bar{f}_1[l, m] &:= \mathbb{P}(L_1 = l, \mu_{F_1} \neq m), \\ \bar{f}_+[l, m] &:= \mathbb{P}(L_i = l, \mu_{F_i} \neq m) \text{ for any } i \geq 2, \end{aligned}$$

and similarly define $\bar{f}^\ell_\circ$, $\circ \in \{1, +\}$, for fragments whose length is bounded by ℓ.

Lemma 3.30. *For $\circ \in \{1, +\}$,*

$$\bar{f}_\circ[l, m] = u_\circ[l] - f_\circ[l, m],$$

and similarly for $\bar{f}^\ell_\circ[\ell, m]$.

Proof. We have $\bar{f}_\circ[l, m] = \sum_{m' \neq m} f_\circ[l, m'] = \left(\sum_{m' \in \mathbb{Z}} f_\circ[l, m']\right) - f_\circ[l, m] = u_\circ[l] - f_\circ[l, m]$. □

Combinatorial derivation. While the wHMM framework is easily generalizable, we can again give a more combinatorial derivation for the special case of standard cleavage schemes that is somewhat more efficient. It is based on the structural Lemma 3.12.

As before, we start by examining the distribution for the inner part of fragments. The resulting recurrence is mainly the same as the one for the length distribution $u'[l]$, the major difference being the explicit summation over the character and character mass distributions.

Lemma 3.31 (Joint length-mass distribution for inner parts). *Let $\mathcal{I}^l$ be defined as above and let (S^l, μ) be the length-l prefix of a random weighted string. We define*

$$f'[l, m] := \mathbb{P}(S^l \in \mathcal{I}^l, \mu_{S^l} = m).$$

These quantities can be efficiently computed in time $\mathcal{O}(|\Sigma||\Pi|(\mu_{\max} - \mu_{\min}))$ using the initial conditions $f'[0, 0] = 1$ and the recurrence relation

$$\begin{aligned} f'[l, m] &= \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \sum_{m' \in \mathbb{N}} f'[l - 2, m - m'] \cdot \mathbb{P}(\sigma\sigma') \cdot \mathbb{P}(\mu_{\sigma\sigma'} = m') \\ &\quad + \sum_{\sigma \notin \Gamma} \sum_{m' \in \mathbb{N}} f'[l - 1, m - m'] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m'), \end{aligned}$$

where $f'[l, m] = 0$ whenever $l < 0$ or $m < 0$.

We see that $f'[l,m]$ can be written as a sum of two convolutions of the first resp. first two characters of S^l and the remaining suffixes. The convolution is carried out over the mass only.

Proof. In this proof, we combine the results of Lemmas 3.12 and 2.15. We fix l and write X instead of S^l for brevity.

$$\begin{aligned}
& \mathbb{P}(X \in \mathcal{I}^l, \mu_X = m) \\
= \; & \mathbb{P}(X \in \mathcal{I}^l, \mu_X = m, X_1 \in \Gamma, X_2 \in \Pi) \; + \; \mathbb{P}(X \in \mathcal{I}^l, \mu_X = m, X_1 \notin \Gamma) \\
= \; & \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \mathbb{P}(X \in \mathcal{I}^l, \mu_X = m, X_1 = \sigma, X_2 = \sigma') \\
& + \sum_{\sigma \notin \Gamma} \mathbb{P}(X \in \mathcal{I}^l, \mu_X = m, X_1 = \sigma) \\
= \; & \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \mathbb{P}(X_{3:l} \in \mathcal{I}^{l-2}, \, \mu_{X_{3:l}} + \mu_{X_{1:2}} = m, \, X_1 X_2 = \sigma\sigma') \\
& + \sum_{\sigma \notin \Gamma} \mathbb{P}(X_{2:l} \in \mathcal{I}^{l-1}, \, \mu_{X_{2:l}} + \mu_{X_1} = m, \, X_1 = \sigma)
\end{aligned}$$

Conditioning on X_1, X_2 now yields

$$\begin{aligned}
= \; & \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \mathbb{P}(X_{3:l} \in \mathcal{I}^{l-2}, \mu_{X_{3:l}} + \mu_{X_{1:2}} = m \,|\, X_1 X_2 = \sigma\sigma') \cdot \mathbb{P}(X_1 X_2 = \sigma\sigma') \\
& + \sum_{\sigma \notin \Gamma} \mathbb{P}(X_{2:l} \in \mathcal{I}^{l-1}, \mu_{X_{2:l}} + \mu_{X_1} = m \,|\, X_1 = \sigma) \cdot \mathbb{P}(X_1 = \sigma) \\
= \; & \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \mathbb{P}(X_{3:l} \in \mathcal{I}^{l-2}, \mu_{X_{3:l}} = m - \mu_{\sigma\sigma'}) \cdot \mathbb{P}(X_1 X_2 = \sigma\sigma') \\
& + \sum_{\sigma \notin \Gamma} \mathbb{P}(X_{2:l} \in \mathcal{I}^{l-1}, \mu_{X_{2:l}} = m - \mu_\sigma) \cdot \mathbb{P}(X_1 = \sigma) \\
= \; & \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} f'[l-2, m - \mu_{\sigma\sigma'}] \cdot \mathbb{P}(X_1 X_2 = \sigma\sigma') \\
& + \sum_{\sigma \notin \Gamma} f'[l-1, \mu_\sigma] \cdot \mathbb{P}(X_1 = \sigma)
\end{aligned}$$

The time complexity of the first sum is bounded by $|\Gamma||\Pi|$ and of the second sum by $|\bar{\Gamma}|$. Both are bounded by $|\Sigma||\Pi|$. Summing over all possible values of μ_σ and $\mu_\sigma + \mu_{\sigma'}$, respectively in the sums gives another factor of $(\mu_{\max} - \mu_{\min})$ and concludes the proof. □

Extending Lemma 3.31 to whole fragments using Lemma 3.11, we finally get the length-mass distributions of fragments of S. As before, we first look at the first fragment.

Lemma 3.32 (Joint length-mass distribution of the first fragment). *Consider $f_1[l,m] = \mathbb{P}(L_1 = l, \mu_{F_1} = m)$, the joint length-mass distribution of the first fragment. It*

can be computed recursively in time $\mathcal{O}(|\Gamma|(\mu_{\max} - \mu_{\min}))$ *from* $f'[l, m]$ *by adjusting the suffix:*

$$f_1[l, m] = \sum_{\sigma \in \Gamma} \sum_{m' \in \mathbb{N}} f'[l - 1, m - m'] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m').$$

For finite strings, we get

$$f_1^\ell[\ell, m] \;=\; f'[|S|, m] \;+ \sum_{\sigma \in \Gamma} \sum_{m' \in \mathbb{N}} f'[\ell - 1, m - m'] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m').$$

Proof. The proof is similar to that of Lemma 3.31 and is therefore omitted. As before, the summation over the masses is in fact a finite sum. □

The distributions for the following fragments are again i.i.d. for infinite strings S and have to be conditioned on the remaining length of S for finite strings.

Lemma 3.33 (Joint length-mass distribution for following fragment). *The distribution for the fragments* $f_+[l, m] = \mathbb{P}(L_i = l, \mu_{F_i} = m)$ *for any* $i \geq 2$ *can now be given using* $f'[l, m]$ *and adjusting the prefix and suffix part for the fragment. Let* $i \geq 2$ *with* $l < |S| - T_{i-1}$*. Then* $f_+[l, m]$ *can recursively be computed from* $f'[l, m]$ *in* $\mathcal{O}(|\bar{\Gamma} \cap \bar{\Pi}||\Gamma||\Pi|(\mu_{\max} - \mu_{\min}))$*:*

$$\begin{aligned} &f_+[l, m] = \\ &\sum_{\sigma \in \bar{\Gamma} \cap \bar{\Pi}} \sum_{\sigma' \in \Gamma} \sum_{m' \in \mathbb{N}} f'[l - 2, m - m'] \cdot \mathbb{P}(\sigma\sigma') \cdot \mathbb{P}(\mu_{\sigma\sigma'} = m') \\ &+ \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \sum_{\sigma'' \in \Gamma} \sum_{m' \in \mathbb{N}} f'[l - 3, m - m'] \cdot \mathbb{P}(\sigma\sigma'\sigma'') \cdot \mathbb{P}(\mu_{\sigma\sigma'\sigma''} = m') \end{aligned}$$

For finite strings,

$$\begin{aligned} &f_i^\ell[\ell, m] = \\ &\sum_{\sigma \in \bar{\Gamma} \cap \bar{\Pi}} \sum_{m' \in \mathbb{N}} f'\left[|S| - T_{i-1} - 1, m - m'\right] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m') \\ &+ \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \sum_{m' \in \mathbb{N}} f'\left[|S| - T_{i-1} - 2, m - m'\right] \cdot \mathbb{P}(\sigma\sigma') \cdot \mathbb{P}(\mu_{\sigma\sigma'} = m'). \end{aligned}$$

Proof. The proof is similar to that of Lemma 3.31 and is therefore omitted. □

Corollary 3.34 (Joint length-mass distribution for simple schemes). *For simple cleavage schemes,*

$$f'[l, m] = \sum_{\sigma \notin \Gamma} f'[l - 1, m - \mu_\sigma] \cdot \mathbb{P}(C = \sigma),$$

i.e., $f'[l, m]$ *is just the* l*-time convolution of the mass distribution of non-cleavage characters. Further,*

$$\begin{aligned} f_1[l, m] = f_+[l, m] \;&=\; \sum_{\sigma \in \Gamma} f'[l - 1, m - \mu_\sigma] \cdot \mathbb{P}(C = \sigma), \\ f_1^\ell[\ell, m] = f_+^\ell[\ell, m] \;&=\; f'[l - 1, m] + \sum_{\sigma \in \Gamma} f'[l - 1, m - \mu_\sigma] \cdot \mathbb{P}(C = \sigma). \end{aligned}$$

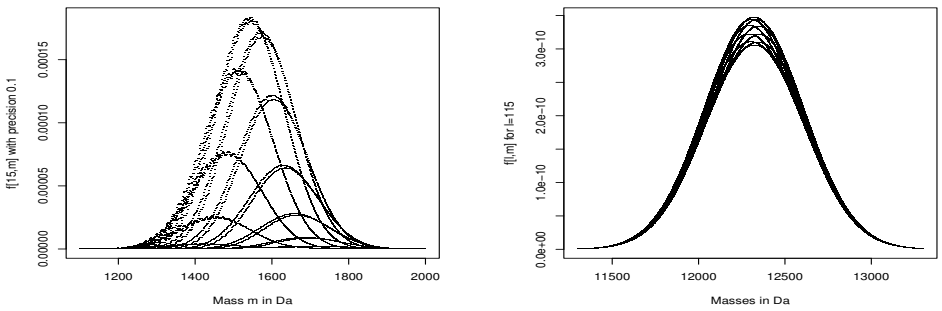

Figure 5: $f[l, m]$ for $l = 15$ (left) and $l = 115$ (right), precision $\Delta = 0.1$, tryptic digestion.

Results. Using the SwissProt database amino acid frequencies and the cleavage scheme of Trypsin, we obtain the first fragment's joint length-mass distribution f_1 shown in Figure 5 for length $l = 15$ (left) and $l = 115$ (right). We would like to stress that in each figure, only one graph is shown. It can be seen that combinatorial effects cannot be neglected and a normal approximation seems not to be accurate even for greater fragment lengths. The distribution for the following fragments looks similar and is not shown here.

3.5 Fragment Mass Distribution

In Section 3.4, we derived the joint length-mass distributions for fragments and re-derived their length distribution by taking the marginal. In the same way we can derive the distribution of fragment masses regardless of their length by taking the other marginal. Let us denote these marginal distributions by re-using f in a straightforward way:

$$\begin{aligned} f_1[m] &:= \mathbb{P}(\mu_{F_1} = m), \\ f_+[m] &:= \mathbb{P}(\mu_{F_i} = m) \text{ for any } i \geq 2 \end{aligned}$$

Proposition 3.35 (Fragment mass distribution). *The distribution of fragment masses is given by*

$$f_\circ[m] = \sum_{l \in \mathbb{N}} f_\circ[l, m].$$

Note that although the sum runs over the whole set of natural numbers, all but finitely many entries of $f_\circ[l, m]$ will be zero. See Proposition 5.4 for m-dependent bounds $B^-(m)$ and $B^+(m)$ on l such that $f_\circ[l, m] = 0$ for all $l < B^-(m)$ and all $l > B^+(m)$.

4 Occurrence of Masses in Random Weighted Strings

One of the goals of this report is to develop a data independent method to compute the significance of a protein identification by peptide mass fingerprints. To compute the significance, it is often necessary to know how likely it is to see a certain mass in a

fingerprint. This results in the question of how likely it is that a weighted string (S^ℓ, μ) of given length ℓ has a fragmentation $\mathcal{F}_S^\ell$ under a cleavage scheme (Γ, Π) that contains at least one fragment of some mass m.

We define the number of fragments of mass m in a finite random weighted string ℓ as

$$N^\ell(m) := \Big|\{F \in \mathcal{F}_S^\ell : \mu_F = m\}\Big|$$

Clearly, $0 \leq N^\ell(m) \leq N^\ell$ and $\sum_{m\in\mathbb{Z}} N^\ell(m) = N^\ell$.

We call the probability

$$p[\ell, m] := \mathbb{P}(N^\ell(m) \geq 1)$$

of a string S^ℓ having at least one fragment of mass m the *mass occurrence probability* of mass m.

Computing $p[\ell, m]$ can be done using the probability of the complementary event $\{N^\ell(m) = 0\}$ that no fragment of mass m occurs in the fragmentation of S^ℓ. We can express the occurrence probability in terms of the fragment mass distributions by

$$\bar{p}[\ell, m] := \mathbb{P}(N^\ell(m) = 0) = \mathbb{P}(\mu(F_1^\ell) \neq m, \ldots, \mu(F_{N^\ell}^\ell) \neq m)$$

We also define $\bar{p}_+[l, m]$ as the probability that a string of length l that looks like a suffix of S^ℓ and structurally starts with a "following" fragment, does not have a fragment of mass m. $\bar{p}_+$ is different from $\bar{p}$ as the fragment length-mass distributions for the first and the "following" fragments differ.

Lemma 4.1 (Mass occurrence). *Let (S^ℓ, μ_S) be a finite random weighted string of length ℓ cleaved with the cleavage scheme (Γ, Π) and let again $\mathcal{F}_S^\ell$ be the resulting fragmentation. The probability that S^ℓ does not contain a fragment of mass m can then be computed as a convolution over string length using the recurrence*

$$\bar{p}[\ell, m] \;=\; \sum_{l=1}^{\ell} \bar{p}_+[\ell - l, m] \cdot \bar{f}_1^\ell[l, m]$$

This recurrence using $\bar{f}_1^\ell[l, m]$ can be written in terms of $\bar{f}'[l, m]$ by

$$\bar{p}[\ell, m] \;=\; \bar{f}'[\ell, m] + \sum_{l=1}^{\ell} \bar{p}_+[\ell - l, m] \sum_{\sigma\in\Gamma} \sum_{m'\in\mathbb{N}} f'[l-1, m-m'] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m')$$

Proof. The main observation for the proof is that although the fragment masses are not independent, as we deal with finite string length, the mass of the first fragment becomes independent of the remaining fragments' masses if L_1 is known. Note that for $L_1 = 0$,

the product is zero.

$$\begin{aligned}
& \mathbb{P}(\mu(F_1^\ell) \neq m, \ldots, \mu(F_{N^\ell}^\ell) \neq m) \\
= & \sum_{l=1}^{\ell} \mathbb{P}(\mu(F_1^\ell) \neq m, \ldots, \mu(F_{N^\ell}^\ell) \neq m, L_1 = l) \\
= & \sum_{l=1}^{\ell} \mathbb{P}(\mu(F_1^\ell) \neq m, \ldots, \mu(F_{N^\ell}^\ell) \neq m \,|\, L_1 = l) \cdot \mathbb{P}(L_1 = l) \\
= & \sum_{l=1}^{\ell} \mathbb{P}(\mu(F_2^\ell) \neq m, \ldots, \mu(F_{N^\ell}^\ell) \neq m \,|\, L_1 = l) \cdot \mathbb{P}(L_1 = l, \mu(F_1^\ell \neq m)) \\
= & \sum_{l=1}^{\ell} \bar{p}_+[\ell - l, m] \cdot \bar{f}_1^\ell[l, m]
\end{aligned}$$

The second formulation using only $\bar{f}'[l, m]$ follows directly from Lemma 3.32. □

To compute the mass occurrence probability in suffices, the same kind of argument applies. Care has to be taken to consider the boundary effects in the $\bar{f}'[l, m]$ formulation.

Lemma 4.2 (Mass occurrence in suffices). *Let (S^ℓ, μ_S) be a finite random weighted string of length ℓ cleaved with the cleavage scheme (Γ, Π) and let again $\mathcal{F}_S^\ell$ be the resulting fragmentation. As the length-mass distribution for fragments F_k^ℓ, $k \geq 2$, have the same distribution, we can just write L for the length of any suffix starting from $T_k + 1$, $k \geq 2$.*

$$\bar{p}_+[L, m] \;=\; \sum_{l=1}^{L} \bar{p}_+[L - l, m] \cdot \bar{f}_+^L[l, m]$$

This recurrence using $\bar{f}_+[l, m]$ is equivalent to the following recurrence using only $\bar{f}'[l, m]$. (Summation over the character mass distributions will be omitted for better readability.)

$$\begin{aligned}
\bar{p}_+[L, m] \;=\; & \sum_{\sigma \in \bar{\Pi} \cap \bar{\Gamma}} \bar{f}'[L - 1, m - \mu_\sigma] \cdot \mathbb{P}(\sigma) && (7) \\
& + \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \bar{f}'[L - 2, m - \mu_{\sigma\sigma'}] \cdot \mathbb{P}(\sigma\sigma') && (8) \\
& + \sum_{l=2}^{L} \bar{p}_+[L - l, m] \sum_{\sigma \in \bar{\Pi} \cap \bar{\Gamma}} \sum_{\sigma' \in \Gamma} \bar{f}'[l - 2, m - \mu_{\sigma\sigma'}] \cdot \mathbb{P}(\sigma\sigma') && (9) \\
& + \sum_{l=2}^{L} \bar{p}_+[L - l, m] \sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \sum_{\sigma'' \in \Gamma} \bar{f}'[l - 3, m - \mu_{\sigma\sigma'\sigma''}] \cdot \mathbb{P}(\sigma\sigma'\sigma'') && (10) \\
& + \bar{p}_+[L - 1, m] \sum_{\sigma \in \Gamma} \mathbb{1}_{\{\mu_\sigma \neq m\}} \cdot \mathbb{P}(\sigma) && (11)
\end{aligned}$$

Proof. The proof for the first part is mainly the same as for Lemma 4.1 and is ommited.

For the formulation in terms of $\bar{f}'[l, m]$, the same argument holds, as we can just replace $\bar{f}_+^L[l, m]$ by it with appropriate correction terms which results in the terms (9) and (10). Care has to be taken for the case that the suffix has only one fragment, namely itself. For this case, the fragment does not have to end on a cleavage character but it might. Ending on a cleavage character it covered by (11) whereas (7) and (8) imply that the fragment does not end on a cleavage character. Then, we again have to distinguish wether the fragment starts with a cleavage character or not to guarantee that it is not ended before reaching length L. □

As before, we get a much simpler recurrence if we only allow simple cleavage schemes.

Lemma 4.3 (Mass occurrence with simple schemes). *For simple cleavage schemes, we have* $p[\ell, m] = p_+[\ell, m]$ *and the recurrence simplifies to*

$$\begin{aligned}\bar{p}[\ell, m] &= \sum_{\sigma \notin \Gamma} \sum_{m' \in \mathbb{Z}} \bar{f}'[\ell - 1, m - m'] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m') \\ &\quad + \sum_{l=1}^{\ell} \bar{p}[\ell - l, m] \sum_{\sigma \in \Gamma} \sum_{m' \in \mathbb{Z}} \bar{f}'[l - 1, m - m'] \cdot \mathbb{P}(\sigma) \cdot \mathbb{P}(\mu_\sigma = m')\end{aligned}$$

Proof. Using Lemma 4.1 and using the fact that $\Pi = \emptyset$ immediately gives the stated result. □

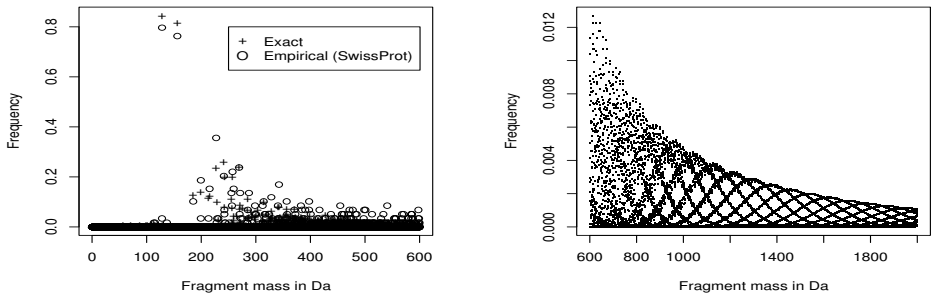

Figure 6: $p[\ell, m]$ for $\ell = 300$, precision $\Delta = 0.1$, tryptic digestion.

Results Comparing the occurrence probability for Trypsin digestion with the empirical SwissProt counterpart again shows a reasonable agreement as seen in Figure 6 (left). The combinatorial effects encountered in the fragment length-mass distributions are still present, as Figure 6 (right) shows. It is again a plot of *one* function. The two most prominent probabilities of about ≈ 0.8 are given to the masses of the cleavage characters K and R. High probabilities (> 0.2) are also given to certain two-character fragments in the range of $100 \ldots 300$ Daltons.

5 Efficient implementation

For practical purposes, we would like to have the distributions and probabilities of the previous sections available in memory to allow a constant time access. This is even more important if we want to read each value several times. In this section, we address these issues and demonstrate how the combinatorial derivations can be used to develop efficient dynamic programming algorithms to compute the entities in question.

We encounter two major problems: First, if we compute the occurrence probabilities $p[\ell, m]$ naively, we need to keep two tables, namely $\bar{f}[l, m]$ and $\bar{p}[\ell, m]$ in memory, both of size $\ell_{\max} \cdot m_{\max}$. Second, such a naive implementation will take up to several hours to compute these tables for realistic problem sizes.

Example 5.1 (Memory consumption). As a first demonstration on how large these tables can actually get, we consider a typical peptide mass fingerprint setting using a protein database. We would like to compute the occurrence probability table $p[\ell, m]$ and keep it in memory for fast lookup. Protein databases typically contain proteins of length 200 to 10.000 characters. Thus, we need $\ell_{\max} = 10,000$ rows. In a PMF setting using MALDI-TOF instruments, the maximal measured mass used is about 3.000 Da. Given an instrument accuracy of about $\Delta = 0.1$ Da, thus one decimal, we need $m_{\max} = 3,000/0.1 = 30,000$ columns. Assuming double precision for each entry (8 bytes each), a total memory of $30,000 \cdot 10,000 \cdot 8 = 24 \cdot 10^9$ bytes (about 2.24 GB) is required. For the computation, the same table size is needed temporarily for the $\bar{f}[l, m]$ table. Thus, to compute the occurrence probability table in this setting, we need about 4.5 GB of main memory, which is currently out of question for desktop computers.

We start this section with the observation that we can find an upper bound for the maximal length of a string of given mass in Section 5.1. Exceeding this bound in computation means that most of our quantities become independent of mass.

We then address the problem of memory consumption in Section 5.2, where we show that only a very small part of $\bar{f}[l, m]$ is needed at a time which can be computed as needed without increase of computation time, and Section 5.3, where we demonstrate how to compress and interpolate the occurrence probability table.

In Sections 5.4 and 5.5 we further explore the combinatorial structure of the recurrences for p, p_+ for the case of simple and general cleavage schemes, respectively, and show how these tables can be computed in time linear in mass and length using dynamic programming by exploiting this structure.

5.1 Length Bounds for Decompositions

The following definition of decomposability is taken from [7].

Definition 5.2 (Decomposition, decomposable). A mass m is called *decomposable over a (random) weighted alphabet* (Σ, μ) if there exist at least one weighted string s over (Σ, μ) such that

$$m = \sum_{i=1}^{|s|} \mu(s_i).$$

Every such string is then called a *decomposition* of m. The *length of a decomposition* is simply the length of the corresponding string.

Example 5.3. Given the weighted alphabet $\Sigma = \{A, B\}$ with $\mu(A) = 2$, $\mu(B) = 4$, the mass $m = 8$ is decomposable. Some possible decompositions are $s = AAAA$, $s' = AAB$ and $s'' = BB$, of length 4, length 3 and length 2, respectively. Thus, the same mass may have decompositions of different length. The mass $m' = 13$ is an example of a non-decomposable mass, as is every other odd integer.

To decide whether a mass is decomposable over a given weighted alphabet and to find one or all decompositions is not a trivial task; see e.g. [7] for an extensive treatment. It is however quite easy to give upper and lower bounds for the length of the decompositions of a mass m.

Proposition 5.4 (Length bounds for decompositions). *Given a weighted alphabet (Σ, μ) and a mass m, a decomposition of m over Σ has at most length*

$$l' \equiv l^{\ell}_{\max}(m) \;:=\; \min\left\{\ell, \left\lceil \frac{m}{\mu_{\min}} \right\rceil\right\},$$

where the ℓ subscript means that the decomposition must have at most ℓ characters and we use the l' notation for readability if mass and length are given from the context. Similarly, the length of a decomposition is lower bounded by

$$l^{\ell}_{\min}(m) \;:=\; \min\left\{\ell, \left\lceil \frac{m}{\mu_{\max}} \right\rceil\right\},$$

given that m is decomposable.

As noted, the lower bound is only valid if the mass m is decomposable. For implementations of the f/f' tables, we can nevertheless start at index $l^{\ell}_{\min}(m)$ for both decomposable and non-decomposable masses, as in the latter case, all entries of these tables are zero (or one) anyway.

Clearly, the upper bound is not tight, as the following example shows.

Example 5.5. For the weighted alphabet $\Sigma = \{A, B\}$ with $\mu(A) = 2$, $\mu(B) = 4$ of the previous example and mass $m = 13$, we have the upper bound $l^{100}_{\max}(13) = 7$. Nevertheless, there simply is no decomposition of $m = 13$ over Σ of whatever length.

5.2 Memory Efficient Computation of $\bar{f}'$

The table $f'[l, m]$ holds the probabilities that the internal part of a fragment has length l and mass m. We note that to compute the occurrence probability tables p and p_+, we only need the mass avoidance probabilities $\bar{f}'[l, m] = u'[l] - f'[l, m]$. All these primed quantities are independent of the position of the fragment in the string.

Mass independence of $\bar{f}'$. We now observe that for any mass m and any string length ℓ, we have that $f'[l, m] = 0$ for $l > l^{\ell}_{\max}(m)$, which means that there are no fragments of mass m and length l in this string simply because m is not decomposable with more than $l^{\ell}_{\max}(m)$ characters. So, the entries in the $\bar{f}'[l, m]$ table become independent of m and we have that $\bar{f}'[l, m] = u'[l]$ for $l > l^{\ell}_{\max}(m)$.

As we have to compute $u'[l]$ anyway, we can store it right away in $\mathcal{O}(\ell_{\max})$ memory, and only need to store the first $l^{\ell_{\max}}_{\max}(m_{\max})$ rows of $\bar{f}'[l, m]$. For the setting described above, with $\mu_{\min} \approx 500$ (scaled by $\Delta = 0.1$) and $m_{\max} = 30,000$, we would get $l^{\ell_{\max}}_{\max}(m_{\max}) \approx 60$ (for $\ell_{\max} \geq 60$).

Using this simple observation allows us to store all necessary information of the $\bar{f}'[l, m]$ table in $30,000 \cdot 60 = 1.800.000$ instead of $30,000 \cdot 10,000 = 3 \cdot 10^8$ entries, meaning about 13 MB instead of 2,300 MB memory consumption.

Block-wise computation of $\bar{f}'$. If we are only interested in the occurrence probability tables and thus only need $\bar{f}'[l, m]$ temporarily, a further reduction of the memory requirement is possible by looking at the recurrence equation for the occurrence probabilities $\bar{p}, \bar{p}_+$. To compute their respective value, only the last $3\,\mu_{\max}$ columns of $\bar{f}'[l, m]$ are needed. The same holds for the recurrence of $\bar{f}'[l, m]$, so we can compute the successive columns of $\bar{f}'$ by looking at the $3\,\mu_{\max}$ precessing columns.

To compute $\bar{p}, \bar{p}_+$ we thus only need to keep at most $3\,\mu_{\max}$ columns of $\bar{f}'$ in memory and can even compute the next column from them. Accessing the columns of $\bar{f}'$ by taking the modulo, exactly the same computation time is needed as if we computed the whole table $\bar{f}'$ and going from there, computing $\bar{p}, \bar{p}_+$.

In total, we only need to allocate a memory block of size $3 \cdot \mu_{\max} \cdot l^{\ell_{\max}}_{\max}(m_{\max})$ for considering strings of length $\leq \ell_{\max}$. For our PMF example, this means for $\Delta = 0.1$, the amino acid alphabet with $\mu_{\max} \approx 2,500$, $\mu_{\min} \approx 500$ and $m_{\max} = 30,000$ a value of $l^{10,000}_{\max}(m_{\max}) \approx \left\lceil \frac{30,000}{500} \right\rceil = 60$. Using double precision with 8 bytes per table entry, we need a total of 3.43 MB to keep the necessary parts of $\bar{f}'[l, m]$ in memory.

5.3 Compression and Interpolation of p

As we do not know in which order the elements of p are accessed and as we usually want to use these probabilities many times in constant time, we have to keep the whole table in memory.

We first note that for both $\bar{p}$ and $\bar{p}_+$, computation of columns can be performed independently as only the previous values in the same column m are needed. We can compute $\bar{p}[\ell, m]$ for some m up to $\ell = \ell_{\max}$ and then immediately store $p[\ell, m] = 1 - \bar{p}[\ell, m]$ for $\ell = 1 \ldots \ell_{\max}$. This means that after completing column m of $\bar{p}_+$, we can immediately compute the corresponding column of p and store it in-place as we will not need this column of $\bar{p}_+$ again. We therefore do not have to keep both the full p and $\bar{p}_+$ in memory but only p (as we want to access its values later on) and one column of $\bar{p}$. Another advantage of this procedure is that to compute the m-th column of p, exactly the same columns of $\bar{f}'$ are needed as for the computation of the m-th column of $\bar{p}$.

Thus, we do not have to recompute any part of $\bar{f}'$ later on.

As seen in this and the previous section, the only main problem concerning memory requirements is that we want every element of p in the main memory. Clearly, this problem can not be solved by trickier computation. Our only possibility to reduce the size of p is therefore compression. Two major observations turn out to be helpful.

Non-decomposable masses. First, we expect some masses not to be decomposable, meaning that every entry in the corresponding column of p is just 0. We may not want to store these useless columns. Unfortunately, the number of such columns is negligible and we would need an additional data structure to keep track of column indexing. We also would have to test every column if the corresponding mass is decomposable. This can be done in constant time during computation by setting a flag whenever a non-zero entry is produced for the first time in a column. We nevertheless do not follow this compression procedure as a first very rough estimate shows far less than 1000 complete zero columns for a maximal mass of $30,000$ (using $\Delta = 0.1$). This means that we would have to keep more than 97% of the columns anyway. However, this low rate of non-decomposable columns may just by a side effect of floating point numerics and has to be studied further. Usually, we would expect nearly every mass above 1,500 Da to be decomposable whereas between 15 and 30 % of the masses below 1,500 Da would be expected to not be decomposable which would lead to a much better compressibility of p in the range of 1-1,500 Daltons.

Smooth dependence of p on ℓ. Another possibility to compress the table comes from the following observation: Although the values of the row entries seem not to have a regularity that can be exploited (see Figure 6), we can nevertheless expect the column entries to have a more smooth behavior. If the corresponding mass is not decomposable, every entry would be zero anyway. If it is decomposable, however, we can expect that from a certain string length on, the probability to see a fragment of such mass increases continously with the string length. This argument can be supported by the following considerations: If the mass m is decomposable as a fragment, meaning there is at least one decomposition of m that has a valid fragment structure (ending on a cleavage character and eventually starting with a non-prohibition character), the probability to see such a fragment in a string is greater than zero from some string length on. (The string must be at least as long as the decomposition.) If the string gets longer, its number of fragments increases and therefore also the chance that a fragment of mass m appears. The occurrence probability is increasing until it reaches its limit of 1 for infinitely long strings.

Interpolation of rows. To exploit this observation, the following procedure can be used to compress the table: We store the first K entries in the m-th column to avoid combinatorial effects and from then on store only every Ω-th entry. We call Ω the *compression factor* of p. We nevertheless have to compute every intermediate entry during computation of p in order to avoid accumulation of approximation errors. If an

intermediate value is accessed afterwards, it can easily be computed in constant time using a linear interpolation between the two nearest entries. In practice, a compression factor about 20-25 starting from $K = 100$ did not result in any noticeable interpolation error compared to the exact values.

5.4 Computation of Simple Cleavage Models

We start with the efficient computation in the case of simple cleavage models, as the main ideas are the same, but the results are much easier to understand than the results for standard schemes. We first stress again, that for $l > l' = \min\{\ell, \lceil \frac{m}{\mu_{\min}} \rceil\} + 1$, the entries in the fragment table $\bar{f}'[l, m]$ become independent of m (and thus independent can be computed without summing over the character mass distribution), as $\bar{f}[l, m] = u'[l]$ in this case. Note that we have to add 1 to the original definition of l' to capture the fact that we look back in the $\bar{f}'[l, m]$ table. Furthermore, as the inner sum is independent of ℓ, the entries $\bar{f}_+[l, m]$ can be precomputed from $\bar{f}'[l, m]$. The memory efficient implementation of 5.2 is thus still possible.

We now state the main result.

Lemma 5.6 (Efficient computation complexity (simple case)). *For simple cleavage schemes, a maximal sequence length $\ell_{\max}$ and a maximal occurrence mass $m_{\max}$, the table of fragment occurrence probabilities $p[\ell, m]$, $1 \leq \ell \leq \ell_{\max}$, $1 \leq m \leq m_{\max}$, can be computed from $\bar{f}'$ in time $\mathcal{O}(\ell_{\max} l' m_{\max} (\mu_{\max} - \mu_{\min}))$ for probabilistically weighted alphabets and in time $\mathcal{O}(\ell_{\max} l' m_{\max})$ for weighted alphabets with constant character weights. As $\mu_{\max}$ is a (small) constant, computation can be done in time linear in maximal sequence length, maximal mass considered and quadratic in required precision.*

Proof. We first observe that although l' is a function of ℓ, it is constant once $\ell > \lceil \frac{m}{\mu_{\min}} \rceil$, which is independent of ℓ. We will write $\bar{f}$ for both $\bar{f}_1$, $\bar{f}_+$ as they coincide for simple cleavage schemes and similarly write $u[l]$ for both $u_1[l]$, $u_+[l]$. We will now fix a mass m and also omit summing over the mass distributions

The main observation is that the value of $\bar{p}[\ell + 1, m]$ (if we lengthen the string by one) can be split into a part involving fragment lengths $l \leq l'$ computable in time $\mathcal{O}(l')$ and a part involving fragment lengths $l > l'$ that has already been computed in the previous step.

$$\begin{aligned}
\bar{p}[\ell+1,m] &= \bar{f}[\ell+1,m] + \sum_{l=1}^{\ell+1} \bar{p}[\ell-l,m]\cdot\bar{f}[l,m] \\
&= \underbrace{\bar{f}[\ell+1,m] + \sum_{l=1}^{l'} \bar{p}[\ell+1-l,m]\cdot\bar{f}[l,m]}_{=:\,\alpha[\ell+1,m]} \\
&\quad + \underbrace{\sum_{l=l'+1}^{\ell+1} \bar{p}[\ell+1-l,m]\cdot\bar{f}[l,m]}_{=:\,\beta[\ell+1,m]}
\end{aligned}$$

Clearly, we have to compute $\alpha[\ell+1,m]$ anyway as it depends on $\bar{f}[l,m]$ for $l<l'$. In the last sum, $\bar{f}[l,m]=u[l]$ is independent of m and we immediately get the following relationship between successive rows of $\bar{f}[l,m]$ for $l>l'$: $\bar{f}[l+1,m]=u[l+1]=u[l]\cdot(1-\gamma)=\bar{f}[l,m]\cdot(1-\gamma)$. Using the index shift $l\to l+1$ in the last sum, we get

$$\begin{aligned}
\bar{p}[\ell+1,m] &= \alpha[\ell+1,m] + \sum_{l=l'}^{\ell} \bar{p}[\ell-l,m]\cdot\bar{f}[l+1,m] \\
&= \alpha[\ell+1,m] + (1-\gamma)\cdot\sum_{l=l'}^{\ell} \bar{p}[\ell-l,m]\cdot u[l] \\
&= \alpha[\ell+1,m] + (1-\gamma)\cdot\bar{p}[\ell-l',m]\cdot u[l'] + (1-\gamma)\cdot\beta[\ell,m]
\end{aligned}$$

Computing $\alpha[L+1,m]$ takes $\mathcal{O}(l'\,(\mu_{\max}-\mu_{\min}))$ time. The second term can be computed in time $\mathcal{O}(\mu_{\max}-\mu_{\min})$ and the third in $\mathcal{O}(1)$ if we stored $\beta[\ell,m]$ in the previous step computing $\bar{p}[\ell,m]$ or in time $\mathcal{O}(l'\,(\mu_{\max}-\mu_{\min}))$ if it has to be recomputed from $\bar{p}[\ell,m]$. Thus, the next row entry of $\bar{p}[\ell,m]$ can be computed in time $\mathcal{O}(l'\,(\mu_{\max}-\mu_{\min}))$ where l' is constant from some small value of ℓ. The dependence on $\mu_{\max}$ comes from the fact that we have to sum over the whole mass distribution of the characters starting from $\mu_{\min}$. The whole p table can thus be computed in time $\mathcal{O}\left(\ell_{\max}\,l'\,m_{\max}\,(\mu_{\max}-\mu_{\min})\right)$ instead of $\mathcal{O}\left(\ell_{\max}^2\,m_{\max}\,(\mu_{\max}-\mu_{\min})\right)$ using the naive implementation. □

5.5 Standard Cleavage Models

Again, we can use the memory efficient implementation of $\bar{f}'[l,m]$. Again computing values in column m of $\bar{p}[\ell,m]$ does only depend on values in the same column. We will use the recurrence equation from Lemma 4.2 for $\bar{p}_+[\ell,m]$.

Lemma 5.7 (Efficient computation complexity (general standard case)). *For general standard cleavage schemes, the table of fragment occurrence probabilities $p[\ell,m]$, $1\le\ell\le\ell_{\max}$, $1\le m\le m_{\max}$, can be computed in time $\mathcal{O}\left(\ell_{\max}\,l'\,m_{\max}\,(\mu_{\max}-\mu_{\min})\right)$. For weighted alphabets with constant character masses, the $(\mu_{\max}-\mu_{\min})$ factor is again constant.*

Proof. We will again omit the summation over the mass distribution and concentrate on a fixed mass m. We first observe that the first, second and last summand in the equation of Lemma 4.2 can be computed in a simple preprocessing step. We will denote them by $\alpha'[\ell, m]$. The recurrence can then be written as

$$\begin{aligned}\bar{p}_+[\ell, m] &= \alpha'[\ell, m] \\ &\quad + \sum_{l=2}^{\ell} \bar{p}_+[\ell - l, m] \underbrace{\sum_{\sigma \in \bar{\Pi} \cap \bar{\Gamma}} \sum_{\sigma' \in \Gamma} \bar{f}'[l-2, m - \mu_{\sigma\sigma'}] \cdot \mathbb{P}(\sigma\sigma')}_{=: \beta_1[l,m]} \\ &\quad + \sum_{l=2}^{\ell} \bar{p}_+[\ell - l, m] \underbrace{\sum_{\sigma \in \Gamma} \sum_{\sigma' \in \Pi} \sum_{\sigma'' \in \Gamma} \bar{f}'[l-3, m - \mu_{\sigma\sigma'\sigma''}] \cdot \mathbb{P}(\sigma\sigma'\sigma'')}_{=: \beta_2[l,m]}\end{aligned}$$

We note that this is the same as

$$\bar{p}_+[\ell, m] \;=\; \alpha'[\ell, m] + \sum_{l=2}^{\ell} \bar{p}_+[\ell - l, m] \cdot \bar{f}_+[l, m]$$

since $\bar{f}_+[l, m] = \beta_1[l, m] + \beta_2[l, m]$ by Lemma 3.33. Now we can again split the sum into a part up to and a part starting from l'. A little care has to be taken that we have to take the original definition of l' plus 3 because of the lookback in the $\bar{f}'$ table. To simplify notations, we will nevertheless denote this value by l'.

Now, for $l > l'$ we have that $\bar{f}_+[l, m] = u_+[l]$, leading to

$$\bar{p}_+[\ell, m] \;=\; \underbrace{\alpha[\ell, m]' + \sum_{l=2}^{l'} \bar{p}_+[\ell - l, m] \cdot \bar{f}_+[l, m]}_{\alpha[\ell,m]} + \underbrace{\sum_{l=l'+1}^{\ell} \bar{p}_+[\ell - l, m] \cdot u_+[l]}_{\beta[\ell,m]}$$

As $\alpha[\ell, m]$ depends on $\bar{f}_+[l, m]$, we have to compute it anyway. This can again be done in time $\mathcal{O}(l' (\mu_{\max} - \mu_{\min}))$ which becomes independent of the string length ℓ once this is large enough.

For the last sum, we can again try to exploit the recurrence equation for $u_+[l]$ from Lemma 3.22.

We take at look at the two immediate successors of $\bar{p}_+[\ell, m]$ for $\ell > l'$:

$$\bar{p}_+[\ell + 1, m] \;=\; \alpha[\ell + 1, m] \;+\; \sum_{l=l'+1}^{\ell+1} \bar{p}_+[\ell + 1 - l, m] \cdot u_+[l]$$

and

$$\bar{p}_+[\ell + 2, m] \;=\; \alpha[\ell + 2, m] \;+\; \sum_{l=l'+1}^{\ell+2} \bar{p}_+[\ell + 2 - l, m] \cdot u_+[l]$$

We can now use the recurrence equation for $u_+[l]$ for $l > l'$ and get

$$\bar{p}_+[\ell, m] = \alpha[\ell, m] + \sum_{l=l'+1}^{\ell} \bar{p}[\ell - l, m] \cdot ((1-\gamma) \cdot u_+[l-1] + \gamma\pi \cdot u_+[l-2]) .$$

Splitting the sum and performing the index shifts $l \to l+1$ and $l \to l+2$, respectively, on the summation index, we get

$$\bar{p}_+[\ell, m] = \alpha[\ell, m] + (1-\gamma) \cdot \sum_{l=l'}^{\ell+1} \bar{p}[\ell + 1 - l, m] \cdot u_+[l] + \gamma\pi \cdot \sum_{l=l'-1}^{\ell+2} \bar{p}[\ell + 2 - l, m] \cdot u_+[l].$$

By extracting the $l', l'-1$ terms and finally end up with

$$\begin{aligned} \bar{p}_+[\ell+2, m] \;=\; & \alpha[\ell+2, m] \\ & + (1-\gamma) \cdot \big(\bar{p}[\ell + 1 - l'] \cdot u_+[l'] \;+\; \beta[\ell+1, m]\big) \\ & + \gamma\pi \cdot \big(\bar{p}[\ell - l' - 1, m] \cdot u_+[l'-1] \;+\; \bar{p}[\ell - l', m] u_+[l'] + \beta[\ell, m]\big) . \end{aligned}$$

Note that the last line actually defines a recurrence equation of second order for the sequence $(\beta[L, m])_{L > l'+2}$. Not surprisingly, this recurrence is simply the recurrence for $u_+[l]$ with some additional correction terms. As in the case of simple cleavage schemes, we can compute $\alpha[\ell+2, m]$ in time $\mathcal{O}(l' \, (\mu_{\max} - \mu_{\min}))$ and the rest in time $\mathcal{O}(\mu_{\max} - \mu_{\min})$ given we stored $\beta[\ell+1, m]$ and $\beta[\ell, m]$ in the two previous steps. As we only need them for the $\ell + 2$-step, they can afterwards be overwritten, so we only need constant extra memory. □

6 Conclusion

We presented a rigorous mathematical model for weighted strings and their random models. We modelled the fragmentation of such strings using restriction type cleavage reactions which gives us a model for biochemical digestion of proteins. This model is particularly applicable to mass spectrometry of proteins and peptides.

We presented weighted HMMs, a general computational framework for length and mass statistics of random weighted strings and their fragments. Our main results for wHMMs are Theorems 3.26 and 3.27. We showed in detail how the distributions of these statistics can be derived from the model for the case of i.i.d. strings and standard cleavage schemes. The model is readily extendable to Markov sequences and general cleavage schemes. Mass modifications of the sequence termini can easily be modeled, and it is also possible to introduce probabilistically missed cleavage sites.

We also derived the same statistics by combinatorical means which in turn led us to recurrence equation for these statistics. Using these recurrences, we were able to develop efficient algorithms to compute the statistics and showed in detail the main ideas of these algorithms. To make these algorithms work on standard desktop machines, we presented several observations and methods to significantly reduce the memory requirements for both computation and usage of the statistics.

Amino acid	Monoisotopic	Average	Frequency (%)
A	71.03711	71.0788	7.85
C	103.00919	103.1388	1.54
D	115.02694	115.0886	5.31
E	129.04259	129.1155	6.61
F	147.06841	147.1766	4.00
G	57.02146	57.0520	6.95
H	137.05891	137.1412	2.27
I	113.08406	113.1595	5.92
K	128.09496	128.1742	5.92
L	113.08406	113.1595	9.63
M	131.04049	131.1925	2.38
N	114.04293	114.1039	4.18
P	97.05276	97.1167	4.83
Q	128.05858	128.1308	3.94
R	156.10111	156.1876	5.33
S	87.03203	87.0782	6.85
T	101.04768	101.1051	5.45
V	99.06841	99.1326	6.73
W	186.07931	186.2133	1.15
Y	163.06333	163.1760	3.06

Table 3: Amino acid masses and frequencies

We implemented these efficient algorithms and compared the statistics of our model to their empirical counterparts by an in-silico digest of the current SwissProt protein sequence database, release 48. A good agreement was observed, showing that our model correctly represents main features of protein fragmentation.

We expect that wHMMs will prove their usefulness for further probability computations in mass spectrometry and assume that they will become a standard tool for significance computations for peptide mass fingerprinting.

Acknowledgments Sebastian Böcker is currently supported by the Deutsche Forschungsgemeinschaft (BO 1910/1-1) within the Computer Science Action Program. The authors would like to thank Matthias Steinrücken for help in running several of the experiments. Jens Stoye has provided valuable input.

A Amino Acid Weights and Frequencies

Table 3 shows amino acids with their monoisotopic and average masses in Daltons and their occurrence frequencies in the SwissProt database resource (October 2005).

B Isotopic Distributions of Amino Acids

The following tables give the isotopic distribution of the twenty amino acids with masses given in Dalton (Da). Due to floating point computations, all entries having probability smaller than 10^{-15} have been discarded.

A		C		D		E	
Mass	Prob.	Mass	Prob.	Mass	Prob.	Mass	Prob.
71	0.9605	103	0.9126	115	0.9453	129	0.9345
72	0.0369	104	0.0423	116	0.0477	130	0.0579
73	0.0024	105	0.0430	117	0.0066	131	0.0071
74	7.6616e-05	106	0.0018	118	0.0002	132	0.0003
75	1.0286e-06	107	0.0001	119	1.7107e-05	133	2.0289e-05
76	6.1448e-09	108	3.7288e-06	120	6.2278e-07	134	8.1159e-07
77	1.4274e-11	109	5.8690e-08	121	1.9031e-08	135	2.5967e-08
78		110	4.6455e-10	122	4.8466e-10	136	6.9806e-10
79		111	1.8300e-12	123	7.8264e-12	137	1.3323e-11
80		112		124	7.0868e-14	138	1.6084e-13

F		G		H		I	
Mass	Prob.	Mass	Prob.	Mass	Prob.	Mass	Prob.
147	0.8977	57	0.9715	137	0.9218	113	0.9280
148	0.0955	58	0.0261	138	0.0735	114	0.0677
149	0.0063	59	0.0021	139	0.0044	115	0.0039
150	0.0003	60	5.2382e-05	140	0.0001	116	0.0001
151	1.1490e-05	61	4.2949e-07	141	5.7429e-06	117	4.5843e-06
152	2.8656e-07	62	1.0877e-09	142	1.0656e-07	118	7.4777e-08
153	4.9956e-09	63	4.3335e-13	143	1.2965e-09	119	7.5925e-10
154	6.1456e-11			144	1.0490e-11	120	4.7716e-12
155	5.3446e-13			145	5.6659e-14	121	1.8215e-14

K		L		M		N	
Mass	Prob.	Mass	Prob.	Mass	Prob.	Mass	Prob.
128	0.9245	113	0.9280	131	0.8919	114	0.9439
129	0.0710	114	0.0677	132	0.0619	115	0.0508
130	0.0042	115	0.0039	133	0.0431	116	0.0049
131	0.0001	116	0.0001	134	0.0027	117	0.0002
132	5.2244e-06	117	4.5843e-06	135	0.0001	118	8.2942e-06
133	9.2048e-08	118	7.4777e-08	136	6.4221e-06	119	2.5212e-07
134	1.0437e-09	119	7.5925e-10	137	1.5678e-07	120	4.6671e-09
135	7.6866e-12	120	4.7716e-12	138	2.2935e-09	121	4.9743e-11
136	3.6910e-14	121	1.8215e-14	139	2.0549e-11	122	3.0047e-13
				140	1.1251e-13		

P		Q		R		S	
Mass	Prob.	Mass	Prob.	Mass	Prob.	Mass	Prob.
97	0.9390	128	0.9332	156	0.9177	87	0.9582
98	0.0574	129	0.0610	157	0.0772	88	0.0372
99	0.0033	130	0.0054	158	0.0047	89	0.0043
100	0.0001	131	0.0002	159	0.0002	90	0.0001
101	3.0288e-06	132	1.0671e-05	160	6.6036e-06	91	5.9524e-06
102	3.8920e-08	133	3.4447e-07	161	1.3197e-07	92	1.5975e-07
103	2.8699e-10	134	7.5143e-09	162	1.7773e-09	93	2.0739e-09
104	1.1622e-12	135	1.0319e-10	163	1.6475e-11	94	1.2303e-11
		136	8.7952e-13	164	1.0673e-13	95	2.8691e-14

T		V		W		Y	
Mass	Prob.	Mass	Prob.	Mass	Prob.	Mass	Prob.
101	0.9473	99	0.9387	186	0.8745	163	0.8956
102	0.0477	100	0.0577	187	0.1160	164	0.0956
103	0.0047	101	0.0033	188	0.0088	165	0.0081
104	0.0001	102	0.0001	189	0.0004	166	0.0005
105	7.6228e-06	103	3.0682e-06	190	2.0753e-05	167	2.4322e-05
106	2.2625e-07	104	3.9820e-08	191	6.4260e-07	168	9.3068e-07
107	3.8904e-09	105	2.9865e-10	192	1.4733e-08	169	2.8074e-08
108	3.6329e-11	106	1.2488e-12	193	2.5257e-10	170	6.3634e-10
109	1.7559e-13			194	3.2661e-12	171	1.0547e-11
				195	3.2176e-14	172	1.2671e-13

C Restriction Enzymes

Table 4 shows some known restriction enzymes and their cleavage site patterns. Cleavage occurs after the cleavage character, but not before the prohibition character (unless stated otherwise). The last column of the table indicates whether the enzyme's behavior is captured by our model.

References

[1] Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003.

[2] A.Frank and P.Pevzner. Pepnovo: de novo peptide sequencing via probabilistic network modeling. *Anal Chem.*, 15:964–973, 2005.

[3] A. Bairoch and B. Boeckmann. The swiss-prot protein sequence data bank. *Nucleic Acids Res.*, 20:2019–2022, 1992.

[4] Nikhil Bansal, Mark Cieliebak, and Zsuzsanna Lipták. Efficient algorithms for finding submasses in weighted strings. In *Proc. of the Fifteenth Annual Combinatorial*

Enzyme	Cleaves after	except before	covered?
arg C	R	P	+
asp N	before D		+
chymotrypsin	E,(L, M),W,Y	P,after PY	- (PY)
cyanogen bromide	M		+
Glu C (basic)	E	P or E	- $(\Gamma \cap \Pi \neq \emptyset)$
Glu C (acidic)	D or E	D or E	- $(\Gamma \cap \Pi \neq \emptyset)$
Lys C	K		+
pepsin (high activity)	F or L		+
pepsin (low activity)	A, E, F, L, Q, W, Y		+
proteinase K	A, C, F, G, M, S, W, Y		+
trypsin	K or R	P	+

Table 4: Enzymes and their cleavage behavior

Pattern Matching Symposium (CPM 2004), volume 3109 of *LNCS*, pages 194–204. Springer, 2004.

[5] Patrick Billingsley. *Probability and Measure*. Wiley Interscience, New York, 3rd edition, 1995.

[6] Sebastian Böcker and Hans-Michael Kaltenbach. Mass spectra alignments and their significance. In Alberto Apostolico, Maxime Crochemore, and Kunsoo Park, editors, *Combinatorial Pattern Matching*, volume 3537 of *Lecture Notes in Bioinformatics*, pages 429–441. Springer, 2005.

[7] Sebastian Böcker and Zsuzsanna Lipták. Efficient mass decomposition. In *Proc. of ACM Symposium on Applied Computing (ACM SAC 2005)*, pages 151–157, Santa Fe, USA, 2005.

[8] Mark Cieliebak, Thomas Erlebach, Zsuzsanna Lipták, Jens Stoye, and Emo Welzl. Algorithmic complexity of protein identification: Combinatorics of weighted strings. *Discrete Applied Mathematics*, 137(1):27–46, 2004.

[9] Jacques Colinge, Alexandre Masselot, and Jérôme Magnin. A systematic statistical analysis of ion trap tandem mass spectra in view of peptide scoring. In *Proc. of WABI 2003, Budapest, Hungary*, volume 2812 of *Lecture Notes in Computer Science*, pages 25–38. Springer, 2003.

[10] Nathan Edwards and Ross Lippert. Generating peptide candidates from amino-acid sequence databases for protein identification via mass spectrometry. In *Proc. of the* 2^{nd} *International Workshop on Algorithms in Bioinformatics (WABI)*, pages 68–81, 2002.

[11] William J. Henzel, Colin Watanabe, and John T. Stults. Protein idendification: The origins of peptide mass fingerprints. *J.Am.Soc.Mass Spectrometry*, 14:931–942, 2003.

[12] D.J.C. Pappin, P. Hojrup, and A.J. Bleasby. Rapid identification of proteins by peptide-mass fingerprints. *Current Biology*, 3(6):327–332, 1993.

[13] D. Perkins, D. Pappin, D. Creasy, and J. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20:3551–3567, 1997.

[14] Mireille Régnier. A unified approach to word occurrence probabilities. *Discrete Applied Mathematics*, 104:259–280, 2000.

[15] Gesine Reinert, Sophie Schbath, and Michael S. Waterman. Probabilistic and statistical properties of words: An overview. *Journal of Computational Biology*, 7:1–46, 2000.

[16] S. Robin and J.-J. Daudin. Exact distribution of word occurrences in a random sequence of letters. *Journal of Applied Probability*, 36:179–193, 1999.

[17] Yunhu Wan and Ting Chen. A hidden markov model based scoring function for mass spectrometry database search. In *Proc. of RECOMB 2005*, volume 3500 of *LNCS*, pages 342–356. Springer, 2005.

[18] I-Jeng Wang, Christopher P. Diehl, and Fernando J. Pineda. A statistical model of proteolytic digestion. In *Proceedings of IEEE CSB 2003*, pages 506–508, Stanford, California, 2003.

[19] Michael S. Waterman. *Introduction to Computational Biology.* CRC Press, Boca Raton, first edition, 1996.

[20] Abraham J. Wyner. More on recurrence and waiting times. *The Annals of Applied Probability*, 9:780–796, 1999.

[21] Wenzhu Zhang and Brian T. Chait. Profound: an expert system for protein identification using mass spectrometric peptide mapping information. *Anal. Chem.*, 72(11):2482–2489, 2000.

Bisher erschienene Reports an der Technischen Fakultät
Stand: 2005-11-21

94-01 Modular Properties of Composable Term Rewriting Systems
(Enno Ohlebusch)

94-02 Analysis and Applications of the Direct Cascade Architecture
(Enno Littmann, Helge Ritter)

94-03 From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction
(Robert Giegerich, Stefan Kurtz)

94-04 Die Verwendung unscharfer Maße zur Korrespondenzanalyse in Stereo Farbbildern
(André Wolfram, Alois Knoll)

94-05 Searching Correspondences in Colour Stereo Images – Recent Results Using the Fuzzy Integral
(André Wolfram, Alois Knoll)

94-06 A Basic Semantics for Computer Arithmetic
(Markus Freericks, A. Fauth, Alois Knoll)

94-07 Reverse Restructuring: Another Method of Solving Algebraic Equations
(Bernd Bütow, Stephan Thesing)

95-01 PaNaMa User Manual V1.3
(Bernd Bütow, Stephan Thesing)

95-02 Computer Based Training-Software: ein interaktiver Sequenzierkurs
(Frank Meier, Garrit Skrock, Robert Giegerich)

95-03 Fundamental Algorithms for a Declarative Pattern Matching System
(Stefan Kurtz)

95-04 On the Equivalence of E-Pattern Languages
(Enno Ohlebusch, Esko Ukkonen)

96-01 Static and Dynamic Filtering Methods for Approximate String Matching
(Robert Giegerich, Frank Hischke, Stefan Kurtz, Enno Ohlebusch)

96-02 Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language
(Alois Knoll, Bernd Hildebrand, Jianwei Zhang)

96-03 Correctness in System Engineering
(Peter Ladkin)

96-04 An Algebraic Approach to General Boolean Constraint Problems
(Hans-Werner Güsgen, Peter Ladkin)

96-05 Future University Computing Resources
(Peter Ladkin)

96-06 Lazy Cache Implements Complete Cache
(Peter Ladkin)

96-07 Formal but Lively Buffers in TLA+
(Peter Ladkin)

96-08 The X-31 and A320 Warsaw Crashes: Whodunnit?
(Peter Ladkin)

96-09 Reasons and Causes
(Peter Ladkin)

96-10 Comments on Confusing Conversation at Cali
(Dafydd Gibbon, Peter Ladkin)

96-11 On Needing Models
(Peter Ladkin)

96-12 Formalism Helps in Describing Accidents
(Peter Ladkin)

96-13 Explaining Failure with Tense Logic
(Peter Ladkin)

96-14 Some Dubious Theses in the Tense Logic of Accidents
(Peter Ladkin)

96-15 A Note on a Note on a Lemma of Ladkin
(Peter Ladkin)

96-16 News and Comment on the AeroPeru B757 Accident
(Peter Ladkin)

97-01 Analysing the Cali Accident With a WB-Graph
(Peter Ladkin)

97-02 Divide-and-Conquer Multiple Sequence Alignment
(Jens Stoye)

97-03 A System for the Content-Based Retrieval of Textual and Non-Textual Documents Based on Natural Language Queries
(Alois Knoll, Ingo Glöckner, Hermann Helbig, Sven Hartrumpf)

97-04 Rose: Generating Sequence Families
(Jens Stoye, Dirk Evers, Folker Meyer)

97-05 Fuzzy Quantifiers for Processing Natural Language Queries in Content-Based Multimedia Retrieval Systems
(Ingo Glöckner, Alois Knoll)

97-06 DFS – An Axiomatic Approach to Fuzzy Quantification
(Ingo Glöckner)

98-01 Kognitive Aspekte bei der Realisierung eines robusten Robotersystems für Konstruktionsaufgaben
(Alois Knoll, Bernd Hildebrandt)

98-02 A Declarative Approach to the Development of Dynamic Programming Algorithms, applied to RNA Folding
(Robert Giegerich)

98-03 Reducing the Space Requirement of Suffix Trees
(Stefan Kurtz)

99-01 Entscheidungskalküle
(Axel Saalbach, Christian Lange, Sascha Wendt, Mathias Katzer, Guillaume Dubois, Michael Höhl, Oliver Kuhn, Sven Wachsmuth, Gerhard Sagerer)

99-02 Transforming Conditional Rewrite Systems with Extra Variables into Unconditional Systems
(Enno Ohlebusch)

99-03 A Framework for Evaluating Approaches to Fuzzy Quantification
(Ingo Glöckner)

99-04 Towards Evaluation of Docking Hypotheses using elastic Matching
(Steffen Neumann, Stefan Posch, Gerhard Sagerer)

99-05 A Systematic Approach to Dynamic Programming in Bioinformatics. Part 1 and 2: Sequence Comparison and RNA Folding
(Robert Giegerich)

99-06 Autonomie für situierte Robotersysteme – Stand und Entwicklungslinien
(Alois Knoll)

2000-01 Advances in DFS Theory
(Ingo Glöckner)

2000-02 A Broad Class of DFS Models
(Ingo Glöckner)

2000-03 An Axiomatic Theory of Fuzzy Quantifiers in Natural Languages
(Ingo Glöckner)

2000-04 Affix Trees
(Jens Stoye)

2000-05 Computergestützte Auswertung von Spektren organischer Verbindungen
(Annika Büscher, Michaela Hohenner, Sascha Wendt, Markus Wiesecke, Frank Zöllner, Arne Wegener, Frank Bettenworth, Thorsten Twellmann, Jan Kleinlützum, Mathias Katzer, Sven Wachsmuth, Gerhard Sagerer)

2000-06 The Syntax and Semantics of a Language for Describing Complex Patterns in Biological Sequences
(Dirk Strothmann, Stefan Kurtz, Stefan Gräf, Gerhard Steger)

2000-07 Systematic Dynamic Programming in Bioinformatics (ISMB 2000 Tutorial Notes)
(Dirk J. Evers, Robert Giegerich)

2000-08 Difficulties when Aligning Structure Based RNAs with the Standard Edit Distance Method
(Christian Büschking)

2001-01 Standard Models of Fuzzy Quantification
(Ingo Glöckner)

2001-02 Causal System Analysis
(Peter B. Ladkin)

2001-03 A Rotamer Library for Protein-Protein Docking Using Energy Calculations and Statistics
(Kerstin Koch, Frank Zöllner, Gerhard Sagerer)

2001-04 Eine asynchrone Implementierung eines Microprozessors auf einem FPGA
(Marco Balke, Thomas Dettbarn, Robert Homann, Sebastian Jaenicke, Tim Köhler, Henning Mersch, Holger Weiss)

2001-05 Hierarchical Termination Revisited
(Enno Ohlebusch)

2002-01 Persistent Objects with O2DBI
(Jörn Clausen)

2002-02 Simulation von Phasenübergängen in Proteinmonoschichten
(Johanna Alichniewicz, Gabriele Holzschneider, Morris Michael, Ulf Schiller, Jan Stallkamp)

2002-03 Lecture Notes on Algebraic Dynamic Programming 2002
(Robert Giegerich)

2002-04 Side chain flexibility for 1:n protein-protein docking
(Kerstin Koch, Steffen Neumann, Frank Zöllner, Gerhard Sagerer)

2002-05 ElMaR: A Protein Docking System using Flexibility Information
(Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)

2002-06 Calculating Residue Flexibility Information from Statistics and Energy based Prediction
(Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)

2002-07 Fundamentals of Fuzzy Quantification: Plausible Models, Constructive Principles, and Efficient Implementation
(Ingo Glöckner)

2002-08 Branching of Fuzzy Quantifiers and Multiple Variable Binding: An Extension of DFS Theory
(Ingo Glöckner)

2003-01 On the Similarity of Sets of Permutations and its Applications to Genome Comparison
(Anne Bergeron, Jens Stoye)

2003-02 SNP and mutation discovery using base-specific cleavage and MALDI-TOF mass spectrometry
(Sebastian Böcker)

2003-03 From RNA Folding to Thermodynamic Matching, including Pseudoknots
(Robert Giegerich, Jens Reeder)

2003-04 Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt
(Sebastian Böcker)

2003-05 Systematic Investigation of Jumping Alignments
(Constantin Bannert)

2003-06 Suffix Tree Construction and Storage with Limited Main Memory
(Klaus-Bernd Schürmann, Jens Stoye)

2003-07 Sequencing from compomers in thepresence of false negative peaks
(Sebastian Böcker)

2003-08 Genalyzer: An Interactive Visualisation Tool for Large-Scale Sequence Matching – Biological Applications and User Manual
(Jomuna V. Choudhuri, Chris Schleiermacher)

2004-01 Sequencing From Compomers is NP-hard
(Sebastian Böcker)

2004-02 The Money Changing Problem revisited: Computing the Frobenius number in time $O(k\, a_1)$
(Sebastian Böcker, Zsuzsanna Lipták)

2004-03 Accelerating the Evaluation of Profile HMMs by Pruning Techniques
(Thomas Plötz, Gernot A. Fink)

2004-04 Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection
(Ferdinando Cicalese, Peter Damaschke, Ugo Vaccaro)

2004-05 Compressed Representation of Sequences and Full-Text Indexes
(Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, Gonzalo Navarro)

2005-01 Overlaps Help: Improved Bounds for Group Testing with Interval Queries
(Ferdinando Cicalese, Peter Damaschke, Libertad Tansini, Sören Werth)

2005-02 Two batch Fault-tolerant search with error cost constraints: An application to learning
(Ferdinando Cicalese)

2005-03 Searching for the Shortest Common Supersequence
(Sergio A. de Carvalho Jr., Sven Rahmann)

2005-04 Counting Suffix Arrays and Strings
(Klaus-Bernd Schürmann, Jens Stoye)

2005-05 Alignment of Tandem Repeats with Excision, Duplication, Substitution and Indels (EDSI)
(Michael Sammeth, Jens Stoye)