Freie Universität Berlin

Fachbereich Mathematik u. Informatik
Studiengang Bioinformatik

Prediction of Post-translational Modifications of Proteins from 2-DE/MS Data

Vorhersage post-translationaler Modifikationen anhand von 2-DE/MS Daten

Masterarbeit
zur Erlangung des akademischen Grades
"Master of Science (M.Sc.)"

Bearbeiter: Axel Rack
M.-Nr.: 3579100

Betreuer: Dr. K.-P. Pleißner
Prof. Dr. K. Reinert

Acknowledgments

First of all, I would like to thank Dr. Klaus-Peter Pleissner for suggesting the topic of this thesis and for providing the possibility to work at the Max Planck Institute for Infection Biology. I very much appreciated his advice and the valuable discussions we had. I am indebted to Prof. Dr. Knut Reinert for helpful advice and suggestions. I am also grateful to Prof. Dr. Sebastian Böcker for suggestions concerning the implementation of the algorithm. I would also like to thank Dr. Peter R. Jungblut and Dr. Robert Stein for insightful discussions.

Zusammenfassung

Die lebende Zelle ist eine komplexe Einheit, die aus Nukleinsäuren, Proteinen und anderen Biomolekülen besteht, welche zusammen ein dynamisches Netzwerk bilden. Die Entschlüsselung dieses Netzwerks ist für viele Wissenschaftler verschiedener Disziplinen von großem Interesse. Durch die Sequenzierung zahlreicher Genome gelang ein großer Schritt in Richtung des Verständnis der fundamentalen Elemente der Zelle — der Gene. Im Menschen existieren ungefähr 20 000 bis 25 000 Gene, die mehr als eine Million Proteine codieren. Die Komplexität auf der Ebene der Proteine wird herbeigeführt durch alternatives Spleißen und co- und posttranslationale Modifkationen. Diese produzieren mehrere Proteinspezies pro Transkript. Modifikationen sind in der Regulation der zellulären Prozesse absolut notwendig, und sie verursachen die Aktivierung oder Inaktivierung von Enzymen oder sogar von ganzen Signalwegen.
Die Gesamtheit aller Proteine in einer Zelle, zu einem bestimmten Zeitpunkt und unter bestimmten biologischen Umständen, wird Proteom genannt. Die Analyse des Proteoms wird als Proteomics bezeichnet. Ein Forschungsgebiet der Proteomics ist die Identifikation von Proteinen und ihren posttranslationalen Modifikationen. Peptide Mass Fingerprinting ist eine oft benutzte und erprobte Methode, um mit Hilfe von Massenspektrometrie und einer Proteinsequenz Datenbank Proteine durch ihre Aminosäuresequenz zu identifizieren. Diese Methode vergleicht experimentelle (gemessene) Massenpeaks mit theoretischen Massen, die von einem Protein in der Sequenz Datenbank errechnet werden. Die Masse eines modifizierten Proteins unterscheidet sich von der Masse seines nicht-modifizierten Gegenstücks. Deshalb muss dieser Massenunterschied berücksichtigt werden, wenn Peptide Mass Fingerprinting zur Detektion von Potein Modifikationen angewendet wird.
In der vorliegenden Arbeit wurde ein neuer Algorithmus entwickelt und implemeniert, der in Peptide Mass Fingerprint Daten Proteinmodifikationen identifiziert. Der Algorithmus betrachtet den Prozeß der Vorhersage von Proteinmodifikationen als ein erweitertes Geldwechsel-Problem, das mit bestimmten Kombinationen von Modifikationen versucht, einen Massenunterschied zu erklären. Im Gegensatz zu den existierenden Algorithmen, ist der hier präsentierte Algorithmus nicht in der Anzahl der zu berücksichtigen Modifikationen beschränkt. Desweiteren ist der Algorithmus effizient, da er für eine Liste von Modifikationen nur ein Mal alle Kombi-

nationen der Modifikationen berechnet, unabhängig von der Anzahl der Abfragen. Die Häufigkeiten des Auftretens von Proteinmodifikationen sind allerdings kaum bekannt, so dass keine verlässliche Aussage über die Qualität der Identifikationen getroffen werden kann. Trotzdem ist dieser neue Ansatz ein verheißungsvoller Schritt, um mehr über die Komplexität der Proteine zu erfahren.

Abstract

The living cell is a complex entity consisting of nucleic acids, proteins, and other biomolecules that form an interrelated and dynamic network. The unraveling of this network is of great interest for scientists of different disciplines. With the sequencing of the genome a step was made to the understanding of the fundamental elements of the cells — the genes. In humans, approximately 20,000 to 25,000 genes exist which encode about more than one million proteins. This complexity at the protein level is a result of alternative splicing and co- and post-translational modifications producing several protein species per transcript. Modifications are essential to the regulation of cellular processes and account for the activation or deactivation of enzymes and whole signaling pathways. The entirety of all proteins present in a cell at a fixed point of time and under particular biological conditions is called proteome, and the analysis of it is proteomics. One particular area of interest in proteomics is the identification of proteins and their post-translational modifications. Peptide mass fingerprinting is an established method and has proved useful to identify proteins by their amino acid sequence using mass spectrometry and protein sequence databases. This method relies on the idea of comparing experimental (measured) mass peaks to theoretical (calculated) masses, the latter being generated from a protein in a sequence database. As the mass of a modified protein differs from the mass of its unmodified counterpart, this mass distance is to be considered when detecting protein modifications with peptide mass fingerprinting.
In the work described here, a novel algorithm was developed and implemented that allows for the identification of protein modifications from data derived by peptide mass fingerprinting. The algorithm transformed the process of predicting protein modifications to an extended Money Changing Problem of finding suitable combinations of modifications that explain the observed peak mass distances. Unlike common computational approaches the algorithm presented here will not be restricted in the number of modifications to be considered. Furthermore, this algorithm is efficient by calculating for a given list of modifications the combinations of modifications only once, independent of the number of queries. Although there exist hardly any frequencies of protein modifications, which turns the validation of the results very difficult, this novel approach is a promising step towards the unraveling of protein complexity.

Contents

1 Introduction

1.1 Proteomics

A living cell is a complex system of a multitude of proteins, nucleic acids and other molecules that participate in numerous interrelated molecular processes, forming a dynamic network. The prospect of unraveling this complex network was promoted by the sequencing of the genomes of a broad variety of organisms from bacteria to eukaryotes. Finally, the sequencing efforts reached a major milestone when draft sequences of the complete human genome were published (Venter et al., 2001; Lander et al., 2001). Investigation of these sequences gave rise to an approximate number of 30,000 to 35,000 protein-encoding genes — just about one-third higher than that reported for the simple organism *Caenorhabditis elegans* in 1998. Three years after the publication of the first draft sequences, the International Human Genome Sequencing Consortium revised the former assumption of the number of genes on the basis of an improved version of the original sequences to an even lower approximation of only 20,000 to 25,000. This relatively low number of genes in the human genome is, however, assumed to be complemented by a number of probably more than one million different protein species as a result of alternative splicing and co- and post-translational modifications (Jensen, 2004).

Indeed, the mere number of genes allows no simple assumption of a cell's complexity, which has been discussed by Claverie (2001). Even a simple mechanism like "on" and "off" regulation of genes introduces an enormous combinatorial complexity at the level of transcripts, which is further increased by alternative splicing producing an average of 5 to 6 splicing variants per gene (Jensen, 2004). Furthermore, the complexity at the protein level is even higher because co- and post-translational modifications give rise to several protein species per transcript. These modifications are essential to the regulation of cellular processes and account for the activation or deactivation of enzymes and whole cellular signaling pathways, as in the case of phosphorylation (Leevers and Marshall, 1992; Colgan et al., 1998; Livolsi et al., 2001).

For all the intense interest in genomic research and the insights into gene expression patterns that our knowledge of the genome has generated, we are still left in the dark about exactly where and when all these genes are expressed in the form of proteins. In contrast to the virtually static genome, the proteins present in a cell change constantly in response to environmental factors or developmental stages. Understanding the function of proteins and the molecular processes in which they are involved is essential to understanding their role in the cell. This can only be achieved by direct study of the proteins expressed in a cell, which is the subject of proteomics.

The term "proteome" was coined by Wilkins et al. (1997) to indicate the *PROTEins expressed by a genOME or tissue*. It defines a variable feature of an organism, comprising all proteins that are present in a cell, an organ, or an organism under particular biological conditions and at a fixed point in time. One particular area of interest in proteomics is the identification of proteins and their post-translational modifications (PTMs) that account for much of the biological complexity of the proteome. The combination of protein separation by two-dimensional gel electrophoresis (2-DE) and mass spectrometry (MS) is a well-established method allowing for the identification of post-translationally modified proteins (Jensen, 2004). But, the quantitative and qualitative characterization of PTMs on a large scale still presents a considerable challenge in proteomics (Mann and Jensen, 2003; Jensen, 2004).

1.2 Classical Approach of Protein Identification

The classical workflow of the analysis of expressed proteins in a cell or tissue comprises three major steps (Wilkins et al., 1997):

1. Protein separation
2. Protein analysis
3. Protein identification and characterization

The two classical techniques that contribute to the proteomics workflow are two-dimensional polyacrylamide gel electrophoresis (2D-PAGE) (Klose, 1975; O'Farrell, 1975) for protein separation and identification, and MS for protein identification, and characterization.

1.2.1 Two-Dimensional Polyacrylamide Gel Electrophoresis

The 2D-PAGE technique allows for the simultaneous separation of thousands of proteins (Klose and Kobalz, 1995) in an electric field. In the first dimension, proteins are isoelectrically focused on a polyacrylamide gel, which means that they are separated in a pH gradient until they reach a stationary position — their isoelectrical point (pI). The pI is the pH at which a protein has a net charge of zero. The second dimension separates the proteins according to their molecular weight (m_w). The proteins, already separated in the first dimension, are now separated orthogonally in the presence of sodium dodecyl sulphate (SDS). This anionic detergent binds to the backbone of the zero-net-charge proteins, and as a result, all proteins have a uniform negative charge. Since all the proteins also have the same charge density, they are separated by their m_w during electrophoresis because of the sieving effect of the gel pores.
The result of a 2D-PAGE is a protein map on which proteins are separated according to pI and mass. In order to visualize the proteins on the gel, different techniques are applied. Among these are staining with dyes (e.g. Coomassie Blue), fluorescence (e.g. Sypro Ruby) or chemoluminescence detection, immunohistochemical staining (Western-Blot), or radio-isotope marking (Thébault et al., 2001). After visualization, proteins can be detected as spots on the gel (Fig. 1.1). For the purpose of protein identification, these stained spots, ideally corresponding to isolated proteins, are excised from the gel and further analyzed by MS.

1.2.2 Mass Spectrometry

MS is a method that analyzes molecules of different molecular masses and has become one of the most important techniques used for analysis, characterization and identification of proteins in proteomics (Mann and Pandey, 2001). In principle, a mass spectrometer is an instrument that measures the mass-to-charge ratio (m/z ratio) of free ions under vacuum conditions (Beckey, 1969). A mass spectrometer consists of three essential parts (Fig. 1.2):

1. An ion source in which the analyte is ionized and transforms into the gaseous phase.
2. A mass analyzer separating the ions by m/z ratio.

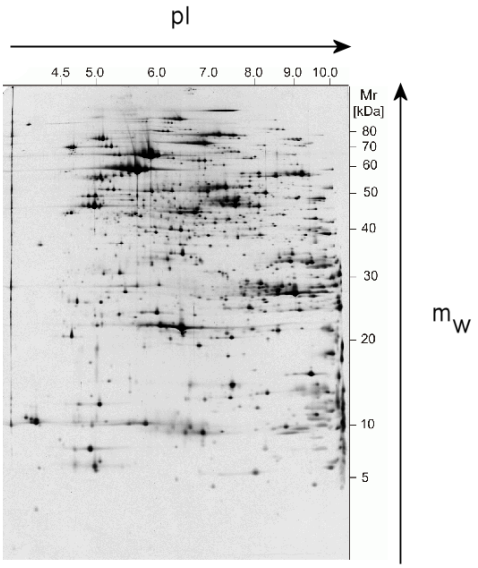

Figure 1.1: Protein map of the intracellular proteins of *Helicobacter pylori* 26695 obtained by 2D-PAGE. The proteins were separated according to pI (x-axis) and molecular weight (y-axis). Subsequently, the spots have been visualized by silver staining. The image has been published in the 2D-PAGE database (Pleissner et al., 2004) at the Max Planck Institute for Infection Biology available at `http://web.mpiib-berlin.mpg.de/cgi-bin/pdbs/2d-page/extern/index.cgi`.

3. A detector which records the charge that is induced when an ion passes by or the current that is produced when it hits a surface.

The sample is introduced into the ion source either as a solid, a liquid, or in the gaseous phase, depending on the sample and the kind of mass spectrometer used. In the case of protein analysis, the proteins are digested by proteolytic enzymes before being introduced to the mass spectrometer. For sample ionization the following methods can be used : bombardment with photons, thermal or electrical energy, molecules, ions, or electrons. The ionization results in a stream of ions leaving the source and then being accelerated in an electric/ magnetic field on their way to the mass analyzer. This field separates the ions by their m/z ratio. Subsequently, the separated ions are directed to the detector. The result of a mass spectrometric measurement is a mass spectrum (Brunneé, 1987) that allows to infer information about which ions, i.e. at which m/z ratio, and in what relative amount these ions were detected. In mass spectrometry, the mass m of ions is typically given in Dalton, that is $\frac{1}{12}$th of the mass of a ^{12}C atom. The charge z is given in the fundamental unit of charge. Although a mass spectrometer typically measures the mass-to-charge ratio of a molecule, this ratio is often loosely referred to as "mass". However, this is only true if an ion carries a single positive charge. Note that a molecule with a monoiso-

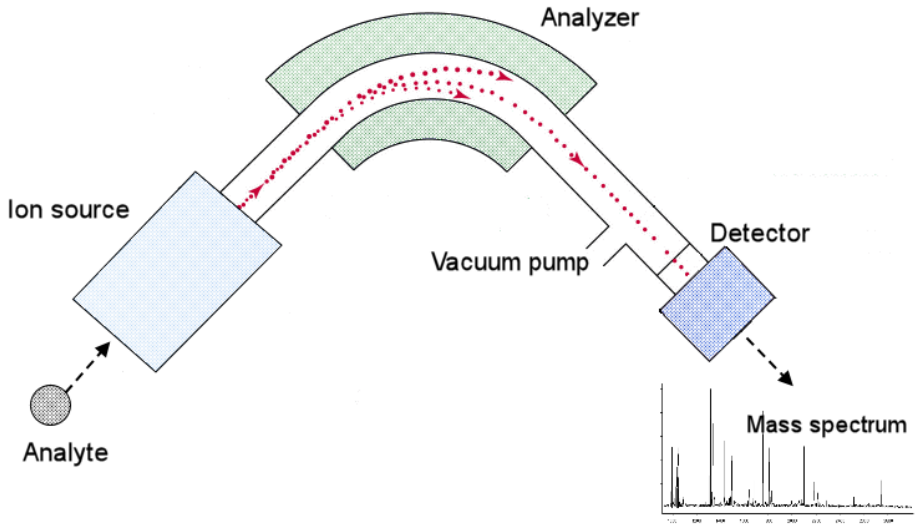

Figure 1.2: Scheme of a typical mass spectrometer. The ion source produces ions (red dots) of the analyte molecules which are accelerated in an electric field. Subsequently, the ions reach the separator, which bends the flight path of the ions in a magnetic or an electric field depending on their mass and charge. Ions that are too heavy or too light are deflected and are thus not detected. The flight paths of all other ions are modulated only slightly, so that they hit the detector and their m/z ratio is recorded.

topic mass of M=2000 Da can, for instance, produce an $[M+ H]^+$ monoisotopic peak at $m/z = 2001$ but also an $[M+ 2H]^{2+}$ monoisotopic peak at $m/z = 1001$.

Several types of mass spectrometers exist which differ in their ionization, their separation, and their ion-detection methods. The most common ionization sources for the analysis of proteins are matrix-assisted laser desorption/ionization (MALDI; Karas and Hillenkamp, 1988) and electron ionization (ESI; Fenn et al., 1989).
The MALDI method requires the sample to be embedded in a crystalline structure of a matrix. For the desorption and ionization of the sample, an ultra-violet or an infrared laser beam is used. The ESI method uses a hypodermic needle through which liquid, containing the analyte, is pumped at a high voltage. This causes the liquid to electrostatically disperse (electrospray), resulting in small droplets which quickly evaporate, passing their charge on to the analyte molecules.
Although many different types of analyzers exist, such as time-of-flight or ion-trap, they all rely on the principle of using magnetic and/or electric fields to deflect the ions and to separate them according to their m/z ratio.
For example, the time-of-flight (TOF) analyzer separates ions, accelerated in an electrostatic field, based on the time it takes them to travel a fixed flight path in a field-free region (flight tube). The traveling time for each ion depends on its m/z ratio.

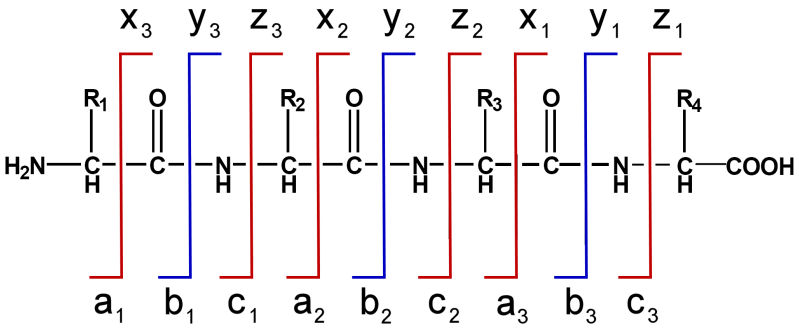

Figure 1.3: MS/MS fragmentation patterns. A peptide backbone can break at NH-CH, CH-CO, and CO-NH bonds, as indicated by red and blue lines. Each type of fragmentation produces one charged and one uncharged fragment. Mass measurement of the charged fragments by MS produces different ion series, and the mass difference between adjacent ions of one series ideally corresponds to the mass of an amino acid which allows to derive the peptide sequence.

The tandem MS (MS/MS) technique extends the previously described MS procedure. A tandem mass spectrometer consists of two analyzers separated by a fragmentation chamber (also called collision cell). This chamber fractions selected ions from the first analyzer by the transfer of energy, usually by collision with an inert gas. These fragments are subsequently separated in the second analyzer. The rationale behind MS/MS lies in the characteristic types of fragmentation that occur when the peptide backbone is broken by collision at the NH-CH, CH-CO, and CO-NH bonds (Roepstorrf and Fohlmann, 1984, Fig. 1.3). Each of these breakages yields one uncharged and one charged fragment, and in an ideal case, each of the bonds of a peptide backbone is destroyed in some portion of the sample molecules. The result are N-terminal ion series (a, b, c ions) and C-terminal ion series (x, y, z ions) that contain all prefixes and all suffixes of the peptide sequence. The most abundant type of fragmentation occurs at CO-NH bonds, corresponding to b and y ions, and the sequence of amino acids can be directly inferred by investigation of the mass differences of adjacent ions of either b or y series. In practice, the MS/MS procedure is far from ideal which is reflected in incomplete ion series and noisy spectra. Thus, the *de novo* interpretation of MS/MS spectra to derive a proteins sequence is complicated.

The major difference between MS and MS/MS spectra lies in information content. MS spectra have more of a compositional character, because they contain information about the masses of peptides, which is determined by a peptide's composition of amino acids. On the other hand, MS/MS spectra have a sequential character, and ideally they can depict the complete sequence of a peptide by the masses of

its fragments. Both methods are efficiently applied to protein and PTM identification, but MS/MS is the more powerful method, since it allows to derive information about the exact position of PTMs in the sequence; and interesting approaches for the study of "unknown" PTMs using alignment of MS/MS spectra have recently been published (Pevzner et al., 2000, 2001; Tsur et al., 2005). However, the interpretation of MS/MS spectra is much more complicated than that of MS spectra used for peptide mass fingerprinting.

1.3 Peptide Mass Fingerprinting

Peptide mass fingerprinting (PMF), first mentioned by Pappin et al. (1993), is one of the foundational methods which has led, and is still fueling, the advance of proteomics (Henzel et al., 2003). It allows identification of proteins by their amino acid sequence using MS and protein sequence databases (DBs; Fig. 1.4).
PMF relies on the generation of a characteristic "mass fingerprint" of a protein. This fingerprint is obtained by digestion of a purified protein with a sequence-specific protease, usually trypsin, and subsequent mass spectrometric analysis of these peptides, typically by MALDI–TOF. A mass fingerprint can be used to identify a protein via its amino acid sequence by matching the experimental peptide mass peaks against a set of calculated, theoretical peptide mass peaks from the entries in a protein sequence DB. As early as 1993, research groups demonstrated that for each entry in a protein sequence DB, or translation of a nucleotide sequence DB, peptide masses could be calculated by the use of the expected cleavage specificity of the enzyme employed in the experiment (James et al., 1993; Pappin et al., 1993; Yates et al., 1993).

There are, however, several prerequisites that must be met to apply PMF successfully. First, it is required that the analyzed sample contains a purified protein. Insufficient separation of sample proteins or, for example, contamination with keratin or protease autolysis products can lead to an adulterated mass fingerprint. Second, the protein under study must be represented in the sequence DB to be identified. Moreover, the interpretation of peptide mass fingerprints is complicated by the suboptimal efficiency of proteases that may cause some of the cleavage sites in a protein to remain uncleaved. Consequently, the set of peptides and subsequently the mass fingerprint will not match the perfectly cleaved theoretical mass fingerprint of the corresponding protein sequence in the DB. However, PMF algorithms

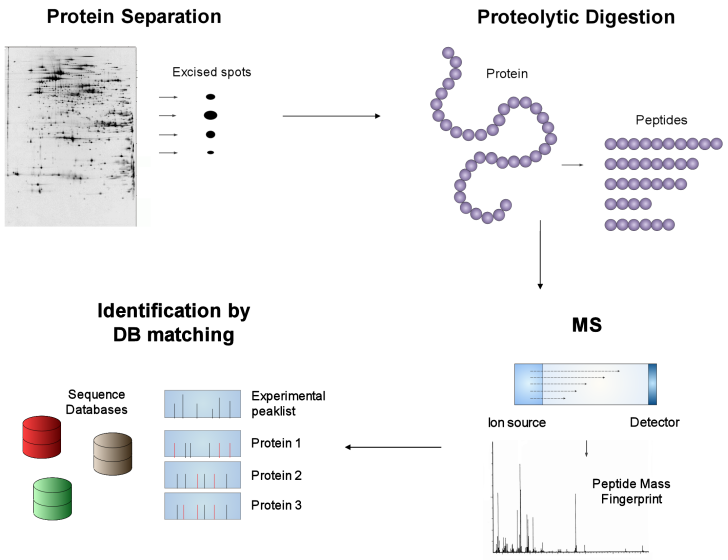

Figure 1.4: Protein identification by peptide mass fingerprinting. The protein sample is separated by 2D-PAGE or other separation methods. Subsequently, the purified proteins are excised and digested with a proteolytic enzyme. The obtained peptides are measured in a mass spectrometer and the resulting peptide mass fingerprint of a protein is used to identify the protein by comparing it with the entries of a sequence DB.

commonly account for this by generating theoretical masses that correspond to peptides with up to a certain number of missed cleavage sites. One of the elementary drawbacks of PMF, however, is the inability to distinguish peptide homologues, i.e. peptides with a different amino acid sequence but with a similar mass. Additionally, masses of peptides that were subject to post-translational modification deviate from the masses of their unmodified forms that are calculated from the DB proteins and cannot be matched directly.

Nevertheless, PMF is a powerful method that proved useful for rapid, large-scale protein identification, because a protein can often be reliably identified by only a subset of its peptide masses. As a result, the identification of proteins is quite robust against irregularities in the experimental mass fingerprints, for instance, by mass changes of peptides due to post-translational modification (Henzel et al., 2003).

1.3.1 Identification of Protein Modifications

The actual protein identification from sequence DBs is usually a matter of pairwise peak comparisons between mass peaks of an experimental mass fingerprint and the

calculated masses in a theoretical mass list (generated from a protein in the DB). Due to the limited measurement accuracy of mass spectrometers, an experimental mass m_E and a theoretical mass m_T are considered equal if they are not further apart from each other than a tolerance ϵ:

$$|m_E - m_T| \leq \epsilon$$

Using the above definition, the similarity of two masslists can be described by the shared peaks count (spc), which is defined as the number of experimental mass peaks that match a theoretical mass. Current PMF algorithms, such as Mascot (Perkins et al., 1999) or ProFound (Zhang and Chait, 2000), actually use a more sophisticated strategy that assigns probability-based scores to each single match, and therefore produces pairs of matching masses of different quality. The total similarity of a mass fingerprint and a protein from the DB is then given by the combined scores of all matches. Nevertheless, the basic idea of matching is the same and works well in the case of unmodified proteins. It fails, however, if a peptide is modified, i.e., one or more of its amino acids were subject to a gain or loss of atoms.
Usually, a protein modification δ is reflected in a measurable mass change denoted by $m(\delta) \mapsto \mathbb{R}$, which is the sum of the masses of all newly introduced atoms minus the sum of the masses of all removed atoms. Consequently, the distance between the masses of a peptide in its modified and its unmodified form is given by $M_\Delta = \sum_i m(\delta_i)$, the sum of the mass changes of each modification occurring on that peptide. Taking this into consideration, an experimental mass peak and a theoretical peptide mass match if their mass difference $m_E - m_T$ is equal to an M_Δ produced by any combination of modifications that can occur on the peptide:

$$|m_E - m_T - M_\Delta| \leq \epsilon.$$

The usual procedure of DB search algorithms is to generate all masses for a theoretical peptide that might occur using all the different combinations of variable modifications, i.e. modifications that may or may not be present (Fig. 1.5). Subsequently, the resulting extended list of theoretical masses is compared to the experimental mass fingerprint, and even if only a few variable modifications are considered, the number of theoretical masses becomes quite large. Matthiesen et al. (2005) have described x, the number of possibilities to modify a peptide with a length of l residues, by

$$x = \prod_{i=1}^{l} (V_i + 1),$$

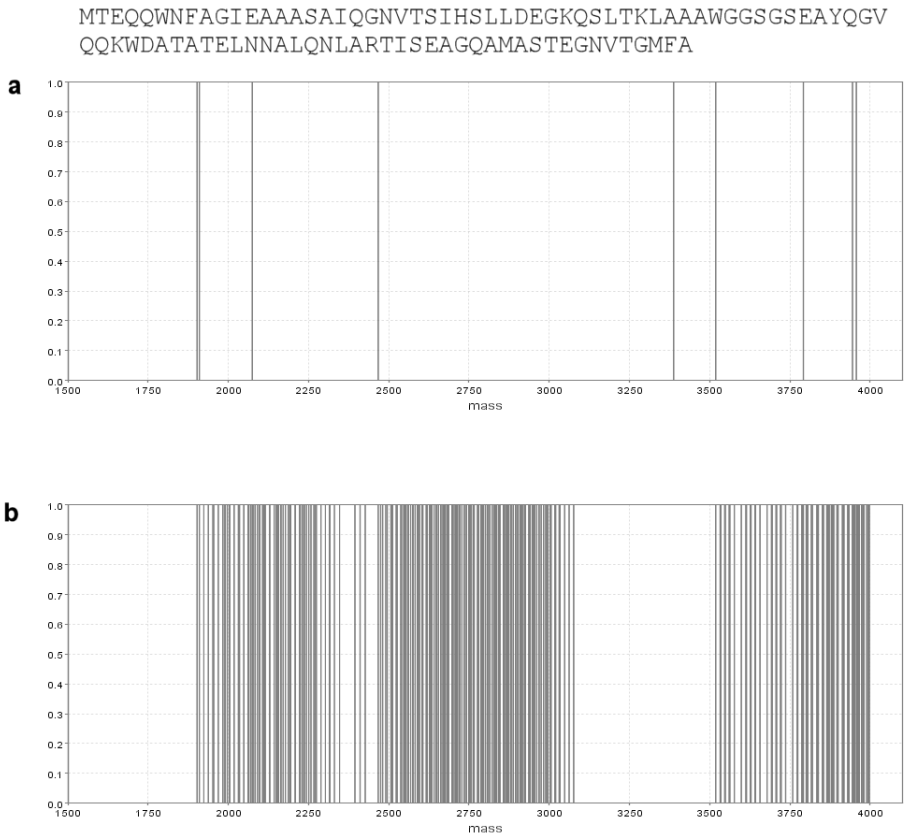

Figure 1.5: Combinatorial explosion introduced by variable modifications. Two theoretical mass fingerprints of the sequence shown on top of the figure. The digestion was done *in silico* using trypsin cleavage rules. Peptides were calculated with a maximum of 2 missed cleavages, a length of at least 4 residues, and a mass between 800 Da and 4000 Da. **a)** 9 theoretical masses, calculated without variable modifications. **b)** 338 theoretical masses, calculated with a few modifications: oxidation of M; phosphorylation of S, T and Y; sulphation of Y; nitration of Y; carbamidomethylation of C; carboxymethylation of C; methylation of K; and acetylation of K.

where V_i is the number of different modifications that may occur on the i-th residue. Clearly, it is infeasible to iterate over the x possible combinations of modifications for each peak–peak comparison, especially if many modifications are considered. Thus, it is a common procedure in PMF algorithms to restrict either the number of modifications that is allowed on a peptide, or to consider only a few modifications in order to avoid the combinatorial explosion.

1.3.2 Existing Tools for the Detection of Protein Modifications

Apart from the common PMF algorithms that carry out protein identification by searching sequence DBs, there are two popular tools, FindMod (Wilkins et al., 1999)

and FindPept (Gattiker et al., 2002), that detect variable PTMs or single amino-acid substitutions. These tools compare an experimental mass fingerprint with a single theoretical mass list, calculated from the expected protein identification. The rationale is to increase the number of mass assignments between the experimental and theoretical mass list by allowing a large number of modifications. The DB search algorithms have to match an experimental mass fingerprint against a large set of proteins in a DB, so that it is infeasible to include high numbers of modifications in a search. However, obtaining correct identifications is often still possible.
Both FindMod and FindPept are available at the Expasy[1,2] server. The former performs PTM detection assuming sequence-specific cleavage, whereas the latter assumes unspecific cleavage of a protein. The detection of PTMs is done by simply iterating over all possibilities to modify a peptide, and even though these tools compare an experimental mass fingerprint with only a single theoretical mass list, the number of modifications that is allowed in a search is still limited. FindMod, for instance, includes about 60 modifications in a search, if at most two modifications per peptide are allowed, and only 12, if at most three modifications per peptide are considered. Additionally, it finds single amino-acid substitutions.

There are data sources that provide highly accurate mass changes for a large number of modifications that have been observed on proteins. For example, the current release of the Unimod DB of protein modifications for mass spectrometry[3] provides a compilation of 508 different modifications[4] including PTMs, single amino-acid substitutions or chemical modifications introduced by experimental techniques. Nevertheless, none of the PTM detection approaches allows for an exhaustive analysis of protein modifications using PMF data by taking into account all known modifications.

1.4 Goals of this study

PTMs influence the function of proteins and account for a large part of the complexity of the proteome (Jensen, 2004). PMF allows investigation and identification of protein modifications. However, common computational approaches for protein identification (e.g. Mascot) or PTM detection (e.g. FindMod) are restricted in the

[1] http://expasy.org/tools/findpept.html
[2] http://expasy.org/tools/findmod/
[3] http://www.unimod.org
[4] The Unimod compilation of modifications is also used by Mascot.

number of modifications that they can consider to avoid combinatorial explosion for the comparison of masses.
This thesis will present a novel algorithmic approach to detect PTMs or other protein modifications by comparing an experimental and a theoretical mass fingerprint. The problem of PTM detection will be discussed from a combinatorial point of view, by investigating its relationship to a simple "Money-Changing Problem". Currently, there exist hardly any comprehensive data about the frequencies of protein modifications, making validation of results especially difficult. Therefore, the main goal of this study will be computational efficiency, allowing for an exhaustive search of protein modifications in PMF data.

2 Algorithm

2.1 Protein Modification as a Money Changing Problem

In the mass spectrometric context, the modification of proteins can be defined as a set of events that lead to a measurable mass difference M_Δ between a protein and its unmodified counterpart. Assume that there is a set of k different modifications $\{\delta_1, ..., \delta_k\}$ for which the single mass changes $A = \{m(\delta_1), ..., m(\delta_k)\} = \{a_1, ..., a_k\}$ are given by a mass function $m(\delta) \mapsto \mathbb{R}$. Then, any mass difference M_Δ that is truly the result of one or more modifications can be expressed as a k-tuple with non-negative integer entries, where the i-th entry denotes the number of modifications δ_i. In the following, each k-tuple will be referred to as a decomposition or compomer $c := (c_1, ..., c_k) \in (\mathbb{N}_0)^k$ of M_Δ over A, its length by $|c| = \sum_{i=1}^{k} c_i$, and its mass, reflecting the mass change introduced to a protein, by $m(c) = \sum_{i=1}^{k} c_i a_i$. Finally, the "empty compomer" denotes the compomer with length $|c| = 0$ and mass $m(c) = 0$. The concept of compomers has been introduced in a different context to describe the composition of biomolecules over weighted alphabets of smaller building blocks like amino acids or nucleotides (Böcker, 2004; Böcker and Liptak, 2004). However, it can also be used to describe the composition of modifications that results in a mass change M_Δ of a protein, and it can even be used to describe any composition of numbers or things that can be assigned a number. For the sake of clarity, the modifications are hereafter referred to only by their mass changes $A = \{a_1, ..., a_k\}$.

The above definitions oversimplify the "real" situation because modifications cannot occur arbitrarily at each residue of a protein, and consequently not all compomers with a mass $m(c) = M_\Delta$ make biological sense. The modification of a protein is a chemical process that leads to changes in the structure and atomic composition of amino acids, and is restricted to specific residues at specific locations in a protein. The following categories of modifications can be distinguished (Perkins et al., 1999):

1. Modifications which affect a residue anywhere in a protein or peptide.

Table 2.1: Descriptors of the specificity of protein modifications based on Unimod. A complete specificity definition comprises one descriptor from the site domain and one from the position domain.

Descriptor	Domain	Description
Amino acid	Site	One of the 20 amino acids.
C-term	Site	Any C-terminal amino acid.
N-term	Site	Any N-terminal amino acid.
Anywhere	Position	Residue can be anywhere in the sequence.
C-term	Position	Residue must be C-terminal of a protein or peptide.
N-term	Position	Residue must be N-terminal of a protein or peptide.
Protein C-term	Position	Residue must be C-terminal of a protein.
Protein N-term	Position	Residue must be N-terminal of a protein.

2. Modifications which affect a protein or peptide terminus, independent of the terminal residue.
3. Modifications which affect a specific residue that is located at a protein or peptide terminus.

Therefore, the specificity of any single modification can be characterized by (i) a site descriptor, defining what can be modified, and (ii) a position descriptor, defining where the modification can occur. Both site and position classifiers are chosen from a controlled vocabulary (Table 2.1) in accordance to the Unimod DB, and each modification is assigned at least one pair of site and position definitions. As an example, Unimod describes two particular instances of deamidation as pairs (R, "Anywhere") and (F, "Protein N-term"). The former declares that it can occur on arginine residues anywhere in a protein sequence, the latter that it can also occur on phenylalanine residues located at the N-terminal end of a protein.

The problem that arises in protein identification is to find out whether a theoretical reference mass m_T of an unmodified peptide and an experimentally observed mass m_E (of which the underlying peptide is unknown) essentially represent the same peptides that differ only by some number of modifications giving rise to a mass distance $m_E - m_T = M_\Delta$. The set of all compomers that explain this mass distance is defined as $C(M_\Delta) := \{c \mid m(c) = M_\Delta\}$ over A and the total number of decompositions is given by the cardinality of $C(M_\Delta)$, which will in the following be denoted $\gamma(M_\Delta) := |C(M_\Delta)|$. Finally, M_Δ is called decomposable over A if at least one compomer has a mass equal to the observed mass change, i.e. $\gamma(M_\Delta) \geq 1$. According to these definitions the problem of protein identification with modifications can be formulated as finding $C(M_\Delta)$, i.e. finding all compomers with $m(c) = M_\Delta$ which

corresponds to a changing problem. The search space $(\mathbb{N}_0)^k$, however, is huge if many modifications are considered, even though the compomers are restricted by the theoretical peptide sequence and the specificities of the modifications. This combinatorial complexity prevents direct iteration over all possible ways to combine the k possible modifications. To find a way to compute $C(M_\Delta)$ it is useful to investigate the simpler Money Changing Problem (MCP).

Problem 2.1 (Money Changing Problem)**.** Given a positive integer M and k positive integers $A = \{a_1, ..., a_k\}$, are there non-negative integers c_i such that $\sum_{i=1}^{k} a_i c_i = M$?

The MCP is NP-hard (Lueker, 1975) if A and M vary, but it is possible to compute a solution in pseudo-polynomial time using a dynamic programming (DP) algorithm (Martello and Toth, 1990). A variation of this algorithm has been sketched by Böcker and Liptak (2004). It can be used to solve the MCP and a set of related integer decomposition problems: the "Find-One" problem asking for one way to represent M as the sum of positive integers c_i, and the "Find-All" problem asking for all solutions.
Unfortunately, the algorithms can not be applied directly for the problem of identifying protein modification compomers given a mass difference M_Δ for two reasons: First, a modification can reduce or increase the mass of a protein or peptide; and an appropriate algorithm has to deal with positive and negative a_i and M_Δ. Second, the compomers correspond to distinct combinations of modifications, and at least some compomers with $m(c) = M_\Delta$ will violate the specificity constraints. The decision of which number and which combinations of modifications can be attached to a peptide sequence is a combinatorial problem and makes the determination of correct compomers in a biological sense even more intricate.

To anticipate a little: it is possible to adapt the idea of the DP approaches for the MCP and derive a new DP algorithm that is suitable to compute modification compomers for a peak distance M_Δ. Due to complexity reasons, the DP algorithm is defined for integers (like the DP algorithm for the MCP). The abstraction of the problem to integers is straightforward: Typically, the masses obtained in a mass spectrometric measurement are real numbers that are accurate up to a decimal place that depends on the measurement process. Therefore, they can be treated as integers that capture a certain precision of the real mass if multiplied by a precision $p \in \mathbb{R}$. The conversion can be calculated for any mass value m by $m^* = \text{round}(\frac{1}{p}m)$. For example, $p = 0.1$ takes into account the first decimal place and the tendency of the second decimal place of a mass value. By treating all mass values as integers the

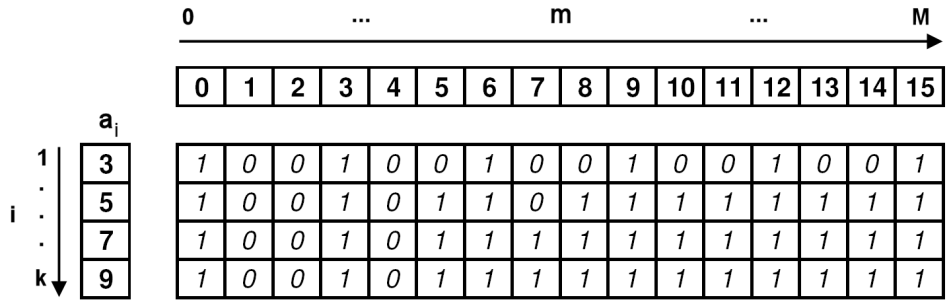

Figure 2.1: Example Boolean table of the original DP algorithm. The table B has been constructed for a query $M = 15$ and the set of input integers $A = \{3, 5, 7, 9\}$. i corresponds to the row indices and m to the column indices $0 \leq m \leq M$. A value $B_{i,m} = 1$ denotes m can be decomposed over $\{a_1, ..., a_i\}$.

"Protein Modification Money Changing Problem" (PMMCP) can be formulated as an extended MCP in the following way:

Problem 2.2 (Protein Modification Money Changing Problem)**.** Given an integer mass distance M_Δ and a set of k modification integer mass changes $A = \{a_1, ..., a_k\}$, are there non-negative integers c_i such that $\sum_{i=1}^{k} a_i c_i = M_\Delta$ and all c_i fulfill the modification constraints?

The next section gives a short introduction to the variation of the original DP algorithm solving the MCP. This algorithm is used as a basis to discuss the extended DP algorithm that solves the PMMCP in section 2.3.

2.2 Dynamic Programming for the Money Changing Problem

The DP algorithm sketched by Böcker and Liptak (2004) computes the decompositions of an integer query M over a set of integers $A = \{a_1, ..., a_k\}$ in two steps. First, it constructs a Boolean table B with rows $i = 1, ..., k$ and columns $m = 0, ..., M$ that stores for each integer $0 \leq m \leq M$ whether it is decomposable. Subsequently, B allows to solve the MCP and the related integer-decomposition problems for M by a simple backtracking procedure.

Construction of the DP Table

The DP algorithm relies on two simple considerations about the sums of positive integers that are used to recursively define whether a positive integer is decomposable:

1. $A = \{a_1\}$. Clearly, explaining one integer (m) by the sum of another integer (a_1) is only possible if m is a multiple of a_1. Thus, m is decomposable if and only if $m \bmod a_1 = 0$.
2. $A = \{a_1, ..., a_i\},\ i > 1$. It is possible to explain an integer m by the sum of other positive integers A in two cases: Either m is the sum of a combination of multiples of the integers $\{a_1, ..., a_{i-1}\}$ and a_i is not part of the sum, or $m - a_i$ is the sum of a combination of multiples of the integers $\{a_1, ..., a_i\}$, and m can therefore be simply explained by adding another a_i.

On the basis of these rules, the algorithm constructs B (Fig. 2.1) up to the query M, such that

$$B_{i,m} = 1 \quad \Leftrightarrow \quad m \text{ is decomposable over } \{a_1, ..., a_i\}.$$

B is initialized by setting $B_{i,0} = 1$ for all $1 \leq i \leq k$ in the first column, and $B_{1,m} = 1$ if and only if $m \bmod a_1 = 0$ in the first row. For $i, m > 1$ the table is computed recursively:

$$B_{i,m} = \begin{cases} B_{i-1,m} & \text{if } m < a_i, \\ B_{i-1,m} \vee B_{i,m-a_i} & \text{otherwise.} \end{cases}$$

The time and memory to construct B are $\mathcal{O}(kM)$. Even though it can be used to solve the decomposition problems for all queries from 1 to M without the need for re-computation, this may be impractical for large k or M. To overcome this obstacle, Böcker and Liptak (2004) have presented a sophisticated DP algorithm that constructs a different data structure called an extended residue table. It can be constructed independent of the query M using $\mathcal{O}(ka_1)$ runtime and memory costs.

Backtracking

In a second step, B allows to find the decompositions of the query M over $\{a_1, ..., a_k\}$ by straightforward recursive backtracking. Each decomposition corresponds to a path in B that contains only values $B_{i,m} = 1$, starting at $B_{k,M}$ and ending in any cell $B_{i,0}$, $1 \leq i \leq k$. The algorithm begins with a single lookup of $B_{k,M}$ to check if M can be decomposed ($Bk, M = 1$). If this is the case, it tries to add an integer a_k to

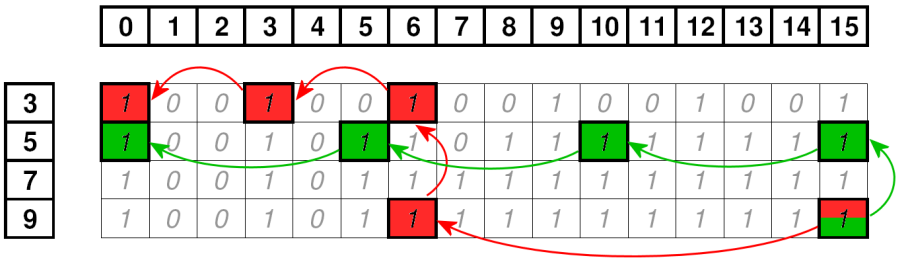

Figure 2.2: Example of backtracking of the original DP algorithm. Two possible backtracking paths of the table B constructed for $A = \{3,5,7,9\}$ and $M = 15$ are highlighted. Green: Path corresponding to the decomposition $5+5+5$. Red: Path corresponding to the decomposition $9+3+3$.

the solution by "going left" to $B_{k,m-a_k}$, or to skip a_k by "going up" to row $B_{k-1,m}$. In both cases, the algorithm checks whether the newly reached cells have a value of 1 and repeats the procedure of path extension to $B_{i-1,m}$ and $B_{i,m-a_i}$ for each $B_{i,m} = 1$ that is visited. One decomposition run is completed, if a path reaches column 0. Fig. 2.2 shows an example of the backtracking procedure in a table B constructed for $A = 3,5,7,9$ and $M = 15$. Two possible backtracking paths corresponding to the decompositions $15 = 9+3+3$ and $15 = 5+5+5$ are highlighted. Note that the backtracking algorithm produces the same results whatever order of $\{a_1,...,a_k\}$ is used for the construction of B because addition is a commutative operation and eventually all integers A will be considered in row k.
By backtracking or single lookups the algorithm can compute the solution of several decomposition problems with the following costs:

- **Existence:** Whether M is decomposable over A can be evaluated in constant time by a single lookup of $B_{k,M}$.
- **Find-One:** Finding any decomposition of M over A requires at most $k + \frac{M}{\min(A)}$ steps. Consequently, the worst-case time to find a single decomposition is $\mathcal{O}(k + \frac{1}{\min(A)}M)$.
- **Find-All:** The time to compute all decompositions of M over A by backtracking is the number of decompositions ($\gamma(M)$ in compomer notation) times the worst-case cost of computing one decomposition. Consequently, all witnesses can be computed in $\mathcal{O}(\gamma(M)(k + \frac{1}{\min(A)}M))$ time.

As indicated by Böcker and Liptak (2004), the Existence and Find-One problems can also be solved with a Boolean vector that is constructed up to M in construction time $\mathcal{O}(kM)$. The number of decompositions can be obtained in the last row of a table of

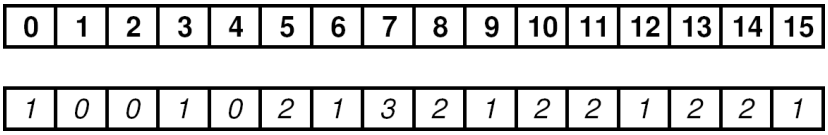

Figure 2.3: Compressed vector representation of the Boolean table of the original DP algorithm. The vector F is constructed for the query $M = 15$ and integers $A = \{3, 5, 7, 9\}$. A value F_m denotes the row index of the first appearance of a "1" in column m of the original DP table.

integers E with rows $i = 1, ..., k$ and columns $m = 0, ..., M$. This table is initialized by $E_{i,0} = 1$ and afterwards computed recursively by $E_{i,m} = E_{i-1,m} + E_{i,m-a_i}$.
It is worth adding to Böcker and Liptak's suggestions that the Find-All problem can as well be solved by constructing a vector of integers F up to a query M in worst-case runtime $\mathcal{O}(kM)$. F_m stores the smallest i such that m is decomposable over $\{a_1, ..., a_i\}$, or 0 if no such i exists (Fig. 2.3), and consequently

$$F_m > 0 \quad \Leftrightarrow \quad m \text{ is decomposable over } \{a_1, ..., a_i\} \text{ for all } i \geq F_m.$$

F is a compressed representation of B which is correct because if m is decomposable over $\{a_1, ..., a_i\}$, it immediately follows that m is also decomposable over $\{a_1, ..., a_j\}$ for all i, j with $i < j \leq k$.
F is constructed by initializing $F_0 = 1$ and $F_m = 0$ for $1 \leq m < \min(A)$. For $m \geq min(A)$, F_m is computed recursively:

$$F_m = \min \{i \in \{1, 2, ..., k\} \mid m \geq a_i \wedge 0 < F_{m-a_i} \leq i\}.$$

2.3 Dynamic Programming for the Protein Modification Money Changing Problem

The DP algorithm that computes the PMMCP proceeds similarly to the simple DP approach presented for the MCP: First, it recursively constructs a DP table D by defining $D_{i,m}$ via previously computed results $D_{i-1,m}$ and $D_{i,m-a_i}$. In a second step D is used to compute the decompositions for a query M_Δ. Similarly to the Boolean table B of the original algorithm the table D stores in $D_{i,m}$ whether a mass m can be decomposed over $\{a_1, ..., a_i\}$. However, due to the greater complexity of the

PMMCP D stores non-Boolean values and must be computed in a different, more specific way in contrast to the table B of the original algorithm. The new DP procedure and the required preconditions are specified in the following.

2.3.1 Constraints on the use of Protein Modifications

In section 2.1 a simple model of the specificity of protein modifications has been introduced that restricts the set $C(M_\Delta)$ to those compomers that correspond to combinations of modifications that might occur on a given protein sequence. To recapitulate briefly: The specificity of each modification a_i is defined as a set of rules that define the residues and positions in a protein that can be subject to that modification. The specificities, however, are not unique to a single modification so that an amino acid might be modified differently.
To emphasize the influence of the specificity constraint on the DP algorithm, consider how to decide whether single modifications can be added to a protein or peptide sequence one after another: Adding any one modification a_i to an unmodified sequence is easy because it can be placed arbitrarily on any residue that matches its specificity rules. Whether additional modifications can be added to the sequence, however, depends on the way other modifications have been distributed among the residues beforehand. In the worst-case it may be impossible to add a new modification or a change in the position of one or more already attached modifications may be required. Either way, all possibilities to distribute the previously attached modifications to the sequence have to be tested before a_i is accepted or rejected.
Obviously, the PMMCP does not exhibit optimal substructure if the modifications cannot be used arbitrarily to decompose a mass change m because the optimal choice of placing a modification on a sequence depends on the subsequent modifications. As a result, the DP approach would become useless because it relies on the optimal substructure of the problem to define further stages of the solution by previously computed (optimal) solutions. Consequently, the only way to integrate the specificity constraints to restrict $C(M_\Delta)$ is in the backtracking procedure by filtering out compomers that do not match the specificity constraints of a given sequence.
To define the DP algorithm for positive and negative modifications a_i in a subsequent step, however, it is necessary to place some kind of restriction on the compomers. Fortunately, it is no problem to define a global upper bound $|c| \leq n$ for the length of the compomers, and to define the table D recursively such that $D_{i,m}$ stores information whether m can be decomposed over $\{a_1, ..., a_i\}$ using at most n modifi-

cations. Put another way, n is an upper bound for the number of modifications that are allowed on a protein or peptide at a time.

The value of $D_{i,m}$ is defined recursively by $D_{i-1,m}$ and $D_{i,m-a_i}$. It is not necessary to consider n, if the former defines the outcome of $D_{i,m}$ because no additional modification a_i is required to decompose a mass distance m. In the latter case, one a_i has to be used to extend any decomposition of $m - a_i$, and the algorithm has to decide whether an extension is possible if $|c| \leq n$. It is sufficient for the algorithm to know whether there is any compomer c over $\{a_1, ..., a_i\}$ with $m(c) = m - a_i$ and $|c| < n$, or else whether the length of the shortest compomer that decomposes $m - a_i$ over $\{a_1, ..., a_i\}$ is smaller than n, which would guarantee that at least one compomer can be extended.
In general, the algorithm has to keep track of the lengths of the shortest compomers for every mass distance m that is represented in D. Using an additional integer vector T with a length according to the number of columns in D, it is possible to store in T_m the current minimum for each m in the table. If no decomposition of m exists, set $T_m = \infty$.
The computation of D has to be carried out row by row to ensure that T_m contains the minima for the modifications $\{a_1, ..., a_{i-1}\}$ from the previous row, while T_{m-a_i} contains those computed in the current row using the modifications $\{a_1, ..., a_i\}$. T is updated each time a cell $D_{i,m}$ is computed and is constructed in the following way: First, set $T_0 = 0$ because $m = 0$ can always be decomposed by the empty compomer with $|c| = 0$, and for $m \geq 1$ set $T_m = \infty$ by convention. Subsequently, D can be constructed row by row by DP and a cell $D_{i,m}$ is set to decomposable if $T_m = \min(T_{m-a_i} + 1, T_m) \leq n$.

2.3.2 Negative Mass Changes

The upper bound $|c| \leq n$ for the length of the compomers defines an upper bound $M_{max} = \max(A)n$ and a lower bound $M_{min} = \min(A)n$ for the mass distance that can be explained by sums of the modifications A such that any query $M_\Delta < M_{min}$ or $M_\Delta > M_{max}$ is not decomposable. If A contains both negative and positive a_i, then $M_{min} < 0$ and $M_{max} > 0$, and the table D must comprise a range of masses including $m = 0$. The query M_Δ, however, is either positive or negative, such that D only has to be constructed for either one of the cases. Assume without loss of generality that $M_\Delta > 0$ because every decomposition of a negative M_Δ can be also obtained

by changing the sign of M_Δ and correspondingly of all a_i prior to computation. In order to build an appropriate table D for positive and negative a_i that considers all possible decompositions of M_Δ two prerequisites must be met:

1. The number $\gamma(M_\Delta)$ of compomers that decompose M_Δ must be finite. Consider the case that the partial mass of at least one compomer c with $m(c) = M_\Delta$ equals 0, then those entries c_i that contribute to the partial mass and consequently $|c|$ can become infinitely large while $m(c)$ remains constant. Each of the so obtained compomers c adds to the set $C(M_\Delta)$ and computation of the table D would become impossible because infinitely many columns would have to be constructed to include all solutions. By way of example, think of $8 = 4 + 4$ but also $8 = 4 + 4 + 4 - 2 - 2$ and so forth. In section 2.3.1 a restriction has been presented that defines an upper bound $|c| \leq n$ and thus directly restricts $\gamma(M_\Delta)$.
2. All compomers must be represented in the table D. Clearly, one cannot explain a positive number merely by the sum of negative numbers, meaning that the computation of D must not start with $a_1 < 0$. For all subsequent computations, the algorithm proceeds row by row extending $\{a_1, ..., a_{i-1}\}$ by one modification a_i, and only masses m which are decomposable over $\{a_1, ..., a_{i-1}\}$ or $m = 0$ can be used to explain any subsequent m that is to be decomposed over $\{a_1, ..., a_i\}$. The masses m that can be decomposed using positive and negative a_i therefore depend on the decompositions using positive a_i. Furthermore, the algorithm must fill the table D up to M_{max} to make sure that all compomers including negative a_i are represented. To process the negative a_i after all positive modification mass changes, A has to be sorted according to sign such that $i < j$ if $a_i > 0 \wedge a_j < 0$.

Although the above prerequisites allow construction of D as a Boolean table, this is actually not very clever. The reason lies in the negative mass changes a_i itself that allow to take detours during backtracking. Detours which lead to paths corresponding to compomers with $|c| > n$ cannot be avoided and are not recognized until reaching the n-th cell of a backtracking path if Boolean storage is used (Fig. 2.4a). Instead, if the algorithms knows for each $D_{i,m}$ by how many modifications a compomer must at least be extended to create a complete compomer when adding a_i, detours can be avoided. If no valid compomer exists, the number of required extensions is defined as infinite (Fig. 2.4b). The number of modifications that must at least be added to a compomer when using an a_i can be obtained for each m during the construction of D from the vector of minima T that has been introduced in

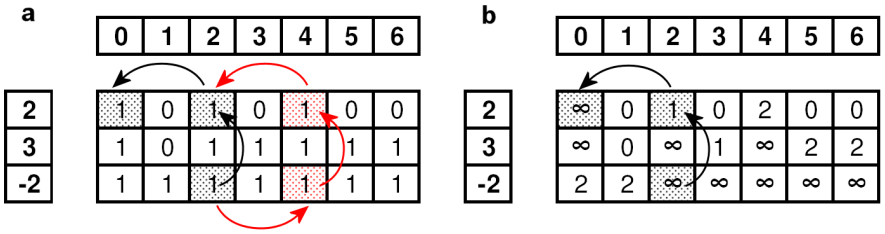

Figure 2.4: Influence of different storage schemes on the backtracking behavior of the new DP algorithm. Both matrices have been constructed for the input modification mass changes $A = \{2, 3, -2\}$ and an upper bound of $|c| \leq n = 2$. **a)** Boolean table. Using a negative a_i (red arrows) can lead to wrong backtracking paths ($|c| > n$). **b)** Integer table. Detours leading to paths corresponding to compomers with $|c| > n$ are avoided by storing the lowest number of extensions that are required to include an a_i in a valid compomer ($|c| \leq n$), or infinity if no such compomer exists.

section 2.3.1. Additionally, because $m - a_i > m$ if $a_i < 0$, rows that correspond to negative mass changes have to be iterated over starting at $m = M_{max}$ down to $m = 0$ to consider the previously computed results correctly. The whole construction process of the integer table D by DP is presented in more detail in section 2.3.3 including the recursive definitions.

2.3.3 Definition of the Algorithm

The new DP algorithm requires the following input: An upper bound $|c| \leq n$, the modification mass changes $A = \{a_1, ..., a_k\}$ and the peak distance M_Δ from which $M_{max} = max(A)n$ is calculated. Without loss of generality the algorithm is defined for $M_\Delta > 0$ as detailed in section 2.3.2. It solves the PMMCP by first constructing an integer table D with rows $i = 1, .., k$ and columns $m = 0, ..., M_{max}$, and it subsequently uses this table to find the decompositions of M_Δ over A by backtracking.

Construction of the Table

Prior to construction, A is sorted by sign such that $i < j$ if $a_i > 0 \wedge a_j < 0$. Subsequently, the integer table D is constructed so that $D_{i,m}$ stores the minimal number of modifications needed to decompose m over $\{a_1, ..., a_i\}$ if one a_i is used, or 0 if m is not decomposable over $a_1, ..., a_i$. If m cannot be decomposed by extending $m - a_i$ but is already decomposable over $\{a_1, ..., a_{i-1}\}$, define $D_{i,m} := \infty$, and consequently

$$D_{i,m} > 0 \quad \Leftrightarrow \quad m \text{ is decomposable over } \{a_1, ..., a_i\}.$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	∞	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	∞	0	∞	0	∞	1	0	2	0	0	2	0	0	0	0	0	0	0	0
9	∞	0	∞	0	∞	∞	0	∞	0	1	∞	2	0	0	2	0	0	0	2
-2	2	0	∞	2	∞	∞	0	2	0	∞	∞	∞	0	0	∞	0	0	0	∞

Figure 2.5: Example integer table constructed according to the new DP algorithm. The table D has been constructed for the modification mass changes $A = \{2, 5, 9, -2\}$, an upper bound $n = 2$, and applying a computational precision $p = 1$. Non-zero cells are highlighted.

D (Fig. 2.5) is constructed row by row up to mass M_{max} to allow for negative a_i and the upper bound n: Rows are computed from left to right iterating over $m = 1, 2, ..., M_{max}$ if $a_i > 0$, and from right to left iterating over $m = M_{max}, M_{max} - 1, ..., 0$ if $a_i < 0$. An auxiliary vector T is also constructed up to M_{max} in parallel so that for the computation of row D_i, T_m stores the minimal number of modifications needed to decompose m over $\{a_1, ..., a_i\}$ if m has already been computed in that row, or else the according minimum over $\{a_1, ..., a_{i-1}\}$. T is initialized by setting $T_0 = 0$, and for $m \geq 1$ setting $T_m = \infty$. For $1 \leq i \leq k$, D is computed recursively:

$$D_{i,m} = \begin{cases} T_{m-a_i} + 1 & \text{if } m \geq a_i \wedge T_{m-a_i} < n, \\ \infty & \text{if } T_m \leq n \wedge (m < a_i \vee T_{m-a_i} \geq n), \\ 0 & \text{otherwise.} \end{cases}$$

The vector T, storing the column minima, is updated each time a value $D_{i,m}$ has been computed by

$$T_m = \min(T_m, T_{m-a_i} + 1).$$

D can be constructed in runtime and memory $\mathcal{O}(kM_{max})$, which is essentially the same as for the original DP algorithm. It can be used to solve the PMMCP for all mass changes $0 \leq M_\Delta \leq M_{max}$.

Backtracking

The backtracking procedure is similar to that of the original DP algorithm presented in section 2.2. According to the updated definition of D, each compomer corresponds to a path in D that contains only values $D_{i,m} > 0$ and has a length of at most n, starting at D_{k,M_Δ} and ending in a cell $D_{i,0}$ of the first column. Whether a path

can be extended by the use of a_i is evaluated by comparing the current path length l with the value $D_{i,m}$: If $D_{i,m} > 0$ and $n - l \leq D_{i,m}$, at least one path exists that uses $D_{i,m-a_i}$ and corresponds to a compomer with $|c| \leq n$ and $m(c) = M_\Delta$.
The algorithm begins with a single lookup of D_{k,M_Δ} to check if M can be decomposed. If $D_{k,M_\Delta} > 0$, it tries to add an integer a_k to the solution by going left to $D_{k,M_\Delta - a_k}$, or to skip a_k by going up to row $D_{k-1,m}$. The algorithm checks in both directions if the cells are set to decomposable and repeats the procedure of path extension to $D_{i-1,m} \ \wedge \ D_{i,m-a_i}$ for each $D_{i,m} > 0$ that is visited. One decomposition run is completed, if a path reaches column 0. The different costs to solve the relevant integer decomposition problems by backtracking are:

- **Existence:** Whether M_Δ is decomposable over $\{a_1, ..., a_k\}$ can be evaluated in constant time by a single lookup of D_{k,M_Δ}.
- **Find-One:** Finding one decomposition of M_Δ over $\{a_1, ..., a_k\}$ has a worst-case running time of $\mathcal{O}(k + n)$. If the highest number of specificity definitions per modification is s, it takes additional $\mathcal{O}(s^n)$ time to decide whether the compomer is valid. Consequently, the complete backtracking costs for the worst-case are given by $\mathcal{O}(k + n + s^n)$.
- **Find-All:** The time to compute all decomposition of M_Δ over A by backtracking is the number of decompositions $\gamma(M_\Delta)$ times the worst-case cost of computing one decomposition. Consequently, all decompositions can be computed in $\mathcal{O}(\gamma(M_\Delta)(k + n))$ time. Let s be the largest set of specificity definitions of a modification. The overall backtracking cost in the worst-case is then given by $\mathcal{O}(\gamma(M_\Delta)(k + n) + \gamma(M_\Delta)s^n)$.

2.4 Efficient Dynamic Programming

The modified DP algorithm constructs D in $\mathcal{O}(kM_{max})$ to construct the table D. For small precisions like $p < 0.01$ and large numbers of modificiations per peptide n in parallel the runtime and memory consumption become impractical. On the other hand, it is advisable to use a computational precision as high as the input mass measurements can reliably provide.

To improve the efficiency of preprocessing, one can make use of the sparse structure of D which is especially the case for a moderate upper bound n and a small precision p.By way of example, consider the set of modifications $A = 2, 5, 9, -2$, and

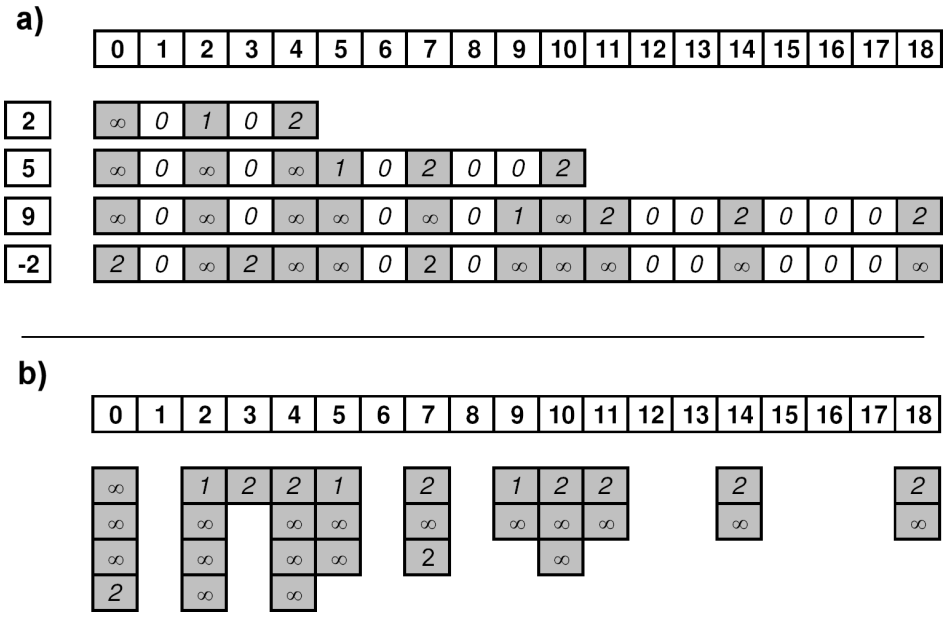

Figure 2.6: Ragged storage of the new DP table. Both examples have been constructed for the input modification mass changes $A = \{2, 3, -2\}$ and upper bound of $|c| \leq n = 2$. **a)** Row-by-row storage avoiding trailing zeros of each row. **b)** Column-by-column storage saving leading zeros of each column. Non-zero cells are highlighted.

parameters $n = 2$, $p = 1$. Preprocessing then produces a table that contains 30 non-zero values of a total of 76 values (see fig. 2.5). If $p = 0.1$, the number of cells in D becomes ten times higher, whereas the number of non-zero values stays constant.
At this point remember that the algorithm requires an ordering of the set of modifications only by sign, such that $i < j$ if $a_i > 0$ and $a_j < 0$. Clearly, each modification alone can change the mass of a peptide by at most na_i, but the largest m that can be decomposed in row i depends on the already decomposable mass changes m in the previous rows $1 \ldots i-1$. Nevertheless, it can be guaranteed to reach a maximum m of na_i for all $a_i > 0$ in row i, if A is sorted such that $a_i = \max(\{a_1, ..., a_i\}$ for all $a_i > 0$. Using this ordering, D can be computed more efficiently in ragged array representation.

2.4.1 Ragged Array Storage of the Dynamic Programming Table

Using a sorted set of modifications, a considerable amount of memory and time might be saved by constructing D as a ragged array (vector of vectors), i.e. each row or column is stored and treated as a separate entity that can have its own size. Both column-by-column and row-by-row representations of D are possible but produce

different results: The row-by-row representation (Fig. 2.6a) excludes all zero values of row i for $m > na_i$ (trailing zeros). The column-by-column representation (Fig. 2.6b) is more efficient for the structure of D because it avoids storage of leading zeros in each column, which also includes trailing zeros in rows. However, to access $D_{i,m}$, the index i in a column m has to be computed by $i_m = i - (k - l(m))$, where $l(m)$ denotes the length of column m.
The concept of ragged arrays allows to efficiently implement the table D and thereby reduce the average runtime and memory with only little computational overhead in the column-by-column case. Nevertheless, ragged-array storage is no improvement of the algorithm itself and is still inefficient for small p as it requires iteration over all columns in the worst-case. However, by using additional data structures it is possible to construct D with running time independent of the computational precision. The precise procedure will be sketched in the next section.

2.4.2 Efficient Construction of the Dynamic Programming Table

In the worst case, the DP algorithm for the PMMCP iterates over the whole range of indices $1 \leq m \leq M_{max}$ to fill a row i, even if most of the values $D_{i,m}$ will be zero. However, according to the idea of the algorithm, any new decomposition can only be obtained by extending an existing decomposition that requires less than n modifications.
Taking this into consideration, D could be efficiently computed assuming sparse storage of D because the algorithm can then simply iterate over all columns m that contain non-zero values in row $i - 1$. The generic concepts of sparse arrays or matrices, however, are accompanied by large runtime penalties for getting and setting values if the number of non-zero elements that are to be stored is not known beforehand. Instead, these concepts are mainly suitable for efficient iteration over their non-zero elements, which is useful for mathematical matrix computations. As an example, see for instance Gundersen and Steihaug (2004). For the DP algorithm presented here that frequently sets values in D, the runtime penalty would exceed the benefit of sparse storage.
Nevertheless, the algorithm can use arbitrary storage and maintain a linked list of column indices m, for which a decomposition has already been computed. This list is then used as an iterator over the non-zero elements, and for each m a new decomposition $m + a_i$ is appended to the end of the list if it is not yet contained. Initially, before the computation starts in the first row of D, $m = 0$ which can always be de-

composed by the empty compomer is added as a first element to the linked list. To allow for negative a_i which require row i to be filled from "right to left", a doubly-linked list can be used that can be iterated over from start to end and vice versa. Additionally, the algorithm must ensure that $D_{i,m-a_i}$ is computed prior to $D_{i,m}$. It is thus necessary to keep the linked list in ascending order and to insert each new m at its correct position. The latter can be achieved by using an auxiliary queue structure, that serves as a buffer: Values $m + a_i$ that extend m and are not yet in the linked list will be added to the queue first. Before proceeding with the next m in the linked list, it is compared to the first element in the queue. If this queue element is lesser (greater in the case of reversed-order iteration) than the next m, it is inserted into the linked list at the current position and links to m afterwards. The algorithm directly continues with this new element and a row has been filled completely if the last element of the linked list has been processed and no more elements are left in the queue.

All the operations, insert for the linked list and peek for the queue, can be performed in constant time. It is also possible to check in constant time whether a value m is in the linked list, by a lookup of T_m: If $T_m < \infty$, there is at least one way to decompose m and if so, the value m must have already been added to the list. The two auxiliary data structures use additional memory $\mathcal{O}(2M_{max})$ in the worst case. The overall worst-case runtime for the construction of D does not change in comparison to the easier DP algorithm, except for a constant factor. On the other hand, the average runtime should have improved much, if D is sparse.

3 Implementation

3.1 Programming Language and Environment

A test version of the DP algorithm been implemented using the Java™2 Standard Edition Development Kit (JDK) 1.4.1 and JDK 1.5. Even though Java can hardly compete with other languages like C or Fortran for solving computationally intense problems, it offers great ease of programming and has been improved constantly to provide better performance. Moreover, it offers excellent object-oriented design, platform independence and the comprehensive and elaborate standard application programming interface (API) that facilitated the integration and internal management of auxiliary data that is required for the algorithm. Additionally, a graphical user interface (GUI) has been implemented that takes a protein sequence and an experimental mass fingerprint as input and applies the DP algorithm to detect matching peaks considering a list of possible modifications.

Programming was done using the Netbeans™Integrated Development Environment (IDE) 5.5 by Sun Microsystems Inc. on a Microsoft Windows®XP professional platform (standard PC with Intel®Pentium®4 cpu 2.8GHz, 1024mb main memory).

3.2 External Classes and Class Libraries

The JDOM™1.0 API was used to extract the modification datasets from the Unimod DB XML output (for further details see section 3.3). It offers extensive features to access, manipulate and output XML data with Java. The class packages and a more comprehensive description can be accessed at the project homepage[1].

To efficiently store the DP table D, a concept for sparse arrays has been used (McCluskey, 1999). Although proposed in the context of the Java programming language, it can be used similarly in all object-oriented programming languages.

[1] http://www.jdom.org

3.3 Data Sources

Datasets of protein modifications, atomic and amino acid masses, and enzymatic cleavage rules used in the implementation have been extracted from either the Unimod database or the Mascot help pages.

1. **Protein modifications:** UniMod offers the possibility to retrieve a complete database image in XML format, which was taken and parsed for relevant information. A total of 508 modifications has been extracted to be used in the algorithm, each with monoisotopic and average mass change, and site and position specifications.
2. **Other mass data:** For the calculation of the theoretical mass list of the input protein, tables of the atomic and amino acid masses have been compiled. Atomic masses were taken from the Unimod help page[2] and amino acid masses from the Mascot help page[3]. The list of all considered atoms and amino acids is given in the appendix (Tables A.2, A.3).
3. **Proteolytic enzymes:** Cleavage recognition patters of different proteolytic enzymes were taken from the Mascot help page to be used for theoretical digestion. For each enzyme, rules were formulated as a regular expression and the cut position was marked by a "`#`"-sign. Enzymes with more than one cleavage recognition pattern were described as a list of regular expressions for practical purposes. As an example, trypsin cuts C-terminal to arginine (R) or lysine (K) if not followed by proline (P), such that the regular expression rules for trypsin are formulated as `K#[^P],R#[^P]`. A list of all enzymes supported by the digestion module can be found in the appendix (Table A.1).

3.4 A Tool for the Detection of Protein Modifications

The DP algorithm for the PMMCP presented in section 2.3 allows efficient calculation of the combinations of modifications that can give rise to an observed mass distance M_Δ. The algorithm can be used in a typical PMF approach to detect protein modifications (Fig. 3.1) by comparing an experimental mass fingerprint and a putative protein identification that is expected to underly the experimental data.

[2] http://www.unimod.org/masses.html
[3] http://www.matrixscience.com/help/aa_help.html

The idea of the procedure is to calculate a theoretical mass fingerprint of the sequence of the unmodified input protein as a reference. Subsequently, any combination of a list of known protein modifications is used to find matches for each pair of an experimental mass m_E and a theoretical mass m_T with distance $m_E - m_T = M_\Delta$ by applying the DP algorithm for the PMMCP. The procedure comprises the following steps:

1. **Theoretical digestion:** The input protein is theoretically digested to obtain a reference mass fingerprint containing the masses of the unmodified peptides.
2. **Precomputation of decompositions:** Based on an input list of modifications, the DP algorithm constructs a decomposition table D that contains all mass distances that can be obtained by combinations of modifications.
3. **Identification of protein modifications:** For each mass distance observed between masses of the experimental mass fingerprint and the theoretical reference masses, D is queried to obtain potential combinations of modifications that may explain the observed mass distance.

3.4.1 Theoretical Digestion

In the first step, a theoretical mass fingerprint is calculated from the input reference protein. Theoretical digestion is computationally demanding when variable modifications are considered because every unmodified theoretical peptide can have many modified counterparts for which a mass has to be calculated. However, for the above described approach it is sufficient, and even desired, to have a theoretical mass fingerprint corresponding to the unmodified peptides of the input protein for subsequent comparison. A straightforward digestion algorithm was developed that first gathers all cleavage positions in the protein sequence and subsequently iterates over pairs of these positions to obtain the corresponding peptide sequences (Fig. 3.2). Subsequent to peptide construction, the algorithm then calculates the theoretical mass of each peptide based on its amino acid sequence.

1. **Gathering of cleavage indices:** Each of the cleavage recognition patterns, given a proteolytic enzyme, is used to gather a sorted list L of cleavage positions of the input protein sequence S with length $|S|$ in ascending order. Subsequently, the start position 0 and the terminal position $|S|$ are added, and finally, L contains all cleavable sequence positions l_i, and $l_i \leq l_j$ if $i \leq j$.

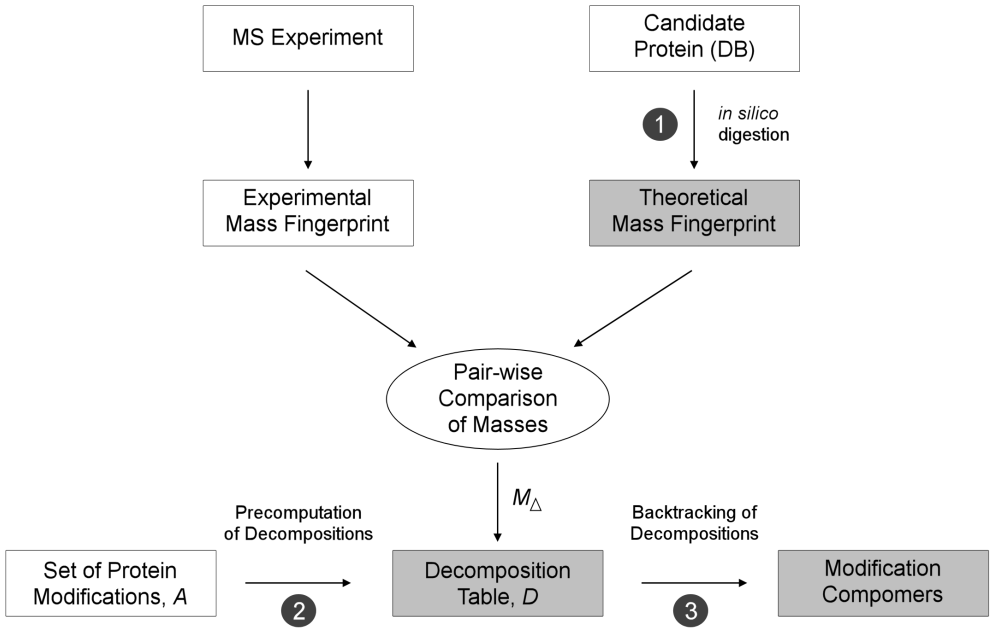

Figure 3.1: Principal approach of the detection of protein modifications using PMF data. The protein modification identification process comprises 3 steps. 1. An input protein is used as a reference to obtain a theoretical mass fingerprint that is subsequently compared to the experimental mass fingerprint. 2. All possible ways to combine modifications from a list of known modifications are precomputed by DP and stored in a table, D. 3. Masses of experimental and theoretical mass fingerprints are compared pair-wise, and combinations of modifications explaining an observed mass distance are obtained by backtracking in D.

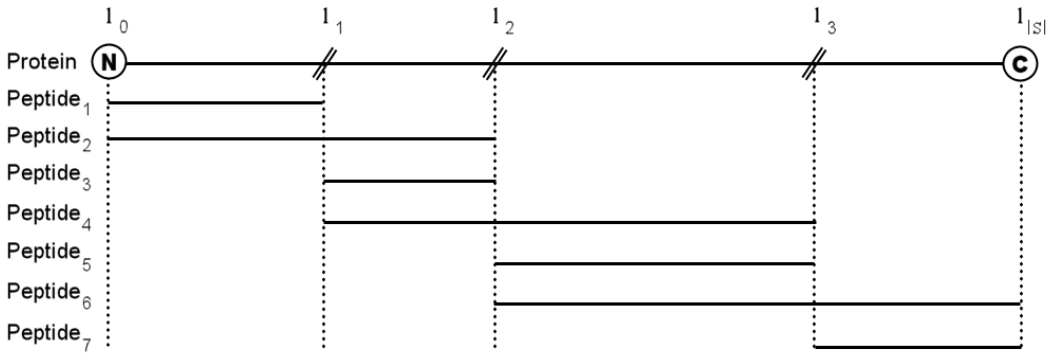

Figure 3.2: Illustration of the theoretical digestion process. Schematic of the procedure of the theoretical digestion algorithm. L_0 to $L_{|S|}$ mark the cleavage positions in a protein sequence S with length $|S|$. The peptides that are calculated by the algorithm if one missed cleavage site is allowed are indicated by black lines below the protein sequence (top line). N and C denote the N-terminal and C-terminal end of the protein, respectively.

2. **Peptide construction:** Theoretical peptides are generated by iterating over the positions in L with two indices, i_{start} and i_{end}. Initially, i_{start} points at l_0 and i_{end} at l_1. The algorithm proceeds by storing the peptide between positions l_0 to l_1 and increases a missed cleavages counter. While this counter is lower than a threshold for the maximum of missed cleavages and i_{end} does not point to the last element in L, i_{end} is incremented so that it points to the next cleavage position. Otherwise, the missed cleavages counter is set to 0, i_{start} is incremented, and the procedure is repeated. The algorithm terminates if i_{start} points at the last element in L.

Real proteolytic enzymes often show suboptimal cleavage efficiency which means that some of the possible cleavage positions remain intact. The algorithm accounts for that by allowing a maximum number of missed cleavages. When set to 2, for example, the algorithm produces all peptides with 0, 1 and 2 missed cleavage sites. Additionally, if the protein sequence starts with a methionine residue at the N-terminal end, two peptides are created, one including the initial residue, and one excluding it.

3.4.2 Precomputation and Identification of Protein Modifications

Two decomposition tables have to be constructed to allow backtracking in the case of positive and negative mass distances M_Δ, respectively. Both tables are constructed on the basis of an input set of modifications A, a user-defined upper bound for the maximum number of modifications per peptide n, and an appropriate computational accuracy p. The construction of the tables relies on the principles mentioned in section 2.3.

Putative modifications of the peptides underlying the experimental mass fingerprint are identified by backtracking in the previously constructed tables for each pairwise mass distance M_Δ between the experimental and theoretical masses. However, mass spectrometric measurements are not perfectly accurate such that the uncertainty of mass values and similarly of peak distances M_Δ needs to be considered. Instead of computing only the compomers of M_Δ, the algorithm builds the interval $[M_\Delta - \epsilon, M_\Delta + \epsilon]$ and calculates the compomers for all masses inside this interval by backtracking. Unfortunately, the limits of the tolerance interval can exceed the lowest mass 0 and the largest mass M_{max}. In these cases, only that part of the interval can be used that is inside the range 0 and M_{max}. The "quality" of a compomer

is assessed by its deviation from an observed M_Δ and its length $|c|$ such that fewer modifications in an explanation and compomers with smaller deviation from an observed mass distance are considered to be better.

3.4.3 Parameterization

The various steps of the identification procedure are controlled by the following set of parameters:

- **Input list of modifications:** The input list of modifications is used to construct the decomposition table. While a large list is important to perform an exhaustive search for protein modifications, it becomes more likely to explain a mass distance M_Δ simply by chance.
- **Computational accuracy p:** The computational accuracy defines the discretization of the real mass values on the integer scale. p should be chosen to capture the relevant decimal places of the experimental masses.
- **Maximum number of modifications per peptide n:** Defines the upper bound $|c| \leq n$ for the modification compomers. Choosing a large n increases the number of modification compomers that may explain an observed mass distance M_Δ.
- **Mass error ϵ:** The extent of an interval around the mass difference M_Δ in which other mass distances are considere equal. The defintion of ϵ accounts for inaccuracies of the mass measurements.

3.4.4 User Interface

A graphical user interface (GUI) has been implemented that allows to run the algorithm. The main window (Fig. 3.3) allows users to enter the input mass list and protein sequence, as well as the parameters required for the theoretical digestion and the DP procedure. However, the algorithm can also be run without using the GUI to provide the ability to process batches of peaklists in a row. The results can be stored as tab-separated tables in ASCII format and can be used for further processing as needed.

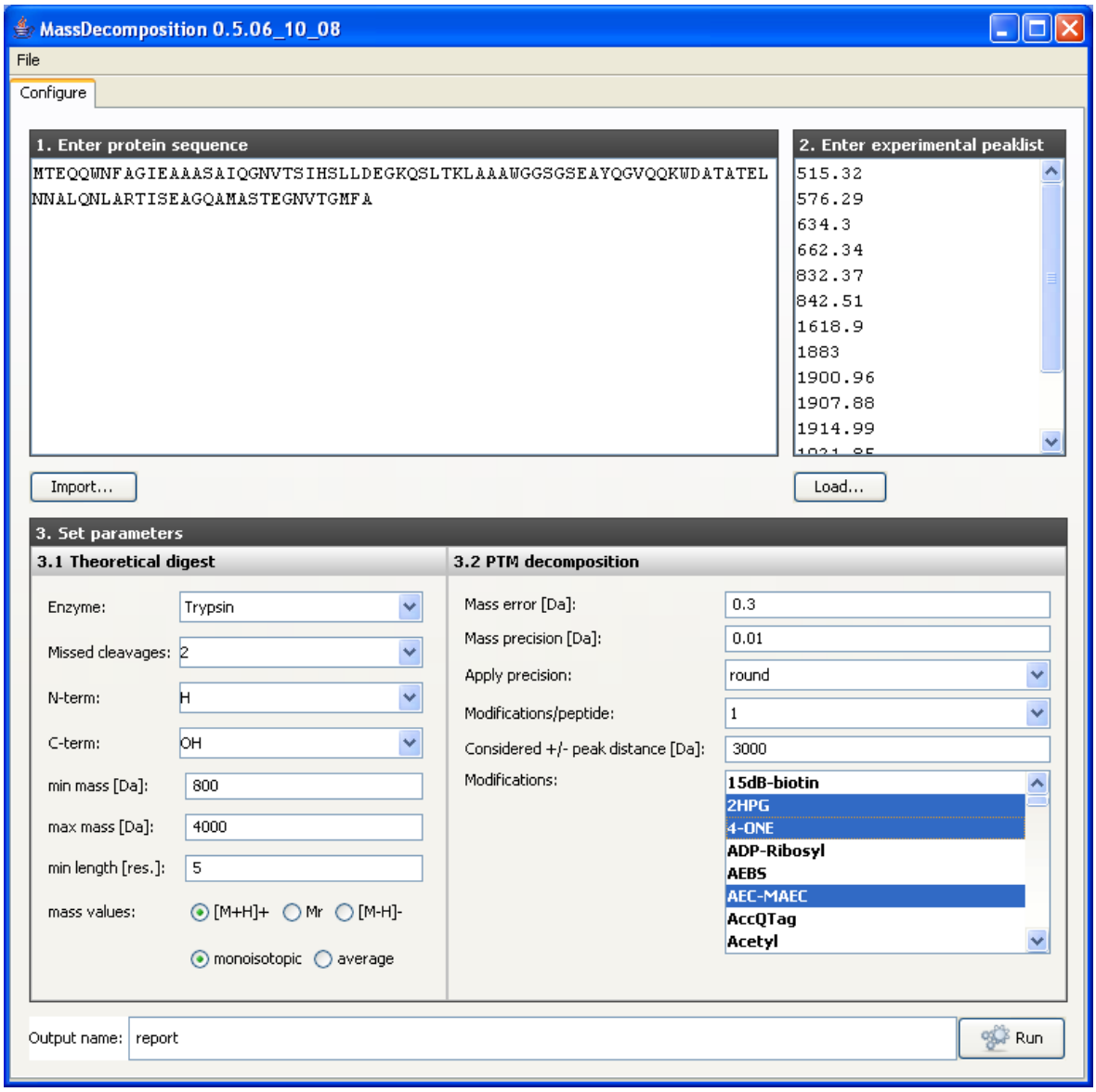

Figure 3.3: Main window of the GUI to run and configure the DP algorithm. The main window provides a top-level interface to configure and run the algorithm to detect putative protein modifications by comparing an experimental mass fingerprint with a protein sequence.

4 Evaluation

4.1 Testing Environment

The DP algorithm solves a combinatorial problem in a deterministic way. Each result, corresponding to a combination of modifications, is ranked by the number of modifications required for a decomposition and the deviation from the observed mass distance. This ranking is not based on biological background data like, for instance, PTM frequencies which makes the investigation of the predictive capabilities of this approach impossible. Therefore, the algorithm will only be evaluated under the the aspect of computational performance.

The performance of the algorithm has been evaluated on an Intel®Pentium®4 CPU (2.8GHz clock speed, 1024 MB main memory) running Windows®XP professional. The evaluation was based on a large dataset (in the following referred to as "Unimod dataset") with all 508 entries included in the current release of Unimod comprising PTMs, single amino-acid substitutions, and experimental artifacts. The smallest monoisotopic mass change (-129.057849 Da) in the dataset corresponds to the substitution of tryptophan by glycine whereas the largest mass change (1768.6395170 Da) is observed for dHex(1)Hex(5)HexNAc(4). However, most of the mass changes lie in the range between -100 Da and 500 Da (Fig. 4.1) and the majority of protein modifications induces a positive mass change to a protein.
The influence of the computational precision p on the separability of modifications by their mass change is shown in Fig. 4.2. It reveals high numbers of non-unique modification mass changes for large computational precisions. The best separability is observed for $p \leq 0.001$ with 55 non-unique modification mass changes. Therefore, a small computational accuracy has to be applied to reduce the number of false positive compomers due to redundancy in the protein modification mass changes.

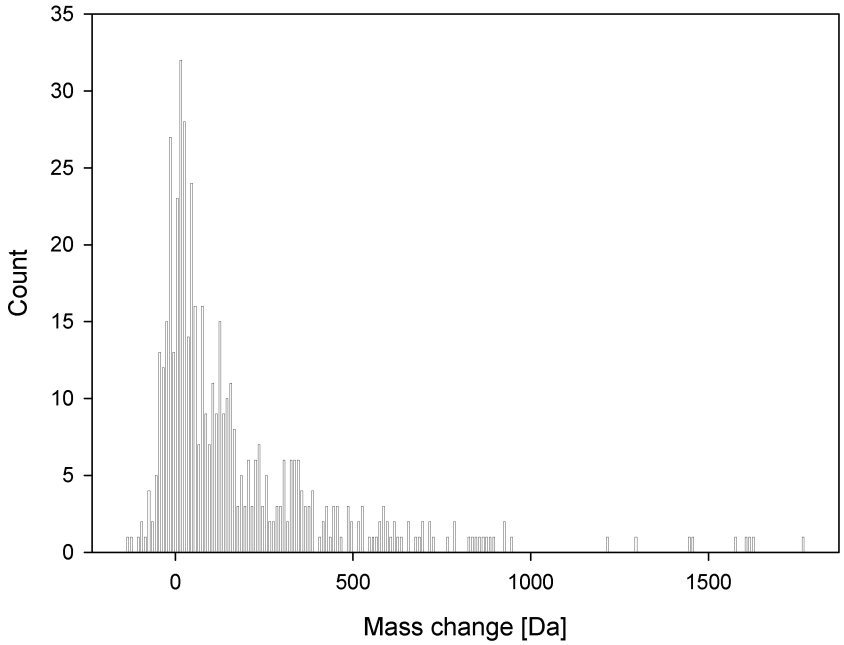

Figure 4.1: Distribution of modification mass changes in the Unimod dataset. The histogram has been computed with a bin size of 10 Da.

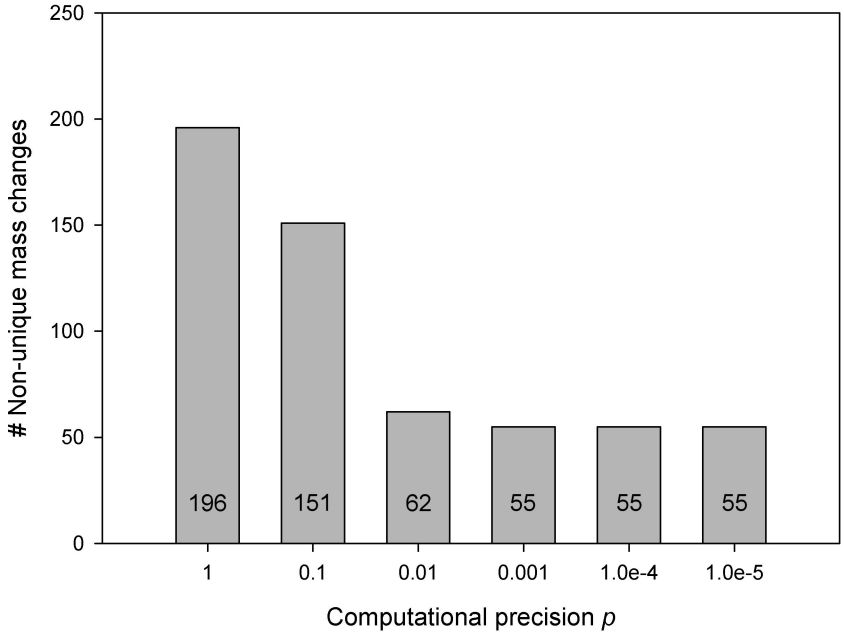

Figure 4.2: Numbers of non-unique modification mass changes in the Unimod dataset. Modifications with non-unique mass change have been counted after applying different computational precisions p. The exact number of non-unique mass changes is indicated on each bar.

Table 4.1: Runtimes for the construction of the DP table using the Unimod dataset. p: Computational precision; $t_{n=x}$: Construction time (measured in ms) in the case that x modifications per peptide were allowed.

p	$t_{n=1}$	$t_{n=2}$	$t_{n=3}$	$t_{n=4}$	$t_{n=5}$
1.0	3.8556	12.8513	19.0837	20.9966	25.7742
0.1	37.8858	75.5808	106.3276	135.4023	169.9714
0.01	254.3822	533.4976	785.7779	1142.5357	1430.2937

4.2 Construction of the Dynamic Programming Table

The size of the DP table depends on the computational precision p, the upper bound $|c| \leq n$ that restricts the maximum number of modifications per peptide, and the set of modifications A that is used for the construction. The influence of parameters p and n on the construction time of the DP table has been investigated.
The cost for the construction of the DP table increases linearly in dependence of n and p (Figs. 4.3, 4.4; Table 4.1). Note that a change from $p = 0.1$ to $p = 0.01$ is a 10-fold increase in precision and causes the runtime to be roughly 10-fold higher. Construction of the DP table takes less than 170 ms for a precision $p \geq 0.1$ allowing up to 5 modifications per peptide. Applying a computational precision of $p = 0.01$ requires up to 1430 ms to construct the DP table with $n = 5$. However, the DP table needs to be constructed only once for any particular set of modifications which puts the high runtimes using small precisions into perspective.

4.3 Backtracking

The results obtained by backtracking are restricted by the peptide sequence underlying a theoretical mass that is compared to an experimental mass. It is impossible to measure a general trend in the number of decompositions for an observed mass distance M_Δ because peptides comprising only a few of the 20 amino acids can be subject to fewer modifications than peptides with a large variety of different amino acids. Therefore, the backtracking performance was investigated for a particular example using the sequence of the 6 kDa early secretory antigenic target (ESAT-6) [accession number: Q540D8] of *Mycobacterium tuberculosis*. This sequence was theoretically digested allowing no missed cleavage sites and subsequently, the resulting theoretical mass list was compared to the experimental peaklist that gave rise to the identification (Fig. 4.5).

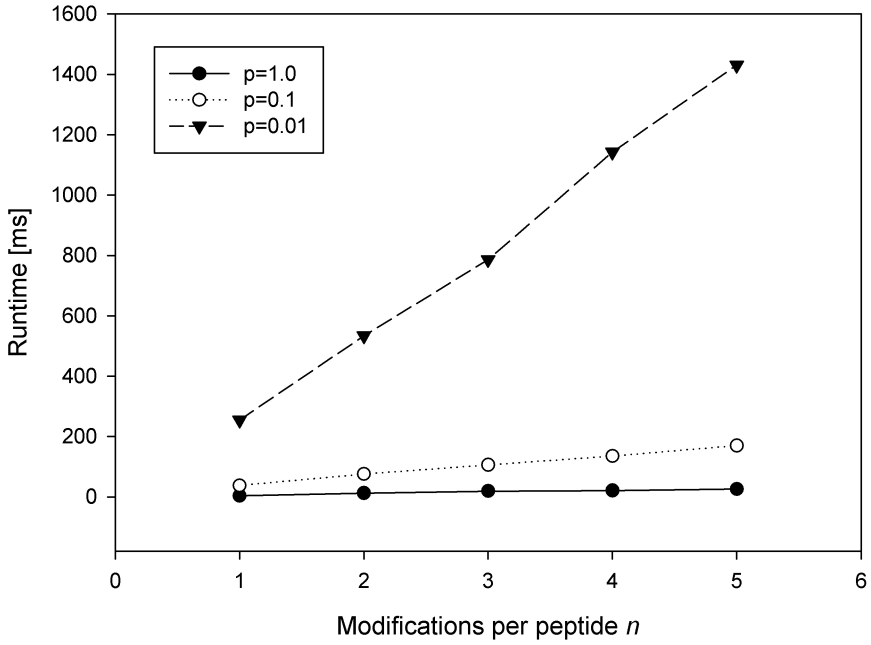

Figure 4.3: Runtime for the construction of the DP table applying different computational precisions. The runtime has been measured as the average of 15 construction runs.

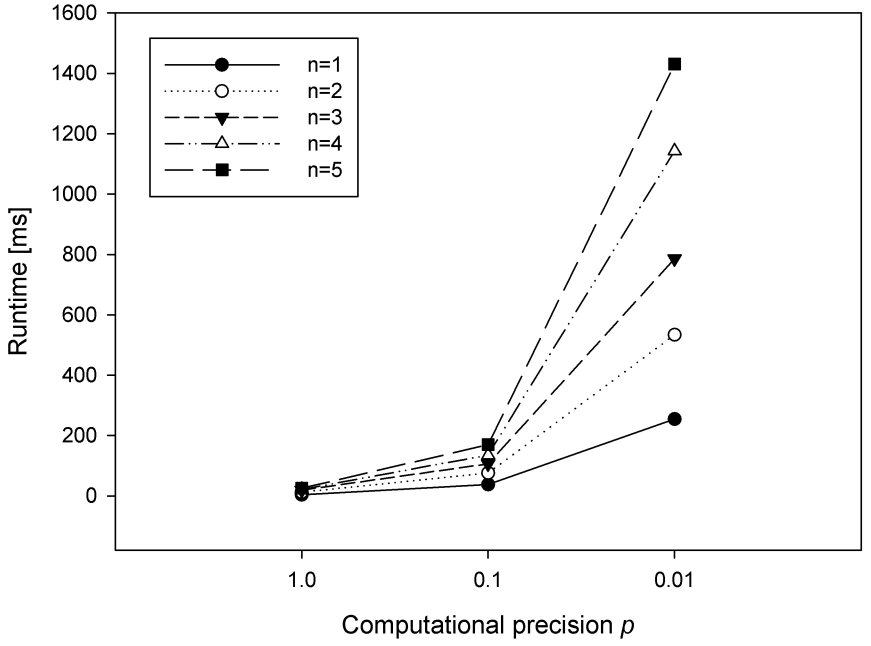

Figure 4.4: Runtime for the construction of the DP table applying different upper bounds n. The runtime has been computed as the average of 15 construction runs.

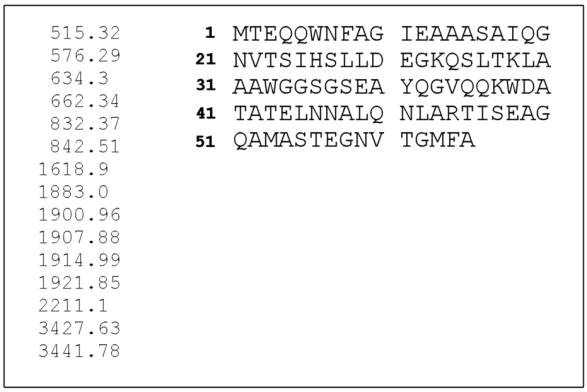

Figure 4.5: Protein sequence of the 6 kDa early secretory antigenic target of *Mycobacterium tuberculosis*. Left: Experimental peaklist of a protein that was identified as ESAT-6. Upper right: Protein sequence of ESAT-6.

Table 4.2: Runtime, spc, and number of decompositions for backtracking with precisions $p = 1.0$, $p = 0.1$, and $p = 0.01$. The mass error has been set to $\epsilon = 0$. n: Maximum number of modifications per peptide; t: Backtracking time [ms]; spc: Shared peaks count between experimental and theoretical mass fingerprint; c_f: Total number of decompositions for all matches with sequence constraints applied; c_{uf}: Total number of compomers for all matches without sequence constraints applied.

	$p = 1.0$				$p = 0.1$				$p = 0.01$			
n	t	spc	c_f	c_{uf}	t	spc	c_f	c_{uf}	t	spc	c_f	c_{uf}
1	78	12	64	64	62	6	36	36	62	0	0	0
2	79	18	3988	7252	78	14	1672	3424	93	9	120	212
3	594	19	321340	614104	438	16	109874	218322	328	12	11172	21912

The runtime to compare the experimental mass list and the theoretical mass list calculated from the sample sequence, the shared peaks count, and the total number of compomers for all assignments were measured in dependence of the parameter n for computational precisions $p = 1$, $p = 0.1$, and $p = 0.01$ (Table 4.2). The maximum number of modifications per peptide is the main influencing parameter of the backtracking performance because $|c| \leq n$ directly restricts the number of compomers that can be used for a decomposition and the total number of compomers increases with increasing n. For the same n, the number of compomers is significantly higher in the case of larger precisions than in the case of smaller precisions which can be attributed to the higher redundancy in the modification mass changes if small precisions are applied. However, the total number of decompositions is too high to allow reasonable interpretation of the single compomers in all cases such that it is infeasible to decide whether a combination of modifications is likely to be correct or if it matches an observed mass distance only by chance.

5 Discussion

5.1 Algorithm

The algorithm presented in this thesis addresses the problem of PTM detection by formulating the Protein Modification Money Changing Problem. This problem asks for combinations of protein modifications giving rise to a mass change observed between an experimental and a theoretical mass. Other approaches like FindMod avoid the integration of large numbers of modifications in parallel, because the combinatorial explosion makes it infeasible to iterate over all possible combinations of modifications. The purpose of the novel algorithm is to allow for high numbers of modifications to be searched in parallel by limiting the search space to all combinations of modifications that produce an observed mass change. This has been achieved by investigating the classical Money Changing Problem which asks for the decomposition of a non-negative integer over a set of other non-negative integers. This similar but simpler problem can be solved in pseudo-polynomial time using DP (Böcker and Liptak, 2004). Although the original DP algorithm cannot be applied to the Protein Modification Money Changing Problem directly, it was possible to extend the idea of the algorithm to allow for negative numbers (modification mass changes and mass distances) and a restriction of the combinations by allowing at most n of these numbers in a decomposition. In the context of protein modifications this corresponds to an upper bound for the number of modifications that are allowed per peptide.
The general proceeding of the algorithm is to precompute the decompositions of all mass changes in the range between 0 and M_{max}, the maximum mass change that is determined by the set of modifications and the upper bound n, by DP. This precomputation yields a decomposition table that allows to iterate exclusively over all combinations of modifications that give rise to a mass change M_Δ by backtracking. However, filtering of the biologically correct decompositions according to the specificity of protein modifications has to be done after backtracking. This is a considerable drawback because it may require to investigate all possible decompositions of a mass distance to decide whether there exists a solution that agrees to the biological

constraints. If a large precision and a large n is chosen, the backtracking runtime will become impractical.

Nevertheless, the algorithm can solve the combinatorial problem of PTM detection efficiently for large numbers of modifications if $p \geq 0.01$ and $n \leq 3$ while taking into account a multiple of the modifications used by FindMod. However, although the algorithm can include many modifications in a search, the results are difficult to interpret because hardly any information about the frequencies of modifications is available to calculate the quality of the decompositions.

5.2 Assessment of Decompositions

The vast number of possibilities that explain a mass distance simply by chance cannot be distinguished from those that might in fact be the result of modifications. Therefore, it is necessary to introduce a scoring method that can rank the large number of putative results to allow for a reasonable application of the algorithm. One reliable way to achieve this might be to assign a weight score to each combination of modifications that is based on the frequencies (weights) of occurrences of single modifications. However, no comprehensive dataset is available that allows to infer the frequencies of all known protein modifications. Probably, more insight will be gained from blind search approaches, for example by Tsur et al. (2005), that can predict known and unknown protein modifications and may allow to find and annotate PTMs on a large-scale.
A simpler way of validation is by application of modification rules that describe properties that may hinder or contribute to the occurrence of particular modifications. FindMod, for example, uses sequence patterns that were frequently observed in correspondence to sites of post-translational modifications. Moreover, an approach to detect single residue PTMs by investigation of consensus sequence motives has been implemented in the AutoMotif server Plewczynski et al. (2005). Nevertheless, the set of rules of both approaches is far from being comprehensive and captures mainly the most common and best studied protein modifications like phosphorylation, acetylation, or methylation. Additionally, rule-based scoring does only allow for a rudimentary ranking of different results because a rule can either be accepted or rejected.

5.3 Outlook

The DP algorithm allows for the computation of decompositions of observed mass distances between experimental and theoretical masses efficiently if the computational precision and the number of modifications per peptide is limited ($p \geq 0.01$, $n \leq 3$). Runtimes suggest that it is suitable to be applied on a small scale, i.e., when only one experimental mass fingerprint is compared to one or a few theoretical mass lists. It is not suitable to be applied in time-critical applications which require to consider a large number of theoretical mass lists, as for example, in DB search methods. To further improve the runtime of the algorithm it is necessary to integrate the specificity constraints of the protein modifications already in the construction of the DP table. While this would require to build a separate table for each theoretical peptide that is used for comparison, the performance penalty may be only little: It is likely that only a subset of modifications can occur on a peptide so that the construction of each separate table is more efficient because less rows of the table have to be computed.
Nevertheless, the algorithm has little practical value for the prediction of PTMs without an appropriate scoring procedure. Further efforts have to be made to combine the efficient calculation of the PMMCP with means of interpreting the quality of detections to prove this approach useful.

6 Summary

In this work, a novel algorithmic approach was presented that allows to detect PTMs or other protein modifications by comparing an experimental and a theoretical mass fingerprint. The algorithm is based on a variation of a DP algorithm for the classical MCP which has been applied for the decomposition of biomolecules over alphabets of amino acids or nucleotides. Here, the problem of PTM detection is formulated as an extended MCP which is called the Protein Modification Money Changing Problem: For an observed mass distance between an experimental and a theoretical mass, find all combinations of protein modifications that give rise to that mass difference. The original algorithm could not be applied directly, because it is unable to deal with non-negative numbers. The novel algorithm allows for decompositions of positive and negative numbers (mass distances M_Δ) over a set of positive and negative numbers (protein modification mass changes $A = \{a_1, ..., a_k\}$) and integrates an upper bound n for the number of modifications that can be used to decompose a mass distance.

The combinatorial complexity of PTM detection was addressed using this improved DP algorithm to precompute all combinations of modifications that can result in mass distances between 0 and a maximum mass $M_{max} = \max(A)n$. Subsequently, for any observed M_Δ in this range, only the precomputed results are iterated over to find those combinations of protein modifications that agree with the biological constraints induced by the peptide sequence underlying the theoretical mass. Although the algorithm allows to compute PTM candidates efficiently, even with high numbers of modifications in parallel, the number of results is usually too large to be interpreted. Therefore, it will be essential to integrate a scoring method that evaluates the quality of the results either by probabilities to observe particular combinations of modifications, or by expert rules that indicate the quality of an explanation. At the current time, however, there are no methods or data to evaluate the likelihood of observing a particular modification for all, or at least for a large part of known protein modifications.

List of Figures

List of Tables

Abbreviations

2-DE	Two-dimensional electrophoresis
2D-PAGE	Two-dimensional polyacrylamide gel electrophoresis
API	Application programming interface
DB	Database
DP	Dynamic programming
ESAT-6	6 kDa early secretory antigenic target
ESI	Electrospray ionization
IDE	Integrated development environment
JDK	Java SE development kit
MALDI	Matrix assisted laser desorption ionization
MCP	Money Changing Problem
MS(/MS)	(Tandem)Mass spectrometry
PMF	Peptide mass fingerprinting
PMMCP	Protein Modification Money Changing Problem
PTM	Post-translational modification
spc	Shared peaks count
TOF	Time of flight
XML	Extensible markup language

Bibliography

Böcker, S. (2004). Sequencing from compomers: using mass spectrometry for dna *de novo* sequencing of 200+ nt. *J Comput Biol*, 11(6):1110–1134.

Böcker, S. and Liptak, S. (2004). Efficient mass decomposition. In *20th Annual ACM Symposium on Applied Computing in Santa Fe, New Mexico*.

Beckey, H. D. (1969). Field desorption mass spectrometry: A technique for the study of thermally unstable substances of low volatility. *International Journal of Mass Spectrometry and Ion Physics*, 2:500–502.

Brunneé, C. (1987). The ideal mass analyzer: fact or fiction. *International journal of mass spectrometry and ion processes*, 76:125–237.

Claverie, J. M. (2001). Gene number. what if there are only 30,000 human genes? *Science*, 291(5507):1255–1257.

Colgan, D. F., Murthy, K. G., Zhao, W., Prives, C., and Manley, J. L. (1998). Inhibition of poly(a) polymerase requires p34cdc2/cyclin b phosphorylation of multiple consensus and non-consensus sites. *EMBO J*, 17(4):1053–1062.

Fenn, J. B., Mann, M., Meng, C. K., Wong, S. F., and Whitehouse, C. M. (1989). Electrospray ionization for mass spectrometry of large biomolecules. *Science*, 246(4926):64–71.

Gattiker, A., Bienvenut, W. V., Bairoch, A., and Gasteiger, E. (2002). Findpept, a tool to identify unmatched masses in peptide mass fingerprinting protein identification. *Proteomics*, 2(10):1435–1444.

Gundersen, G. and Steihaug, T. (2004). Data structures in java for matrix computations. *Concurrency Computat.: Pract. Exper.*, 16:799–815.

Henzel, W. J., Watanabe, C., and Stults, J. T. (2003). Protein identification: the origins of peptide mass fingerprinting. *J Am Soc Mass Spectrom*, 14(9):931–942.

International Human Genome Sequencing Consortium (2004). Finishing the euchromatic sequence of the human genome. *Nature*, 431:931–945.

James, P., Quadroni, M., Carafoli, E., and Gonnet, G. (1993). Protein identification by mass profile fingerprinting. *Biochem Biophys Res Commun*, 195(1):58–64.

Jensen, O. N. (2004). Modification-specific proteomics: characterization of post-translational modifications by mass spectrometry. *Current Opinion in Chemical Biology*, 8:33–41.

Karas, M. and Hillenkamp, F. (1988). Laser desorption ionization of proteins with molecular masses exceeding 10,000 daltons. *Anal Chem*, 60(20):2299–2301.

Klose, J. (1975). Protein mapping by combined isoelectric focusing and electrophoresis of mouse tissues. a novel approach to testing for induced point mutations in mammals. *Humangenetik*, 26(3):231–243.

Klose, J. and Kobalz, U. (1995). Two-dimensional electrophoresis of proteins: an updated protocol and implications for a functional analysis of the genome. *Electrophoresis*, 16(6):1034–1059.

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., Funke, R., Gage, D., Harris, K., Heaford, A., Howland, J., Kann, L., Lehoczky, J., LeVine, R., McEwan, P., McKernan, K., Meldrim, J., Mesirov, J. P., Miranda, C., Morris, W., Naylor, J., Raymond, C., Rosetti, M., Santos, R., Sheridan, A., Sougnez, C., Stange-Thomann, N., Stojanovic, N., Subramanian, A., Wyman, D., Rogers, J., Sulston, J., Ainscough, R., Beck, S., Bentley, D., Burton, J., Clee, C., Carter, N., Coulson, A., Deadman, R., Deloukas, P., Dunham, A., Dunham, I., Durbin, R., French, L., Grafham, D., Gregory, S., Hubbard, T., Humphray, S., Hunt, A., Jones, M., Lloyd, C., McMurray, A., Matthews, L., Mercer, S., Milne, S., Mullikin, J. C., Mungall, A., Plumb, R., Ross, M., Shownkeen, R., Sims, S., Waterston, R. H., Wilson, R. K., Hillier, L. W., McPherson, J. D., Marra, M. A., Mardis, E. R., Fulton, L. A., Chinwalla, A. T., Pepin, K. H., Gish, W. R., Chissoe, S. L., Wendl, M. C., Delehaunty, K. D., Miner, T. L., Delehaunty, A., Kramer, J. B., Cook, L. L., Fulton, R. S., Johnson, D. L., Minx, P. J., Clifton, S. W., Hawkins, T., Branscomb, E., Predki, P., Richardson, P., Wenning, S., Slezak, T., Doggett, N., Cheng, J. F., Olsen, A., Lucas, S., Elkin, C., Uberbacher, E., Frazier, M., Gibbs, R. A., Muzny, D. M., Scherer, S. E., Bouck, J. B., Sodergren, E. J., Worley, K. C., Rives, C. M., Gorrell, J. H., Metzker, M. L., Naylor, S. L., Kucherlapati, R. S., Nelson, D. L., Weinstock, G. M., Sakaki, Y., Fujiyama, A., Hattori, M., Yada, T., Toyoda, A., Itoh, T., Kawagoe, C., Watanabe, H., Totoki, Y., Taylor, T., Weissenbach, J., Heilig, R., Saurin, W., Artiguenave, F., Brottier, P., Bruls, T., Pelletier, E., Robert, C., Wincker, P., Smith, D. R., Doucette-Stamm, L., Rubenfield, M., Weinstock, K., Lee, H. M., Dubois, J., Rosenthal, A., Platzer, M., Nyakatura, G., Taudien, S., Rump, A., Yang, H., Yu, J., Wang, J., Huang, G., Gu, J., Hood, L., Rowen, L., Madan, A., Qin, S., Davis, R. W., Federspiel, N. A., Abola, A. P.,

Proctor, M. J., Myers, R. M., Schmutz, J., Dickson, M., Grimwood, J., Cox, D. R., Olson, M. V., Kaul, R., Raymond, C., Shimizu, N., Kawasaki, K., Minoshima, S., Evans, G. A., Athanasiou, M., Schultz, R., Roe, B. A., Chen, F., Pan, H., Ramser, J., Lehrach, H., Reinhardt, R., McCombie, W. R., de la Bastide, M., Dedhia, N., Blöcker, H., Hornischer, K., Nordsiek, G., Agarwala, R., Aravind, L., Bailey, J. A., Bateman, A., Batzoglou, S., Birney, E., Bork, P., Brown, D. G., Burge, C. B., Cerutti, L., Chen, H. C., Church, D., Clamp, M., Copley, R. R., Doerks, T., Eddy, S. R., Eichler, E. E., Furey, T. S., Galagan, J., Gilbert, J. G., Harmon, C., Hayashizaki, Y., Haussler, D., Hermjakob, H., Hokamp, K., Jang, W., Johnson, L. S., Jones, T. A., Kasif, S., Kaspryzk, A., Kennedy, S., Kent, W. J., Kitts, P., Koonin, E. V., Korf, I., Kulp, D., Lancet, D., Lowe, T. M., McLysaght, A., Mikkelsen, T., Moran, J. V., Mulder, N., Pollara, V. J., Ponting, C. P., Schuler, G., Schultz, J., Slater, G., Smit, A. F., Stupka, E., Szustakowski, J., Thierry-Mieg, D., Thierry-Mieg, J., Wagner, L., Wallis, J., Wheeler, R., Williams, A., Wolf, Y. I., Wolfe, K. H., Yang, S. P., Yeh, R. F., Collins, F., Guyer, M. S., Peterson, J., Felsenfeld, A., Wetterstrand, K. A., Patrinos, A., Morgan, M. J., de Jong, P., Catanese, J. J., Osoegawa, K., Shizuya, H., Choi, S., Chen, Y. J., Szustakowki, J., and Consortium, I. H. G. S. (2001). Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921.

Leevers, S. J. and Marshall, C. J. (1992). Activation of extracellular signal-regulated kinase, erk2, by p21ras oncoprotein. *EMBO J*, 11(2):569–574.

Livolsi, A., Busuttil, V., Imbert, V., Abraham, R. T., and Peyron, J. F. (2001). Tyrosine phosphorylation-dependent activation of nf-kappa b. requirement for p56 lck and zap-70 protein tyrosine kinases. *Eur J Biochem*, 268(5):1508–1515.

Lueker, G. (1975). Two np-complete problems in nonnegative integer programming. Technical report, Department of Electrical Engineering, Princeton University.

Mann, M. and Jensen, O. N. (2003). Proteomic analysis of post-translational modifications. *Nat Biotechnol*, 21(3):255–261.

Mann, M. and Pandey, A. (2001). Use of mass spectrometry-derived data to annotate nucleotide and protein sequence databases. *Trends Biochem Sci*, 26(1):54–61.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and Sons.

Matthiesen, R., Trelle, M. B., Højrup, P., Bunkenborg, J., and Jensen, O. N. (2005). Vems 3.0: algorithms and computational tools for tandem mass spectrometry based identification of post-translational modifications in proteins. *J Proteome Res*, 4(6):2338–2347.

McCluskey, G. (1999). Thirty ways to improve the performance of your java™programs. Online Article.

O'Farrell, P. H. (1975). High resolution two-dimensional electrophoresis of proteins. *J Biol Chem*, 250(10):4007–4021.

Pappin, D. J., Hojrup, P., and Bleasby, A. J. (1993). Rapid identification of proteins by peptide-mass fingerprinting. *Curr Biol*, 3(6):327–332.

Perkins, D. N., Pappin, D. J. C., Creasy, D. M., and Cotrell, J. S. (1999). Probability-based protein identification by searching sequence databases using mass spectrometry data. *Eletrophoresis*, 20:3551–3567.

Pevzner, P. A., Dancík, V., and Tang, C. L. (2000). Mutation-tolerant protein identification by mass spectrometry. *J Comput Biol*, 7(6):777–787.

Pevzner, P. A., Mulyukov, Z., Dancik, V., and Tang, C. L. (2001). Efficiency of database search for identification of mutated and modified proteins via mass spectrometry. *Genome Res*, 11(2):290–299.

Pleissner, K.-P., Eifert, T., Buettner, S., Schmidt, F., Boehme, M., Meyer, T. F., Kaufmann, S. H. E., and Jungblut, P. R. (2004). Web-accessible proteome databases for microbial research. *Proteomics*, 4(5):1305–1313.

Plewczynski, D., Tkacz, A., Wyrwicz, L. S., and Rychlewski, L. (2005). Automotif server: prediction of single residue post-translational modifications in proteins. *Bioinformatics*, 21(10):2525–2527.

Roepstorrf, P. and Fohlmann, J. (1984). Proposal for a common nomenclature for sequencing ions in mass spectra of peptides. *Biomedical Mass Spectrometry*, 11:601–631.

Thébault, S., Machour, N., Perrot, F., Jouenne, T., Lange, C., Hubert, M., Fontaine, M., Tron, F., and Charlionet, R. (2001). Objet et évolution méthodologique de l'analyse protéomique. *médecine/sciences*, 17:609–618.

The C. elegans Sequencing Consortium (1998). Genome sequence of the nematode *C. elegans*: A platform for investigating biology. *Science*, 282:2012 – 2018.

Tsur, D., Tanner, S., Zandi, E., Bafna, V., and Pevzner, P. A. (2005). Identification of post-translational modifications via blind search of mass-spectra. *Proc IEEE Comput Syst Bioinform Conf*, pages 157–166.

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A., Gocayne, J. D., Amanatides, P., Ballew, R. M., Huson, D. H., Wortman, J. R., Zhang, Q., Kodira, C. D., Zheng,

X. H., Chen, L., Skupski, M., Subramanian, G., Thomas, P. D., Zhang, J., Miklos, G. L. G., Nelson, C., Broder, S., Clark, A. G., Nadeau, J., McKusick, V. A., Zinder, N., Levine, A. J., Roberts, R. J., Simon, M., Slayman, C., Hunkapiller, M., Bolanos, R., Delcher, A., Dew, I., Fasulo, D., Flanigan, M., Florea, L., Halpern, A., Hannenhalli, S., Kravitz, S., Levy, S., Mobarry, C., Reinert, K., Remington, K., Abu-Threideh, J., Beasley, E., Biddick, K., Bonazzi, V., Brandon, R., Cargill, M., Chandramouliswaran, I., Charlab, R., Chaturvedi, K., Deng, Z., Francesco, V. D., Dunn, P., Eilbeck, K., Evangelista, C., Gabrielian, A. E., Gan, W., Ge, W., Gong, F., Gu, Z., Guan, P., Heiman, T. J., Higgins, M. E., Ji, R. R., Ke, Z., Ketchum, K. A., Lai, Z., Lei, Y., Li, Z., Li, J., Liang, Y., Lin, X., Lu, F., Merkulov, G. V., Milshina, N., Moore, H. M., Naik, A. K., Narayan, V. A., Neelam, B., Nusskern, D., Rusch, D. B., Salzberg, S., Shao, W., Shue, B., Sun, J., Wang, Z., Wang, A., Wang, X., Wang, J., Wei, M., Wides, R., Xiao, C., Yan, C., Yao, A., Ye, J., Zhan, M., Zhang, W., Zhang, H., Zhao, Q., Zheng, L., Zhong, F., Zhong, W., Zhu, S., Zhao, S., Gilbert, D., Baumhueter, S., Spier, G., Carter, C., Cravchik, A., Woodage, T., Ali, F., An, H., Awe, A., Baldwin, D., Baden, H., Barnstead, M., Barrow, I., Beeson, K., Busam, D., Carver, A., Center, A., Cheng, M. L., Curry, L., Danaher, S., Davenport, L., Desilets, R., Dietz, S., Dodson, K., Doup, L., Ferriera, S., Garg, N., Gluecksmann, A., Hart, B., Haynes, J., Haynes, C., Heiner, C., Hladun, S., Hostin, D., Houck, J., Howland, T., Ibegwam, C., Johnson, J., Kalush, F., Kline, L., Koduru, S., Love, A., Mann, F., May, D., McCawley, S., McIntosh, T., McMullen, I., Moy, M., Moy, L., Murphy, B., Nelson, K., Pfannkoch, C., Pratts, E., Puri, V., Qureshi, H., Reardon, M., Rodriguez, R., Rogers, Y. H., Romblad, D., Ruhfel, B., Scott, R., Sitter, C., Smallwood, M., Stewart, E., Strong, R., Suh, E., Thomas, R., Tint, N. N., Tse, S., Vech, C., Wang, G., Wetter, J., Williams, S., Williams, M., Windsor, S., Winn-Deen, E., Wolfe, K., Zaveri, J., Zaveri, K., Abril, J. F., Guigó, R., Campbell, M. J., Sjolander, K. V., Karlak, B., Kejariwal, A., Mi, H., Lazareva, B., Hatton, T., Narechania, A., Diemer, K., Muruganujan, A., Guo, N., Sato, S., Bafna, V., Istrail, S., Lippert, R., Schwartz, R., Walenz, B., Yooseph, S., Allen, D., Basu, A., Baxendale, J., Blick, L., Caminha, M., Carnes-Stine, J., Caulk, P., Chiang, Y. H., Coyne, M., Dahlke, C., Mays, A., Dombroski, M., Donnelly, M., Ely, D., Esparham, S., Fosler, C., Gire, H., Glanowski, S., Glasser, K., Glodek, A., Gorokhov, M., Graham, K., Gropman, B., Harris, M., Heil, J., Henderson, S., Hoover, J., Jennings, D., Jordan, C., Jordan, J., Kasha, J., Kagan, L., Kraft, C., Levitsky, A., Lewis, M., Liu, X., Lopez, J., Ma, D., Majoros, W., McDaniel, J., Murphy, S., Newman, M., Nguyen, T., Nguyen, N., Nodell, M., Pan, S., Peck, J., Peterson, M., Rowe, W., Sanders, R., Scott, J., Simpson, M., Smith, T., Sprague, A., Stockwell, T., Turner, R., Venter, E., Wang, M.,

Wen, M., Wu, D., Wu, M., Xia, A., Zandieh, A., and Zhu, X. (2001). The sequence of the human genome. *Science*, 291(5507):1304–1351.

Wilkins, M., Williams, K., and Appel, R. (1997). *Proteome Research: New Frontiers in Functional Genomics*. Springer.

Wilkins, M. R., Gasteiger, E., Gooley, A. A., Herbert, B. R., Molloy, M. P., Binz, P.-A., Ou, K., Sanchez, J.-C., Bairoch, A., Williams, K. L., and Hochstrasser, D. F. (1999). High-throughput mass spectrometric discovery of protein post-translational modifications. *J. Mol. Biol.*, 289:645–657.

Yates, J. R., Speicher, S., Griffin, P. R., and Hunkapiller, T. (1993). Peptide mass maps: a highly informative approach to protein identification. *Anal Biochem*, 214(2):397–408.

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

A Reference Data

A.1 Proteolytic Enzymes

Table A.1: Proteolytic enzymes included in the current implementation. Each enzyme is defined by a set of cleavage recognition patterns which are formatted as comma-separated lists of regular expression–like strings. Letters correspond to the one-letter amino acid symbols and '#' marks the cleavage position.

Name	Rules
Trypsin	`K#[^P],R#[^P]`
Arg-C	`R#P`
Asp-N	`#B,#D`
Asp-N_ambic	`#D,#E`
Chymotrypsin	`F#[^P],Y#[^P],W#[^P],L#[^P]`
CNBr	`M#`
Formic_acid	`D#`
Lys-C	`K#[^P]`
Lys-C/P	`K#`
PepsinA	`F#,L#`
Tryp-CNBr	`K#[^P],R#[^P],M#[^P]`
TrypChymo	`F#[^P],Y#[^P],W#[^P],L#[^P],K#[^P],R#[^P]`
Trypsin/P	`K#,R#`
V8-DE	`B#[^P],D#[^P],E#[^P],Z#[^P]`
V8-E	`E#[^P],Z#[^P]`

A.2 Amino Acid Masses

Table A.2: Amino acid masses used in the theoretical digestion procedure. Monoisotopic and average masses are given in Dalton.

Name	3-Letter	1-Letter	Mono. Mass	Avg. Mass	Composition
Alanine	Ala	A	71.03712	71.08	C3H5NO
Arginine	Arg	R	156.10112	156.19	C6H12N4O
Asparagine	Asn	N	114.04293	114.10	C4H6N2O2
Asparticacid	Asp	D	115.02695	115.09	C4H5NO3
Cysteine	Cys	C	103.00919	103.14	C3H5NOS
Glutamicacid	Glu	E	129.04260	129.12	C5H7NO3
Glutamine	Gln	Q	128.05858	128.13	C5H8N2O2
Glycine	Gly	G	57.02147	57.05	C2H3NO
Histidine	His	H	137.05891	137.14	C6H7N3O
Isoleucine	Ile	I	113.08407	113.16	C6H11NO
Leucine	Leu	L	113.08407	113.16	C6H11NO
Lysine	Lys	K	128.09497	128.17	C6H12N2O
Methionine	Met	M	131.04049	131.19	C5H9NOS
Phenylalanine	Phe	F	147.06842	147.18	C9H9NO
Proline	Pro	P	97.05277	97.12	C5H7NO
Serine	Ser	S	87.03203	87.08	C3H5NO2
Threonine	Thr	T	101.04768	101.10	C4H7NO2
Selenocysteine	SeC	U	150.95364	150.03	C3H5NOSe
Tryptophan	Trp	W	186.07932	186.21	C11H10N2O
Tyrosine	Tyr	Y	163.06333	163.18	C9H9NO2
Valine	Val	V	99.06842	99.13	C5H9NO

A.3 Atomic Masses

Table A.3: Atomic masses used in the theoretical digestion procedure. Monoisotopic and average masses are given in Dalton.

Symbol	Name	Mono. Mass	Avg. Mass
H	Hydrogen	1.007825035	1.00794
H2	Deuterium	2.014101779	2.014101779
Li	Lithium	7.016003	6.941
C	Carbon	12	12.0107
C13	Carbon13	13.00335483	13.00335483
N	Nitrogen	14.003074	14.0067
N15	Nitrogen15	15.00010897	15.00010897
O	Oxygen	15.99491463	15.9994
O18	Oxygen18	17.9991603	17.9991603
F	Fluorine	18.99840322	18.9984032
Na	Sodium	22.9897677	22.98977
P	Phosphorous	30.973762	30.973761
S	Sulfur	31.9720707	32.065
Cl	Chlorine	34.96885272	35.453
K	Potassium	38.9637074	39.0983
Ca	Calcium	39.9625906	40.078
Fe	Iron	55.9349393	55.845
Ni	Nickel	57.9353462	58.6934
Cu	Copper	62.9295989	63.546
Zn	Zinc	63.9291448	65.409
Br	Bromine	78.9183361	79.904
Se	Selenium	79.9165196	78.96
Mo	Molybdenum	97.9054073	95.94
Ag	Silver	106.905092	107.8682
I	Iodine	126.904473	126.90447
Au	Gold	196.966543	196.96655
Hg	Mercury	201.970617	200.59

Affidavit

I hereby declare that the following master thesis "Prediction of Post-translational Modifications of Proteins from 2-DE/MS Data" has been written only by the undersigned and without any assistance from third parties.

Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Berlin, 24.01.2007

Axel Rack