Molecular Formula Identification using High Resolution Mass Spectrometry

Algorithms and Applications in Metabolomics and Proteomics

Dissertation
zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)
vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena
von Dipl.-Ing. Anton Pervukhin
geboren am 29. Juli 1982 in Tscheljabinsk

Gutachter:

1. **Prof. Dr. Sebastian Böcker, Friedrich-Schiller-Universität Jena**
2. **Prof. Dr. Jens Stoye, Universität Bielefeld**

Tag der öffentlichen Verteidigung: 8. Dezember 2009

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706

Zusammenfassung

Wir untersuchen mehrere theoretische und praktische Aspekte der Identifikation der Summenformel von Biomolekülen mit Hilfe von hochauflösender Massenspektrometrie.

Durch die letzten Forschritte in der Instrumentation ist die Massenspektrometrie (MS) zur einen der Schlüsseltechnologien für die Analyse von Biomolekülen in der Proteomik und Metabolomik geworden. Sie misst die Massen der Moleküle in der Probe mit hoher Genauigkeit, und ist für die Messdatenerfassung im Hochdurchsatz gut geeignet. Eine der Kernaufgaben in der MS-basierten Proteomik und Metabolomik ist die Identifikation der Moleküle in der Probe. In der Metabolomik unterliegen Metaboliten der Strukturaufklärung, beginnend bei der *Summenformel* eines Moleküls, d.h. der Anzahl der Atome jedes Elements. Dies ist der entscheidende Schritt in der Identifikation eines unbekannten Metabolits, da die festgelegte Formel die Anzahl der möglichen Molekülstrukturen auf eine viel kleinere Menge reduziert, die mit Methoden der automatischen Strukturaufklärung weiter analysiert werden kann. Nach der Vorverarbeitung ist die Ausgabe eines Massenspektrometers eine Liste von Peaks, die den Molekülmassen und deren Intensitäten, d.h. der Anzahl der Moleküle mit einer bestimmten Masse, entspricht. Im Prinzip können die Summenformel kleiner Moleküle nur mit präzisen Massen identifiziert werden. Allerdings wurde festgestellt, dass aufgrund der hohen Anzahl der chemisch legitimer Formeln in oberen Massenbereich eine exzellente Massengenaugkeit alleine für die Identifikation nicht genügt. Hochauflösende MS erlaubt die Bestimmung der Molekülmassen und Intensitäten mit hervorragender Genauigkeit.

In dieser Arbeit entwickeln wir mehrere Algorithmen und Anwendungen, die diese Information zur Identifikation der Summenformel der Biomolekülen anwenden. Im ersten Teil stellen wir einen Ansatz zur Bestimmung der Summenformel eines Metabolits durch seine Masse und die natürliche Verteilung seiner Isotopen vor. Wir führen den Begriff "Isotopenmuster" ein und zeigen die Methoden für dessen schnelle Berechnung. Wir evaluieren unseren Algorithmus auf mehreren experimentellen Datensätzen und erreichen vielversprechende Ergebnisse mit geringem Fehleranteil für die Moleküle unter 1 000 Da für orthogonale Flugzeitmassenspektrometrie. Des Weiteren haben wir eine Methode entwickelt, um die Aminosäuresequenz eines unbekanntes Proteins aus seiner Summenformel sich herzuleiten. Wir formulieren das Problem als mehrdimensionales Equality-Constrained-Integer-Knapsack-Problem, und präsentieren effiziente Methoden der Maßreduktion, um alle Problemlösungen aufzuzählen.

Im zweiten Teil entwickeln wir mehrere Anwendungen, die unsere algorithmischen Ansätze implementieren und für die Analyse der MS-Daten kleiner Biomoleküle angewandt werden können. Wir präsentieren Decomp, eine web-basierte Anwendung für die Massenzerlegung über einen beliebigen Alphabet, und zeigen ihre Anwendbarkeit als Teil eines Software-Werkzeuges CompNovo für die *de-novo*-Sequenzierung von Peptiden durch Tandem-MS. Schließlich stellen wir die Java-basierte Software SIRIUS vor, die unsere Algorithmen zur Identifikation der Summenformel von Metaboliten implementiert, und mit einer leicht bedienbaren graphischen Benutzeroberfläche kombiniert.

Acknowledgements

This work would not have been possible without the support of many people.

First of all, I would like to thank Prof. Dr. Sebastian Böcker, who has been a great supervisor over these years, sharing lots of ideas, providing dozens of useful insights into problems, dedicating much time for his students, and simply supporting them at all levels. For me, working with Sebastian has been an incomparable and very valuable experience. Also, I would like to thank Prof. Dr. Jens Stoye at Bielefeld University, who has been for me an example of a brilliant organizer and supportive mentor. Working with Jens in the group Genome Informatics in Bielefeld, I could particularly appreciate the opportunity to study in a motivated and yet very friendly atmosphere, in which things were getting managed as if by themselves.

I am grateful to the Deutsche Forschungsgemeinschaft (DFG), which has financed me within the Computer Science Action Program (BO 1910/1).

I would like to express my gratitude to Dr. Michael Jung from the bj-diagnostik GmbH, Gießen, for his kindness and immense support over the period of time before beginning this PhD thesis.

I wish to acknowledge Dr. Dirk Evers, Silke Kölsch, and the whole International NRW Graduate School in Bioinformatics and Genome Research, where I had an opportunity to study during the first one and a half years at Bielefeld University.

I express many thanks to Dr. Hans-Michael Kaltenbach who has been a great office mate in Bielefeld. I am also grateful to Dr. Zsuzsanna Lipták, who has been a great co-author and, in some sense, an elder tutor for me in my first publications. I have been learning from Zsuzsa how to write clear and well-formulated papers with grammatically correct English. I thank Marcel Martin, Henner Sudek, and Matthias Steinrücken who helped me a lot in getting acquainted with realities of studying at Bielefeld University. I also wish to thank Heike Samuel for her kindness and support during my first days at my first German university.

For the very useful and successful cooperation, I wish to acknowledge the following scientists: Dr. Matthias Letzel at Bielefeld University, Dr. Steffen Neumann at Leibniz Institute of Plant Biochemistry (IPB) Halle, and Andreas Bertsch at Tübingen University. I wish to thank Henning Mersch and Jan Krüger for their help with installing Decomp at the Bielefeld University Bioinformatics Server (BiBiServ).

As during this PhD thesis, I had to change the location of my study, I would like to thank Nicole Hinz, Frank Mäurer, and Anke Truß who helped me a lot to easily change the university, and to continue research at Friedrich-Schiller-University Jena.

In particular, I would like to thank my office neighbor Thasso Griebel, who joined our newly created group at Jena University. I feel myself lucky that he landed at my office, so that we could share a lot of ideas and have many useful discussions, in particular, regarding the software design and development. Without his support in the initial stage of creating SIRIUS, its release would have been much more complicated matter. I thank Florian Rasche who came recently and could quickly adopt himself in our group, and he is going to take over in the further development and maintenance of SIRIUS. I also thank Martin Engler and Franzisca Hüfsky for their help with SIRIUS.

I wish to acknowledge Frank Mäurer for his help in correcting student assignments, and Anke Truß and Quang Bao Anh Bui for being great coworkers in holding various exercises and seminars for students. And special thanks to Kathrin Schwotka, our beautiful secretary, for her kindness and patience in helping us to get done all administrative duties that typically accompany the academic research.

I thank Malte Brinkmeyer, Thasso Griebel, Florian Rasche, and Anke Truß for proofreading parts of this thesis.

Finally, I would like to thank my parents, Gennady and Zareen, and my sister Vedanta, who have accepted my long term leaves, and eventual move to Germany. I am deeply grateful to them for their love and support, whenever being necessary.

Contents

1 Introduction **1**

1.1 Structure of the Thesis 2

2 Biological Background **5**

2.1 Atoms and Molecules 5

2.2 Proteomics 8

2.3 Metabolomics 11

2.4 Mass Spectrometry 12

2.4.1 Instrumentation 12

2.4.2 Experimental Workflow 18

2.5 Mass Spectrometry Data Analysis 21

2.5.1 Types of Mass Spectrometric Analysis 21

2.5.2 Computational Problems 22

2.5.3 Protein Identification using Databases 24

3 Decomposition Algorithms **27**

3.1 Integer Mass Decomposition 27

3.1.1 Definitions and Problems 27

3.1.2 Enumerating Integer Decompositions 29

3.2 Decomposing Real-valued Masses 31

3.2.1 Approximating Number of Decompositions 33

4 Molecular Formula Identification of Metabolites **37**

4.1 Isotope Patterns 38

4.1.1 Isotope Species 38

4.1.2 Isotopic Distributions 39

4.2 Computing Isotope Patterns 41

4.2.1 Folding Isotopic Distributions 41

4.2.2 Folding Peak Masses 42

4.3 Scoring Candidate Molecules 44

4.3.1 Estimating Probabilities of Peak Masses 45

4.3.2 Estimating Probabilities of Peak Intensities 46

4.4 Experimental Results 46

4.4.1 Identification Rates 48

5 Deriving Peptide Compositions 53
5.1 Peptide Molecular Formula Decomposition Problem 54
5.1.1 Related Problems and Solutions 55
5.1.2 Multi-dimensional Integer Knapsack 57
5.2 Generating Decomposition Matrices and a Mixed Matrix Approach 59
5.3 Experimental Results 60
5.3.1 Selecting Good Decomposition Matrices 60
5.3.2 Comparison with Other Methods 61
5.4 Summary 62
5.5 Best Six Matrix Pairs 63

6 Application Tools and Cases 71
6.1 DECOMP 71
6.1.1 Introduction 71
6.1.2 Implementation and Use 72
6.2 Application Case with CompNovo 75
6.2.1 Existing Approaches for De Novo Peptide Sequencing by Tandem MS 77
6.2.2 Algorithm Overview 79
6.2.3 Experimental Results 82
6.3 Rdisop 84
6.3.1 Introduction 84
6.3.2 Implementation and Use 85

7 SIRIUS 91
7.1 System Architecture 92
7.1.1 Domain Objects 92
7.1.2 Functional Logic Layer 94
7.1.3 Presentation Layer 96
7.2 Application Workflow 98
7.2.1 Preparing Input Data 98
7.2.2 Analysis Preparation and Parameter Input 99
7.2.3 Analyzing Algorithm Results 101
7.3 Summary 103

8 Conclusion 105

1 Introduction

> The process of scientific discovery is, in effect, a continual flight from wonder.
>
> Albert Einstein (1879-1955)

All life on this planet basically depends on three types of molecules: DNA, RNA, and proteins. In macroscopic terms, one could visualize a cell roughly analogous to the university library, where DNA are the book shelves holding all information about how the cell works, RNA are the librarians who transfer books, small pieces of this knowledge to students representing proteins that perform the actual tasks of the cell. In fact, this is rather an oversimplified view: There exist other types of molecules, such as metabolites that play a crucial role in maintaining the cell's structure as well as in the cell growth.

The sequencing of numerous genomes[1] in the last two decades has stimulated impressive advances in the biological sciences, but emphasized our limited understanding about the "building blocks" of biological systems. The post-genomic era has progressed by shifting the focus of bioinformatics research from structural towards functional genomics: attention has grown to products of the later steps of protein expression and general functioning of the cell. For a better understanding of the biochemical and biological mechanisms in complex systems, it is necessary to comprehend an organism's response to a conditional perturbation at the transcriptome, proteome and metabolome levels. The respective areas are referred to as *transcriptomics*, *proteomics*, and *metabolomics*, while the eventual goal of interpretation of the whole system with its complexities is referred to as *systems biology.*

Mass spectrometry has been one of the workhorse analytical tools over the last quarter century. Mass spectrometers are extensively used by pharmaceutical companies to analyze small and thermostable compounds for drug research. With rapid advances in instrumentation and capability to perform high-throughput analysis, mass spectrometry has advanced by leaps and bounds and become a central analytical technique for protein and metabolite research and for the study of biomolecules in general.

One of the key challenges in mass spectrometry-based proteomics and metabolomics is to establish the identity of sample molecules. In metabolomics each individual compound *(metabolite)* needs to undergo structural elucidation, starting from the elemental composition or *molecular formula*, i.e., the number of atoms of each element. This is an essential step in identifying a metabolite, since a fixed formula reduces the number of possible molecular structures to a much smaller set that can be further examined for automatic structure elucidation of molecules. After preprocessing, the output of a mass

---

[1] `http://www.genomenewsnetwork.org`

spectrometer is a list of peaks which corresponds to the masses of the sample molecules and their abundances, i.e., the amount of sample compounds with a certain mass. In principle, molecular formulas of small molecules can be identified using only accurate masses. However, due to many chemically possible formulas in higher mass regions, it has become evident that excellent mass accuracy alone is insufficient for the identification purposes.

High resolution mass spectrometry allows us to determine the isotope pattern of sample molecules with outstanding accuracy. In this work, we develop several algorithms that extensively make use of this information for the identification of the molecular formula of sample molecules. To achieve this, we use and extend recently developed efficient techniques for decomposition of integer and real-valued masses. We present a web-based tool DECOMP for solving integer and real-valued mass decomposition problems over any arbitrary alphabet including several common "bioalphabets", such as amino acids, nucleotides, and chemical elements most frequently occurring in nature. We demonstrate the applicability of our decomposition approach for *de novo* sequencing of peptides using tandem mass spectra. Moreover, we present a novel algorithm to go one step further and use the information about the molecular formula to derive the amino acid sequence of an unknown peptide. To generate all amino acid sequences from a peptide's molecular formula, our method efficiently solves a joint decomposition of a set of queries on the number of elements that each amino acid consists of. The performance of our approach has been evaluated on both simulated and real proteomics data. Furthermore, we introduce Rdisop, a new R package for *de novo* identification of molecular formulas solely from masses and isotope patterns measured by high resolution mass spectrometers. Finally, we present the java-based software application, called SIRIUS, that implements all of our algorithms for identification of the molecular formula of metabolites, and combines them with an easy-to-use graphical user interface.

Parts of this dissertation thesis have been published in advance [15, 17, 16, 20], and one further paper is accepted for publication [12]. DECOMP is freely accessible at the Bielefeld University Bioinformatics Server[2] (BiBiServ). Rdisop is distributed as a part of the Bioconductor project, and is publicly accessible at the project website[3]. SIRIUS is publicly available for download at the project website[4], and is distributed for various operating systems including Unix/Linux, Windows and Mac OS.

1.1 Structure of the Thesis

This thesis consists of eight chapters. In Chapter 2, an introduction into the basic terms and principles in computational molecular biology, in particular, the emerging fields of proteomics and metabolomics, is given. We introduce mass spectrometry (MS) and its biotechnological aspects, and outline the computational problems that arise in its context.

---

[2] `http://bibiserv.techfak.uni-bielefeld.de/decomp/`

[3] `http://www.bioconductor.org/packages/2.4/bioc/html/Rdisop.html`

[4] `http://bio.informatik.uni-jena.de/sirius/`

The thesis is organized as follows: The first part (Chapters 3–5) is devoted to the theoretical concepts of our research. In Chapter 3, we start by a brief detour to existing approaches for decomposing integer masses, a question that is frequently encountered in the MS data analysis. We define the relevant problems and introduce available computational solutions, which are further used as a basis in all our algorithms and applications. We also show how to extend the integer decomposition techniques for the analysis of real-valued MS data.

Chapter 4 presents a new approach for *de novo* identification of molecular formulas of metabolites, which uses the mass decomposition techniques, and incorporates further information, such as *isotopic abundance* data. We introduce an *isotope pattern* and related notions, and present methods for the fast simulation of isotope patterns, which is important for the analysis of larger molecules where the search space increases rapidly. We evaluate our approach on several experimental datasets using different MS techniques, and obtain very promising results with only a tiny proportion of false identifications for molecules below 1 000 Da for orthogonal time-of-flight mass spectrometry.

In Chapter 5, we present a new algorithm that employs the molecular formula information to efficiently infer the amino acid composition of an unknown peptide. We formulate the problem as a joint set of decomposition queries based on the number of elements that each amino acid consists of, and present a dimension reduction technique to reduce a multi-dimensional problem to a one-dimensional instance. We also provide an experimental evaluation of the algorithm performance, both on simulated data and peptides from experimental mass spectra.

The second part (Chapters 6 and 7) is devoted to the practical aspects of our work. In Chapter 6, we present several application tools that implement our algorithmic approaches, and can be utilized for the MS data analysis of small sample molecules. We present DECOMP, a web-based application to find decompositions of a given mass over any arbitrary alphabet. Being designed to solve integer and real-valued mass decomposition problems, DECOMP is well suited both for the interpretation of MS data and for solving instances of Money Changing Problem. We then demonstrate its applicability as an essential part of another algorithm, called CompNovo, for *de novo* peptide sequencing using tandem mass spectrometry. We also present Rdisop, a new R package for the analysis of small sample molecules using an accurate isotope pattern information.

In Chapter 7, we present the java-based software SIRIUS with the implementation of all our algorithms for *de novo* identification of molecular formula of metabolites using high resolution MS data. We describe the architecture and provide some technical details on implementations and employed technologies. Well-defined structure and management system of our software allow a simple integration of new computation and visualization functionalities to the application. We further describe a basic application workflow and outline the cornerstones of the data and analysis preparation. We discuss a set of SIRIUS features for the interpretation of the computational results including visualization, data export, and searching for molecular formulas in biological databases.

Chapter 8 concludes the thesis by recalling the main results and presenting an outlook on further applications of the molecular formula analysis for the identification of unknown sample molecules.

2 Biological Background

We start by providing some basic physical and chemical background, which we further use throughout this work. Since the motivation of our work as well as its applications and results have their origin in proteomics and metabolomics, this chapter will give an overview of the central terms and principles of these two emerging fields of scientific research. Main focus of our work also lies in the context of mass spectrometry, thus we will briefly describe the basics of mass spectrometry, and outline the computational challenges that appear in its context. This chapter cannot be a self-contained introduction to mass spectrometry or to the fields of proteomics and metabolomics, and for further information, the interested reader might have a look at any of the relevant textbooks, e.g., [57, 34, 125].

2.1 Atoms and Molecules

Atoms and Isotopes. The word "atom" comes from the Greek word *átomos*, which means uncuttable: Atoms are basic units of matter that cannot be decomposed chemically. Atoms consist of a central *nucleus* surrounded by negatively charged electrons. The atomic nucleus is made up from positively charged protons and neutrons which have no charge. Atoms contain the same number of protons and electrons, and thus have no charge; if this charge is broken, the resulting particle is called an *ion*.

Atoms are classified according to the number of protons and neutrons in their nucleus. The total number of protons and neutrons is called the *nominal mass*, or the *nucleon number* of an atom. The number of protons is referred to as the *atomic number* and determines the chemical element of an atom. Atoms with equal atomic numbers share the same chemical behavior and cannot be distinguished chemically. The elements most abundant in living beings are hydrogen (symbol H) with atomic number 1, carbon (C, atomic number 6), nitrogen (N, 7), oxygen (O, 8), and to lesser extent, phosphor (P, 15), and sulfur (S, 16). Throughout this work, we will refer to this set as CHNOPS.

Unlike the fixed number of protons, the number of neutrons of an element can vary: Atoms with equal number of protons and electrons but different number of neutrons define the *isotopes* of an element. For example, carbon has two isotopes that occur in nature, ^{12}C and ^{13}C (the preceding superscript denotes the nucleon number): ^{12}C is comprised of six protons, six electrons, and six neutrons, whereas ^{13}C carries six protons, six electrons, and seven neutrons.[1] The lightest isotope, such as ^{1}H or ^{12}C, is also called *monoisotopic*. Note that the term "monoisotopic" is sometimes referred to as the most

---

[1] In fact, carbon has the third radioactive isotope ^{14}C. Radioactive isotopes are usually ignored in mass spectrometric analysis.

element (symbol)	isotope	abundance %	mass (Da)	av. mass (Da)
hydrogen (H)	^{1}H	99.985	1.007825	
	^{2}H	0.015	2.014102	1.007976
carbon (C)	^{12}C	98.890	12.0	
	^{13}C	1.110	13.003355	12.011137
nitrogen (N)	^{14}N	99.634	14.003074	
	^{15}N	0.366	15.001090	14.006727
oxygen (O)	^{16}O	99.762	15.994915	
	^{17}O	0.038	16.999131	
	^{18}O	0.200	17.999160	15.999305
phosphor (P)	^{31}P	100	30.973762	30.973762
sulfur (S)	^{32}S	95.020	31.972071	
	^{33}S	0.750	32.971459	
	^{34}S	4.210	33.967867	
	^{36}S	0.020	35.967081	32.064388

proton (p^+, $^{1}H^{+}$) 1.00728 Da, neutron (n) 1.008665 Da, electron (e^-) 0.00054 Da

Table 2.1: Natural isotopic distribution: Relative abundance of isotopes and their masses in Dalton.

abundant isotope, but in this work we use this term to refer to the isotope with the smallest mass. Isotopes of each element are found in nature with certain abundance: For example, the relative abundance of the monoisotopic carbon isotope ^{12}C is 98.89 %, whereas the isotope ^{13}C has the relative abundance of 1.11 %. For abundances of other isotopes most commonly occurring in nature, see Table 2.1.

Masses of atoms are measured in *Dalton* (Da), or equivalently in *unified atomic weight units* (u). According to *International Union of Pure and Applied Chemistry* (IUPAC), one Dalton is defined as 1/12 of the mass of one atom of the ^{12}C isotope. In fact, due to the mass contained in the binding energy of the atom's nucleus, an atom with n protons and neutrons has a mass, which roughly, but not exactly, equals to n Da. This illustrates the *mass defect*, the difference between the atom's nominal mass and the sum of masses of the constituting protons, neutrons, and electrons. For example, the total mass of 6 protons, 6 neutrons, and 6 electrons is equal to 12.09596 Da, while the ^{12}C isotope has a mass of exactly 12.0 Da, a difference of about 0.8 %. The *average mass* of an element is the expected mass computed with respect to the relative abundances of isotopes. For example, the average mass of carbon is 0.98890 times the mass of ^{12}C plus 0.01110 times the mass of ^{13}C. For the masses of elements listed above, see Table 2.1, while the detailed list of all chemical elements can be found in [4].

Molecules. A *molecule* is a stable group of two or more atoms joined together by chemical bonds of shared pairs of electrons. A *molecular formula* indicates the exact number of atoms that compose a molecule. Compared to the chemical formula that may provide information about the structure and types of chemical bonds, the molecular formula only reflects the amount of atoms in the molecule. For example, the chemical formula of the amino acid alanine $CH_3CH(NH_2)COOH$ implies a chain of three carbon atoms, with an α-carbon (see Section 2.2) surrounded by an amino group (NH_2), a carboxyl group (COOH), and a methyl group (CH_3), whereas the molecular formula $C_3H_7NO_2$ tells us only that the molecule contains 3 carbons, 7 hydrogens, 1 nitrogen, and 2 oxygens. Molecules can have different amount of protons and electrons, thus being positively or negatively *charged*. Molecules that pick up a net electric charge are called *ions*, or in the context of mass spectrometry, they are often referred to as *molecular ion adducts*.

Chemical Bonding Rules. Atoms in the molecule are held together by the chemical bonds formed through the pairs of electrons that atoms can share with their neighbours. The number of chemical bonds formed by atoms is often referred to as the *valence* of an element. For example, in natural compounds carbon has valence 4, oxygen 2, and hydrogen 1. However, for many elements, the valence varies depending on the molecule to which this element belongs. For example, phosphor sometimes forms 5 chemical bonds and sometimes only 3. For the physical existence of a molecule as a set of interconnected elements, certain chemical rules, such as a valence balance and unsaturation, must hold. In its stable state, the molecule is an electrically neutral entity with the same number of protons and electrons. Therefore, the valence states of all its constituting atoms must be balanced. For example, if one atom would have valence $+1$, meaning it lacks one electron, and another atom would have valence -1, meaning it possesses an additional electron, then a bond between these two atoms would be formed to complement their unbalanced valence states.

Another prerequisite of the legitimate compound is the number and types of bonds that must be present. A molecule only containing single bonds is called *saturated*, while the presence of multiple bonds is known as *unsaturation*. The *degree of unsaturation* (DU) [96] formula, also known as *rings-plus-double-bonds equivalent* (RDBE) [114], helps to determine the number of different types of bonds in the compound, and therefore can be used for validation of molecular formulas.

In 1951, Senior [107] proposed a graph-theoretical concept to unify the aspects of valence balance and unsaturation. Senior's theorem outlines the three prerequisites for the existence of a molecular graph:

1. The sum of valences or the total number of atoms having odd valences is even;
2. The sum of valences is greater than or equal to twice the maximum valence;
3. The sum of valences is greater than or equal to twice the number of atoms minus one.

The first condition corresponds to the valence balance, whereas the third condition accounts for connectivity of a molecular graph requiring the number of independent cycles in the graph to be nonnegative. The second prerequisite ensures the non-existence of small molecules such as CH_2.

There exist other chemical criteria, such as a *nitrogen rule*, or a *Lewis octet rule*, but they are either rarely used for mass spectrometric analysis, or can be seen as a subset of Senior's conditions.

2.2 Proteomics

The word "proteome" was coined in 1994 to denote the total amount of PROTEins expressed by a genOME, cell, tissue or organism [91]. More specifically, it is the set of expressed proteins at a given time under defined conditions. Unlike the genome, the proteome is very dynamic: Proteins are constantly formed and degraded in different cell types and their expression is influenced by the environment and events going on inside the cell, such as cell division and growth state. Proteins participate in almost all cellular processes: They are part of the cell membrane, form enzymes that perform biochemical reactions, and serve as receptors and transmitters in signal transduction. Identification of all the proteins in a cell or an organism is one of the major tasks of *proteomics* – the study of the proteome.

Protein Structure. Proteins are *polypeptides*: They form a chain of *amino acid* molecules linked together in a certain order by peptide bonds.

There are 20 different naturally occurring amino acids, all sharing the same common structure: A central carbon atom (often denoted by α (alpha) carbon) surrounded by an amino group (HN_2), a carboxyl group (COOH), a hydrogen atom and a side chain specific for the amino acid. The generalized structure of an amino acid is depicted in Figure 2.1. Each amino acid can be denoted either by its full name, symbol, or three-letter abbreviation, see Table 2.2. Note that *leucine* (L) and *isoleucine* (I) have identical molecular formula and usually cannot be distinguished by mass spectrometry. In the following, we thus consider these two amino acids as one, and talk about 19 standard amino acids.

From the structure shown in Figure 2.1, it is obvious that an amino acid looses a water molecule (H_2O) when it is included in the chain. Therefore, when amino acids are part of a chain, they are commonly referred to as *residues*. The bonds between two consecutive amino acid residues in the chain are amide bonds, and are denoted as *peptide bonds*. Correspondingly, a short chain of amino acids is usually referred to as a *peptide*. Each peptide has an an unbounded amine group, called the *N-terminus*, and an unbounded carboxyl group, called the *C-terminus*. By convention, the peptide sequence is written as a string of residues (usually denoted by symbols) from the N-terminus to the C-terminus.

The peptide sequence is the most fundamental attribute of a protein and therefore referred to as its *primary structure*: If the sequence is known, the protein is considered identified. To determine the complete peptide sequence is, however, not a straightforward process.

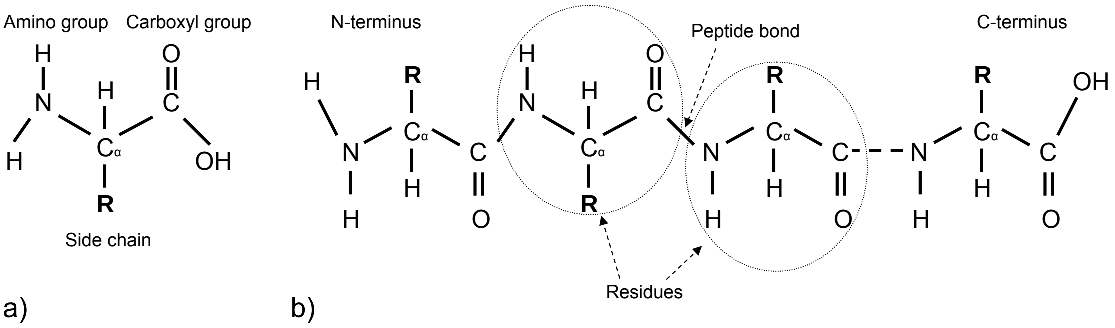


Figure 2.1: a) All amino acids have the same common structure. Different amino acids vary in their side chains. b) A polypeptide is formed by linking together multiple amino acids residues.

Protein Synthesis. In modern molecular biology, the word “genome” refers to the hereditary information of an organism encoded in several *deoxyribonucleic acid* (DNA) molecules, each a double-stranded polymer built from four different *nucleic acids*, or *nucleotides*: adenine (symbol A), cytosine (C), guanine (G) and thymine (T). The genome is mainly static and the same for all cells of an organism. The primary structure of a protein is encoded in a gene, a coding region of the genome.

Protein synthesis is the creation of proteins using DNA and *ribonucleic acid* (RNA) molecules. RNA molecules are very similar to DNA molecules, except that they are single-stranded, use a different ribose in the backbone and have the base *uracil* (U) instead of *thymine* (T). To build a new protein, the gene that encodes this protein must be transcribed and translated. During the transcription, messenger ribonucleic acid (mRNA) molecules are formed that contain a complementary copy of the genetic information. Similar to the genome and the proteome, the *transcriptome* is the set of all mRNA molecules present in a cell at a certain time. Like the proteome, the transcriptome is dynamic.

In the *translation* process, the mRNA molecules, containing the genetic instructions to make the protein, are first brought to the ribosomes, large ribonucleoproteins that are responsible for building the protein’s primary structure from amino acids. This process is done with the help of transfer RNA (tRNA) that transfers the amino acids to the amino acid chain. The actual translation is done by the ribosomes that read the mRNA sequence of nucleic acids and for each triplet of nucleotides, or *codon*, add the encoded amino acid to the so far formed amino acid chain. Codons are consecutive and non-overlapping, that is the next triple of nucleotides after each codon corresponds to the next codon. The start and the end of the translation process are triggered by special start and stop codons. When the ribosome reaches the stop codon, indicating the end of the mRNA sequence, it releases the protein that then folds to its final structure.

This flow of information in a cell,

$$\text{DNA} \quad \rightarrow \quad \textit{transcription} \quad \rightarrow \quad \text{RNA} \quad \rightarrow \quad \textit{translation} \quad \rightarrow \quad \text{protein}$$

amino acid	symb.	TLC	molecular formula	mass (Da)	
				mono.	average
Alanine	A	Ala	$C_3H_5N_1O_1$	71.0371	71.0788
Cysteine	C	Cys	$C_3H_5N_1O_1S_1$	103.0091	103.1448
Aspartic acid	D	Asp	$C_4H_5N_1O_3$	115.0269	115.0886
Glutamic acid	E	Glu	$C_5H_7N_1O_3$	129.0425	129.1155
Phenylalanine	F	Phe	$C_9H_9N_1O_1$	147.0684	147.1766
Glycine	G	Gly	$C_2H_3N_1O_1$	57.0214	57.0520
Histidine	H	His	$C_6H_7N_3O_1$	137.0589	137.1412
Isoleucine	I	Ile	$C_6H_{11}N_1O_1$	113.0840	113.1595
Lysine	K	Lys	$C_6H_{12}N_2O_1$	128.0949	128.1742
Leucine	L	Leu	$C_6H_{11}N_1O_1$	113.0840	113.1595
Methionine	M	Met	$C_5H_9N_1O_1S_1$	131.0404	131.1986
Asparagine	N	Asn	$C_4H_6N_2O_2$	114.0429	114.1039
Proline	P	Pro	$C_5H_7N_1O_1$	97.0527	97.1167
Glutamine	Q	Gln	$C_5H_8N_2O_2$	87.0320	87.0782
Arginine	R	Arg	$C_6H_{12}N_4O_1$	156.1011	156.1876
Serine	S	Ser	$C_3H_5N_1O_2$	87.0320	87.0782
Threonine	T	Thr	$C_4H_7N_1O_2$	101.0476	101.1051
Valine	V	Val	$C_5H_9N_1O_1$	99.0684	99.1326
Tryptophan	W	Trp	$C_{11}H_{10}N_2O_1$	186.0793	186.2133
Tyrosine	Y	Tyr	$C_9H_9N_1O_2$	163.0633	163.1760

Table 2.2: Amino acid residues with 3-letter-code (TLC), symbol, molecular formula, monoisotopic (mono.) and average masses. The molecular formulas for the residues are given without terminal H and OH groups.

is referred to as *the central dogma of molecular biology.* In fact, this is rather an oversimplified view on the transfer of biological information in a cell: Today, we know other factors, such as an alternative splicing of introns, DNA and RNA editing, post-translational modifications (PTM), and others, that make this process more complicated, resulting in a variability of protein molecules translated from a single gene. Here, we omit further details.

Although DNA, RNA and proteins are the main players in the cell's life circle and the three primary types of molecules that biologists study, there are other types of molecules, such as metabolites that play a crucial role in maintaining the cell's structure. In the next section, we address metabolites in more detail.

2.3 Metabolomics

Metabolites are the end products of cellular regulatory processes, and their expressions can be considered as the ultimate response of biological systems to genetic or environmental changes [42]. *Metabolomics* deals with the systematic study of the metabolism of both endogenous and exogenous metabolites present in biological systems. The word "metabolism" is derived from the Greek word *metabolè*, meaning change, which is applicable as metabolism is in a constant and rapid flux. In analogy to genome and proteome, *metabolome* refers to the total amount of metabolites present in a biological system that participate in metabolic reactions such as growth, maintenance and normal function [57]. Metabolites are organic and inorganic species, mostly of small molecular mass. Metabolites are usually divided into two categories: *primary* metabolites are directly involved in growth, development, or reproduction of an organism, whereas *secondary* metabolites are not directly involved in these processes, but have other important biological functions. Endogenous metabolites are biochemically synthesized or catabolized within the cell or organism, whereas exogenous metabolites have an external origin, as is the case for pharmaceuticals and food nutrients consumed by humans.

A typical analytical procedure in metabolomics can be divided in three major steps: First, a biological sample must be collected without causing any change of the metabolome: the metabolite molecules must be extracted and eventually undergo chemical derivatization. Second, the metabolites are separated by chromatographic techniques, such as gas chromatography (GC) or high performance liquid chromatography (HPLC), see Section 2.4. Finally, metabolites are identified and quantified using Mass Spectrometry, or Nuclear Magnetic Resonance (NMR).

Although metabolomics is closest to the phenotype in the "omics" cascade (see Figure 2.2), there currently exists no single-instrument platform that can analyze all metabolites, and the tools for the comprehensive examination of the metabolome are still emerging [13]. In Table 2.3, a list of different strategies is given that are currently employed. The complexity of the metabolome is due to the broad variety of compounds, such as lipids, carbohydrates, vitamins, and others, which comprise the metabolome. These compounds constitute a diverse set of molecular structures when compared to the proteome (string of 20 standard amino acids) and transcriptome (string of 4 nucleotides). The majority of metabolites contain only six elements CHNOPS that most frequently occur in living beings. However, metabolites can also contain other elements, such as chlorine (Cl), fluorine (F), and others [72]. This results in a wide variety of *chemical* (molecular weight, polarity, solubility) and *physical* (volatile) properties of metabolites becoming a challenge for the comprehensive biochemical analysis.

One of the major applications of metabolomics is the detection of biomarkers that change as an indicator of the presence of a disease in an individual biological system. The biomarkers for certain diseases, such as for reversible myocardial ischemia, can be found on metabolomic, rather than on genomic or proteomic level [103]. Also, metabolomics has been employed in a variety of other health applications including pharmacology, pre-clinical drug trials, clinical chemistry, and others. Moreover, the concept of individualized health, such as nutrition or tailored pharmacological treatment based on

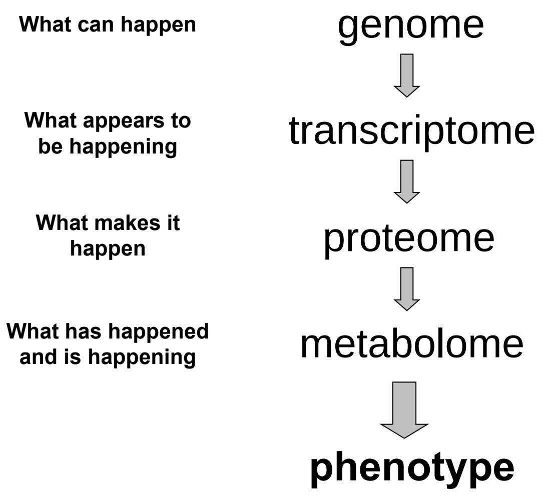


Figure 2.2: The “Omics” cascade contains several levels of the biological system, which can be analyzed with regard to the response of the system to disease, genetic, and environmental changes. Of these levels, the metabolome is the most predictive of phenotype.

metabolic phenotype will strongly rely on metabolomics technology [124].

2.4 Mass Spectrometry

Mass spectrometry is an analytical technique that has played a key role in emerging of proteomics and metabolomics into mainstream science. In the remainder of this chapter, an introduction to the field of mass spectrometry from both experimental and computational points of view is given.

According to Siuzdak [110], a mass spectrometer is “*an analytical device that determines the molecular weight of chemical compounds by separating molecular ions according to their mass-to-charge (m/z) ratio*”. As mentioned before, molecular weights are measured in *Dalton* (Da), or equivalently in *unified atomic weight units* (u). One Dalton equals one atomic weight unit (amu) of $1.66 \cdot 10^{-24}$ g, which is approximately the molecular weight of one proton. The *mass-to-charge* is measured in *Thompson* (Th).

2.4.1 Instrumentation

Very schematically, a mass spectrometer consists of three main parts as shown in Figure 2.3: the *ionization source*, the *mass analyzer*, and the *detector*. Following the sample introduction, the analyte is ionized in an ionization source, either operating at atmospheric or vacuum pressures. The generation of charged molecules is necessary to enable ion manipulation based on its *mass-to-charge* (m/z) ratio. The sample is then transferred to the mass analyzer that separates the components of the sample in space or time

Metabolomics	Identification and quantification of as many of the compounds present in a metabolome sample as possible.
Metabolic profiling	Identification and approximate quantification of a large set of metabolites, generally related by one or more specific metabolite classes, within a metabolite sample. This strategy can be also described as *metabolite profiling* or *untargeted analysis*, and provides metabolite identifications where the compounds of biological interest are not known *a priori*.
Targeted analysis	Identification and precise quantification of a single or highly-related small set of metabolites within a metabolome sample.
Metabolic fingerprinting	High-throughput generation of a global snapshot, or fingerprint, of a metabolome sample without regard for the individual compounds that it contains. Identification and quantification is limited and the strategy is employed for discrimination of samples from different biological origins.

Table 2.3: Typical strategies employed in metabolomic analysis.

according to their m/z ratios. After separation, ions are registered by a detector either physically as an ion current or by the detection of orbital frequencies as image currents. Finally, the mass spectrometer's output – a *mass spectrum* – is acquired by a connected computer. A mass spectrum is a diagram, with the position of the peak along the horizontal axis that ideally indicates the presence of the molecule with the corresponding *m/z* value in the sample, and the height of the peak, referred to as *intensity* or *relative abundance*, which is proportional to the amount of molecules with this m/z value.

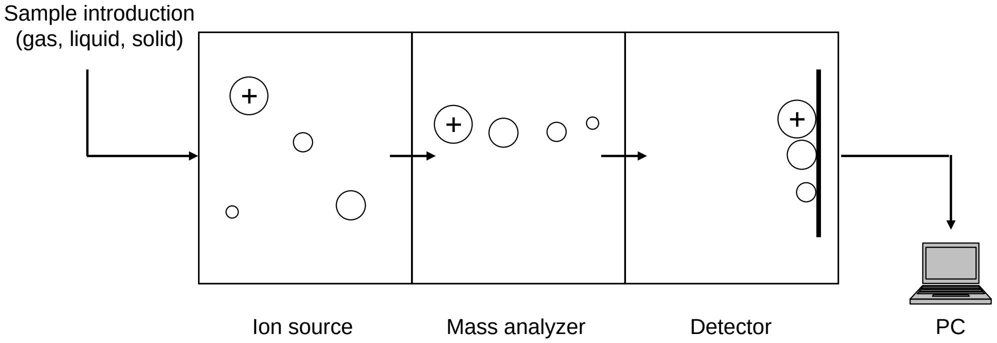


Figure 2.3: Schematic principle of the mass spectrometer

In the following, we briefly describe the main ionization sources and mass analyzers. A comprehensive survey with more details about instrumentation can be found in [54].

Ionization Sources. Although invented in the late 19^{th} century, mass spectrometry (MS) for a long time was restricted to small and thermostable molecules due to the lack of proper techniques to “softly” ionize and transfer ionized molecules from the condensed phase to the gas phase without excessive fragmentation. This situation was changed in the late 1980s, with the development of two techniques for the routine and general accumulation of molecular ions of entire biomolecules, namely *matrix-assisted laser desorption/ionization* (MALDI) [61, 68] and *electrospray ionization* (ESI) [39, 126] that allowed the ionization of higher-value molecules such as proteins. For the developments of these invaluable MS technologies, Koichi Tanaka (MALDI) and John B. Fenn (ESI) were given the Nobel Prize in Chemistry in 2002.

In MALDI, the analytes are dissolved with a matrix solution and a small drop (microliter volume) is spotted on a metal plate and allowed to dry. After drying, the metal plate is transferred to the vacuum system of the mass spectrometer and a laser beam shots onto a matrix, with wave length specific to the matrix. The absorbed energy causes the matrix to evaporate, releasing the enclosed analyte and ionizing it. The ionization is typically singly-charged protonation, meaning that the ionized molecules carry one additional proton. Finally, directed by the electric field, ionized molecules are transported to the mass analyzer. Since only a part of the sample and the matrix is used with each laser shot, the same sample can be measured multiple times. Moreover, MALDI plates can be kept and re-used in later experiments, allowing interruptions in the analysis.

In ESI, the analyte is dissolved in a liquid solvent, and is sprayed from a tiny highly charged needle or capillary into a strong electromagnetic field, resulting in a condensate of small aerosol droplets with a charged surface. The droplets are brought into the vacuum system of mass spectrometer, and as the solvent in the droplets evaporates, they get smaller and smaller, increasing the electric field on the surfaces. When the electric field becomes strong enough, charged molecules desorb from the surfaces. These conditions usually result in multiply protonated ions. Finally, influenced by the electric field, ions are brought to the mass analyzer. In contrast to MALDI, ESI relies on continuous supply of dissolved analytes, which makes it particular suited for coupling with liquid chromatography, see Section 2.4.2.

Mass Analyzers and Detector. Mass analyzers separate ionized molecules, influenced by the electromagnetic field according to their m/z ratios. Varying in physical principles and performance standards, there exist several types of mass analyzers that are employed in proteomics and metabolomics. Here, we first introduce several widely used technologies. Since most of our work was performed on high resolution mass spectrometers, we then introduce several novel techniques, employment of which is increasingly becoming a routine technique in modern laboratories, that allows for a precise and comprehensive analysis of complex biological systems [81].

The simplest and most frequently used mass analyzers are *time-of-flight* (TOF) analyzers, *quadropoles* and *ion traps.* TOF analyzers are usually coupled with MALDI, whereas quadropoles and ion traps come along with an ESI source.

In a TOF analyzer, the ionized analyte is accelerated by an electric field for a certain distance. Due to the same force of the electric field applied for all ions, the velocity of a particular ion depends only on its mass and charge. Assuming the same singly-charged type of ions produced by MALDI, an ion's velocity depends only on its mass. After acceleration, ions drift in the field-free tube towards detector. Clearly, the time needed to pass the tube depends on velocity. When a particular ion hits the detector at the end of the tube, the flight time is registered, and the mass of an ion is calculated. The principle of a linear MALDI-TOF instrument is illustrated in Figure 2.4.

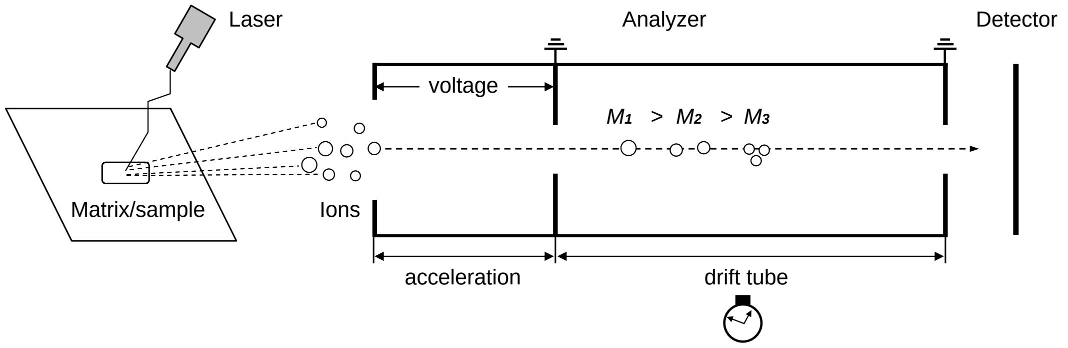


Figure 2.4: Illustration of a linear MALDI-TOF mass spectrometer.

Quadrupole analyzer consists of four circular metallic rods arranged parallel to each other. Each opposite pair of rods is connected together electrically, and a high frequency alternating voltage applied on the rods produces a high oscillating electromagnetic field between them. Ions passing through the field will move in spiral trajectories with the radius depending on the m/z value of a particular ion and on the offset voltage of the field. For a specific strength and frequency of applied voltage, only ions within a particular, very confined m/z range will reach the detector, whereas others will be deflected.

Ions traps (IT) function by a similar principle as quadrupole, but the ions are captured in a confined space of a vacuum system or tube. Ion traps exist in both linear and 3D form, while the latter is often referred to as a *quadrupole ion trap* (QIT), or *Paul trap*. The name is credited to its inventor Wolfgang Paul, who shared the Nobel Prize in Physics in 1989 for this work. One can think of a QIT trap as a quadrupole with two of the rods forming the endcap electrodes and a third bent into the ring electrode between them. Similar to quadrupole, the ions are moved by specific trajectories within the trap by applying an alternating voltage that results in an oscillating electric field inside the trap. Applying certain voltage causes ions of a particular m/z range to be released from the trap; the ejected ions are subsequently recordered by the detector.

The detector is typically build as an electron multiplier, and measures the signal by converting the kinetic energy of arriving ions into an electric current. The ion current is registered in equal intervals of time. Since the strength of the current depends on the quantity of ions, the ion current is referred to as the *intensity* of that particular m/z value.

Mass Accuracy and Resolution. Experimentally measured masses, or m/z values inevitably have some errors or uncertainties. The total error of mass measurements is given as either an absolute, or relative value. An absolute error is typically provided in Dalton, or *milli-mass unit* (mmu), where 1 mmu equals 0.001 Da. A relative error is commonly given in *parts-per-million* (ppm), and is also referred to as a *mass accuracy.* The mass accuracy of 10 ppm for measuring a molecule with mass 1 000 Da is equivalent to an absolute error of 0.01 Da.

The term *mass resolution*, or *resolving power* describes an ability of mass spectrometer to discriminate between several peaks with small differences in their masses. The resolution (R) of mass m is calculated as $R = \frac{m}{\Delta m}$, with two common ways to determine Δm residing in two corresponding IUPAC definitions: the *ρ percent valley* definition and the *peak width* definition.

In the ρ percent valley definition, Δm is defined as the smallest spacing between two equally intense peaks with a valley between them, while the height of the valley at its lowest point is bigger than ρ percent of either peaks. The typical value for ρ is 10 %.

In the peak width definition, Δm is defined as the width of the peak at a height which is a specified portion of the maximum peak height. According to IUPAC, it is recommended to always use one of the three values: 50 %, 5 % or 0.5 %. A most common standard is to measure the full width of the peak at half of its maximum height, also abbreviated as “FWHM”. The higher values of mass resolution indicate a better separation of peaks.

Because of imperfections in ionization sources and mass analyzers, charged molecules with equal m/z values never reach a detector at exactly the same time. For example, in MALDI-TOF mass spectrometers the ions are not shot from exactly the same spot on the matrix, therefore being spread out in 3D space when they enter the electric field. Thus, their velocities after acceleration may slightly differ. As a result, hitting the detector at slightly different times, two ions with equal m/z values are registered as one “merged” peak of a certain width. This behavior consequences in worser resolution – ions of similar m/z values may not be distinguished, and in lower mass accuracy – the actual m/z value may not be detected exactly, but only with some tolerance.

Ion Cyclotron Resonance and Orbitrap Mass Analyzers. The introduction and commercialization of *Fourier transform-ion cyclotron resonance* (FT-ICR) mass spectrometers with external ionization sources [108] was a breakthrough in terms of the mass accuracy and resolving power of mass spectrometers, with a completely new principle to determine the masses of charged molecules.

As its name implies, a FT-ICR analyzer is equipped with a cyclotron that accelerates charged molecules to high energies. The cyclotron contains a Penning trap (a magnetic field with electric plates), and the ions are moving in circles around this magnetic field directed by an applied voltage. Under the influence of an oscillating electric field that is placed perpendicular to the magnetic field, the ions are excited to a larger cyclotron radius. The frequency (or angular velocity) of a particular ion depends on its m/z value and the strength of the magnetic field applied. Instead of a detector, a pair of plates is used to record a signal from ions that pass close by. Since the ions are moving in packets, the frequencies of several ions are recorded simultaneously, resulting in a superposition of sine waves. The frequency of a particular ion is extracted from this "interlaced" diagram by applying a Fourier transformation. With individual frequencies extracted, it is then straightforward to derive the corresponding m/z values.

The resolution and mass accuracy of FT-ICR analyzer are extremely high. However, its employment adds a sizeable economic burden mostly due to the costs of the superconducting magnet.

Recently, a new type of mass analyzer, called Orbitrap [56], was introduced. Like FT-ICR, Orbitrap follows the same principle to measure the frequency of rotating ions, but instead of a magnet, an oscillating electric field is used for the acceleration of charged particles.

An Orbitrap consists of an outer and inner coaxial electrode that create an electrostatic field. The ions are injected into the field between electrodes, and influenced by centrifugal forces, start moving around the inner electrode. Additionally, the ions move back and forth forming a harmonic oscillation in orbits along the axis of the electrostatic field. The oscillating frequency of a particular ion is inversely proportional to its m/z value. Like in FT-ICR analyzer, the m/z value of a single molecule is derived from the superposition of sine waves of several ions by applying a Fourier transformation. Thus, an Orbitrap analyzer provides a resolution and mass accuracy similar to FT-ICR, but saves on the usage of the expensive superconducting magnet. The design of Orbitrap analyzer is patented by Thermo Scientific[2].

The development of new mass analyzers catalyzed further improvements of instrumentation by assembling complex multistage instruments specifically suited for various experimental setups. One typical example is a *hybrid quadrupole time-of-flight* (Q-TOF) mass spectrometer that combines a quadrupole and TOF analyzer. Employing a TOF analyzer in the last stage allows to achieve a higher mass resolution, and fastens the analysis, which makes this design better suited for protein and proteome analysis [3].

Tandem Mass Spectrometry. As its name implies, the principle of tandem mass spectrometry is based on two mass analyzers in tandem. In the first mass analyzer, a particular ion is selected based on its m/z ratio. In the second analyzer, the selected peptide ion undergoes a fragmentation step, resulting in the *fragment ions* of this peptide ion. In the second analyzer, masses of the fragment ions are recorded. The spectrum containing measured m/z values and intensities of the fragment ions is called an *MS/MS spectrum*, or *product ion spectrum*. The peptide ion selected for fragmentation is referred to as a *precursor ion*, or *parent ion*, whereas the fragment ions are called *product ions*, or *daughter ions*. A typical experimental setup for tandem MS includes LC/ESI interfaced with several quadrupole analyzers. The precursor ions are selected in the first quadrupole, and undergo fragmentation through *collision-induced dissociation* (CID) in the second quadrupole, while the third quadrupole measures the resulting MS/MS spectrum.

In CID, the precursor ion is accelerated in vacuum by some electrical potential, and enter the collision chamber, which contains chemically inert gas molecules, typically he-

---

[2] `http://www.thermo.com/`

lium, nitrogen, or argon. In the chamber, the precursor ion repeatedly collides with gas molecules resulting in the increase of its potential energy, until a certain fragmentation threshold is achieved, and the precursor is fragmented into the product ions. For peptides, the fragmentation typically occurs at the peptide bond (see Figure 2.1), which results in the creation of *b-ions* and *y-ions*. The series of b-ions represents a sequence of increasing m/z values from the N-terminus, with each ion differing from the previous one by the mass of one amino acid. The y-ions are complementary to b-ions, and are created on the C-terminus. The fragmentation of other ion types may also occur, i.e., through loss of ammonia, water, or immonium ion formation. Further details about the ion types, and other fragmentation techniques used in tandem MS will be discussed in Chapter 6.

2.4.2 Experimental Workflow

Any mass spectrometry-based workflow in proteomics and metabolomics usually consists of three major experimental stages. First, samples are extracted from the biological source, and optionally undergo further biochemical treatment. For proteins, this includes purification from other cell molecules, such as DNA and metabolites by the means of precipitation. For metabolites, the sampling procedure requires the rapid inhibition of metabolic activity, referred to as *quenching*, followed by the subsequent release of molecules into the suitable medium to facilitate cell lysis. In a second step, the resulting complex mixture has to be separated into its individual components. There exist three major separation techniques currently employed in the mass spectrometric analysis: *2D-gel electrophoresis* (2D-GE), *liquid chromatography* (LC), and *gas chromatography (GC)*, where the former two are used in proteomics, and the latter two are used in metabolomics. Followed by the analysis in a mass spectrometer, in the last step, the identification and quantification of the sample molecules is performed. The sample identity can be determined either through database search, or *de novo*, i.e., without prior knowledge.

In the remainder of this section, we briefly describe two major techniques involved in the sample separation: 2D-gel electrophoresis and liquid chromatography, while the topic of the sample identification will be addressed in Section 2.5.

2D-gel Electrophoresis. A gel represents a matrix of various spot size, with each spot containing a cross-linked polymer. As its name implies, 2D-GE is used to separate the individual components of a sample in two dimensions, based on two properties of the molecule. For proteins, these properties are usually a molecular mass and isoelectric point. Separation of proteins on isoelectric point is performed by isoelectric focusing, where a pH gradient is employed for separation. The proteins are loaded onto a gel matrix with immobilized pH gradient, and the matrix is placed in an electric field. Charged proteins start moving towards negative or positive electrodes until they reach a particular pH value within the gradient, neutralizing their charges. Since the separation on isoelectric point is usually insufficient to separate all proteins, the separation based on another property – molecular mass, follows. For mass-proportional size separation, proteins have to be first denatured into a linear form and negatively charged. This necessary preprocessing is achieved by treating proteins with *sodium dodecyl sulfate* (SDS). Built from *polyacrylamide* (PA) for this separation technique, gel forms a network of pores that allows the easier transit of smaller molecules than of larger ones. Directed by a current applied to the second dimension, proteins start moving towards the positive electrode with smaller molecules migrating faster. Since the size of a molecule is considered to be sufficiently proportional to its mass, proteins are separation with respect to their molecular weights. Gel electrophoresis with SDS and PA is usually called *SDS-PAGE*. Finally, the proteins are colored by silver or coomassie blue, making them visible in the gel. The spots of interest are then picked from the gel either manually or by picking robot, and are subjected to the mass spectrometric analysis.

Protein Digestion. After separation, proteins have to be biochemically dissociated or *digested*, i.e., undergo a site-specific cleavage using a *protease*. Proteases are enzymes that cut a peptide bond with a release of one water molecule, see Section 2.2. The most common protease in proteomics is *trypsin*: Trypsin cleaves after each arginine (R) or lysine (K), unless followed by a proline (P). Trypsin is well-suited for mass spectrometry-based proteomics due to several reasons: It allows a high specificity of digestion with relatively few missed cleavages, and none or very few cleavages at unexpected positions. Furthermore, trypsin is easily obtained and applicable in most experimental setups, and the resulting peptides follow the “Goldilocks” phenomena: they are neither too long, nor too short, with an average length of 11 residues.

The process of proteins separation using 2D-gel and the subsequent digestion of each instance individually is usually expensive, slow, and error-prone. Instead, another set of techniques under the common term *chromatography* has emerged, where one first digests the protein mixture as a whole, and then performs the separation of individual digested peptides. Digestion of all proteins simultaneously significantly speeds up the process, but, on the other hand, makes the subsequent data analysis harder.

Liquid Chromatography. The term “chromatography” defines a set of techniques that are used to separate a mixture into the individual components. The separation is performed by introducing the sample into a *mobile phase*, or solvent, that moves through a *stationary phase*, or packing, and the individual molecules are interacting with either the stationary or mobile phase. The more the component interacts with a stationary phase, the longer it takes for transition. The characteristic time of a molecule to pass through this system is called a *retention time*.

Column chromatography is the most widely used type of chromatography for separation of organic compounds. The stationary phase in column chromatography contains a tube made from metal, glass or a synthetic matter. Based on the type of the mobile phase, two methods of column chromatography exist: *gas chromatography* and *liquid chromatography*. In gas chromatography, gas is used for the mobile phase, and a liquid, gum, or elastomer for the stationary phase. The instruments for gas chromatography are simpler in use, and are more efficient in separation than the ones for liquid chromatography. However, being widely employed in metabolomics, gas chromatography cannot be used for separation of non-volatile or thermally labile compounds, such as proteins. Therefore, in the following, we skip further details about gas chromatography and describe its counterpart – liquid chromatography, that can be applied both in proteomics and metabolomics.

In liquid chromatography, a liquid is used for the mobile phase, and a porous material for the stationary phase. Originally, the mobile phase moved through the stationary phase solely by the gravitation forces. However, to achieve a more practical flow, speed, and a better separation, high-pressure pumps have been introduced some decades ago. Therefore, the technique is often referred to as *High-pressure liquid chromatography*, or *High-Performance liquid chromatography* (HPLC), and abbreviations HPLC and LC are often used interchangeably.

In Figure 2.5, a schematic principle of HPLC is depicted. The components appear in the column as *bands*, each containing a large number of molecules to be separated. Passing through the column, different molecules interact with the stationary phase differently, thus receiving the individual moving rates. Placed at the outlet of the column, the detector registers the eluted components, and their representation is visualized as a diagram, called *chromatogram*, by the connected computer. The chromatogram can be interpreted as a two-dimensional spectrum with a retention time along the horizontal axis, and the intensities of the measured components along the vertical axis. Each chromatographic peak can contain dozens of individual species that further subjected to the mass spectrometer. For the convenient MS analysis in the subsequent step, LC is usually coupled in-line with liquid-based ESI: The analyte emerging from the outlet of the column is directly introduced to the ESI ionization.

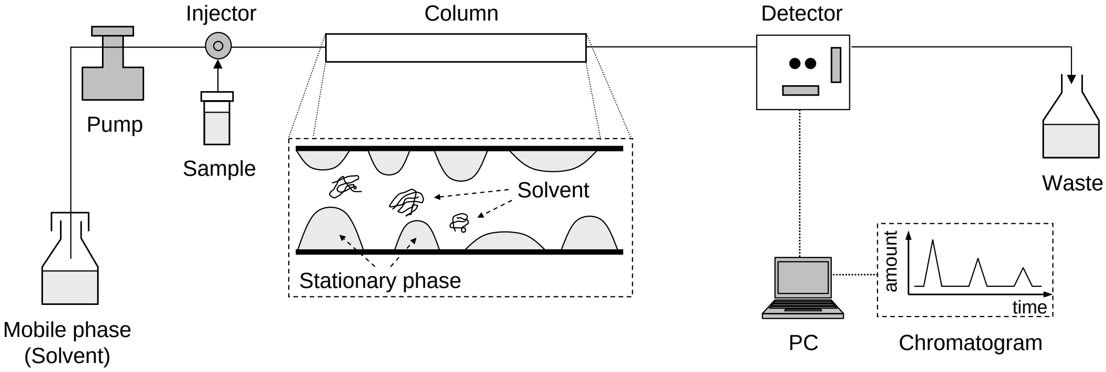


Figure 2.5: Illustration of the HPLC principle.

So far, we have described biochemical and biotechnological aspects of the mass spectrometry-driven analysis. In the next section, we address the topic of analyzing the outcome of the mass spectrometric experiment, pointing out the existing approaches to perform the data analysis, and computational challenges that arise in this context.

2.5 Mass Spectrometry Data Analysis

One of the major tasks in mass spectrometry-based proteomics and metabolomics is the identification of sample molecules. The identification is typically achieved by comparing the measured mass spectra to a database, or *de novo*, that is deducing the sample identity from the mass spectra without any prior knowledge. Clearly, the data analysis depends on the the type of the measured mass spectra.

For a general overview of identification approaches, see the review articles [29, 32, 83, 91]. A detailed overview of identification algorithms can be found in [109], whereas in [91, 83] shorter reviews of more recent advances are given. A detailed discussion of available PFF algorithms is provided in [85]. Further information on experimental and computational methods employed in proteomics can be found in books by Patzkill [92], Snyder [113] (experimental), and Eidhammer *et al.* [34] (computational).

2.5.1 Types of Mass Spectrometric Analysis

Peptide Mass Fingerprinting. Peptide mass fingerprinting (PMF) is a method to identify proteins by comparing its constituent peptide masses to the masses of peptide sequences derived from the database. In the first step, an intact unknown protein is cleaved in fragments (peptides) by a proteolytic enzyme, commonly trypsin. Masses of the peptides are then measured by a mass spectrometer, and compared to the theoretical masses of peptide sequences from the database. Theoretical masses are derived from the database sequences that are generated by *in silico* digestion of all database entries. After comparison, the best matched list of masses with the corresponding protein sequence is returned as identification. In Figure 2.6, a schematic procedure of PMF is depicted. A common instrumentation for PMF approach is a MALDI ionization source, coupled with a TOF mass analyzer.

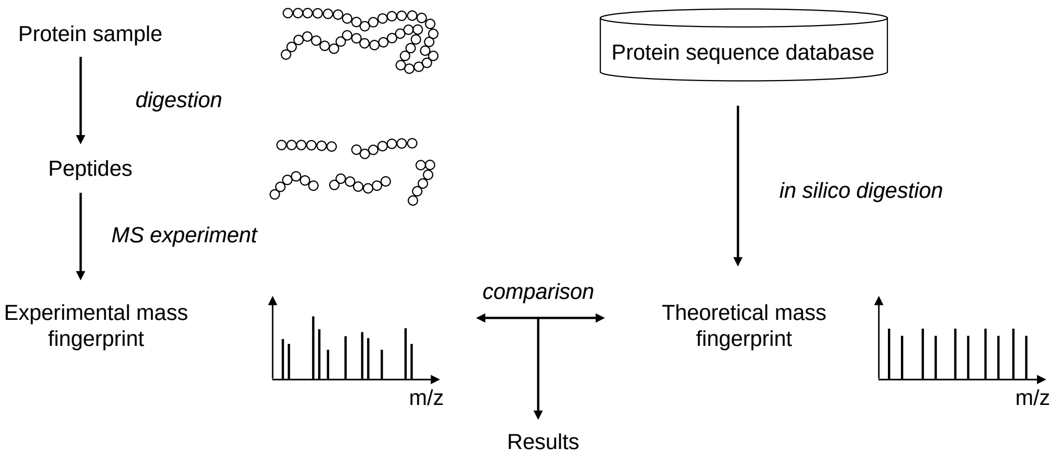


Figure 2.6: Protein identification with PMF.

The premise behind the PMF approach is that every protein has a unique set of peptides and, hence, unique peptide masses. In practice, however, different peptides often have the same or similar masses, requiring other approaches to identify such peptides. A stronger indicator for peptide identity is the peptide sequence itself. Although two peptide sequences from two different proteins may be the same, this overlap occurs in practice much less than on the mass level. Using tandem mass spectrometry, the information about a peptide sequence, or a so-called *peptide fragment fingerprint*, can be obtained.

Peptide Fragment Fingerprinting. For identification, the measured MS/MS spectrum of the fragmented peptide can be compared to the theoretical spectra of every peptide in the database, and the peptide that results in a best match to the measured spectrum can be reported. Clearly, there can be millions of peptides in a sequence database, therefore it is often impractical to inspect every peptide. Instead, some filtering is used to restrict the search to only those peptides that can lead to a potential matching. Most commonly used filters are the *precursor mass* and *specificity* of the enzyme used for digestion. The result of the mass filtering is a mass interval that defines the constraints on the peptides allowed for further comparisons. If the matching of mass spectra leads to the identification of several peptides from one protein, the corresponding database sequence is returned as the protein identification. In Figure 2.7, a schematic illustration of the PFF approach is shown.

An experimental MS/MS spectrum of the fragmented peptide can be also used for *de novo* peptide sequencing. In principle, since the fragmentation of the parent ion leads to the presence of masses of all prefixes and suffixes of the peptide sequence in the measured MS/MS spectrum, one can reconstruct the sequence by finding peaks in the spectrum that lie at mass differences of exactly one amino acid to each other. In practice, however, the quality of the measured spectrum is insufficient to perform such an exhaustive search successfully. Some peaks corresponding to true peptide fragments may be missing, or the background noise may give rise to additional “false” peaks, thus making the correct interpretation of the measured spectrum impossible. These and other complications that are faced while interpreting mass spectra are discussed in the next section, whereas more details on the *de novo* peptide sequencing by tandem MS will be given in Chapter 6 in the context of the corresponding application study.

2.5.2 Computational Problems

Peak List Construction. As mentioned earlier, in a mass spectrometer, each compound is detected not exactly at one point of time, but rather over a tiny time interval. That is, for each peak, a collection of intensities is detected over a very short m/z range rather than one point, at exactly one m/z value. A spectrum containing a complete set of intensity measurements at certain small increments of the m/z value is referred to as a *raw spectrum* or *raw data*. To derive a in a form of *peak list*, the raw data has to be transformed.

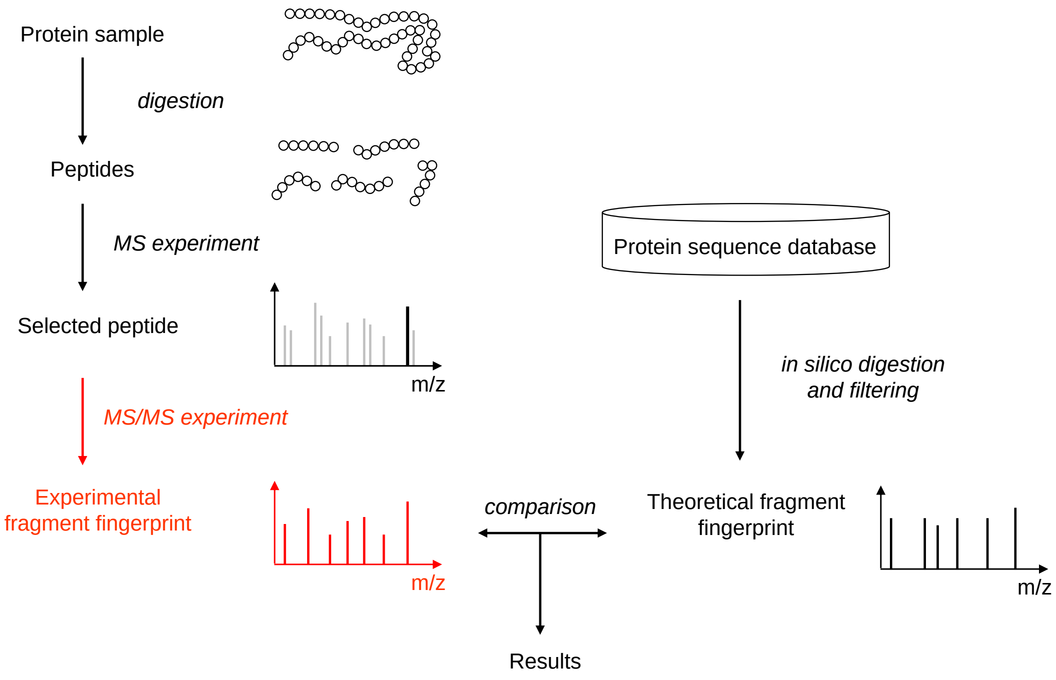

Protein sample
digestion
Peptides
MS experiment
Selected peptide
m/z
MS/MS experiment
Experimental fragment fingerprint
m/z
Protein sequence database
in silico digestion and filtering
comparison
Theoretical fragment fingerprint
m/z
Results

Figure 2.7: Protein identification with PFF.

A peak list is a processed form of the raw data, and its construction typically involves several processing steps. In the first step, the m/z values from the raw spectrum have to be calibrated to eliminate systematic shifts in the measured data. Since the raw data also include signals arisen from different forms of noise, in the second step, the effects of noise have to be removed. This includes a baseline correction, i.e., normalizing the signal level by eliminating a "drift" along the vertical axis, which commonly occurs due to chemical noise. Moreover, a spectrum can be rugged, complicating the recognition of the true peaks amongst noise. Therefore, a smoothing and noise reduction can be performed. Finally, a peak detection or a peak picking procedure is performed, to collect the ion signals that correspond to the sample compounds. A detailed introduction to the peak list construction is out of scope of this introduction; the algorithms for the spectrum processing can be found in [84, 76, 128].

The m/z values from the processed raw spectrum, together with the peak intensities and possibly other features form the peak list. A peak list is typically used as an input for the majority of computational methods.

Additional and Missing Peaks. Ideally, a mass spectrum would only consists of peaks that correspond to the compounds present in the sample. However, the experimental workflow including the sample preparation, measurement by a mass spectrometer, and transformation of a raw spectrum inevitably has some imperfections. The measured peak list, thus, can differ from its ideal counterpart encountering some additional and missing peaks.

Additional peaks mostly appear due to the various types of chemical noise and randomly occurring electronic disturbances. Chemical noise can originate from chemical contaminants that have been unintentionally introduced during the sample preparation. A typical contaminant is a human keratin from the hair or skin. In MALDI-TOF analysis, noise can occur due to the instability of peptides after the ionization by the laser shot, which results in losses of parts of amino acid side chains. The 100 most common contaminants occurring in proteomic experiments can be found in [31]. Sometimes, peaks of very low intensity, stemming from the random noise, are wrongly interpreted as true peaks by the peak picking software.

Another complication are *missing* peaks, i.e., peaks that should be present in a peak list but are not. This happens due to the lack of the analyte ionization, leading to low abundance of the corresponding compound, or errors by the preprocessing algorithms.

2.5.3 Protein Identification using Databases

In a database-oriented approach, each database protein *in silico* undergoes the same experimental modifications, as the protein to be identified. This commonly includes an enzymatic cleavage and possibly further fragmentation. As a result, a theoretical spectrum (or spectra) for each protein in a database is created, and subsequently compared to the experimental data. Finding a best theoretical match is done by scoring each theoretical spectrum against the measured spectrum, where a score corresponds to some degree of similiraty between two spectra. A “hit” defines any score above a certain confidence level, where the top hit is assumed to be the identity of an unknown protein. If no score above a given significance threshold exist (“no hits”), the protein remains unidentified.

Theoretically, any method to compare two spectra can serve as a scoring scheme. The simplest example of such comparison is a *Peak Counting Score* (PCS) that counts a number of peaks that two spectra have in common. Another example of a scoring metric is *sequence coverage*, the fraction of the protein sequence covered by the peaks matched in experimental and theoretical spectra. Scoring techniques that are used in practice usually derive a final score as a combination of several basic scoring functions.

PMF Algorithms. Historically, first mass spectrometry-based algorithms to identify proteins were PMF approaches: In 1993, several methods were published that rank protein sequences by PCS with a certain tolerance on the mass of the measured peak [59, 80, 130]. PMF approaches are still widely used in practice [60]. For the analysis of high quality samples stemming from the well-characterized organisms, PMF is often able to identify proteins with high confidence, particularly for organisms with small genomes.

One of the first advanced scoring schemes proposed by Pappin *et al.* [94] was *MOlecular Weight SEarch* (MOWSE). MOWSE score takes into account several factors such as the number of matched peak masses, mass of the database sequence, and the number of peptides in the database with masses similar to the experimental mass. The latter is derived from the frequency of occurrence of a particular fragment mass of a protein, assuming the non-uniform distribution of the peptide molecular masses in a database. Based on the MOWSE score, two further probabilistic algorithms have been developed, MASCOT [97] and MS-Fit as a component of ProteinProspector [25].

An important probabilistic measure is the *p-value*, the probability that the observed match between the experimental data and the database entry occurs by chance. MASCOT also provides a calculation of the significance of each score based on the p-value. A typically used significance threshold is $p < 0.05$. The significance score is calculated by taking the negative logarithm of p-value multiplied by a constant. Unfortunately, no further details about the actual, proprietary algorithm to calculate a MASCOT score are available.

Another probabilistic algorithm is ProFound [134]. It is based on Bayesian Statistics to score hits using additional background information such as enzyme cleavage information, presence or absence of particular amino acids in the protein, a protein's mass and sequence, and previous experiments on the sample protein. ProFound assumes the Gaussian distribution of mass measurement error, takes into account overlapping and adjacent peptides, and includes additional and missing peaks in its scoring model. As a score, it returns the probability that the protein suggested from the database originates from the measured spectrum.

An extensive list of PMF algorithms can be found in [109]. A recent evaluation of PMF approaches using mass spectra gathered on MALDI-TOF instruments revealed the performance of ProFound to be slightly better than of MASCOT, and both methods were found to be superior to MS-FIT [23].

One of the drawbacks of PMF algorithms is their sensitivity to the database size. Growing number of database sequences has a direct effect on the significance of database hits, increasing the probability that the match between the experimental and theoretical data occurs by chance. Therefore, for protein identification many laboratories employ PMF as a "first run", and if the results remain uncertain, continue with PFF analysis.

PFF Algorithms. Protein fragment fingerprinting is very similar to PMF approach but applied to MS/MS spectra. Peptide sequences are identified by comparing the measured ion spectra to the peptide spectra derived either from a protein sequences database, or from libraries of experimental spectra identified in previous experiments.

Due to the additional information acquired from the fragmentation, MS/MS-based approaches have several advantages over PMF algorithms. Provided significant peptide coverage, identification algorithms are less sensitive to the database size and various side effects such as protein modifications, and can provide results of higher confidence than traditional PMF. Furthermore, to achieve confident results, PFF methods do not require all peptides of a target protein to be found, and can handle analysis of complex peptide mixtures.

One of the earliest and still widely used PFF programs is a commercial software SEQUEST [37,129]. SEQUEST divides the scoring procedure in two steps: the preliminary scoring employs simple metrics including the intensities of matching peaks, and presence and continuity of spectrum ions, to filter out ions that have a very low probability of being a correct peptide. The final scoring evaluates the remaining segments using more sophisticated means, such as a *cross-correlation* function that computes the correlation between the measured and theoretical spectra.

Another popular representative of a non-probabilistic scoring approaches is the open-source program X!Tandem [40], available from the Global Proteome Machine Organization[3]. It employs nonlinear scoring functions to account for the number and intensities of matching peaks in the measured and theoretical spectra. The reported score is converted to an *expectation value*, or *E-value*, which refers to the expected number of peptides with scores equal to or better than the observed score, assuming that peptides match the measured spectrum by random chance.

Originally based on the MOWSE algorithm for PMF search, MASCOT has been further adapted for tandem MS analysis. It uses the parent mass and relative abundances of corresponding peptide masses as search space constraints; however, no further details about the scoring scheme have been published.

A detailed list of the most widely used PFF tools can be found in [85]. A few recent benchmarking studies have compared several publicly available PFF tools on their sensitivity and specificity. The comparison of PFF packages using data collected on the low-resolution ESI ion trap instrument from human samples, revealed better sensitivity of SEQUEST, while MASCOT and X!Tandem were better at distinguishing true positives from false positives [67].

One of the major challenges in tandem MS analysis is the assignment of statistical significance to peptide identifications. Typical estimates of the error rates such as p-value, or E-value, are currently based on empirical assumptions or fitting of the database scores distribution (for example, Gaussian). In 2008, a principally different approach, called MS-GF, was proposed [71], which employs generating functions to establish the theoretical estimates of statistical confidence of spectral identifications. Weighted generating functions are computed by the dynamic programming algorithm, and are further used to derive exact significance scores of spectral matches. The performance of MS-GF was compared to that of another popular open-source package X!Tandem, and was found to be significantly superior.

---

[3] `http://www.thegpm.org/`

3 Decomposition Algorithms

One of the key challenges in analyzing the mass spectrometry (MS) data is the assignment of the individual peaks to the sample compounds they represent. Given a mass of a peak, the obvious way to encounter this problem is to look at which elementary components this mass can be composed of. For the case of integer masses, there exists a lot of work on the *mass decomposition* problems. Since we address these problems in our applications, and use the decomposing of integer masses as a cornerstone in all our implementations, in this chapter, we give a brief overview to the integer mass decomposition problems. Followed by a few extensions of the existing algorithmic solutions, we then show how to integrate the integer decomposition techniques for the analysis of the real-valued MS data.

3.1 Integer Mass Decomposition

From the algorithmic point of view, the problem of finding decompositions with a given integer mass and related issues have been frequently addressed in the literature [19, 18, 77]. Here, we outline the decomposition problems relevant to our applications, and mention their typical solutions. We also briefly describe recently developed, more efficient algorithms that we use as a basis for our implementations, and present a few extensions of these algorithms. A detailed survey in the integer mass decomposition algorithms can be found in, i. e., [19].

3.1.1 Definitions and Problems

Inferring the molecular formula of a sample molecule from its mass M comes to computing a *decomposition* of M over the masses of its individual components, i. e., considering M as a non-negative integer linear combination of these masses. We call a set of individual masses an *alphabet* $\Sigma = \{a_1, \ldots, a_n\}$, where $n = |\Sigma|$ is the *size* of the alphabet. For example, the proteins alphabet consists of 20 standard amino acids, while 4 nucleotides build the DNA alphabet.

Given an alphabet Σ of size n, we define a *decomposition* as a n-tuple $(c_1, \ldots, c_n)$, where each entry $c_i \in N_0$, for $i = 1, \ldots, n$, corresponds to a multiplicity of the i-th element in Σ. For example, over the alphabet of 4 nucleotides $\Sigma = \{A, C, G, T\}$, the strings $ACGC$ and $CCGA$ have the same decomposition $(1, 2, 1, 0)$, or specified using a symbolic sum $\mathrm{A}_1 + \mathrm{C}_2 + \mathrm{G}_1 + \mathrm{T}_0 = \mathrm{A}_1\mathrm{C}_2\mathrm{G}_1$. Decomposition thus can be seen as an *unordered* representation of a molecular sequence: it allows an abstraction from the order of molecular components, instead only counting the number of occurrences of each component in the molecule. We call $|c| = \sum_{i=1}^{n} c_i$ the *length* of the decomposition c, that equals the overall multiplicity of the decomposition entries. For the example above, the length of the decomposition $(1,2,1,0)$ equals 4. Decompositions have been frequently referred to in the literature under different names such as *compomers* [14], *Parikh-vectors* [104], *abelian patterns* [35], to name a few. Here, we skip further description of the properties of decompositions, they can be found in, i. e., [77].

Formally, the *Integer Mass Decomposition Problem* (IMDP) can be stated as follows:

> Given an *alphabet* $\Sigma = \{a_1, \ldots, a_n\}$ with $a_i \in \mathbb{N}$, for $i = 1, \ldots, n$, and a mass $M \in \mathbb{N}$, find all decompositions $(c_1, \ldots, c_n)$ with $c_i \in \mathbb{N}_0$, for $i = 1, \ldots, n$, such that $M = \sum_{i=1}^{n} c_i a_i$.

We call M *decomposable* over the alphabet Σ if at least one such decomposition exists. Different forms of IMDP problem include the following related invariants:

1. Decision Problem: Does a decomposition of M exist?
2. One Instance Problem: Find one decomposition of M?
3. Counting Problem: How many decompositions of M exist?

A decision form of IMDP is known as the *Money Changing Problem*, or *Equality Constrained Integer Knapsack Problem* [1]. The problem is known to be NP-complete, when the input parameters, a mass M and alphabet Σ, vary [79], but can be solved in pseudo-polynomial time by a dynamic programming (DP) algorithm [82]. Two other problems can be also solved by the invariants of the DP algorithm, originally proposed by Gilmore and Gomory [51] for the *Coin Change Problem*, where the decomposition with the *minimal* number of coins, or in our terms, with the minimal length, is sought.

Classical Solutions. For the *Decision Problem*, one builds a one-dimensional boolean table B of size $M+1$, where $B[m] = 1$ means that m is decomposable over the alphabet Σ, otherwise $B[m] = 0$. The table entries are filled using the following simple recurrence: $B[0] = 1$, $B[m] = 0$, for $1 \leq m \leq a_1$; and for $m \geq a_1$,

$$B[m] = \begin{cases} 1 & \text{if there exists } 1 \leq i \leq n \text{ with } m \geq a_i \text{ and } B[m - a_i] = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the solution for the Decision Problem requires $O(nM)$ time and $O(M)$ space. If we are interested only in $B[M]$, we can keep only those table entries that can contribute to the currently computed value in the recurrence; the remaining entries can be overridden. Doing so, the space complexity is decreased to $O(\max_i a_i)$. The *One Instance Problem* can be solved using the same table by backtracking, in time proportional to the length of the decomposition, which can be in the worst case $O(\frac{M}{a_1})$.

For the *Counting Problem*, a two-dimensional integer table C of size $n \cdot (M+1)$ can be constructed, which entry $C[i, m]$ contains the number of decompositions of mass m

over alphabet $\{a_1, \ldots, a_i\}$, for $i = 0, \ldots, n$ and $m = 0, \ldots, M$. The table is initialized with $C[0, 0] = 1$, $C[0, m] = 0$, for $m > 0$, and is filled using the following recurrence:

$$C[i, m] = \begin{cases} C[i-1, m] + C[i, m - a_i] & \text{if } m \geq a_i, \\ C[i-1, m] & \text{otherwise} \end{cases}$$

for $1 \leq i \leq n$. This solution requires $O(nM)$ time and space complexity. If we only want to know the number of decompositions over $\{a_i, \ldots, a_n\}$, for each $m \leq M$, we can compute the table row by row, while the previous row can be discarded. Doing so, reduces the space requirements to $O(M)$. Alternatively, if we are only interested in $C[n, M]$, we can compute the table column by column, while the columns that are unreachable by the recurrence equation can be dropped. Then, the memory complexity is reduced to $O(n\ \max_i a_i)$ and does not depend on M.

For solving the original IMDP problem to find all decompositions with mass M, an intermediate between the above two solutions can be employed: A two-dimensional boolean table A of size is $n \cdot (M + 1)$ is created, where $A[i, m] = 1$ means that m is decomposable over an alphabet $\{a_1, \ldots, a_i\}$, for $i = 0, \ldots, n$ and $m = 0, \ldots, M$, otherwise $A[i, m] = 0$. We initialize the table with $A[0, m] = 1$ if and only if $m \bmod a_1 = 0$, for $0 \leq m \leq a_1$. The table is further filled using the recurrence:

$$A[i, m] = \begin{cases} A[i-1, m] \vee A[i, m - a_i] & \text{if } m \geq a_i, \\ A[i-1, m] & \text{otherwise} \end{cases}$$

for $1 \leq i \leq n$. The construction time is $O(nM)$. We denote $D(M)$ as a set of all decompositions of mass M. Then, $D(M)$ can be generated by a simple backtracking algorithm, which requires $O(nM)$ space and $O(\sum_{c \in D(M)} |c|) = O(\gamma(M) \cdot \frac{M}{a_1})$ running time, where $\gamma(M)$ is the number of decompositions of M.

3.1.2 Enumerating Integer Decompositions

Recently, a novel algorithm for the above problems has been introduced [19], which outperforms the classical dynamic programming algorithm in running times and particularly in space requirements, that are reduced to several orders of magnitude depending on the alphabet in use. Here, we only sketch the algorithm to solve IMDP, further details and evaluations of the algorithms for this and other problems can be found in [19].

Given an alphabet $\Sigma = \{a_1, \ldots, a_n\}$, first a two-dimensional table called *Extended Residue Table* (ER table) of size na_1 is computed in time $O(na_1)$. Each entry $ER(r, i)$, for $r = 0, \ldots, a_1 - 1$ and $i = 1, \ldots, n$, stores the smallest number congruent r modulo a_1, which is decomposable over an alphabet $\{a_1, \ldots, a_i\}$. The last column of the table thus contains the smallest number, which is decomposable over the whole alphabet. The ER table is then used to generate all decompositions of a query mass M using the recursion through the table. As a result, the space requirements of the algorithm is na_1, whereas the running time is proportional only to the table size na_1 and the number of decompositions $\gamma(M)$, with no direct dependency on the input mass M itself.

Decompositions with Elemental Bounds. Often in applications, we know that sample molecules have certain *bounds* on the minimal and maximal amount of elements. For example, if we know that our protein molecule must have at least 4 *cysteins* (C), or at most 2 tryptophans (W), then the peptide sequence $A_3C_5D_4W_3$ is invalid solution, violating the second constraint.

Now, we want to extend the above algorithm for finding all decompositions to make the results compliant with the provided constraints. For the case with minimal bounds, the modification is trivial: We simply subtract the mass that contributes to the minimal bounds from the query mass, and start the backtracking algorithm with the reduced value. After the backtracking is complete, we add each elemental bound to the solution. For example, consider computing all decompositions of mass 100 over the alphabet with the smallest mass 5, with at least 2 copies must be present in the solution: We run the backtracking algorithm for mass 90, and after the results are obtained, we modify them by increasing the first entry in each solution vector by 2.

For the maximal bounds, there exist two possible ways to extend the algorithm: A naive approach is first to generate all decompositions as before, and then, for each decomposition, to check if it fits the constraints. This, however, may generate many decompositions unnecessary. Another possibility is to include the bounds in the backtracking algorithm itself. In fact, the recursive nature of the backtracking process allows us to easily do so: while increasing the decomposition entry for the current alphabet element, we can check if the upper bound is not exceeded. In this case, we enter into the recursion for the next element, otherwise we discard the partial solution. At the end, we apply the same test for the smallest element, before reporting the result. In Figure 3.1.2, we give the pseudo-code of the FIND-ALL-WITH-BOUNDS algorithm, which is a slight extension of the recursive FIND-ALL algorithm, which was proposed in [19] to find all decompositions with no bounds. Similar to the unbounded version, on each recursion step FIND-ALL-WITH-BOUNDS keeps a current query M, a decomposition c, and an index i of the current alphabet element. Additionally, we know the maximal elemental bounds b, and check if they are satisfied for the i-th element (line 11), and for the first element (line 3). If so, we either step into the recursion for the next element, or report the resulting decomposition. Here, we skip further description of the algorithm, referring the interested reader to [19] for a detailed discussion.

Analogously to the unbounded version, the running time complexity of FIND-ALL-WITH-BOUNDS is $O(na_1\gamma(M))$. Clearly, applying the bounds on the resulting decompositions can only decrease the number $\gamma(M)$. Thus, the usage of FIND-ALL-WITH-BOUNDS algorithm can lead to a significant improvements of the actual performance depending on the applied constraints, see Chapter 6.

While the former term na_1 in the running time complexity of the algorithm is fixed for a given alphabet, the number of decompositions $\gamma(M)$ grows rapidly with increasing M. In Section 3.2.1, we show how to approximate this number for various alphabets.

Algorithm FIND-ALL-WITH-BOUNDS **(mass M, index i, compomer c, bounds b)**

```
if $i = 1$ then
    $c_1 \leftarrow M/a_1$;
    if $c_1 \leq b_1$ then
        output $c$; return;
    end if;
end if;
lcm $\leftarrow \mathrm{lcm}(a_1, a_i)$; $\ell \leftarrow \mathrm{lcm}/a_i$;
for $j = 0, \ldots, \ell - 1$ do
    $c_i \leftarrow j$; $m \leftarrow M - ja_i$;
    $r \leftarrow m \bmod a_1$; lbound $\leftarrow$ ERT$(r, i-1)$;
    while $m \geq$ lbound and $c_i \leq b_i$ do
        FIND-ALL-WITH-BOUNDS$(m, i-1, c)$;
        $m \leftarrow m -$ lcm; $c_i \leftarrow c_i + \ell$;
    end while;
end for;
done.
```

Figure 3.1: Algorithm for finding all decompositions that satisfy given upper bounds on the amount of elements. The algorithm is an extended version of FIND-ALL algorithm, originally proposed in [19]. We mark the modified lines with bold type. For a integer M, FIND-ALL-WITH-BOUNDS$(M, n, 0)$ recursively generates all decompositions of M over an alphabet $\{a_1, \ldots, a_n\}$, such that each solution satisfies the elemental upper bounds $\{b_1, \ldots, b_n\}$.

3.2 Decomposing Real-valued Masses

So far, we have learned the algorithmic techniques to decompose integer masses, but in reality, neither masses of biomolecules, nor peak masses from the MS output are integers. We have thus to transform the integer decomposition problem into a problem instance with *real-valued* coefficients, as well accounting for the inaccuracies of the mass spectrometer measurement.

Real Mass Decomposition Problem. To this end, we want to find all molecules with mass M_0 in the range $[M_l, M_u] \subseteq \mathbb{R}^+$, where $M_l := M_0 - \epsilon$ and $M_u := M_0 + \epsilon$, for some mass measurement accuracy ϵ.

Formally, the *Real Mass Decomposition Problem* (RMDP) can be stated as follows:

> Given an alphabet $\Sigma = \{a_1, \ldots, a_n\}$, with $a_j \in \mathbb{R}^+$, for $j = 1, \ldots, n$, of monoisotopic masses of elements, and a mass range $[M_l, M_u] \subseteq \mathbb{R}^+$, find all decompositions $c = (c_1, \ldots, c_n)$ with $c_j \in \mathbb{N}_0$, for $j = 1, \ldots, n$, such that
>
> $$a_1c_1 + a_2c_2 + \cdots + a_nc_n \in [M_l, M_u]. \qquad (3.1)$$

This problem is closely related to a well-known *Integer Knapsack Problem* [70], but instead of maximizing the profit $\sum_{i=1}^{n} c_i a_i$ as in a classical knapsack, RMDP looks for the "profit" that falls into a certain, typically a quite tiny range. Without a loss of generality, we can assume $a_1 < a_2 < \cdots < a_n$.

A naive approach is to compute all vectors c with $c_n = 0$ and $\sum_j a_j c_j \leq M_u$, and then to test if there is some $c_n \geq 0$ such that $\sum_j a_j c_j \in [M_l, M_u]$. This leads to $\Theta(M^{n-1})$ running time, for constant element masses. As an alternative, the preprocessing technique can be used: We can compute all possible decompositions up to some upper bound M, sort them by mass and use binary search; this leads to $\Theta(M^n)$ memory requirement. These methods are unsuitable both in theoretical complexity and in practice. For example, for the alphabet of six elements most abundant in living beings CHNOPS (see Table 2.1), there exist more than $7 \cdot 10^9$ molecular formulas with mass below 1 500 Da.

Instead of straightforward but impractical solutions, we can employ the efficient integer decomposition techniques described before. For this, the input masses need to be scaled to integers using some precision. We define a *blowup factor* $b \in \mathbb{R}$, corresponding to precision $1/b$, and apply a rounding function $\phi(x) := \lceil bx \rceil$ on the coefficients in (3.1).

After modification, our alphabet masses $a'_j := \phi(a_j)$ and upper and lower bounds $M'_l := \phi(M_l)$, $M'_u := \phi(M_u)$ of the mass interval form another RMDP instance:

$$a'_1 c_1 + a'_2 c_2 + \cdots + a'_n c_n \in [M'_l, M'_u], \tag{3.2}$$

but with *integer* coefficients. Clearly, a new and original equations are not equivalent: modifying coefficients introduces rounding errors, resulting in *false positives* (solutions of the integer coefficient equation with mass not in $[M_l, M_u]$) and *false negatives* (solutions with mass in $[M_l, M_u]$ that are not part of the output of the integer coefficient equation). Filtering out false positives is straightforward by examining boundaries of (3.1). We now focus on the more compelling problem of false negative solutions that lack in the integer coefficient equation.

Let us first consider the lower bound M'_l.

Lemma 3.1. *If a solution vector c satisfies $\sum_j a_j c_j \geq M_l$, then $\sum_j a'_j c_j \geq M'_l$.*

Proof. Clearly, $\sum_j a_j c_j \geq M_l$ implies $\sum_j a'_j c_j \geq \sum_j b a_j c_j \geq b M_l$. Now, consider two possibilities: either $\sum_j a'_j c_j = b M_l$, then $b M_l$ is integer, and

$$\sum_j a'_j c_j = b M_l = \lceil b M_l \rceil = M'_l,$$

or $\sum_j a'_j c_j > b M_l$, then

$$\sum_j a'_j c_j \geq b M_l + 1 \geq \lceil b M_l \rceil = M'_l.$$

In both cases, $\sum_j a'_j c_j \geq M'_l$. □

In contrast to unmodified lower bound, the upper bound M'_u has to be increased to guarantee that no solutions of (3.1) are missed. For every element $a_j \in \Sigma$, for $j = 1, \dots, n$, we define relative rounding errors

$$\Delta_j = \Delta_j(b) := \frac{\lceil b a_j \rceil - b a_j}{a_j}.$$

Note that from $0 \le \lceil b a_j \rceil - b a_j \le 1$ follows $0 \le \Delta_j \le \frac{1}{a_j}$. Let $\Delta = \Delta(b) := \max\{\Delta_j\}$ denote the maximal relative rounding error.

Lemma 3.2. *If a solution vector c satisfies $\sum_j a_j c_j \le M_u$, then $\sum_j a'_j c_j \le b M_u + M_u \Delta$.*

Proof. From $\sum_j a_j c_j \le M_u$ follows $\sum_j b a_j c_j \le b M_u$, or $0 \le b M_u - \sum_j b a_j c_j$. Adding $\sum_j a'_j c_j$ to both parts leads to $\sum_j a'_j c_j \le b M_u + \sum_j (a'_j - b a_j) c_j$ and our proof follows from

$$\begin{aligned} 0 &\le \sum_j (a'_j - b a_j) c_j = \sum_j \frac{\lceil b a_j \rceil - b a_j}{a_j} a_j c_j \\ &\le \sum_j \Delta_j a_j c_j \le \Delta \sum_j a_j c_j \le M_u \Delta. \end{aligned}$$

□

So, we find the bounds of the integer coefficient equation as $\lceil b M_l \rceil$ to $\lfloor b M_u + \Delta M_u \rfloor$. One can easily check that these bounds are tight. Before modifying the bounds, we had to decompose $(M_u - M_l) b$ integers, but the modification introduces $M_u \Delta$ extra values to decompose, while this number is independent of the interval size $M_u - M_l$. For example, for the elements CHNOPS and the blowup factor $b = 10^5$, the maximum relative rounding error $\Delta(b) = \Delta_{\mathrm{H}}(b) = 0.4961178$. Thus, for $M = 1000$ we have to decompose 496 integers more. As we will see in the next section, the running time of this approach is dominated by the number of *decompositions* of these integers, and not by the number of integers itself.

In applications, a blowup factor b, or a precision $1/b$ is simply a parameter of the decomposition algorithm and in principle independent of the measurement accuracy ϵ. To avoid rounding error accumulation, precision is typically chosen one to two orders of magnitude smaller than the measurement accuracy.

3.2.1 Approximating Number of Decompositions

The number of decompositions $\gamma(M)$ for an integer mass M over $\{a_1, \dots, a_n\}$ grows rapidly with increasing M, namely with a polynomial of degree $n-1$ in M (Schur's Theorem [127]):

$$\gamma(M) \sim \frac{1}{(n-1)!\, a_1 \cdots a_n} M^{n-1}. \tag{3.3}$$

Note that this is a rather crude approximation of the true number of decompositions as convergence is slow. A closer approximation is given in [9], of which three leading terms are:

$$\frac{1}{a_1 \cdots a_n}\left(\frac{M^{n-1}}{(n-1)!}+\frac{M^{n-2}}{2(n-2)!}\sum_{i=1}^{n} a_i+\frac{M^{n-3}}{4(n-3)!}(\frac{1}{3}\sum_{i=1}^{n} a_i^2+\sum_{i<j} a_i a_j)\right). \tag{3.4}$$

For example, for CHNOPS alphabet this implies that the number of molecules $\hat{\gamma}(M,\epsilon)$ with real mass in the interval $[M, M+\epsilon]$ can be approximated as

$$\begin{aligned}\hat{\gamma}(M,\epsilon) \approx\; & 3.10657 \cdot 10^{-9}\,\epsilon\, M^5 + 8.22867 \cdot 10^{-7}\,\epsilon\, M^4 \\ & + 8.05088 \cdot 10^{-5}\,\epsilon\, M^3.\end{aligned} \tag{3.5}$$

In Figure 3.2, we plot the number of decompositions for masses up to 2 000 Da over CHNOPS elements. It can be seen that Equation (3.5) gives a very good approximation of $\hat{\gamma}(M,\epsilon)$ over CHNOPS elements.

We can use (3.4) to approximate the number of amino acid decompositions $\hat{\gamma}(M,\epsilon)$ with mass in the interval $[M, M+\epsilon]$, over the alphabet with 19 standard amino acids (I and L are indistinguishable):

$$\hat{\gamma}(M,\epsilon) \approx\; 1.12687 \cdot 10^{-55}\,\epsilon\, M^{18} + 2.29513 \cdot 10^{-51}\,\epsilon\, M^{17} + 2.16611 \cdot 10^{-47}\,\epsilon\, M^{16}. \tag{3.6}$$

Unfortunately, approximation (3.6) over the amino acids alphabet is very inaccurate for masses below 10 000 Da. To this end, we use an improved version of (3.6) with eight leading coefficients [74]:

$$\begin{aligned}\hat{\gamma}(M,\epsilon) \approx\; & 1.12687 \cdot 10^{-55}\,\epsilon\, M^{18} + 2.29513 \cdot 10^{-51}\,\epsilon\, M^{17} + 2.16611 \cdot 10^{-47}\,\epsilon\, M^{16} \\ & + 1.25733 \cdot 10^{-43}\,\epsilon\, M^{15} + 5.02375 \cdot 10^{-40}\,\epsilon\, M^{14} + 1.46529 \cdot 10^{-36}\,\epsilon\, M^{13} \\ & + 3.22832 \cdot 10^{-33}\,\epsilon\, M^{12} + 5.48395 \cdot 10^{-30}\,\epsilon\, M^{11}.\end{aligned} \tag{3.7}$$

Note also that the true number of amino acid decompositions is oscillating with high intensity. In Figure 3.3, we plot the number of amino acid decompositions with mass up to M, and the true and approximate number of decompositions with mass M for bin width $\epsilon = 0.001\,\text{Da}$, for masses $0 \leq M \leq 2500$ Da.

It can be seen that the number of decompositions grows very fast with the increasing mass, thus for higher mass regions, it soon becomes impractical to look for all possible interpretations of a peak based on the mass property alone. In the next chapter, we show how to take into account peak intensities, commonly available in a peak list, using the *isotopic* information from sample molecules, and to employ this knowledge for *de novo* identification of the molecular formula of metabolites.

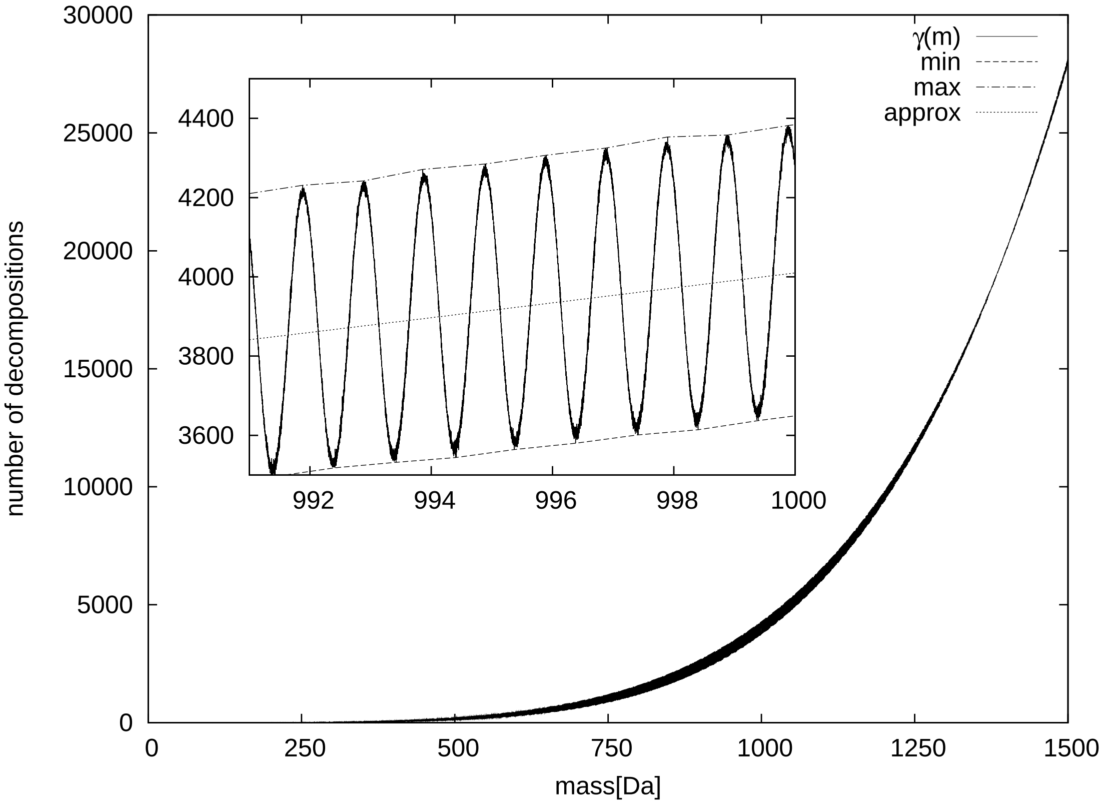


Figure 3.2: Number of decompositions over the elements CHNOPS for intervals of width $\epsilon = 0.001$ Da. Minima and maxima taken in intervals of width 1 Da. True number of decompositions in comparison with approximate (3.5) (approx). As is shown in the inlay, $\hat{\gamma}(M, \epsilon)$ varies with a periodic function of period of about 1 Da.

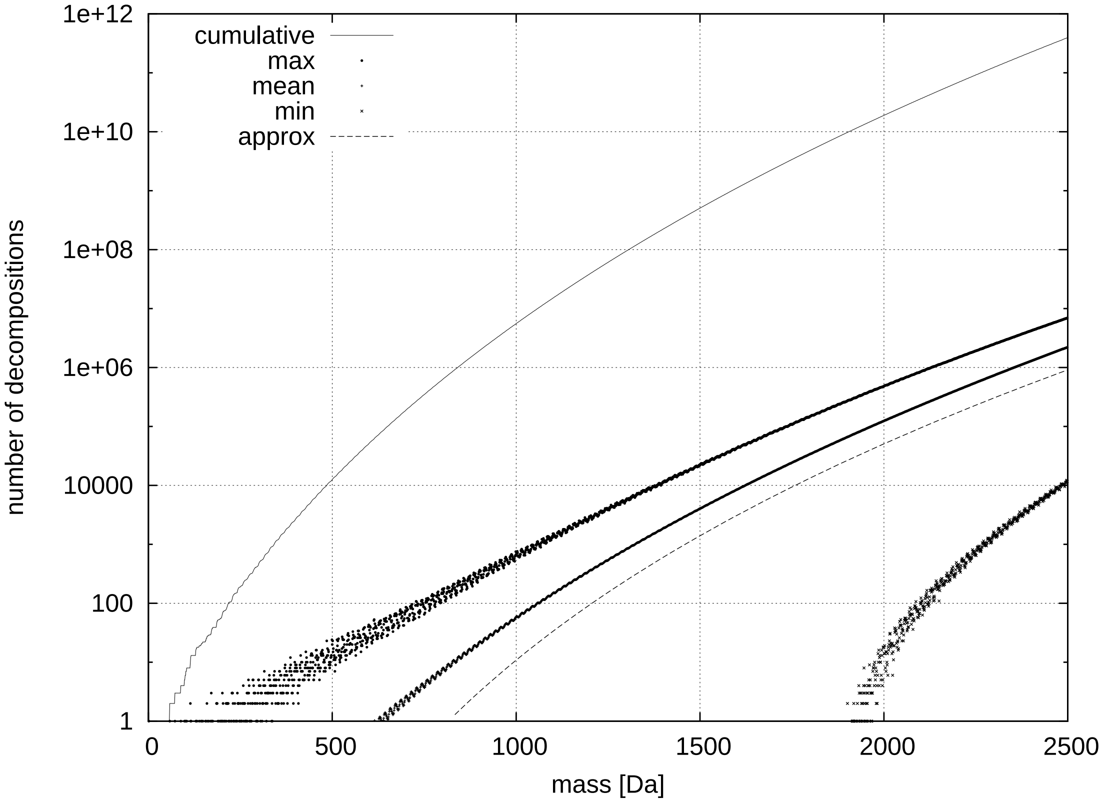


Figure 3.3: Number of amino acid decompositions with mass up to M (*cumulative*), true and approximate (*approx*) number. Approximate number of decompositions from (3.7) with mass M to $M + \epsilon$, for $\epsilon = 0.001\,\mathrm{Da}$. Maxima (*max*), mean, and minima (*min*) of the true number calculated using bin width 1 Da.

4 Molecular Formula Identification of Metabolites

In principle, elemental compositions of small molecules can be identified using only accurate output masses. However, even with high mass accuracy (< 1 ppm), there are many chemically possible formulas obtained in higher mass regions. Recently, it has been shown that this information is, therefore, not sufficient to identify a compound, and other information such as *isotopic abundances*, also needs to be taken into account [72]. Obtaining an accurate isotope pattern from a high resolution mass spectrometer, we want to use this information to identify the molecular formula of a sample molecule.

Related Approaches. To determine the molecular formula, Kind and Fiehn [72] suggested to proceed as follows: First, find all molecular formulas based on the monoisotopic mass of the measured peak list. Then, for every generated candidate molecule, simulate its isotope pattern, and compare it with the measured spectrum. However, the strategy proposed in [72] remained only a suggestion with no implementation or evaluations. A number of other experimental studies using a similar setup have been reported in the literature, e.g., [64, 88, 52]. The most notable results were achieved in [64] using high-performance LC/FTICR-MS interface: almost 500 metabolites in Tomato (*Solanum lycopersicum*) from the tandem MS dataset were claimed to be newly discovered.

Zhang *et al.* [132] focused on a related problem for analyzing peptides, but this method strongly relies on several *ad hoc* criteria regarding allowed molecular formulas and employs a heuristic search. Böcker and Rasche [21] proposed a method for the automated interpretation of metabolite tandem mass spectra but ignore isotope patterns.

Very recently, a new probabilistic approach was proposed [102] for assigning molecular formulas to the peaks in the measured spectrum using a set of possible formulas and putative biochemical transformations. To enumerate over the complete set of assignments of masses to formulas, a Gibbs sampling procedure is employed. Although no references or comparisons to existing approaches are provided, the proposed method is very interesting, and first results on small sample molecules are promising. This method is also able to include isotopic peaks in the model as additional compounds with respect to the monoisotopic mass. However, this can increase the search space dramatically, and make the enumeration of the extended full distribution over the formula assignments infeasible in practice.

Algorithmic Pipeline. Our input is a peak list of masses with normalized intensities that corresponds to the isotope pattern of the sample molecule. We want to find that molecule's formula whose isotope pattern best matches the input. To undertake this task we proceed as follows: First, using the decomposition techniques for real-valued and integer masses described earlier, all molecular formulas are generated that share the monoisotopic mass with the input peak list. Second, to filter out those molecules that cannot exist in nature due to the chemical consideration, Senior rules are applied to remove molecular formulas to which no legitimate molecular structure exists (see Section 2.1 on page 5). Followed by the computation of theoretical isotope patterns for remaining candidate molecules, each simulated isotope pattern is then matched and ranked against the measured peak list, and the pattern with the best score is reported.

The remainder of this chapter is organized as follows: We start by introducing the concept of isotope patterns, and methods to compute the isotope pattern from the molecular formula of a sample molecule. Then, we discuss a scoring scheme to assess the similarity between two isotope patterns for the ranking of candidate molecular formulas. Finally, we evaluate the performance of our approach using several experimental datasets obtained from different high resolution mass spectrometers.

4.1 Isotope Patterns

The isotope patterns of molecules have always been available in MS measurements, but only recently the mass accuracy and resolution of mass spectrometers have become sufficient to deduce molecular formulas from such data.

4.1.1 Isotope Species

The *mass* of a molecule is the sum of masses of its atoms, while its *nominal mass* is the total number of protons and neutrons in these atoms. Clearly, both mass and nominal mass depend on the individual isotopes a molecule is composed of, thus on the *isotope species* of a molecule. We call the isotope species *monoisotopic* if all its constituting isotopes are also monoisotopic. The *monoisotopic (nominal) mass* of the molecule is the sum of (nominal) masses of the monoisotopic species. For example, the molecule of adenine triphosphate (ATP) $C_{10}H_{16}N_5O_{13}P_3$ has a monoisotopic mass of 506.99575 Da, and its monoisotopic nominal mass equals to 507 Da.

The number of isotope species with distinct mass for a molecule with n_H hydrogen, n_C carbon, n_N nitrogen, n_O oxygen, n_P phosphor, and n_S sulfur atoms is

$$(n_H + 1)(n_C + 1)(n_N + 1)\binom{n_O+2}{2}\binom{n_O+3}{3} \tag{4.1}$$

This follows because for an element E with r isotopes, a molecule E_l consisting of l atoms of the element has $\binom{l+r-1}{r-1}$ different isotope species. In general, $\binom{l+r-1}{r-1}$ possibilities exist to distribute l identical objects into r groups. For example, a molecule with l nitrogen atoms (N_l) has $l+1$ isotope species: one consisting only of ^{14}N atoms, one with one ^{15}N atom and $l-1$ ^{14}N atoms, etc. Noticing that distributing a particular element over its isotopes is independent of other elements, the formula (4.1) is obtained by multiplying the number of possibilities for different elements.

For example, the molecule of ATP has 117 810 isotope species. The probability to observe a certain isotope species is a product of the relative abundances of the constituting isotopes. In Table 4.1, the first ten isotope species of ATP and their relative abundances are given.

Formally, we define the mass distribution of an element E by a discrete random variable X_E with finite space $\Omega_E \subseteq \mathbb{N}$: For example, X_{N} with state space $\{14.003074, 15.001090\}$ and

$$\mathbb{P}\big(X_{\mathrm{N}} = 14.003074\big) = 0.99634 \quad \text{and} \quad \mathbb{P}\big(X_{\mathrm{N}} = 15.001090\big) = 0.00366$$

is the random variable of nitrogen. Then, given a molecule consisting of l elements, its *mass distribution* X can be expressed as a sum of independent random variables that correspond to the constituting elements:

$$X := X_1 + \ldots + X_l,$$

where $X_i \sim X_E$, for $i = 1, \ldots, l$, with E being the corresponding element. For example, the mass distribution of a water molecule is the sum of mass distributions of 2 hydrogens and 1 oxygen, $X_{\mathrm{H_2O}} := X_{\mathrm{H}} + X_{\mathrm{H}} + X_{\mathrm{O}}$.

Given the isotope species of two molecules, the isotope species of a combined molecule can be computed straightforward by folding individual species: Species with masses m_1, m_2 and probabilities p_1, p_2 contribute to the isotope species with mass $m_1 + m_2$ and probability $p_1 p_2$ in the combined molecule. After sorting and merging the species with identical mass, the resulting isotope species of the combined molecule are obtained.

^{12}C	^{13}C	^{1}H	^{2}H	^{14}N	^{15}N	^{16}O	^{17}O	^{18}O	^{31}P	nominal mass (Da)	mass (Da)	abundance %
10	0	16	0	5	0	13	0	0	3	507	506.995751	84.9310
10	0	16	0	4	1	13	0	0	3	508	507.992786	1.5599
9	1	16	0	5	0	13	0	0	3	508	507.999106	9.5331
10	0	16	0	5	0	12	1	0	3	508	507.999968	0.4205
10	0	15	1	5	0	13	0	0	3	508	508.002028	0.2038
10	0	16	0	3	2	13	0	0	3	509	508.989821	0.0114
9	1	16	0	4	1	13	0	0	3	509	508.996141	0.1750
10	0	16	0	4	1	12	1	0	3	509	508.997003	0.0077
10	0	15	1	4	1	13	0	0	3	509	508.999063	0.0037
10	0	16	0	5	0	12	0	1	3	509	508.999997	2.2134

Table 4.1: Isotope species of adenine triphosphate (ATP) molecule $\mathrm{C_{10}H_{16}N_5O_{13}P_3}$, sorted by mass. Isotope species with nominal mass ≥ 509 omitted.

4.1.2 Isotopic Distributions

Unfortunately, even the employment of high-resolution MS usually does not allow to resolve isotope species with identical nominal mass. Instead, these species occur as a single peak in the MS output. To simulate the similar behavior while calculating the theoretical peak list, the isotope species with identical nominal mass have to be combined.

Similar to the mass distribution, we define the nominal mass distribution of an element E by a discrete random variable Y_E with finite state space $\Omega_E \subseteq \mathbb{N}$: For example, Y_N with state space $\{14, 15\}$ and

$$\mathbb{P}\big(Y_\mathrm{N} = 14\big) = 0.99634 \quad \text{and} \quad \mathbb{P}\big(Y_\mathrm{N} = 15\big) = 0.00366$$

is the random variable of nitrogen.

Given the molecule with n elements, its nominal mass distribution is $Y := Y_1 + \ldots + Y_l$, where $Y_i \sim Y_E$, for $i = 1, \ldots, l$, with E being the corresponding element. We refer to the resulting distribution of nominal masses as the *isotopic distribution* of a molecule. For the isotopic distribution of ATP molecule, see its relative abundances in Table 4.2. Note the correlation between X and Y: X_E can be seen as a function of Y_E and E.

The peak at the monoisotopic mass is usually referred to as a *monoisotopic* peak, and the subsequent peaks as +1, +2, ... peaks. In an ideal mass spectrum, normalized peak intensities would correspond to the molecule's isotopic distribution. But due to the imperfection of MS instruments, the actual output contains +1, +2, ... peaks that, in fact, are superpositions of the isotope species with identical nominal mass. It is reasonable to assume that the mass of such a superposition peak is the mean mass of all isotope species that contribute to its intensity, that is, the species with the same nominal mass as of the superposition peak [101].

nominal mass (Da)	507	508 (+1)	509 (+2)	510 (+3)	511 (+4)	512 (+5)
abundance %	84.9309	11.7175	2.9653	0.3343	0.0469	0.0044
mean peak mass (Da)	506.995751	507.998347	509.000220	510.002655	511.004629	512.006961

Table 4.2: Isotope pattern of the ATP molecule $C_{10}H_{16}N_5O_{13}P_3$. Peaks with nominal mass ≥ 513 have abundances $< 0.001\,\%$ and omitted.

Formally, we define a mass function $\tilde{\mu} : \mathbb{N} \to \mathbb{R}$ that maps the nominal masses from Table 2.1 on page 6 to the corresponding real masses: $\tilde{\mu}(1) = 1.007825$, $\tilde{\mu}(2) = 2.014102$, ..., $\tilde{\mu}(34) = 33.967867$, $\tilde{\mu}(36) = 35.967081$.

Given a molecule with monoisotopic nominal mass N, let $X := X_1 + \ldots + X_l$ be its mass distribution, and $Y := Y_1 + \ldots + Y_l$ be its isotopic distribution. Then, the mean peak mass of the $+k$ peak is

$$m_k = \frac{1}{\mathbb{P}(Y = n + k)} \sum_{\sum_i n_i = n+k} \mathbb{P}(Y_1 = n_1, \ldots, Y_l = n_l) \cdot \big(\tilde{\mu}(n_1) + \cdots + \tilde{\mu}(n_l)\big). \tag{4.2}$$

Here, the sum is calculated over all l-tuples $(n_1, \ldots, n_l)$ of non-negative integers that satisfy $\sum_{i=1}^{l} n_i = n + k$. We refer to the isotopic distribution together with the mean peak masses as the molecule's *isotope pattern*. See Table 4.2 for the isotope pattern of the ATP molecule.

4.2 Computing Isotope Patterns

The calculation of the molecule's isotope pattern can be divided into two parts: computing the isotopic distribution (relative abundances of the nominal masses), and the mean peak masses of the molecule. Regarding the calculation of isotopic distributions, note that all molecules that consist of elements CHNOPS have isotope distributions that decrease rapidly with increasing mass. Thus, we can limit our attention to calculating the first K non-zero entries of the distribution, for rather small K such as $K = 10$. For example, amongst 11479 entries in the KEGG COMPOUND database [66] (release 42.0) with mass below 3 000 Da, no molecule has an intensity of the +10 peak larger than 0.00007.

4.2.1 Folding Isotopic Distributions

Finding the isotopic distribution of an arbitrary molecule can be further divided into two parts: First, computing the isotopic distributions of molecules E_l consisting of l identical elements E, and then combining these parts by folding distributions from all element types present in the molecule. For example, for a sucrose molecule $\mathrm{C}_{12}\mathrm{H}_{22}\mathrm{O}_{11}$, this means computing individual distributions $Y_{\mathrm{C}_{12}}$, $Y_{\mathrm{H}_{22}}$ and $Y_{\mathrm{O}_{11}}$, and then combining them in the resulting distribution $Y := Y_{\mathrm{C}_{12}} + Y_{\mathrm{H}_{22}} + Y_{\mathrm{O}_{11}}$.

Since the elements hydrogen, carbon, and nitrogen only have two isotopes each, the isotopic distribution of a molecule E_l with $E \in \{\mathrm{H}, \mathrm{C}, \mathrm{N}\}$ follows a binomial distribution:

$$q_k := \mathbb{P}(E_l \text{ has nominal mass } +k) = \binom{l}{k} p^k (1-p)^{l-k}, \tag{4.3}$$

where p is the probability of the +1 isotope. $q_0 = (1-p)^l$, and the values of the q_k can be computed iteratively:

$$q_{k+1} = \frac{l-k}{k+1} \cdot \frac{p}{1-p} q_k \qquad \text{for } k \geq 0, \tag{4.4}$$

The overall calculation time is $O(K + \log l)$ if $\log l$ multiplications is used to calculate q_0.

For elements E with $r > 2$ isotopes, such as oxygen and sulfur, the isotopic distribution of E_l can be computed as follows: Let p_i, for $i = 0, \ldots, r-1$, denote the relative abundance of the i-th isotope. Then,

$$q_k := \mathbb{P}(E_l \text{ has nominal mass } +k\ ) = \sum \tbinom{l}{l_0, l_1, \ldots, l_{r-1}} \cdot \prod_{i=0}^{r-1} p_i^{l_i}, \tag{4.5}$$

where the sum is calculated over all $l_0, \ldots, l_{r-1} \geq 0$ satisfying $\sum_{i=0}^{r} l_i = l$ and $\sum_{i=1}^{r} i \cdot l_i = k$ [63].

The tuples $(l_0, \ldots, l_r)$ satisfying $\sum i \cdot l_i = k$ are the integer partitions of k into at most r parts. To compute all partitions, a greedy algorithm with a simple recursion can be employed. However, this approach faces the problem that the number of summands in (4.5) grows rapidly, at least as a polynomial in k of degree $r-1$ [123].

Alternatively, the isotopic distributions of oxygen O_l and sulfur S_l can be computed by consecutive folding of the corresponding distributions using a Russian multiplication scheme discussed below.

Given two discrete random variables Y and Y' with state spaces $\Omega, \Omega' \subseteq \mathbb{N}$, we can compute the combined distribution of the random variable $Z := Y + Y'$ by folding the distributions as follows:

$$\mathbb{P}(Z = n) = \sum\nolimits_k \mathbb{P}(Y = k) \cdot \mathbb{P}(Y' = n - k). \tag{4.6}$$

Computing the first K values of this sum requires $O(K^2)$ time. To fold isotopic distributions of individual elements, Kubinyi proposed [75] to use a Russian multiplication scheme: We start with the molecule consisting of a single element E, and using formula (4.6) compute the isotopic distributions of molecules with 2, 4, 8, ... atoms. For example,

$$Z_1 + Z_2 + Z_3 + Z_4 + Z_5 + Z_6 + Z_7 = \big((Z_1 + Z_2) + (Z_3 + Z_4)\big) + (Z_5 + Z_6) + Z_7,$$

where Z_i denote independent and identically-distributed (i. i. d.) random variables. Then, we combine these distributions to compute that of E_l. This adds a logarithmic factor to the algorithm's total running time, altogether $O(K^2 \log l)$.

In applications, the distributions of elements with more than two isotopes can be computed not on the fly but during preprocessing. For all $l \leq L$, a lookup table is created that uses $O(KL)$ space for every such element, where L is small: In our applications, the mass range of the analyzed biomolecules usually does not exceed 2 000 Da; 128 oxygen atoms already have a mass of about 2 048 Da, being larger than the relevant upper bound.

After the isotopic distributions of individual elements have been computed (C, H, N) or looked up from memory (O, S), we combine them by folding using formula (4.6). Doing so requires $O(|\Sigma| \cdot K^2)$ time in total.

Note that we can use Fourier transforms of isotopic distributions, and instead of folding the distributions, multiply the Fourier transforms [100]. This can eventually substitute the K^2 factor in the algorithm's running time by the $K \log K$ factor. However, as we restrict ourselves to rather small K such as $K = 10$, this will not speed up the algorithm in practice. Moreover, this approach may face the problem of numerical errors.

4.2.2 Folding Peak Masses

We now move to the more intriguing issue of efficiently calculating the mean peak masses of a distribution. Computing the mean peak mass using formula (4.2) is highly inefficient, because we have to calculate the sum over all isotope species. Pruning techniques have been proposed to speed up computation [131], but they result in a drop of accuracy [101]. We now present a simple approach to compute the mean masses similar to the folding of isotopic distributions:

Given two molecules with monoisotopic nominal masses N and N', and isotopic distributions $Y = Y_1 + \cdots + Y_l$ and $Y' = Y'_1 + \cdots + Y'_L$, respectively. Let m_k and m'_k be the mean peak masses of the $+k$ peaks, and $p_k := \mathbb{P}(Y = N + k)$ and $q_k := \mathbb{P}(Y' = N' + k)$ denote the respective probabilities. Consider the random variable $Z = Y + Y'$ with monoisotopic nominal mass $\tilde{N} = N + N'$.

Theorem 1. *The mean peak mass $\tilde{m}_k$ of the $+k$ peak of the random variable $Z = Y + Y'$ can be computed as:*

$$\tilde{m}_k = \frac{1}{\sum_{j=0}^{k} p_j q_{k-j}} \cdot \sum_{j=0}^{k} p_j q_{k-j} \left(m_j + m'_{k-j} \right) \tag{4.7}$$

Note that $\sum_{j=0}^{k} p_j q_{k-j} = \mathbb{P}(Z = \tilde{N} + k)$.

Proof. Let $\vec{N} = (N_1, \ldots, N_l) \in \mathbb{N}^l$ and $\vec{N'} = (N'_1, \ldots, N'_L) \in \mathbb{N}^L$ be vectors of nominal masses. We denote $\sum \vec{N} := \sum_{i=1}^{l} N_i$ and $\sum \vec{N'} := \sum_{i=1}^{L} N'_i$. Let $\vec{Y} := (Y_1, \ldots, Y_l)$ and $\vec{Y'} := (Y'_1, \ldots, Y'_L)$ be vectors of the input random variables, and note that

$$\mathbb{P}(\vec{Y} = \vec{N}, \vec{Y'} = \vec{N'}) = \mathbb{P}(\vec{Y} = \vec{N}) \cdot \mathbb{P}(\vec{Y'} = \vec{N'})$$

due to the independence of the underlying random variables. Finally, we set $\tilde{\mu}(\vec{N}) = \sum_{i=1}^{l} \tilde{\mu}(N_i)$ and define $\tilde{\mu}(\vec{N'})$ analogously.

Ignoring the normalization factor, we can rearrange formula (4.2) as

$$\mathbb{P}(Z = \tilde{N} + k) \cdot \tilde{m}_k = \sum_{\sum \vec{N} + \sum \vec{N'} = \tilde{N} + k} \mathbb{P}(\vec{Y} = \vec{N}, \vec{Y'} = \vec{N'}) \cdot \left(\tilde{\mu}(\vec{N}) + \tilde{\mu}(\vec{N'}) \right).$$

Now observe that this formula can be divided into two independent sums of the form

$$\sum_{\sum \vec{N} + \sum \vec{N'} = \tilde{N} + k} \mathbb{P}(\vec{Y} = \vec{N}, \vec{Y'} = \vec{N'}) \cdot \tilde{\mu}(\vec{N}) \tag{4.8}$$

and a second summand, where $\tilde{\mu}(\vec{N})$ is substituted with $\tilde{\mu}(\vec{N'})$.

We now concentrate on (4.8):

$$\begin{aligned}
&\sum_{\sum \vec{N} + \sum \vec{N}' = \tilde{N}+k} \mathbb{P}(\vec{Y} = \vec{N}, \vec{Y}' = \vec{N}') \cdot \tilde{\mu}(\vec{N}) \\
&\qquad = \sum_{j=0}^{k} \sum_{\sum \vec{N} = N+j} \sum_{\sum \vec{N}' = N'+k-j} \mathbb{P}(\vec{Y} = \vec{N}) \mathbb{P}(\vec{Y}' = \vec{N}') \cdot \tilde{\mu}(\vec{N}) \\
&\qquad = \sum_{j=0}^{k} \sum_{\sum \vec{N} = N+j} \mathbb{P}(\vec{Y} = \vec{N}) \cdot \tilde{\mu}(\vec{N}) \sum_{\sum \vec{N}' = N'+k-j} \mathbb{P}(\vec{Y}' = \vec{N}') \\
&\qquad = \sum_{j=0}^{k} \sum_{\sum \vec{N} = N+j} \mathbb{P}(\vec{Y} = \vec{N}) \cdot \tilde{\mu}(\vec{N}) \cdot \mathbb{P}(Y_1' + \cdots + Y_L' = N' + k - j) \\
&\qquad = \sum_{j=0}^{k} \mathbb{P}(Y' = N' + k - j) \sum_{\sum \vec{N} = N+j} \mathbb{P}(\vec{Y} = \vec{N}) \cdot \tilde{\mu}(\vec{N}) \\
&\qquad = \sum_{j=0}^{k} q_{k-j} p_j m_j,
\end{aligned}$$

where the last equality follows from the definition of m_j,

$$m_j = \frac{1}{p_j} \sum_{\sum \vec{N} = N+j} \mathbb{P}(\vec{Y} = \vec{N}) \cdot \tilde{\mu}(\vec{N}).$$

Analogously, we can show that

$$\sum_{\sum \vec{N} + \sum \vec{N}' = \tilde{N}+k} \mathbb{P}(\vec{Y} = \vec{N}, \vec{Y}' = \vec{N}') \cdot \tilde{\mu}(\vec{N}') = \sum_{j=0}^{k} q_{k-j} p_j m_j',$$

which concludes the proof of the theorem. ☐

The theorem allows us to “fold” mean peak masses of two distributions to compute the mean peak masses of their sum. This means that we can compute the mean masses as efficiently as the distribution itself, i.e., in $O(K^2 \log l)$ time. This improves on the best present-day approach [101], substituting the linear running time dependence on the number of atoms l by its logarithm.

4.3 Scoring Candidate Molecules

Our task is to distinguish between candidate molecules produced by the decomposition of the monoisotopic mass, and to choose the candidate, whose isotope pattern matches the measured peak list best. In Section 4.1, we have seen how to efficiently calculate the isotope pattern from the molecular formula. Now we want to compare this simulated isotope pattern with the measured peak list by matching the pairs of peaks in both.

To assess the similarity between mass spectra, Zhang and Chait [134] and Zhang *et al.* [133] propose the probabilistic approach employing Bayesian Statistics:

$$\mathbb{P}(\mathcal{M}_j|\mathcal{D},\mathcal{B}) = \frac{\mathbb{P}(\mathcal{M}_j|\mathcal{B})\,\mathbb{P}(\mathcal{D}|\mathcal{M}_j,\mathcal{B})}{\sum_i \mathbb{P}(\mathcal{M}_i|\mathcal{B})\,\mathbb{P}(\mathcal{D}|\mathcal{M}_i,\mathcal{B})},$$

where $\mathcal{M}_i$ are the models (the candidate molecules), $\mathcal{D}$ is the data (the measured peak list), and $\mathcal{B}$ stands for any prior background information.

Concerning the background information, we first employ the prior knowledge about the candidate molecules, by assigning the prior probability $\mathbb{P}(\mathcal{M}_j|\mathcal{B})$ zero for all molecules but the decompositions of the monoisotopic mass. Next, we model the chemical bonding rules by setting the prior probability to zero for molecular formulas that do not satisfy the conditions of the Senior's theorem (see Section 2.1 on page 5). For example, molecules violating Senior's third rule are rare, particularly for natural compounds: less than 0.16 % of substances in the KEGG COMPOUND database violate this rule.

Other priors such as the hetero-to-carbon ratio [73] exist that are typically derived empirically based on the already known compounds. To keep the comparison unbiased for unknown molecules, we deliberately avoid further use of such priors.

Next, we assign probabilities to the measured peak masses and intensities. Let $\mathbb{P}(M_j|m_j)$ denote the probability to detect peak j at mass M_j when its actual mass is m_j, and $\mathbb{P}(f_j|p_j)$ denote the probability to detect peak j with intensity f_j when its actual intensity is p_j. Then, assuming independence, particularly from background information, the probability to observe the measured peak list, given a certain candidate molecule and background information can be calculated as follows:

$$\mathbb{P}(\mathcal{D}|\mathcal{M},\mathcal{B}) = \prod_j \mathbb{P}(M_j|m_j) \prod_j \mathbb{P}(f_j|p_j) \tag{4.9}$$

Note that the peak intensities are, in fact, not independent, since they sum up to one, but the product (4.9) can be viewed as a rough estimate of the true probability.

4.3.1 Estimating Probabilities of Peak Masses

As mentioned in Section 2.5.3, experimentalists typically assume that the mass deviation of a mass spectrometer follows the Gaussian distribution with mean zero. Given the relative mass accuracy α of the measurement (in ppm), we can assign the standard deviation $\sigma_j := \frac{1}{3}\alpha\,10^{-6}\,M_j$ for peak j, assuming that more than 99.7 % of measurements belong to the mass range $[-3\sigma_j, +3\sigma_j]$. However, we notice that peaks of high intensity reveal better mass accuracy than those of low intensity, which can be referred to the complications of extracting the low intensity peaks from the background noise. Our data shows an approximately linear correlation between peak intensity and mass accuracy. To take this into account, we use two mass accuracies α_1 (at full intensity) and α_0 (at minimal intensity), and calculate the standard deviation for peak j as follows:

$$\sigma_j := \tfrac{1}{3}\big(p_j\,\alpha_1 + (1-p_j)\alpha_0\big)\cdot 10^{-6}\,M_j. \tag{4.10}$$

Additionally, imperfections in calibrating mass spectrometers lead to a systematic mass shift between different peak masses of the same spectrum. This occurs frequently even for high resolution MS. To eliminate this error for all masses except the monoisotopic mass, we do not compute the mass differences of the $+1, +2, \dots$ peaks directly but instead, the difference to the monoisotopic mass, $M_j - M_0$ against $m_j - m_0$, for $j \geq 1$.

Given a peak with theoretical mass m_j, we want to estimate the probability to observe a peak at mass M_j in the measured peak list, or, saying strictly, we calculate the probability of detecting a mass difference of $|M_j - m_j|$ or larger. This probability can be computed using the *complementary error function* "erfc":

$$\mathbb{P}(M_j|m_j) = \mathbb{P}(\text{mass difference} \geq x_j) = \operatorname{erfc}\Big(\frac{|x_j|}{\sqrt{2}\,\sigma_j}\Big) = \frac{2}{\sqrt{2\pi}} \int_z^{\infty} e^{-t^2/2} dt \tag{4.11}$$

with $z := \frac{|x_j|}{\sigma_j}$, where $x_0 = (M_0 - m_0)/m_0$ and $x_j = (M_j - M_0 - m_j + m_0)/m_j$, for $j \geq 1$, denote the relative mass differences.

4.3.2 Estimating Probabilities of Peak Intensities

Concerning the peak intensities, a systematic shift in the measured data can be observed: The peaks of high intensity are overestimated, while the peaks with low intensity are underestimated in the measured peak list, see Figures 4.1 and 4.2.

This problem can be attributed to inaccuracies of peak intensity calculation on the preprocessing step: peak picking algorithms typically determine peak intensities using a *signal-to-noise* ratio or height above a certain baseline (see Section 2.5.2 on page 22). The calculation of the baseline, in turn, is done using some empirical estimates that can be imprecise. These inaccuracies have unequal impact on peaks of different intensities. To adjust this systematic bias, we introduce a user-defined parameter *off*. This parameter is added to the measured intensities, which are then re-normalized again. In the next section, when evaluating our algorithm, we provide more details on setting the parameter *off*.

After applying the correction to the measured data, we observe that log ratios between measured and theoretical peak intensity $\log(f_j/p_j)$ are roughly normally distributed. Similar to mass differences, we introduce two precision parameters β_1 (at full intensity) and β_0 (at minimal intensity) to be set (in percent), and compute the standard deviation for peak j as follows:

$$\hat{\sigma}_j := \tfrac{1}{3} \log\left(1 + p_j \tfrac{\beta_1}{100} + (1 - p_j) \tfrac{\beta_0}{100}\right),$$

so that more than 99.7 % of ratios $\log(f_j/p_j)$ belong to the range $[-3\hat{\sigma}_j, +3\hat{\sigma}_j]$. Now, the computation of probability $\mathbb{P}(f_j|p_j)$ can be done analogously to (4.11).

4.4 Experimental Results

Datasets. To evaluate our method, two datasets measured on two instruments were used. Mass spectra with single charge were measured from several organic (macro)molecules, composed of elements CHNOPS. For every such spectrum, the molecular formula

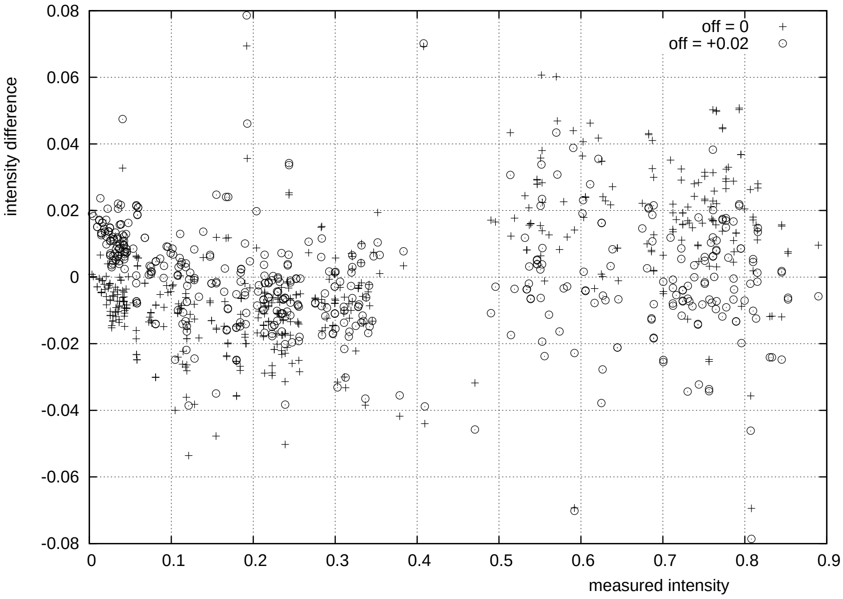


Figure 4.1: Differences between measured and theoretical peak intensities and mass accuracies for the FT-ICR dataset in comparison: measured intensities computed without correction (crosses), measured intensities computed with correction *off* = +0.02 (circles). The difference is computed as measured peak intensity minus theoretical peak intensity.

of the sample molecule is known. The spectra were acquired over a period of two years; the molecules range in mass from 117 to about 1 000 Da. Peak detection and estimation of the peak masses and intensities were performed using vendor software.

The first dataset consists of 153 peak lists. Electrospray ionization (ESI) experiments were conducted using the Fourier Transform Ion Cyclotron Resonance (FT-ICR) mass spectrometer APEX III (Bruker Daltonik GmbH, Bremen, Germany). The FT-ICR MS was equipped with a 7.0 T, 160 mm bore superconducting magnet, infinity cell, and interfaced to an external (nano)ESI ion source. Measured data was externally mass calibrated. The five analysis parameters were chosen as $\alpha_1 = 3$, $\alpha_0 = 6$, $\beta_1 = 10$, $\beta_0 = 90$, and *off* = +0.02.

The second dataset consists of 86 peak lists. ESI experiments were conducted using the oa-TOF mass spectrometer MicrOTOF (Bruker Daltonik GmbH, Bremen, Germany). Quasi-internal mass calibration was used, by measurement of an infused calibrant prior to the compound of interest. For the oa-TOF analysis, the parameters were set to $\alpha_1 = 5$, $\alpha_0 = 6.5$, $\beta_1 = 10$, $\beta_0 = 90$, and *off* = +0.02.

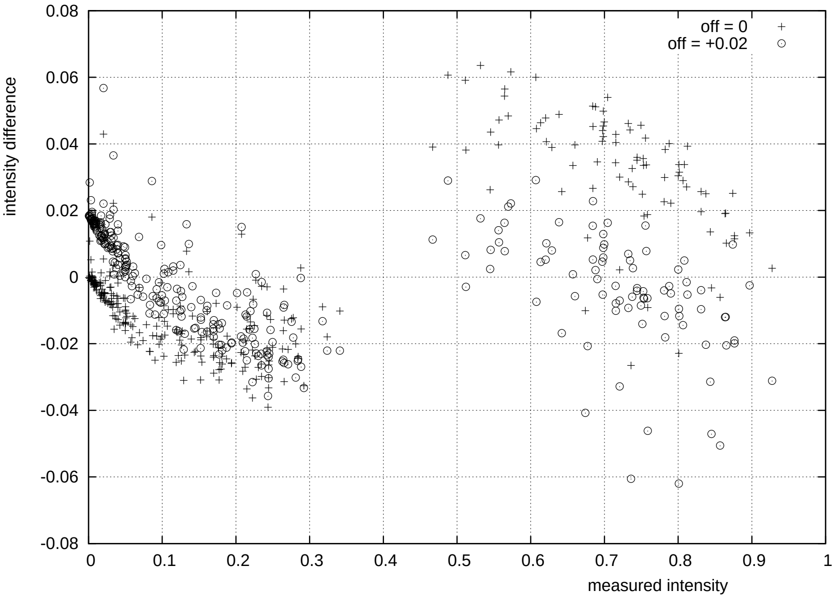


Figure 4.2: Differences between measured and theoretical peak intensities and mass accuracies for the oa-TOF dataset in comparison: measured intensities computed without correction (crosses), measured intensities computed with correction *off* = +0.02 (circles). The difference is computed as measured peak intensity minus theoretical peak intensity.

4.4.1 Identification Rates

Every input peak list consists of masses $M_0, \ldots, M_k$ and intensities $f_0, \ldots, f_k$. For every such peak list, we generated all candidate molecules such that the monoisotopic mass m_0 of the candidate molecule has a relative mass difference of at most α_1 ppm, $|M_0 - m_0| / m_0 \leq \alpha_1 \cdot 10^{-6}$. To do so, we decomposed integer masses using an appropriate scaling of the query mass and alphabet masses, see Section 3.2 on page 31, and filtered out decompositions, whose real-valued mass was outside the allowed mass range. Next, we discarded molecules that disobey Senior's rules on the chemical formation of the molecule. For each remaining molecule, we computed its theoretical isotopic pattern with K leading peaks ($K = 10$), and compared it to the measured peak list. We matched and ranked the molecules based on resulting probabilities. We did not use any other background information to identify the molecule.

For the 153 peak lists in our FT-ICR dataset, 89 resulted in a correct identification; in 86 % of the peak lists, the correct interpretation was found in the TOP 10 explanations. There is a clear correlation between mass and identification accuracy, see Table 4.3. For peak lists below 700 Da, the true interpretation was always found in the TOP 10 explanations, except in one case where it had rank 13.

For 86 peak lists in the oa-TOF dataset the correct molecular formula was found in the TOP 10 interpretations in all but two cases. Moreover, 79 out of 86 compounds were identified correctly, which corresponds to an identification rate of over 90 %, see Table 4.3. Better identification results on the oa-TOF dataset with lower mass accuracy show the crucial importance of including intensity measurements into the candidate evaluation. We note that the intensity accuracy of the oa-TOF instrument is significantly higher than that of the FT-ICR.

mass	no.	rank in output list					no. molecular formulas			
range	pl.	1	2	3–5	6–10	11+	int.	real	chem.	time
200–300	13	11	2	0	0	0	67	37.2	8.6	1.5
300–400	37	28	5	2	2	0	200	109	10.4	4.3
400–500	57	39	6	6	5	1	579.5	318.2	22.8	13.5
500–600	7	5	2	0	0	0	1800.4	990.3	59.6	40.1
600–700	4	3	0	0	1	0	2668.5	1454	37.3	55
700–800	5	0	1	1	1	2	8797.8	4812	247	232
800–900	14	3	2	1	3	5	14781.6	8101	534.6	485
900–1000	16	0	1	1	1	13	31805.7	17448	1570	1281

mass	no.	rank			no. molecular formulas			
range	pl.	1	2–6	11+	int.	real	chem.	time
100–200	7	7	0	0	11.3	7.4	1.9	0
200–300	21	21	0	0	60.1	40.1	4.6	1.4
300–400	27	26	1	0	297.8	199.7	22.6	8.1
400–500	15	14	1	0	725.6	484.1	32.4	17.3
500–600	10	7	2	1	1479.1	988.9	54.3	36
600–700	1	1	0	0	4600	3080	276	130
700–800	2	1	1	0	10336	6909	578	461
800–900	3	2	0	1	18172.3	12146	914.3	757

Table 4.3: Number of correct molecular formulas at specific positions of the output list, for the FT-ICR dataset (top) and the oa-TOF dataset (bottom). We report the number of peak lists in this mass range (no. pl.), as well as the average number of molecular formulas over all molecules in the mass range (no. molecular formulas). We distinguish between the number of integer decompositions (int.), the number of decompositions over real-valued masses (real), and the number of those molecular formulas that satisfy chemical bonding rules (chem.). Finally, we give the average running time in milliseconds per peak list (time).

Regarding the scoring parameter *off*, for both of our datasets we used the same value *off* $= +0.02$. We have also tested the robustness of our method by varying different scoring parameters: Identification accuracy remained relatively stable for small disturbances of parameter values, confer Table 4.4 and Figures 4.3 and 4.4. In applications, parameters can be estimated using a small training set.

β_1	β_0	identification rate rank1 (%)	top10 (%)
5	70	58.82	86.93
15	70	58.82	86.93
5	80	58.82	85.62
10	70	58.17	86.93
10	90	58.17	86.27
15	90	58.17	86.27
20	70	58.17	85.62
5	90	57.52	86.27
5	100	57.52	86.27
10	100	57.52	86.27
15	80	57.52	86.27
20	80	57.52	86.27
20	90	57.52	86.27
5	110	57.52	85.62
10	80	57.52	85.62
20	110	57.52	85.62
10	110	56.86	85.62
15	100	56.86	85.62
15	110	56.86	85.62
20	100	56.86	85.62

β_1	β_0	identification rate rank1 (%)	top10 (%)
5	90	91.86	97.67
5	100	91.86	97.67
10	90	91.86	97.67
10	100	91.86	97.67
15	90	91.86	97.67
15	100	91.86	97.67
20	90	91.86	97.67
20	100	91.86	97.67
5	80	90.70	97.67
5	110	90.70	97.67
10	80	90.70	97.67
10	110	90.70	97.67
15	80	90.70	97.67
15	110	90.70	97.67
20	80	90.70	97.67
20	110	90.70	97.67
5	70	89.53	97.67
10	70	89.53	97.67
15	70	89.53	97.67
20	70	89.53	97.67

Table 4.4: Identification rates for various intensity precisions β_1 (at full intensity) and β_0 (at minimal intensity), for the FT-ICR dataset (left) and the oa-TOF dataset (right). We set mass accuracies α_1 (at full intensity) and α_0 (at minimal intensity) to fixed values: $\alpha_1 = 3$, $\alpha_0 = 6$ for the FT-ICR dataset, and $\alpha_1 = 5$, $\alpha_0 = 6.5$ for the oa-TOF dataset. Parameter *off* is set to +0.02. One can see that our method is very robust to small variations of the parameters.

Running Times. All algorithms were implemented in Java, and executed within the SIRIUS software environment, see Chapter 7. All 239 peak lists were analyzed on a Pentium M 1.5 GHz processor with blowup $b = 5 \cdot 10^4$, using only a few Megabyte of memory. This results in running times of less than 1.3 seconds per peak list for the complete analysis of one peak list. Clearly, running times depend on masses of molecules, again see Table 4.3. Increasing the blowup beyond $5 \cdot 10^4$ increased the running times, presumably because the smaller table can be kept in the processor cache.

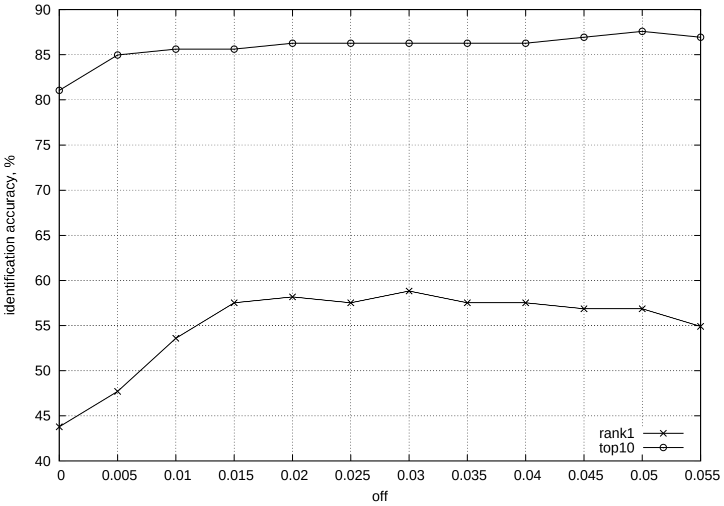


Figure 4.3: Identification rates for various parameters *off*, for the FT-ICR dataset in comparison: Percentage of correct identifications (crosses), percentage of true molecular formulas found in TOP 10 explanations (circles). We set mass and intensity precisions to fixed values: $\alpha_1 = 3$, $\alpha_0 = 6$, $\beta_1 = 10$, $\beta_0 = 90$.

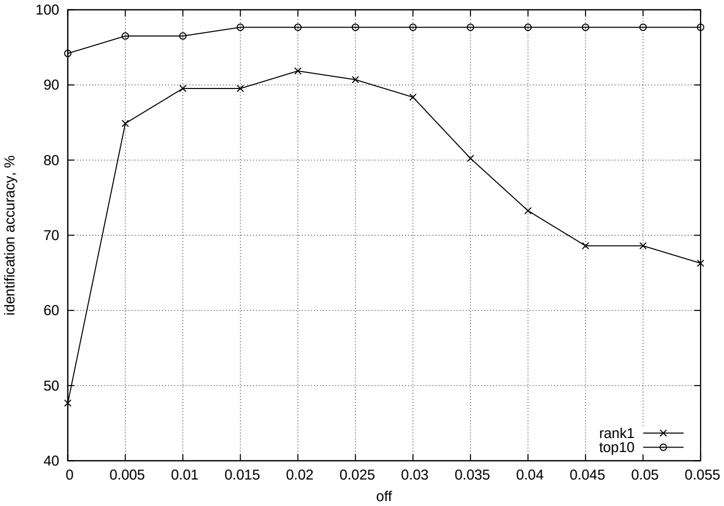


Figure 4.4: Identification rates for various parameters *off*, for the oa-TOF dataset in comparison: Percentage of correct identifications (crosses), percentage of true molecular formulas found in TOP 10 explanations (circles). We set mass and intensity precisions to fixed values: $\alpha_1 = 5$, $\alpha_0 = 6.5$, $\beta_1 = 10$, $\beta_0 = 90$.

5 Deriving Peptide Compositions

In the previous section, we have seen that using only a high quality isotope pattern of a sample molecule, it is possible to identify the molecular formula of unknown metabolite. For proteins, one however encounters the problem that the same molecular formula may correspond to multiple peptides even when abstracting from the order of amino acids in the peptide sequence. In this chapter, we present a novel approach to efficiently derive the amino acid composition of an unknown peptide from its molecular formula.

Novel mass spectrometry techniques allow us to determine the mass of a sample molecule with very high accuracy of 5 ppm, and sometimes below 1 ppm [89, 58]. These techniques are increasingly coupled with high-throughput separation techniques such as (ultra) high performance liquid chromatography (HPLC), and have become one preferred method for the analysis of peptides [55, 112]. This setup becomes of high importance in particular when tackling non-trivial issues in proteomics, such as detecting the post-translational modifications [118], or non-ribosomal peptides that are not directly encoded in the genome [6]. With the advent of new MS instruments (Bruker MicrOTOF, Thermo Electron LTQ Orbitrap, Waters SYNAPT-MS), mass spectra with very high mass accuracy will be routinely acquired for protein identification and quantification in the near future [81].

Given an isotope pattern of an unknown peptide, in theory, one could use an algorithmic procedure similar to the one described above for the identification of metabolites: one can decompose the monoisotopic mass of the peptide to obtain amino acids candidate sequences, and then compare and score candidates using their simulated isotope patterns. However, this approach is infeasible in practice due to the several reasons: First, masses of peptides have to be decomposed over an alphabet of 19 standard amino acids, compared to 6 elements in CHNOPS, which is used for the majority of metabolites. Thus, the number of possible amino acid decompositions is significantly larger than of CHNOPS decompositions, see Section 3.2.1 on page 33: There exist about $3.96 \cdot 10^{11}$ amino acid decompositions with mass up to 2 500 Dalton, see Figure 3.3. Moreover, several peptides can have identical molecular formula even when ignoring the order of amino acids: Besides leucine (L) and isoleucine (I), another example are peptides consisting of two glycine (G) versus a single asparagine (N), both with molecular formula $C_4H_8N_2O_3$ (see Table 2.2 on page 10). Using the above scoring procedure, we cannot distinguish between amino acids compositions with the same molecular formula, and thus, the same isotope pattern. In addition, the sample may be contaminated by metabolite molecules that have a molecular formula which cannot be explained by any peptide. So, instead of determining peptide compositions from the isotope pattern directly, it is much more reasonable first to find out the molecular formula of a sample from its isotope pattern, and then to calculate all amino acid compositions that correspond to this molecular formula. Using the molecular formula of a sample molecule, we can also easily filter out any contaminant, whose molecular formula cannot correspond to any peptide.

Our input is the molecular formula of an unknown peptide. We want find all amino acid compositions that match the given molecular formula. We formulate the problem as a joint decomposition of a set of queries on the number of carbon, hydrogen, and other elements that amino acid residues consist of.

The remainder of this chapter is organized as follows: First, we give a formal definition of a problem and outline its possible solutions. Then, we present our approach based on the dimension reduction technique that transforms an instance of the multi-dimensional decomposition problem to the one-dimensional decomposition problem, which in turn can be efficiently solved using methods presented in Chapter 3. Finally, we provide an experimental evaluation of the algorithm's running time, both on simulated data and peptides from experimental mass spectra. We find that our *mixed matrix* approach is the fastest method for generating solutions, and is about two orders of magnitude faster than the next best algorithm.

5.1 Peptide Molecular Formula Decomposition Problem

Recall that proteins and peptides are built from five elements hydrogen (symbol H), carbon (C), nitrogen (N), oxygen (O), and sulfur (S). For each amino acid its exact molecular formula is known, see Table 2.2.

Formally, the *Peptide Molecular Formula Decomposition Problem* (PMFDP) can be stated as follows:

> Given a molecular formula $b = (b_1, \ldots, b_d)$ and a matrix $A^{d \times n} = (a_{i,j})$ containing the abundances of all elements in amino acid residues, with $b_i \in \mathbb{N}_0$ and $a_{i,j} \in \mathbb{N}^+$, for $i = 1, \ldots, d$ and $j = 1, \ldots, n$, where d is the number of amino acid residues and n is the number of elements that amino acid residues consist of, find all decompositions $x = (x_1, \ldots, x_n)$ with $x_j \in \mathbb{N}_0$, for $j = 1, \ldots, n$, such that $Ax = b$.

If we have $d = n$ many equations, then A is a square matrix, and if its rows are linearly independent, we can compute its inverse A^{-1}. We then only need to check whether $x = A^{-1}b$ has only non-negative integer entries, what can be done in constant time. In the remainder of this chapter, we assume $d \ll n$.

Considering $n = 19$ number of amino acid residues that made up from $d = 5$ elements CHNOS, we thus search for 19-dimensional vector x using the input molecular formula b over the elements CHNOS. Note that only amino acids methionine (M) and cysteine (C) contain sulfur. Thus, we can use a simple technique to further simplify the problem if our input molecular formula contains k sulfur elements: we first distribute these elements between methionine and cysteine, iterating over all k combinations $\mathrm{M}_0\mathrm{C}_k$, $\mathrm{M}_1\mathrm{C}_{k-1}$, …, $\mathrm{M}_k\mathrm{C}_0$. In each case, we reduce the input molecular formula respectively, and compute the decomposions of the resulting formulas over the remaining 17 amino acids A, D, E, F, G, H, K, L, N, P, Q, R, S, T, V, W, Y:

$$\begin{aligned} A &:= \begin{pmatrix} 3 & 4 & 5 & 9 & 2 & 6 & 6 & 6 & 4 & 5 & 5 & 6 & 3 & 4 & 5 & 11 & 9 \\ 5 & 5 & 7 & 9 & 3 & 7 & 12 & 11 & 6 & 7 & 8 & 12 & 5 & 7 & 9 & 10 & 9 \\ 1 & 1 & 1 & 1 & 1 & 3 & 2 & 1 & 2 & 1 & 2 & 4 & 1 & 1 & 1 & 2 & 1 \\ 1 & 3 & 3 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 2 & 2 & 1 & 1 & 2 \end{pmatrix}, \\ b &= \begin{pmatrix} \#\text{ carbon} \\ \#\text{ hydrogen} \\ \#\text{ nitrogen} \\ \#\text{ oxygen} \end{pmatrix}, \quad Ax = b \end{aligned} \tag{5.1}$$

Here, x is a 17-dimensional vector representing the remaining amino acid residues.

Finally, after solutions of (5.1) are obtained, for each of the reduced formulas, we extend each solution vector x by adding the corresponding number of methionine and cysteine, and combine them in our final set of decompositions.

5.1.1 Related Problems and Solutions

Our problem is a special case of a system of linear Diophantine equations. Unlike in PMFDP, the coefficients of matrix A in the linear Diophantine systems can be negative integers. This seemingly negligable relaxation makes a significant impact on the differences in the number of solutions of these two problems: while PMFDP is limited to a finite number of possible explanations, a system of linear Diophantine equations generally has an infinite number of solutions.

To see this, consider an instance of one-dimensional Equality Constrained Integer Knapsack Problem

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b \tag{5.2}$$

where a_j are integer-valued coefficients satisfying $a_j \geq 0$, $b \geq 0$. We search for all solution vectors $x = (x_1, \ldots, x_n)$ such that all x_j are non-negative integers. Note that if there exist indices i, j with $a_i > 0$ and $a_j < 0$, as allowed in a linear Diophantine equation, and if (5.2) has at least one solution, then there is an *infinite* number of solutions.

Our problem can be also seen as d one-dimensional knapsack equations (5.2) that we want to solve simultaneously, and is equivalent to a multi-dimensional Equality Constrained Integer Knapsack Problem. Unlike in a typical formulation of a multi-dimensional Integer Knapsack Problem [70], we are not looking for the maximization of a set of objective functions, but instead for the exact matches of these functions to the set of given queries.

As for a special case of a more general problem, approaches, that exist for solving a system of linear Diophantine equations, can be also employed to solve PMFDP. Most of them have in common to increase the solution vector until an exact hit or a failure is achieved. We call this the *branch-and-bound* algorithm. In the remainder of this section, we describe this and other existing methods to solve PMFDP in some detail, and afterwards present our novel mixed matrix approach to efficiently derive the peptide composition from its molecular formula.

Branch-And-Bound Algorithm One simple method to solve our problem is a branch-and-bound search, which general idea is to add amino acids as long as for each element, the resulting molecule contains at most as many atoms as the input molecule, and to report exact hits.

The algorithm starts by examining all possibilities for the first amino acid alanine and branches, then repeats the same for aspartic acid, and so on until it reaches the last amino acid tyrosine. Then, the algorithm checks whether the number of elements in the current peptide decomposition is equal to the number of elements in the molecular formula, and output exact hits. The algorithm branches using the recursive calls and on each recursion step it maintains a current *decomposition* c, an *index* i of an amino acid, and the current *bounds* on the number of carbon, hydrogen, nitrogen and oxygen in the vector b. In Figure 5.1, we provide the pseudo-code of the algorithm. As an alternative to the branch-and-bound search, we can compute the molecular formulas of all amino acid compositions up to a fixed mass during preprocessing, and employ hashing to efficiently find those decompositions matching the input molecular formula. Unfortunately, due to the large number of possible solutions, in practice this approach suffers from large requirements of memory: Recall that there exist about $3.96 \cdot 10^{11}$

Algorithm BRANCH-AND-BOUND **(decomposition c, index i, bounds b)**

```
if $b_j = 0$ for all $j = 1, \ldots, d$ then
    output $c$;
    return;
end if;
$n_i \leftarrow 0$;
while $b_j >= 0$ for all $j = 1, \ldots, d$ do
    if $i > 1$ then
        $c_i \leftarrow n_i$;
        BRANCH-AND-BOUND$(c, i - 1, b)$;
    else
        $c_1 \leftarrow b_1 / A_{1,1}$;
        if $b_j = A_{j,1} * c_1$ for all $j = 1, \ldots, d$ then
            output $c$;
            return;
        end if;
    end if;
    $b_j \leftarrow b_j - A_{j,i}$ for all $j = 1, \ldots, d$;
end while;
done.
```

Figure 5.1: BRANCH-AND-BOUND algorithm for finding all peptide compositions using the molecular formula. For a molecular formula $b = (b_1, \ldots, b_d)$, where $d = 4$ is the number of bounds, BRANCH-AND-BOUND$(0, k, b)$ will recursively produce all peptide compositions over matrix $A^{d \times k}$ that satisfy b.

amino acid decompositions with mass up to 2500 Da.

Decomposition Algorithms In Chapter 3, we have seen how to efficiently solve an instance of one-dimensional knapsack problem (5.2). Now, as we move to a multi-dimensional problem, a simple algorithm to compute all solutions of $Ax = b$, is to choose one row $i \leq d$ as the master row, then to find all solutions of the one-dimensional knapsack $a_{i,1}x_1 + \cdots + a_{i,n}x_n = b_i$ and, finally, for each solution of the master row to check if it also satisfies the other equations of our multi-dimensional knapsack. We call this the *naïve decomposition* algorithm. Solutions of the master row can be found by backtracing through the Extended Residue (ER) table, as described in Section 3.1.2. However, this involves generating many decompositions unnecessarily.

Instead of discarding the false positives after the completion of the recursive algorithm over the master row, we can introduce several ER tables that correspond to other rows of $Ax = b$, and perform the backtracing over multiple instances simultaneously: The *multiple decomposition* algorithm also picks up a master equation to decompose, but tests during recursion if all other equations of $Ax = b$ besides the master equation can still be satisfied using the current partial solution. If this is no longer possible, then the algorithm stops and discards the current partial solution. Doing so, this approach doesn't generate any false positives, but it requires additional memory to store the data for several decomposition instances.

5.1.2 Multi-dimensional Integer Knapsack

As we have seen in Equation (3.3), the number of mass decompositions over the alphabet of size n increases with a polynomial of degree $n - 1$. It is thus logical to reduce n as much as possible, if we can do so. Actually, the multi-dimensional knapsack allows us to lower n: To this end, we apply a Gaussian elimination to the matrix A to find a lower triangular matrix $L \in \mathbb{R}^{d \times d}$ and an upper triangular matrix $R = (r_{i,j}) \in \mathbb{R}^{d \times n}$ such that $A = LR$. Then, $Ax = b$ if and only if $Rx = L^{-1}b =: b'$, where L^{-1} is known. Every solution of the original equation must henceforth satisfy the *bottom equation* of R,

$$0 \cdot x_1 + \cdots + 0 \cdot x_{d-1} + r_{d,d}x_d + \cdots + r_{d,n}x_n = b'_d \tag{5.3}$$

that has at most $n - d + 1$ non-zero coefficients. Now, our idea is to search for all decompositions of the bottom equation, and for each solution, to test if it is also a solution of $Ax = b$.

Since all coefficients of the matrix A are integers, we can easily assure that after applying Gaussian elimination, the same holds for the coefficients of the output matrix R. On the other hand, we cannot guarantee that $r_{i,j} \geq 0$ holds for all coefficients after Gaussian elimination, even if all entries in the matrix A are non-negative. In particular, there may be negative coefficients in the bottom row of R. But as mentioned above, this means that the bottom equation has an infinite number of solutions, provided that there exists at least one solution. Therefore, we have to avoid that the bottom row of R contains negative coefficients. To find Gaussian eliminations arrangements where this condition holds, we may swap the columns and rows of A: We select a permutation π of the rows of A, and a permutation σ of the columns of A that brings d columns to the front but ignores the remaining $n-d$ columns. We have $d!$ possibilities to select π, and $(n-d+1)\cdots n$ possibilities to select d front rows of A in σ.

We apply the following simple invariant of the Gaussian elimination method in our computations, see the pseudo-code of the algorithm on page 68: Assume that rows and columns of matrix A have been already swapped. For the sake of simplicity, we will compute L^{-1}, instead of L. We initialize $L^{-1} = (l_{i,j}) \leftarrow I$ as the identity matrix and $R \leftarrow A$. We use the row i of matrix R as our master row, and repeat the following procedure for $i = 1, \ldots, d-1$ (lines 3-19): We take the element $r_{i,i}$ in the master row, and the element $r_{i',i}$ below the main diagonal for each row $i' = i+1, \ldots, d$, as our multipliers m_i and $m_{i'}$ (lines 9-10), and iterate through columns $j = i, \ldots, n$, and modify elements in the row i' as following: $r_{i',j} = m_{i'} r_{i,j} - m_i r_{i',j}$. Then, $r_{i',i} = 0$ must hold. Similarly, for columns $j' = 1, \ldots, d$ in the matrix L^{-1}, we modify elements in the row i' such that $l_{i',j'} = m_{i'} l_{i,j'} - m_i l_{i',j'}$. If $r_{i,i} = 0$, this operation reduces the rank of the modified matrix R, which is no longer equivalent to the input matrix. In this case, we discard the partial solution R, L^{-1} and quit the algorithm (lines 4-5). Otherwise, we further check if all entries of the bottom row of R are non-positive: If so, we negate all entries of the bottom row (lines 21-23). At last, we test if $r_{d,j} \geq 0$ holds for all $j = d, \ldots, n$. Otherwise, we discard R, L^{-1}. Applying different permutations on the input matrix A might result in the identical bottom rows in the output matrix R. Thus, at last, we have to filter out matrices with identical bottom rows.

We end up with a set of matrix pairs R, L^{-1} that all can be used to find solutions of the multi-dimensional problem using their bottom rows: Given an input vector b, for one such pair R, L^{-1}, we first apply the row permutation π to b, that was used to generate A. Then, we search all solutions of the equation $Rx = L^{-1}b$ as follows: We compute $b' \leftarrow L^{-1}b$, and apply the decomposition algorithm for the single-dimensional integer knapsack on the bottom equation of $Rx = b'$. For every decomposition $(x_d, \ldots, x_n)$, we iterate over $i = d-1, d-2, \ldots, 1$, and calculate value x_i using row i of $Rx = b'$. We check if $x_i \geq 0$ and if x_i is integer: Otherwise, we discard the partial solution. At last, we apply the inverse column permutation σ^{-1} to the solution x. Doing so for all decompositions of the bottom equation of $Rx = b'$, assures that we find all solutions of $Ax = b$. On the other hand, many decompositions might be generated unnecessarily because these are no solutions of $Ax = b$.

To reduce the number of solutions that are produced "in vain", we apply one further enhancement to the decomposition process: Note that the first row of R contains only positive entries, that can be used as upper bounds for each amino acid during backtracking. On each step of the recursion, we thus check if the current solution of the bottom equation doesn't exceed the upper bound provided by the top equation of $Rx = b'$. In this case, we dynamically update the bound b'_0 and step into the recursion for the next amino acid; otherwise, we discard the partial solution. This enhancement is similar to the one proposed for the bounds on the amount of elements, see Section 3.1.2 on page 29. But here, instead of fixed upper elemental bounds, we have the constraints provided by another query and alphabet. Therefore, we can dynamically update the upper bound depending on the decision we made on each recursion step. This enhancement allowed us to improve the performance of our decomposition algorithm by about 30 % (data not shown).

5.2 Generating Decomposition Matrices and a Mixed Matrix Approach

We have run the Gaussian elimination for all $1 \cdot 2 \cdot 3 \cdot 15 \cdot 16 \cdot 17 = 1370880$ combinations of row and column permutations of A, and generate all possible matrix pairs R, L^{-1}. In 43176 cases, the modification routine successfully produced a bottom row with non-negative entries. After filtering out matrices with duplicate bottom rows, only 19 matrix pairs R, L^{-1} remained, see Section 5.5.

Now, we want to know which of these matrix pairs R, L^{-1} decomposes the fastest. Different bottom equations generate different number of decompositions that are produced unnecessarily and have to be removed. If we use efficient decomposition methods for one-dimensional instances (see Chapter 3), then we can assure that the running time for computing decompositions is, in fact, linear in the number of decompositions. Then, the number of discarded decompositions is a good estimate for the quality of a matrix pair.

In our evaluations we use the notion of *competitive ratio*: the ratio between the number of true solutions and the number of decompositions generated by the bottom equation of a particular matrix pair. Using a training set of molecular formulas to decompose, we can sort out matrices that produce too many additional decompositions, and thus have large competitive ratios. To find the exact number of decompositions that will be generated by the bottom equation of a particular matrix, we can use the dynamic programming techniques mentioned in Chapter 3.

Although well suited for evaluation purposes, in application, one would like to avoid the explicit calculation of the number of decompositions, because this can be very time consuming. But how can we estimate the number of discarded decompositions without actually computing them? To answer this question, we recall that the number of decompositions of some query value over n coprime integers asymptotically behaves like a polynomial of degree $n-1$, see Equation (3.3). Since the value we actually want to decompose using the bottom equation (5.3) is $b'_d = \sum_{k=1}^{d} l_{d,k} b_k$, we define

$$\tilde{l}(b) := \frac{1}{(m-1)!\, r_{n-m+1,d} \cdots r_{n,d}} \left(\sum_{k=1}^{d} l_{d,k} b_k \right)^{m-1} \tag{5.4}$$

as our *indicator*, where m is the number of non-zero elements in the bottom row of the matrix R. As we will see in the next section, there is a clear correlation between this indicator, and the running time of our algorithm.

Now our *mixed matrix* approach proceeds as follows: Given a vector b to decompose, we calculate the indicators $\tilde{l}(b)$ for every matrix pair R, L^{-1}, and select the matrix pair with the smallest indicator. Then, we use the bottom equation of the corresponding matrix R to actually decompose the value $\sum_{k=1}^{d} l_{d,k} b_k$.

5.3 Experimental Results

For evaluations of our method we proceed as follows: We first compute the competitive ratios for all decomposition matrices, and filter out those that produce too many additional solutions. The number of decompositions to be discarded is not a single factor that has an impact on the running times. Additionally, the time required to discard false solutions may vary for different matrices. To better assess the actual performance of a specific matrix pair, we will apply a slight correction to our indicator. We also perform a comparison of the running times of the mixed matrix approach and several other algorithms on simulated and experimental data.

Datasets. We use two datasets in our evaluations: The first dataset consists of 6 000 peptides that have been simulated by in-silico digestion (trypsin) using the Swiss-Prot database (release 56.5); duplicate entries have been removed. We take peptides with masses between 900 and 1 500 Da, and for each mass range of 100 Da, we randomly select 1 000 peptides. The second dataset consists of 99 peptides from *de novo* interpretations of an experimental tandem mass spectra dataset, acquired on quadrupole ion-trap mass spectrometer. The peptides in the second dataset range in mass from about 900 to 2 000 Da.

5.3.1 Selecting Good Decomposition Matrices

To filter out matrix pairs that produce too many additional decompositions, for each pair we compute competitive ratios for all peptides in the simulated dataset. We compute the average competitive ratio over all peptides for ranges of size 100 Da. Competitive ratios of the six best matrix pairs are shown in Figure 5.2. For the remaining 13 pairs, the average competitive ratio is never below 3900 for any pair and any mass range of size 100 Da. See Figure 5.3 for the competitive ratios of seven more matrix pairs.

Finding Final Decomposition Matrix. For further evaluation, we chose the six matrix pairs with the best competitive ratios. These matrix pairs are listed in Section 5.5. For each matrix pair, we compute the indicator $\tilde{l}(b)$ for the input vector b, and compare it with the actual running time of the algorithm. In Figure 5.4, we have plotted these values for all peptides in the mass range 1000–1100 Da from our simulated dataset.

We can see an almost linear correlation between logarithms of indicators and running times. We also observe a slight shift of the intercept of the linear fit over the y-axis for various matrices. This corresponds to the differences in running times required for filtering out additional decompositions. Using a linear fit with the least squares criterion, we derive an affine correction of the indicator from this data, see Figure 5.4. Peptides from other mass ranges reveal a similar correlation (data not shown).

Now, we apply the linear correction to the indicator $\tilde{l}(b)$, and select the matrix with the minimal value. This matrix is then used to actually decompose the input vector b. Clearly, this is not necessarily the matrix that decomposes the fastest for a particular input. We want to evaluate how often we hit the matrix pair with the minimal running

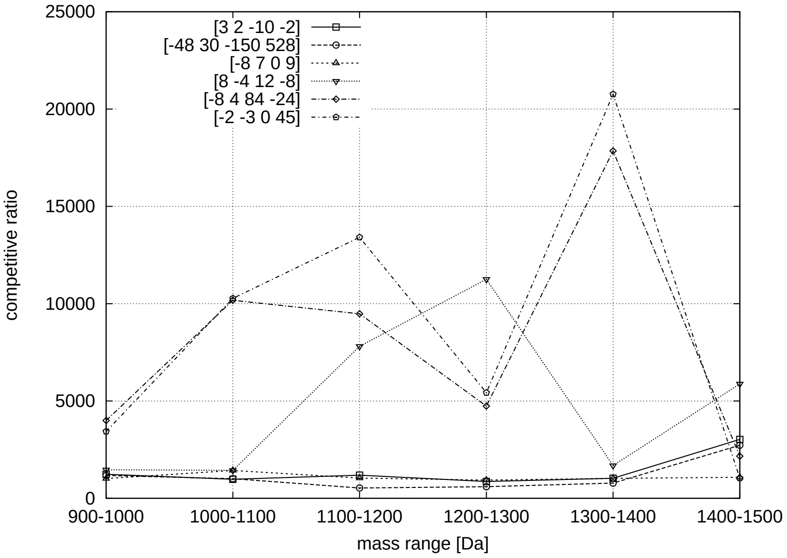


Figure 5.2: Competitive ratios of six matrix pairs R, L^{-1} with the lowest competitive ratios on average for mass ranges of size 100 Da. Matrix pairs are labeled by the last row of the matrix L^{-1}.

times from the six pairs, see Figure 5.5: For the simulated data, we select the fastest matrix pair in more than 91.4 % of the cases, resulting in a total loss of performance of about 1.35 %. Rates for the experimental dataset are similar: in more than 87.5 % of the cases an optimal matrix is chosen, leading to an overall loss of performance of about 1.71 %.

5.3.2 Comparison with Other Methods

Finally, we want to evaluate how good the mixed matrix algorithm performs in comparison with branch-and-bound searching, the naïve decomposition algorithm, and the multiple decomposition algorithm, described in Section 5.1.1. Both the naïve and the multiple decomposition algorithms generate solutions using Extended Residue Table. Note that there exist four possibilities to select one of the rows of the matrix A from (5.1) as the master row for these two methods. We only present the best results of the four variants.

A comparison of the running times of these approaches is shown in Figures 5.6 and 5.7 for the simulated and real data respectively. The performance of the naïve decomposition algorithm was significantly worser than those of all other methods and, therefore, omitted. For both datasets, our mixed matrix approach was greatly superior to the runner-up method, branch-and-bound algorithm: For the simulated dataset, we observe 92-fold speedup over the branch-and-bound searching, whereas the real data is processed 125 times faster on average. The improvement of the running times over the multiple decomposition algorithm is yet better: We observe a 190 and a 218-fold speedup for the

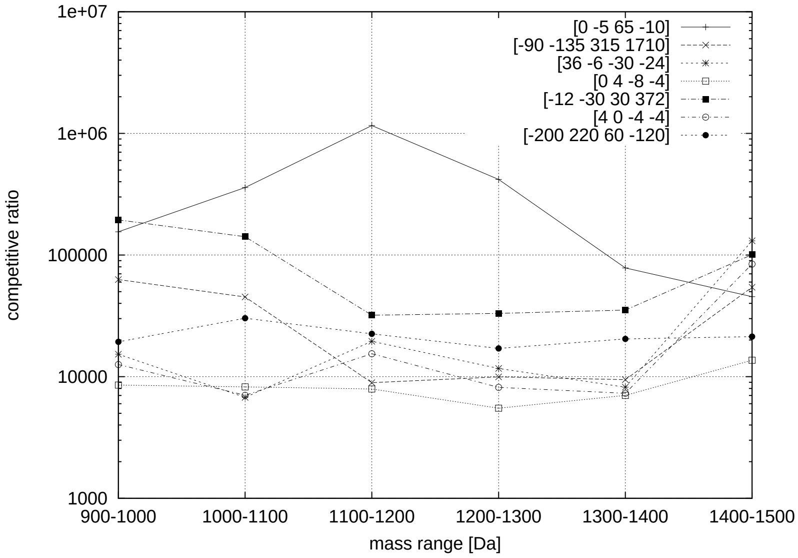


Figure 5.3: Competitive ratios of seven matrix pairs with rather high competitive ratios compared to the matrix pairs depicted in Figure 5.2. Values are calculated on average for mass bins of width 100 Da. Competitive ratios for the remaining six matrices have comparable or higher values than the matrices shown here.

simulated and real data respectively. The performance of the mixed matrix approach is about 33 microseconds per decomposition on average for both datasets. We also observe that the mixed matrix approach significantly outperforms each particular matrix pair alone, see Figure 5.6.

We have also tested the performance of the matrix pairs with five rows, that contain a decomposition query for sulfur. Performance for these matrix pairs was in all cases significantly worser than for the matrix pairs with four rows.

All algorithms were implemented in C++, and the running times were measured on an AMD Opteron-275 2.2 GHz with 6 GB of memory running Solaris 10.

5.4 Summary

We have presented an efficient algorithm to generate all solutions of a multi-dimensional equality constrained integer knapsack problem. We have demonstrated the applicability of our method on the problem of finding all compositions of an unknown peptide using its molecular formula. The results on both simulated and experimental data reveal the outstanding performance of our *mixed matrix* approach, which is about two orders of magnitude faster than the runner-up algorithm. In absolute terms, the average running time of our algorithm is about 33 microseconds per decomposition.

We can further enhance our method by including more matrix pairs to the selection of the matrix pair that is finally used for generating decompositions. Moreover, we can

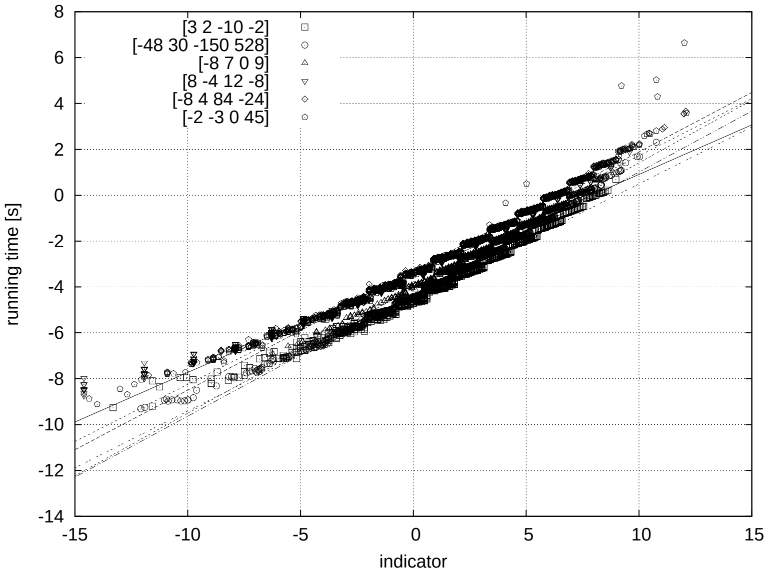


Figure 5.4: Indicators of running times calculated for peptides in the mass range 1000–1100 Da. Values are plotted in a log-log scale. Lines depict the results of the least squares fits.

speed up the decomposition procedure by eliminating identical entries in the bottom row of R, and distributing the resulting value between the amino acids that correspond to the duplicates.

Our mixed matrix approach can be used for any application where all solutions of a multi-dimensional equality constrained integer knapsack [1] are sought. This is needed whenever searching an optimal solution of the knapsack cannot be modeled via a simple linear or quadratic objective function. For example, the method can be used to speed up the search for a molecular formula of an unknown sample molecule, as proposed in [15]. Moreover, our approach is applicable for the decision version of the multi-dimensional knapsack problem. For example, it can be used to find out whether a given molecular formula is originated from the protein molecule. This question is relevant in applications to quickly filter out any contaminants in the protein sample.

5.5 Best Six Matrix Pairs

In the first row of each matrix, we give the permutation of amino acids (for R matrices) or elements (for L^{-1} matrices) that have to be applied before or after using the decomposition matrices. Last rows of the matrices R are divided on the corresponding least common multipliers, see Table 5.1 on page 69.

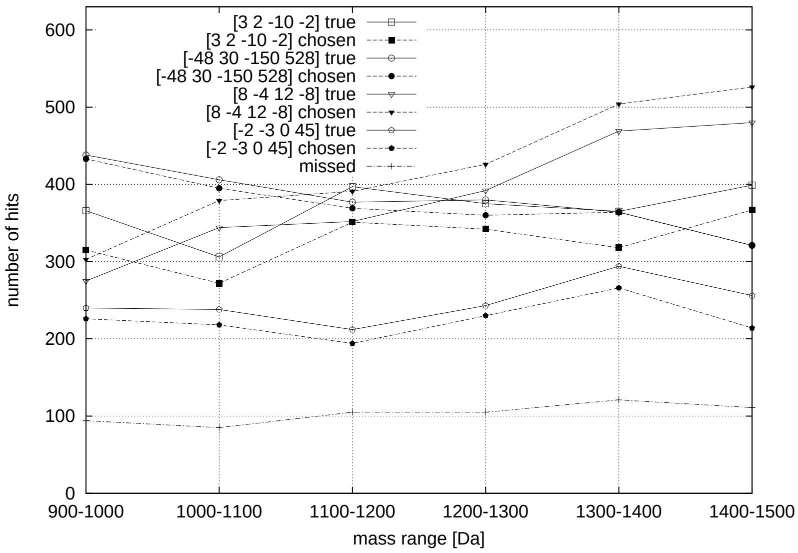


Figure 5.5: Number of hits of a particular matrix chosen by our estimate (chosen), number of times when a particular matrix is actually the fastest (true) and the total number of wrong hits (missed). Number of hits shown for the four, most frequently chosen matrices. Numbers are counted over all peptides in a specific mass range in the simulated dataset.

$$R_1 = \left(\begin{array}{ccccccccccccccccc} \mathrm{G} & \mathrm{H} & \mathrm{R} & \mathrm{F} & \mathrm{A} & \mathrm{D} & \mathrm{K} & \mathrm{L} & \mathrm{N} & \mathrm{P} & \mathrm{Q} & \mathrm{E} & \mathrm{S} & \mathrm{T} & \mathrm{V} & \mathrm{W} & \mathrm{Y} \\ 2 & 6 & 6 & 9 & 3 & 4 & 6 & 6 & 4 & 5 & 5 & 5 & 3 & 4 & 5 & 11 & 9 \\ 0 & 4 & -6 & 9 & -1 & 2 & -4 & -6 & 0 & 1 & -1 & 1 & -1 & -2 & -3 & 13 & 9 \\ 0 & 0 & -2 & 7 & 1 & 2 & 4 & 2 & 0 & 3 & 1 & 3 & 1 & 2 & 3 & 7 & 7 \\ 0 & 0 & 0 & 33 & 7 & 6 & 28 & 20 & 0 & 17 & 7 & 13 & 5 & 12 & 21 & 31 & 31 \end{array}\right)$$

$$L_1^{-1} = \left(\begin{array}{cccc} \mathrm{C} & \mathrm{H} & \mathrm{N} & \mathrm{O} \\ 1 & 0 & 0 & 0 \\ 3 & -2 & 0 & 0 \\ 1 & 0 & -2 & 0 \\ 3 & 2 & -10 & -2 \end{array}\right)$$

$$R_2 = \left(\begin{array}{ccccccccccccccccc} \mathrm{H} & \mathrm{R} & \mathrm{W} & \mathrm{F} & \mathrm{G} & \mathrm{A} & \mathrm{K} & \mathrm{L} & \mathrm{N} & \mathrm{P} & \mathrm{Q} & \mathrm{D} & \mathrm{S} & \mathrm{T} & \mathrm{V} & \mathrm{E} & \mathrm{Y} \\ 6 & 6 & 11 & 9 & 2 & 3 & 6 & 6 & 4 & 5 & 5 & 4 & 3 & 4 & 5 & 5 & 9 \\ 0 & -30 & 17 & 9 & -4 & -9 & -24 & -30 & -8 & -7 & -13 & -2 & -9 & -14 & -19 & -7 & 9 \\ 0 & 0 & -88 & -96 & -4 & -24 & -84 & -60 & -8 & -52 & -28 & -32 & -24 & -44 & -64 & -52 & -96 \\ 0 & 0 & 0 & 18 & 31 & 32 & 35 & 25 & 62 & 29 & 63 & 116 & 76 & 77 & 34 & 117 & 62 \end{array}\right)$$

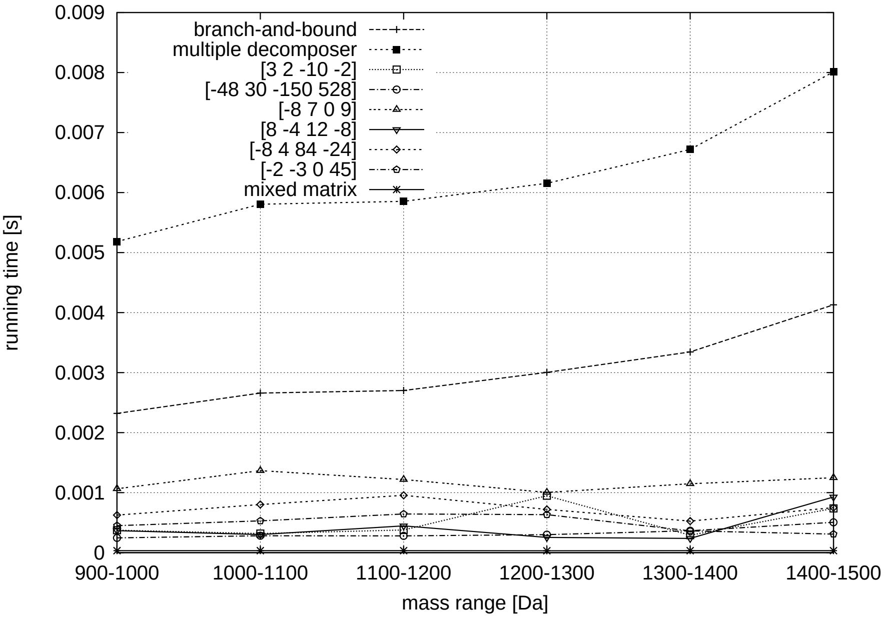


Figure 5.6: Running times of the algorithms (in seconds) for simulated data. Running times are calculated per decomposition and averaged over all peptides in the mass range. We also report the performance of our method with only one particular matrix pair applied.

$$L_2^{-1} = \begin{pmatrix} \text{C} & \text{H} & \text{N} & \text{O} \\ 1 & 0 & 0 & 0 \\ 7 & -6 & 0 & 0 \\ -8 & -6 & 30 & 0 \\ -48 & 30 & -150 & 528 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} \text{F} & \text{W} & \text{Y} & \text{A} & \text{G} & \text{H} & \text{K} & \text{L} & \text{N} & \text{P} & \text{Q} & \text{R} & \text{S} & \text{T} & \text{V} & \text{D} & \text{E} \\ 9 & 11 & 9 & 3 & 2 & 6 & 6 & 6 & 4 & 5 & 5 & 6 & 3 & 4 & 5 & 4 & 5 \\ 0 & 1 & 0 & -2 & -1 & -1 & -5 & -6 & -2 & -2 & -3 & -6 & -2 & -3 & -4 & -1 & -2 \\ 0 & 0 & 9 & 2 & 5 & 1 & -7 & -9 & 10 & 0 & 7 & -9 & 11 & 8 & -4 & 21 & 18 \\ 0 & 0 & 0 & 10 & 7 & 14 & 19 & 27 & 14 & 9 & 17 & 36 & 10 & 13 & 16 & 6 & 9 \end{pmatrix}$$

$$L_3^{-1} = \begin{pmatrix} \text{C} & \text{H} & \text{O} & \text{N} \\ 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & -2 & 9 & 0 \\ -8 & 7 & 0 & 9 \end{pmatrix}$$

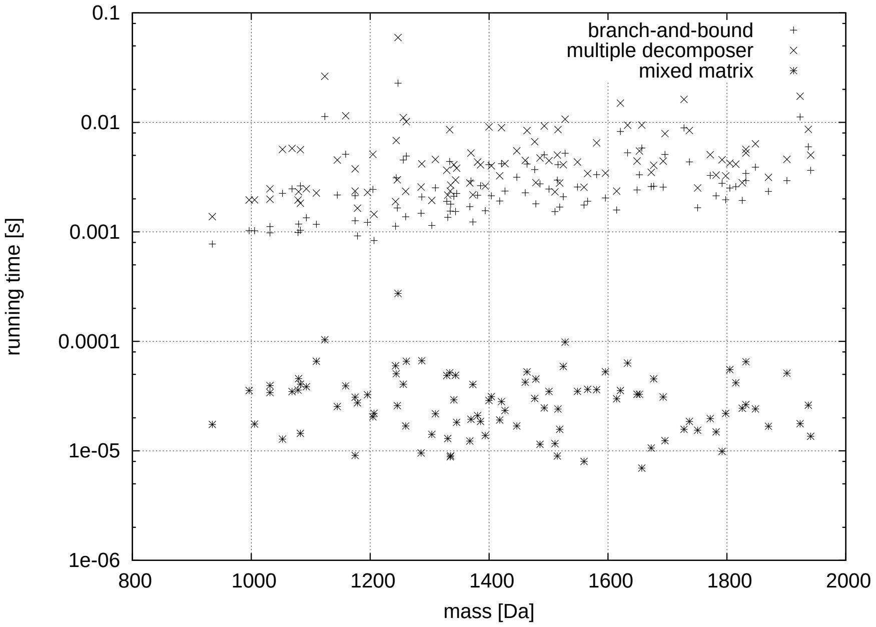


Figure 5.7: Running times of the algorithms (in seconds) for real data.

$$R_4 = \left(\begin{array}{ccccccccccccccccc}
\mathrm{D} & \mathrm{E} & \mathrm{S} & \mathrm{F} & \mathrm{G} & \mathrm{H} & \mathrm{K} & \mathrm{L} & \mathrm{N} & \mathrm{P} & \mathrm{Q} & \mathrm{R} & \mathrm{A} & \mathrm{T} & \mathrm{V} & \mathrm{W} & \mathrm{Y} \\
4 & 5 & 3 & 9 & 2 & 6 & 6 & 6 & 4 & 5 & 5 & 6 & 3 & 4 & 5 & 11 & 9 \\
0 & -3 & -5 & 9 & -2 & 2 & -14 & -18 & -4 & -3 & -7 & -18 & -5 & -8 & -11 & 15 & 9 \\
0 & 0 & -8 & 24 & -8 & -16 & -8 & -24 & -16 & 0 & -16 & -48 & -8 & -8 & -8 & 24 & 24 \\
0 & 0 & 0 & 5 & 1 & 6 & 1 & 2 & 2 & 2 & 2 & 5 & 1 & 0 & 1 & 8 & 4
\end{array}\right)$$

$$L_4^{-1} = \left(\begin{array}{cccc}
\mathrm{C} & \mathrm{H} & \mathrm{N} & \mathrm{O} \\
1 & 0 & 0 & 0 \\
5 & -4 & 0 & 0 \\
8 & -4 & -12 & 0 \\
8 & -4 & 12 & -8
\end{array}\right)$$

$$R_5 = \left(\begin{array}{ccccccccccccccccc}
\mathrm{D} & \mathrm{E} & \mathrm{Y} & \mathrm{F} & \mathrm{G} & \mathrm{H} & \mathrm{K} & \mathrm{L} & \mathrm{N} & \mathrm{P} & \mathrm{Q} & \mathrm{R} & \mathrm{S} & \mathrm{T} & \mathrm{V} & \mathrm{W} & \mathrm{A} \\
4 & 5 & 9 & 9 & 2 & 6 & 6 & 6 & 4 & 5 & 5 & 6 & 3 & 4 & 5 & 11 & 3 \\
0 & -3 & 9 & 9 & -2 & 2 & -14 & -18 & -4 & -3 & -7 & -18 & -5 & -8 & -11 & 15 & -5 \\
0 & 0 & 24 & 24 & -8 & -16 & -8 & -24 & -16 & 0 & -16 & -48 & -8 & -8 & -8 & 24 & -8 \\
0 & 0 & 0 & 3 & 7 & 26 & 7 & 18 & 14 & 6 & 14 & 39 & 4 & 4 & 7 & 12 & 7
\end{array}\right)$$

$$L_5^{-1} = \left(\begin{array}{cccc}
\mathrm{C} & \mathrm{H} & \mathrm{N} & \mathrm{O} \\
1 & 0 & 0 & 0 \\
5 & -4 & 0 & 0 \\
8 & -4 & -12 & 0 \\
-8 & 4 & 84 & -24
\end{array}\right)$$

$$R_6 = \left(\begin{array}{ccccccccccccccccc} \mathrm{F} & \mathrm{K} & \mathrm{Y} & \mathrm{A} & \mathrm{G} & \mathrm{H} & \mathrm{D} & \mathrm{L} & \mathrm{N} & \mathrm{P} & \mathrm{Q} & \mathrm{R} & \mathrm{S} & \mathrm{T} & \mathrm{V} & \mathrm{W} & \mathrm{E} \\ 9 & 6 & 9 & 3 & 2 & 6 & 4 & 6 & 4 & 5 & 5 & 6 & 3 & 4 & 5 & 11 & 5 \\ 0 & -5 & 0 & -2 & -1 & -1 & -1 & -6 & -2 & -2 & -3 & -6 & -2 & -3 & -4 & 1 & -2 \\ 0 & 0 & 45 & 24 & 32 & 12 & 112 & -3 & 64 & 14 & 56 & -3 & 69 & 61 & 8 & -7 & 104 \\ 0 & 0 & 0 & 12 & 16 & 51 & 11 & 21 & 32 & 7 & 28 & 66 & 12 & 8 & 4 & 19 & 7 \end{array}\right)$$

$$L_6^{-1} = \left(\begin{array}{cccc} \mathrm{C} & \mathrm{H} & \mathrm{O} & \mathrm{N} \\ 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ -2 & -3 & 45 & 0 \\ -2 & -3 & 0 & 45 \end{array}\right)$$

Gaussian Elimination Algorithm (matrix $A^{d,n}$)

```
$L^{-1} = (l_{i,j}) \leftarrow I$;
$R = (r_{i,j}) \leftarrow A$;
for $i = 1, \ldots, d-1$ do
    if $r_{i,i} = 0$
        return;
    end if;
    for $i' = i+1, \ldots, d$ do
        if $r_{i',i}! = 0$
            $m_i \leftarrow r_{i,i}$;
            $m_{i'} \leftarrow r_{i',i}$;
            for $j = i, \ldots, n$ do
                $r_{i',j} \leftarrow m_{i'} r_{i,j} - m_i r_{i',j}$;
            end for;
            for $j' = 1, \ldots, d$ do
                $l_{i',j'} \leftarrow m_{i'} l_{i,j'} - m_i l_{i',j'}$;
            end for;
        end if;
    end for;
end for;
if $r_{d,j} \leq 0$ for all $j = d, \ldots, n$ then
    for $j = d, \ldots, n$ do
        $r_{d,j} \leftarrow -r_{d,j}$;
    end for;
end if;
if $r_{d,j} \geq 0$ for all $j = d, \ldots, n$ then
    output $R, L^{-1}$;
    return;
end if;
done.
```

Figure 5.8: Gaussian elimination algorithm for finding a lower triangular matrix $L \in \mathbb{R}^{d \times d}$ and an upper triangular matrix $R \in \mathbb{R}^{d \times n}$ such that $A = LR$, where $A \in \mathbb{R}^{d \times n}$ is the input matrix. The algorithm reports a matrix pair R, L^{-1} such that all entries of the bottom row of R are non-negative.

Last row of the matrix R (modified)														lcm	Last row of L^{-1}			
	1	1	1	1	2	2	2	2	4	5	5	6	8	8	8	-4	12	-8
3	4	4	6	7	7	7	7	12	14	14	18	26	39	8	-8	4	84	-24
	1	2	2	2	3	3	4	5	5	6	6	7	8	4	0	4	-8	-4
	1	1	1	1	1	2	3	3	3	4	6	7	8	4	4	0	-4	-4
5	10	10	16	16	17	22	27	29	33	56	58	73	77	8	-80	108	-76	-48
3	5	6	8	10	10	11	12	14	16	23	29	36	39	40	-200	220	60	-120
1	1	1	3	4	4	5	7	7	8	9	18	18	19	10	20	-15	65	-20
1	1	1	1	2	2	3	4	6	7	7	8	15	19	10	0	-5	65	-10
2	4	7	9	12	13	16	19	27	28	31	32	45	49	90	-90	-135	315	1710
	5	6	7	7	12	13	17	20	21	28	31	31	33	1	3	2	-10	-2
	1	2	2	2	3	5	5	6	7	8	16	18	21	12	36	-6	-30	-24
18	25	29	31	32	34	35	62	62	63	76	77	116	117	12	-48	30	-150	528
1	1	3	4	5	6	7	8	8	9	24	27	29	43	3	21	-12	12	-6
2	5	6	11	15	18	24	33	42	43	48	49	73	79	12	-12	-30	30	372
1	5	6	8	9	10	11	13	14	26	45	78	84	113	1	19	-9	0	-6
1	1	1	1	1	2	2	2	2	4	8	12	13	18	6	18	-9	3	-6
5	9	17	25	27	41	53	81	94	101	106	113	171	183	1	-2	-5	0	72
4	7	7	8	11	12	12	16	19	21	28	32	51	66	2	-2	-3	0	45
6	7	9	9	10	10	13	14	14	16	17	19	27	36	2	-8	7	0	9

Table 5.1: 19 decomposition matrix pairs R, L^{-1} with distinct bottom rows. The last row of the matrix R is shown modified: Values have been divided on the least common multiplier (lcm) and sorted in the increasing order; zero entries have been removed.

6 Application Tools and Cases

In the previous chapters, we have been extensively discussing our algorithms to decompose integer and real-valued mass spectrometry (MS) data. In this chapter, we present several application tools that implement the described algorithms for *de novo* identification of sample molecules. We start by describing DECOMP, a web-based tool for solving integer and real-valued mass decomposition problems over any arbitrary alphabet of biomolecules. Typical examples include amino acids, nucleotides, and the elements CHNOPS most frequently occurring in nature. Then, we demonstrate its applicability as a part of another algorithm, called CompNovo, for *de novo* sequencing of peptides using tandem mass spectrometry. Finally, we present the Rdisop package that integrates more data such as accurate isotopic abundances, for identification of molecular formulas of metabolites.

6.1 DECOMP

6.1.1 Introduction

Assume you have a DNA fragment of mass 1896.33 ± 0.05 Dalton and no other information. What are the possible nucleotide combinations that fall into this mass range? Here, there exist two such DNA molecular formulas: either 6 adenines (amounting to 1878.35 Da, plus 18.01 Da for a water molecule) or 2 cytosines, 1 adenine, and 3 guanines (1878.31 Da plus 18.01 Da). DECOMP helps to efficiently solve this and other related problems.

Recall that the *Real Mass Decomposition Problem* (RMDP) can be stated as follows: Given an alphabet of positive, real-valued numbers $\Sigma = \{a_1, \dots, a_n\}$, a mass M, and a mass error ϵ, find all non-negative integer vectors $c = (c_1, \dots, c_n)$ such that $a_1c_1 + a_2c_2 + \dots + a_nc_n \in [M - \epsilon, M + \epsilon]$. If the alphabet elements are positive integers, and no mass error is allowed, the problem is referred to as the *Integer Mass Decomposition Problem* (IMDP), which decision version is also known as *Money Changing Problem* (MCP), see Chapter 3 for details.

In mass spectrometry applications, three types of alphabets are most prevalent: 19 standard amino acids that make up protein sequences (isoleucine and leucine are indistinguishable), 4 nucleotides that constitute to DNA molecules, and 6 chemical elements (CHNOPS) that most frequently occur in nature, and are the major building blocks of metabolites.

Existing Approaches. From the algorithmic and combinatorial point of view, a significant amount of biochemical and mass spectrometry literature exists on the problem of finding the molecular formula of a sample molecule using its mass, see i.e., [47, 99, 11]. There are many other methods such as [102, 64, 73, 52, 115] that exploit the knowledge about the molecular formulas for the interpretation of mass spectra. In addition, a multitude of software packages exist to compute the molecular formulas, see i.e., Seth[1], ElComp[2], HiRes MS[3], Elemental Composition Calculator[4] and MF finder[5], to name a few. To the best of our knowledge, all these packages employ simple exhaustive strategies to search for all decompositions of a given mass. Since exhaustive algorithm tests all possible solutions up to the input mass, its performance drops dramatically with the increasing input mass, if, at all, it becomes possible to complete the task successfully. Recall that there exist $5.1 \cdot 10^8$ molecular formulas with mass up to 1500 Da over 19 standard amino acids.

Additionally, some of these packages are available for one operating system only (Seth and Elemental Composition Calculator for Windows), whereas others are confined to a certain type of alphabet (HiRes MS and MF finder to the masses of the common chemical elements).

We have developed Decomp, a new application tool to find decompositions of a given mass over any arbitrary alphabet. It implements the efficient algorithms for solving integer and real-valued mass decomposition problems, and is equipped with an user-friendly web-based interface to handle both types of problems conveniently. Thus, Decomp is easily applicable for both the interpretation of MS data, i. e., for calculating all molecular formulas with a given molecular mass from various types of samples, and for finding solutions of MCP problems.

Since the algorithmic part has been discussed in detail in Chapter 3, here we focus on the implementation and the front-end features of our software.

6.1.2 Implementation and Use

Decomp's algorithms are implemented in C++ and run as a stand-alone program on the Bielefeld University Bioinformatics Server[6] (BiBiServ). To submit a request to Decomp interactively, a simple web-based interface has been written using a plain HTML, supplied with a few JavaScript functions for the validation of the input parameters. After submission, the program is processed on the dedicated server, and after the computation is complete, results can be retrieved as a text file.

Web Services. Data retrieval from the biological online resources is typically realized by HTML links to the web pages or by regular, manual downloads and a subsequent integration of the obtained data. This is both error-prone and time-consuming. The

---

[1] `http://www.zebra-crossing.de/software/`

[2] `http://medlib.med.utah.edu/masspec/`

[3] `http://hires.sourceforge.net/`

[4] `http://www.wsearch.com.au/`

[5] `http://www.chemcalc.org/`

[6] `http://bibiserv.techfak.uni-bielefeld.de/decomp/`

introduction of Web Services offers an opportunity to overcome this difficulty and to perform the data exchange in much more reliable and faster way.

Web Services represent software interfaces that interact via network using XML-based messages. The structure of these messages, which contain query requests or the corresponding results, can be described using a *simple object access protocol* (SOAP). Any software that is implemented in a programming language, which supports a SOAP interface can retrieve data directly from that service. Doing so, the end-user does not even realize that the results come from another service.

Web Services have been widely used for the data integration and exchange between heterogeneous systems [87,8], and have been recently implemented by well-known databases at European Bioinformatics Institute (EBI) [98], and KEGG database [66].

Decomp offers a Web Service client, which can be used from the command line for batch processing and other non-interactive uses. The interface is written in Java, and is available upon request.

Parameter Input. Decomp is designed to tackle the decomposition problems for both integer and real-valued queries. Therefore, in an initial step the appropriate submission form has to be selected. Since we want to keep the user interface as simple as possible, specifying only a few parameters is necessary for submission; others are either optional, or provided with reasonable default values.

One required parameter is the query mass or masses that can be entered manually, or the file for upload can be specified. Another prerequisite for the submission is to define the alphabet to be used for the decomposition. For the analysis of MS data, several predefined alphabets of the common types of sample molecules such as amino acids, nucleotides, and CHNOPS are supplied, and the user can select between the monoisotopic and average isotopic mass distribution. Alternatively, a custom alphabet can be entered manually or uploaded from the file. Specifying these two parameters is already enough to successfully submit the input form.

Additionally for the real-valued case, the allowed mass error can be set as an absolute (Da) or relative (ppm) deviation from the query mass. Another optional parameter is the computational precision that corresponds to the blowup factor, described in Section 3.2. It defines the way how the algorithm rounds the real values to integers. Decomp provides an option to calculate the computational precision automatically depending on the alphabet in use, which in most cases allows user to remain unconcerned about this setting.

For the query upload, Decomp supports the majority of the most common MS file formats such as mzData, mzXML, and others. To be recognized as a supported format, the file to upload should have an appropriate extension, i. e., mgf for the Mascot Generic Format. Other supported file formats can be found in Decomp's manual[7].

The user can specify the constraints on the minimal and maximal amount of elements in the output molecule. Often in *de novo* interpretations of tandem MS data, a partial tag or other sample preparation information can suggest such constraints. Specifying the

---

[7] `http://bibiserv.techfak.uni-bielefeld.de/decomp/manual.html`

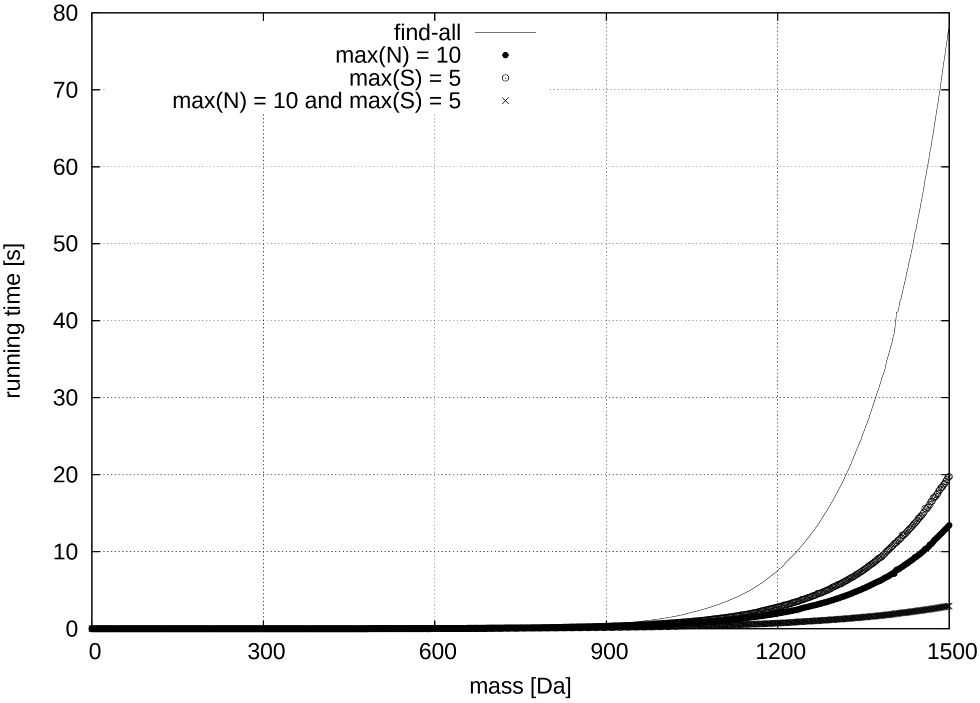


Figure 6.1: Running times of the algorithm FIND-ALL (line) and FIND-ALL-WITH-BOUNDS (points) for various elemental bounds in comparison. Decompositions with mass M to $M + \epsilon$, for $\epsilon = 0.0001$ Da, computed over the alphabet CHNOPS.

upper and lower bounds for each element allows user to confine the search space to only valid decompositions, e.g., those containing at most 4 sulfur (S), or at least 1 carbon (C). Moreover, providing the bounds on the number of elements can greatly speed up the decomposition process: We have compared the performance of FIND-ALL and FIND-ALL-WITH-BOUNDS algorithms, described in Section 3.1.2, for various elemental constraints, and observed the significant improvement in the running times of the latter algorithm depending on the constraints provided, see Figure 6.1. To specify the constraints, the same notation as for decompositions is used: For example, the minimum constraints CH_2 mean that the output molecule must contain at least 1 carbon (C) and 2 hydrogen (H) atoms.

Regarding the modifications of the query mass, DECOMP separates these in two groups: the modifications of the entire molecule and the post-translational modifications (PTM). The former apply for any type of sample molecule. The interface provides the list of predefined modifications those that commonly occur due to the ionization process or sample preparation. Typical examples are protonation and addition of a water molecule. The latter modify only protein molecules, and can be further divided into fixed and variable modifications: *Fixed* modifications apply to all copies of the corresponding amino acid. For example, selecting modification `Phosphorylation (S, +80 Da)` means that all serines (`S`) have been phosphorylated. In contrast, *variable* modifications apply to some, but not necessarily all copies of a relevant amino acid. For instance, choosing modification `Acetylation (M, +42 Da)` means that any methionine (`M`) in the molecule can be acetylated. The modified amino acid is then specifically marked in the output. Note that introducing variable modifications significantly increases the search space, as the alphabet size increases, therefore they should be used sparsely.

DECOMP's results are by default ranked with regard to the deviation from the query mass. Several output and filtering parameters allows user to further customize the output: for each decomposition, the appearance of the actual mass and deviation in the output can be specified. Furthermore, the number of solutions to show can be restricted to a certain threshold. Finally, for the atom alphabet, an additional filter for molecular formulas can be activated that discards irrelevant solutions based on the Senior conditions, described in Section 4.1. Doing so guarantees that only chemically valid compounds will be shown in the output. In Figure 6.2, the submission form for decomposing the real-valued masses is depicted.

The submission view for the integer queries in general resembles its real-valued counterpart, but keeps only parameters relevant for solving integer mass decomposition problems. In particular, the user can select between (1) finding all solutions (default), (2) finding one solution, (3) getting the number of solutions, or (4) deciding whether a query is decomposable.

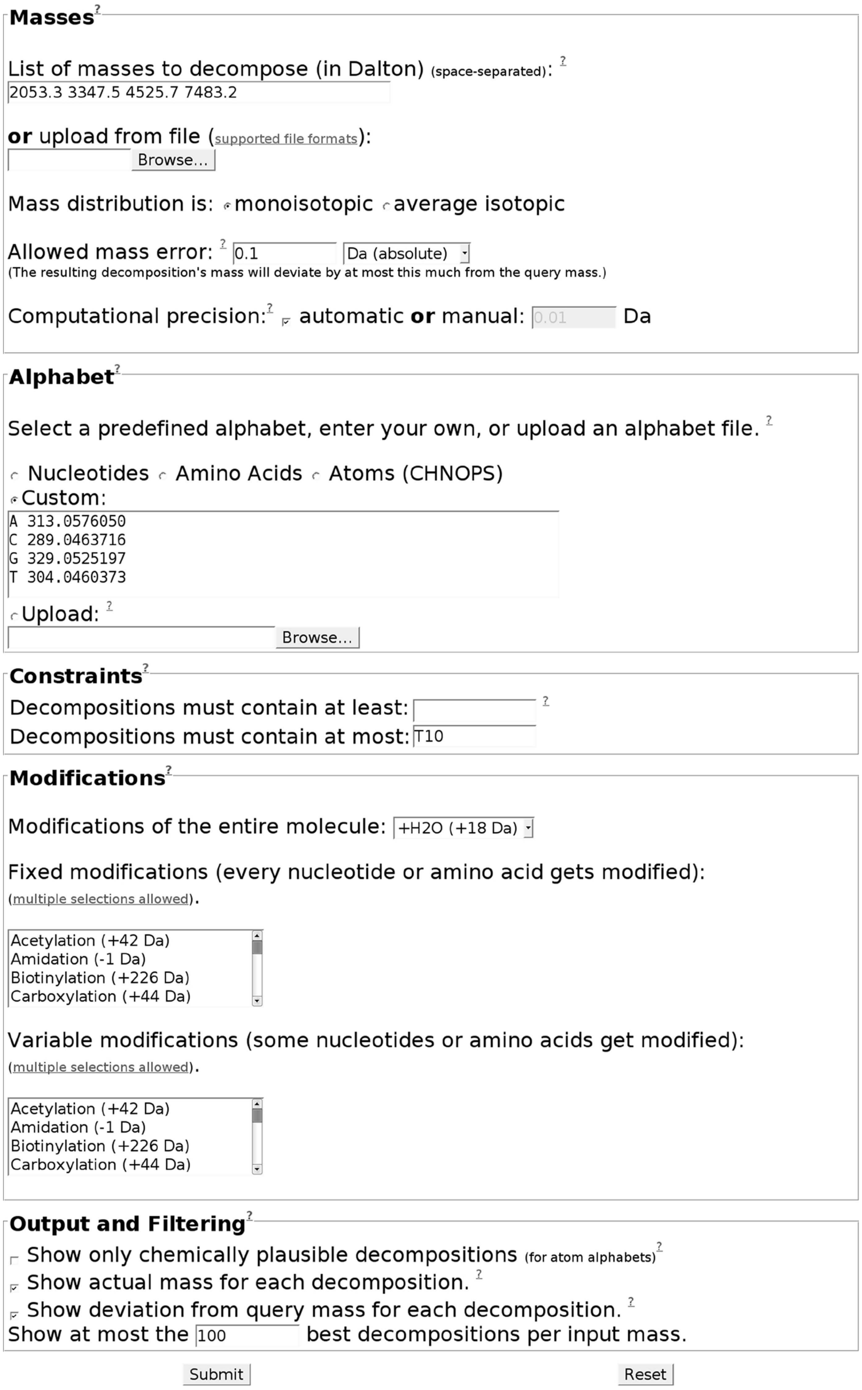
Summary. We have presented DECOMP, a fast and simple tool for solving various mass decomposition problems, both for facilitating in MS data analysis, and for solving instances of MCP problems. An easy-to-use design of our software allows users to quickly submit a task by providing a very few required parameters. DECOMP is supplied with a detailed manual[8] that serves as an extensive guide to all nuances of the submission process and relevant parameters. The results of the program can be used either independently or as a part of a larger identification pipeline of unknown sample molecules.

6.2 Application Case with CompNovo

In this section, we show the applicability of DECOMP as a part of a combinatorial algorithm, called CompNovo, for *de novo* sequencing of peptides using different fragmentation types of tandem mass spectra. We start with the overview of the existing algorithms for *de novo* sequencing of peptides using tandem mass spectrometry. Followed by a brief introduction into several techniques recently appeared for the fragmentation of the tandem mass spectra, we present CompNovo, a novel approach that makes use of the complementary information available in the spectra of different fragmentation types for *de novo* peptide identification. CompNovo is being developed by Andreas Bertsch (University Tübingen) and coworkers, and is described in [12].

---

[8] `http://bibiserv.techfak.uni-bielefeld.de/decomp/manual.html`



Masses?

List of masses to decompose (in Dalton) (space-separated): ?
2053.3 3347.5 4525.7 7483.2

or upload from file (supported file formats):
Browse...

Mass distribution is: ● monoisotopic ○ average isotopic

Allowed mass error: ? 0.1 Da (absolute)
(The resulting decomposition's mass will deviate by at most this much from the query mass.)

Computational precision:? ☑ automatic **or** manual: 0.01 Da

Alphabet?

Select a predefined alphabet, enter your own, or upload an alphabet file. ?

○ Nucleotides ○ Amino Acids ○ Atoms (CHNOPS)
● Custom:

```
A 313.0576050
C 289.0463716
G 329.0525197
T 304.0460373
```

○ Upload: ?
Browse...

Constraints?

Decompositions must contain at least: ?
Decompositions must contain at most: T10

Modifications?

Modifications of the entire molecule: +H2O (+18 Da)

Fixed modifications (every nucleotide or amino acid gets modified):
(multiple selections allowed).

Acetylation (+42 Da)
Amidation (-1 Da)
Biotinylation (+226 Da)
Carboxylation (+44 Da)

Variable modifications (some nucleotides or amino acids get modified):
(multiple selections allowed).

Acetylation (+42 Da)
Amidation (-1 Da)
Biotinylation (+226 Da)
Carboxylation (+44 Da)

Output and Filtering?

☐ Show only chemically plausible decompositions (for atom alphabets)?
☑ Show actual mass for each decomposition. ?
☑ Show deviation from query mass for each decomposition. ?
Show at most the 100 best decompositions per input mass.

Figure 6.2: Submission form for the real-valued mass decomposition problem.

6.2.1 Existing Approaches for De Novo Peptide Sequencing by Tandem MS

The identification of peptide sequences is one of the major goals in mass spectrometry-based proteomics. Approaches that exploit the knowledge of tandem mass spectra are the current mainstream of the high-throughput protein and peptide identification [83]. These approaches can be divided into *Peptide fragment fingerprinting* (PFF) algorithms, where the reference database with protein sequences is used for the interpretation of the measured spectra as shown in Section 2.5.3, and *de novo* sequencing methods, that infer the knowledge about the peptide sequence without prior information. While the former approaches remain the standard strategy for identifying peptides and proteins in complex mixtures, there exist cases, when no appropriate sequence database is available, or when target databases are subjected to sequencing errors, or composed of homologous sequences in the case of cross-species identification. For these cases, *de novo* sequencing, i.e., the determination of the peptide sequence directly from the experimental spectrum, may outperform the database searching [91], and become the method of choice for the successful inference of the peptide identity. Moreover, as a result of the interpretation of the experimental spectra, *de novo* sequencing methods often compute short sequences of length 5–7 amino acids known as *tags*, that can be further used for searching against the database. This mixture of *de novo* sequencing and database searching is often referred to as *tag-based* approaches. Since tag-based methods in the first step of finding putative tags employ exactly the same techniques as *de novo* sequencing, here, we concentrate on the *de novo* algorithms. The overview of the existing tag-based approaches can be found, i.e., in [83].

A wide variety of methods exists for *de novo* peptide sequencing, i.e., [7, 121, 26, 24]. Typically, their algorithmic procedure can be divided into two main parts: generation of candidate sequences and scoring. For the first part, the majority of algorithms, i.e., [26,24,122,10,45], employ a notion of the *spectrum graph*, which nodes represent the spectrum ions and edges represent the differences between m/z ratios of two ions that correspond to the masses of one of the standard amino acids. The candidate sequences are then computed by traversing this graph.

The algorithm NovoHMM [43] utilizes the hidden Markov model (HMM) to directly score the generated sequence. The *de novo* identification problem has been also formulated as the integer linear program (ILP) [30], and this approach demonstrated an excellent performance compared to other *de novo* algorithms for the data measured on high-quality mass spectrometers.

Scoring of candidate sequences against the measured spectrum has also been the subject of intensive investigations. This is due to the crucial importance of the scoring procedure for the overall performance of the database-oriented and *de novo* algorithms. A multitude of different probabilistic scoring schemes, discriminant analysis approaches, and statistical models have been published, i.e., [26, 5, 36]. Even a kinetic fragmentation model to predict complete MS/MS spectra of peptides has been used to score the similarity of peptide sequences to the experimental spectrum [135].

Novel Fragmentation Techniques. Although some *de novo* algorithms presented outstanding results on the high-quality spectra, where the majority of the theoretical ions are available, the overall performance of this technique until recently remained poor compared to the database-oriented methods. One of the reasons for this inferiority is the quality of the spectra, in particular, the missing fragment ions obtained from the most widely used *collisionally-induced dissociation* (CID) of the precursor ions. Using low-energy CID as a fragmentation technique allows for the determination of the peptide sequence from the typical ion series, mainly b-ions and y-ions resulting from the cleavage of the peptide bond.

More recently, CID has been complemented by electron-induced fragmentation methods such as *electron capture dissociation* (ECD) [136] and *electron transfer dissociation* (ETD) [117]. Both these techniques have in common that the fragmentation of multiply protonated peptide ions is induced upon transfer of electrons to them. In ECD fragmentation, low-energy electrons are directly introduced to the peptide ions, resulting in fragmentation of the precursor in different types of fragment ions compared to those produced by other fragmentation methods, such as CID or *infrared multi-photon dissociation* (IRMPD). However, low efficiencies of ECD technique and other experimental difficulties hinder its utilization on the wide scale. Usually, ECD fragmentation is confined with FT-ICR instruments.

In ETD mode, radical anions derived from, e.g. anthracene or fluoranthene molecules by chemical ionization, function as electron donors for positively charged peptide ions, resulting in the spectra similar to those obtained by ECD. Whereas CID spectra contain b-ions and y-ions and many neutral losses to form a complex mixture of ions, ETD spectra commonly show only c-ions and z-ions as fragment ions. Hence, CID and ETD techniques mostly contain complementary information, leading to a more complete coverage of a peptide sequence with fragment ions.

Horn *et al.* [62] proposed a method for *de novo* sequencing of the entire protein utilizing complementary information obtained from CID and ECD spectra on a high-resolution FT-ICR instrument. Utilizing the very accurate mass measurements of FT-ICR-MS/MS, the same group introduced a linear *de novo* sequencing method, which begins with the most reliable fragment ion and generates the sequence by simply adding masses of possible amino acids [105]. However, the proposed greedy strategy is unlikely to generalize well beyond the expensive and rare CID/ECD FT-ICR setup. Moreover, this approach may be an overkill, since FT-ICR allows reliable *de novo* sequencing even without the complementary CID/ECD fragmentations, as shown in [46].

Very recently, Datta *et al.* [27] presented a method for *de novo* sequencing of peptides including CID and ETD spectra at the same time. In particular, a machine learning approach was proposed for learning dependencies between peak types, to improve on a naïve Bayes classifier. However, the performance of this approach heavily relies on the scoring combined spectra, whereas the focus of our work was on the generation of peptide candidates.

Here, we present CompNovo, a novel combinatorial algorithm for *de novo* sequencing using CID and ETD spectra from the same peptide. Recent improvements in instrumentation made available the ion trap mass spectrometers that are capable to produce CID and ETD spectra from the same precursor ion in a tiny interval of time [117]. Thus, the focus of CompNovo has been on developing an efficient method for *de novo* sequencing on medium-quality mass spectrometers such as ion traps. For generation of candidate peptides, CompNovo exploits a divide-and-conquer algorithm and the mass decomposition techniques described above. The algorithm evaluation shows that employing complementary information obtained from both CID and ETD spectra can improve the *de novo* identification significantly, and lead to the identification rates that are superior to those obtained by previous approaches using CID spectra only.

6.2.2 Algorithm Overview

The algorithm consists of two steps: candidate generation and candidate scoring. Followed by spectrum preprocessing, a divide-and-conquer algorithm recursively divides the spectrum into smaller units until the segment is small enough to be the input for the mass decomposition algorithm that produces all amino acid compositions for a given mass segment. Using this list of compositions, all permutations of sequences are computed, and are then scored against the experimental spectra. For each segment the best scoring sequences are kept and marked as candidates for this segment. After processing all segments, the final list of candidates is created and is scored against the measured spectra. The candidates scoring consists of two steps: First, very simple spectra are generated from the candidate sequences and scored against the experimental spectra. The best candidates are then used in a second scoring step to generate more detailed theoretical spectra (including additional ion series, losses, and isotope clusters), which are in turn scored against the experimental spectra. The complete pipeline of the CompNovo algorithm is schematically depicted in Figure 6.3.

In the following, we will discuss individual parts of the algorithm in more detail.

Spectrum Preprocessing. The goal of the preprocessing step is to provide the true y-ions of the CID spectrum for the subsequent divide-and-conquer algorithm. To find as many true y-ions as possible and to reduce the chance of selecting other types of ions from the spectrum, all peaks present in the spectrum are scored using the properties described below. The initial score is the peak's intensity. For the subsequent scoring steps, the fact that ETD spectra contain abundant peaks of the precursor in almost all cases is used, which allows for calculation of the peptide mass the within the mass tolerance of the tandem MS scan. First, for each peak its isotope pattern is compared to the theoretical isotope distribution based on the molecular mass of the fragment ion and on the average elemental composition of peptides with respect to carbon, hydrogen, oxygen, and nitrogen. A simple correlation is used as a multiplicative scoring for this comparison. Doubly charged ions with isotope peaks that correlate well with the theoretical isotope distribution are converted to singly charged ions and added to the spectrum, or the intensity is added to the corresponding singly charged ion if already present in the spectrum.

For further ranking of y-ions, a simplified version of the probabilistic InsPecT [119] scoring scheme is used. The scores of the *witness* set of an ion, i.e., the set of all ions

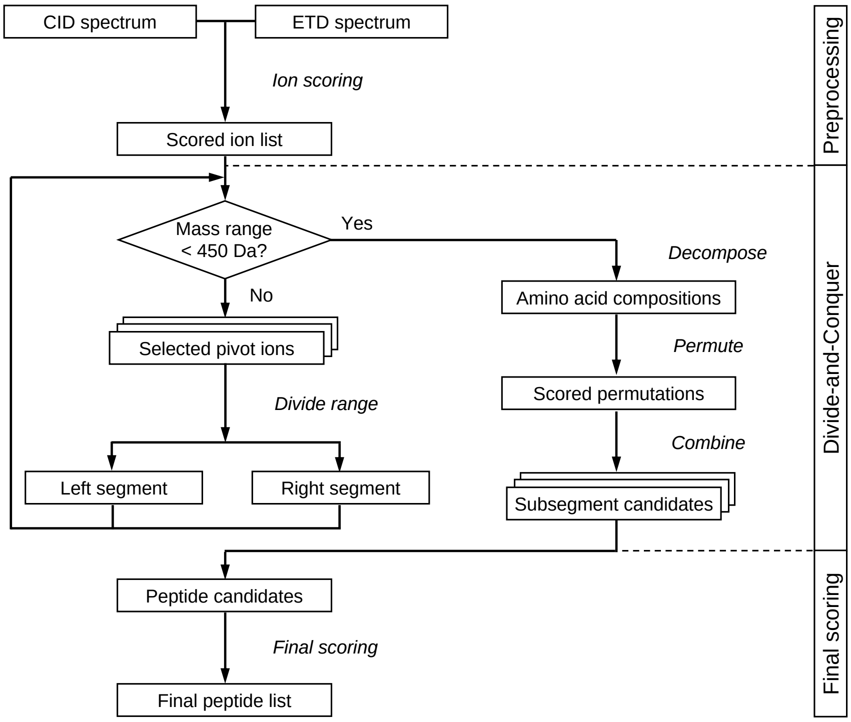


Figure 6.3: Illustration of the CompNovo workflow.

that support this y-ion, are used to increase the score. Additionally, the c/z-ions of the ETD spectrum are used to provide further evidence that a proposed ion is a true y-ion. All intensities of witness ions are weighted by a factor based on the deviation between the expected and calculated mass positions, and are then added to the intensity of the y-ion.

Finally, for each ion and its counterpart with respect to the precursor, it is checked if a mass decomposition is possible for these ions. If no valid decompositions exist for a given ion, its score is set to zero. All ions with positive scores can be used as pivot ions in the following divide-and-conquer step.

Divide-and-conquer algorithm. Peptide sequence candidates are generated using a divide-and-conquer approach, similar to the one proposed in [135]. The main distinction is that, CompNovo uses y-ions as pivot ions, instead of using b-ions. The idea of the divide-and-conquer algorithm is to recursively divide the spectrum into smaller sub spectra until a subspectrum is small enough to be used by the mass decomposition algorithm to compute all possible peptide sequences. Pivot ions are used to divide the spectrum segment into two smaller subsegments.

The algorithm starts with a highly scored putative y-ion that divides the original spectrum into two segments. The lower m/z segment corresponds to the suffix of the peptide sequence and the upper m/z segment corresponds to the prefix of the peptide sequence. If a newly created segment is larger than a given threshold, it is further divided into two subsegments using a pivot ion. The same process is repeated for several pivot ions, to ensure that the correct sequence is generated, i.e., at least one of the selected pivot ions is a true y-ion.

If the subsegment is small enough, the mass decomposition algorithm is employed to compute all possible amino acid decompositions. For each decomposition all possible permutations of sequences are generated. To reduce the number of generated sequences, for each permutation its theoretical spectrum is built and scored against the experimental spectrum; only the best candidates are kept. For subsegments located somewhere in the middle of the original spectrum, a prefix and suffix are added for the generation of the theoretical spectra. CID and ETD spectra are handled independently, and then both scores are added to get a similarity between a sequence and measured spectrum pair.

After the generation of all subsegment candidates is completed, the results are merged by combining every candidate for the left segment with every candidate for the right segment. The merged set is again ranked and reduced to only the best candidates. The entire process is repeated for several ions and the results are reported for the final scoring of the full-length sequence candidates, see Figure 6.3.

Scoring Candidates. At last, the candidates generated by the divide-and-conquer step are scored against the experimental spectra. For this, theoretical CID and ETD spectra of the peptides are generated, which are then compared with the experimental spectra and ranked according to the similarity. To consider the discrepancies in the expected and observed ion positions, the following weighting factor $f_{i,j}$ for peak i in the first spectrum and peak j in the second spectrum is used:

$$f_{i,j} = \frac{\epsilon - |P_i - P_j|}{|P_i - P_j|},$$

where P_i and P_j denote respective peak positions, and ϵ is the mass tolerance allowed for the scoring. Let I_i and I_j denote the intensities of the corresponding peaks. The similarity of two spectra is then computed as follows:

$$s = \frac{\sum_{i,j} \sqrt{I_i \cdot I_j \cdot f_{i,j}}}{\sqrt{\sum_i I_i \cdot \sum_j I_j}}.$$

Since the number of candidates can be very large, the scoring procedure is divided into two parts: For the calculation of the first score all possible pairs of ions are considered, and those that achieve low ranking are filtered out. For this preliminary comparison very simple theoretical spectra are created that consist only of b/y-ions for CID spectra and c/z-ions for ETD spectra. For the second, final score, only the best ion pairs are taken, so that each ion can have at most one counterpart in the other spectrum. The pairs are determined using a spectrum alignment algorithm, where the number of paired ions is maximized and the sum of position distances is minimized. The intensities of ions are ignored by this algorithm. Since for the second score less candidates are compared, the calculation of the theoretical CID spectra is enhanced with more properties of the spectrum, such as a-ions and neutral losses from b- and y-ions. Additionally, doubly charged b- and y-ions are taken into account. Finally, a ranked list of candidate scores is created and reported as the output of the algorithm.

6.2.3 Experimental Results

Experimental Data Generation. For the parameter estimation and evaluations of CompNovo two datasets were used. For the generation of the first, training dataset, nine known proteins, obtained from Sigma (St. Louis, MO) and Fluka (Buchs, Switzerland) were partitioned into two mixtures, and tryptically digested. The resulting peptide mixtures were then separated using capillary ion-pair reversed-phase HPLC (IP-RP-HPLC) and subsequently identified by electrospray ionization tandem mass spectrometry (ESI-MS/MS). On-line ESI-MS/MS detection was carried out on a quadrupole ion-trap mass spectrometer with electron transfer dissociation capability (Model HCTultra PTM Discovery System, Bruker Daltonics, Bremen, Germany). Tandem mass spectra were generated using sequential CID and ETD fragmentation of the same precursor ion. Singly charged analytes were automatically excluded. For the second dataset of a real life sample, the proteins extracted from *Sorangium cellulosum* [106], a soil living bacteria, were used. The experimental setup used for the generation of the second dataset was the same as described above.

Data Preprocessing. Using the experiments described, training and benchmark datasets were generated. Since no currently available database search engines support the use of both CID and ETD spectrum pairs during the identification process and identification using only ETD spectra showed significantly poorer performance with very few identifications found, only CID spectra were used for the annotation of peptides in both datasets.

CID spectra from the eight HPLC runs of the known protein mixture were identified by MASCOT [97], version 2.1.03. Hits to spectra of doubly and triply charged precursors with scores larger than 10 were used. To avoid bias towards specific sequences, spectra corresponding to the same combination of the peptide sequence and charge were allowed at most once in the dataset. Peptide sequences with molecular mass above 2000 Da were excluded. This procedure resulted in a training dataset of 156 pairs of CID/ETD spectra of doubly charged peptides. Only seven pairs of triply charged peptides were obtained and were handled separately.

The benchmark dataset was identified using MASCOT, version 2.1.03, OMSSA [49], version 2.1.1, and X!Tandem [40], version 07-07-01. The target protein database included 9384 protein sequences from *S. cellulosum* and 86 protein sequences of trypsin and keratin. To ensure that peptides included in the dataset were correctly identified, all identifications were extensively verified. Further details about verification procedure can be found in [12]. Out of $38,261$ MS/MS spectrum pairs, the verification routine kept $2,190$ spectrum pairs of doubly charged and 54 of triply charged peptide sequences, and these were included in the benchmark dataset.

Both datasets have been deposited in the Proteomics Identification Database (PRIDE) [65] and are publicly available under the accession numbers #8689 and #8690.

Parameter Estimation. After the determination of the algorithm parameters using the training set, they were applied on the benchmark dataset. The threshold for the maximal mass that can be decomposed was set to 450 Da, and for each subsegment the 40 best amino acids compositions were kept. The computation of the divide-and-conquer algorithm was performed nine times using nine different pivot ions, if available. For the scoring candidates, the 250 top-ranking peptides were kept after the preliminary scoring, and were rescored again using the enhanced CID spectra. The fragment mass tolerance ϵ allowed for scoring was set to 0.4 Da.

Comparison with Other De Novo Algorithms. To evaluate the performance of CompNovo, its results on the benchmark dataset were compared with other publicly available *de novo* algorithms such as PepNovo [45] (version v2.00) and LutefiskXP [122] (version 1.0.5a). Note that both these packages use only CID spectra for the analysis, therefore CompNovo initially has a benefit through using both, CID and ETD spectra. To evaluate the improvements that a pair of CID and ETD spectra provides over CID spectra alone, a slightly modified version of CompNovo, called CompNovoCID, was evaluated, that uses the same algorithm but analyses only CID spectra.

The results of this comparison are summarized in Table 6.1 with the number of fully identified peptide sequences and the identifications with deviations of one to three missed amino acids compared to the correct sequence. One can see that for all types of identifications CompNovo demonstrates the improvements in performance compared to other packages. The identification rates range from 28.5 % for correct sequences to 61.8 % for sequences with 3 missed residues, while the total number of correctly identified amino acids reaches 75.6 %. Moreover, the results show the crucial importance of including the ETD spectra in the analysis. Although the CID version of the algorithm already outperforms other algorithms in all categories, this is the inclusion of ETD data that makes the accuracy of CompNovo's identifications greatly superior: While CompNovoCID finds only 9.9 % fully correct sequences, CompNovo makes 28.5 % correct identifications, which corresponds to roughly 3-fold improvement in the identification rates upon the inclusion of ETD spectra. Identification performances for the triply charged peptides were much lower: As the consequence of the decreased coverage of the cleavage sites, the identification rates of CID-based algorithms were decreased dramatically. Out of 54 triply charged peptides the benchmark dataset, CompNovo identifies 35 % of the amino acids correctly, whereas PepNovo assigns the correct residues in 15.2 % of the cases. Note, however, that in the current experimental setup 54 triply charged peptides constitute to only about 2.5 % of the spectra in the benchmark dataset, which might be too small amount for the rigorous evaluation.

Here, we skip further discussion of the other aspects of the algorithm evaluation, such as a number of correctly predicted short subsequences, or the influence of the charge state of the precursor ion on *de novo* sequencing; further details can be found in [12].

	LutefiskXP	PepNovo	CompNovoCID	CompNovo
correct peptides	0.0	2.2	9.9	28.5
within 1 residue	0.0	3.2	10.1	29.3
within 2 residues	1.6	11.5	25.4	54.0
within 3 residues	2.9	19.0	32.1	61.8
total correct residues	6.9	48.1	56.0	75.6

Table 6.1: Identification rates for different *de novo* sequencing programs for the benchmark dataset in comparison. Rates are given as a proportion to the total amount of 2 190 spectrum pairs in the dataset. For each spectrum pair, only the best scoring peptide sequences were considered from each *de novo* algorithm.

Summary. We have presented CompNovo, an efficient algorithm for *de novo* peptide sequencing using the medium-quality fragmentation data from CID and ETD spectra in combination. Using CompNovo, the applicability of the previously described mass decomposition techniques has been demonstrated, for the efficient candidate generation in the divide-and-conquer procedure. The CompNovo algorithm is available for download as a part of the software package OpenMS [116], which offers an open-source C++ framework for LC/MS data management and analyses[9].

6.3 Rdisop

6.3.1 Introduction

Suppose you are given two peaks with masses 285.075375 and 286.079064 Da and the corresponding intensities 82.03 % and 17.97 %, and the mass accuracy of your instrument is about 3 ppm. If you were to use DECOMP to try to find out what molecule this peak list may represent, you would though find all seven candidates that have the monoisotopic mass in the given mass range correctly, but the true molecule $C_{16}H_{12}O_5$ with monoisotopic mass 284.068475 Da (plus 1.007278 Da for H^+) would be only the fourth candidate with respect to the deviation of the monoisotopic mass. Clearly, the mass information alone does not suffice for the unambiguous identification of a compound even for higher mass accuracies, i.e., < 1 ppm, [72]. Rdisop helps to efficiently solve this problem, by taking into account an additional information provided by the isotope pattern of the molecule for the better identification of molecular formula of metabolites. For the above example Rdisop can unequivocally find the correct molecular formula, and clearly distinguish it from other candidates.

There exists a wide variety of tools for identification of small molecules [17,73,52,115]. However, some of them have been designed to account only for the peak masses during the identification process [17, 115], while others show significant problems usability and

---

[9] `www.openms.de`

data exchange [73,64]. Moreover, none of the existing solutions have been equipped with the built-in functionality for the statistical analysis of the results.

We have developed Rdisop, a new package to determine the molecular formula of metabolite solely from its accurate mass and isotope pattern. It employs efficient techniques for the decomposition of molecular masses and simulation of the isotope patterns, and integrates them into a freeware statistical package R[10], which in turn provides a excellent environment for the subsequent statistical data analysis. Employing an algorithmic core written in C++, that has been enhanced with an extensive R functionality, Rdisop combines the fastness of the C++ implementations with the modularity and comprehensiveness of statistical analysis using R.

Rdisop does not provide a graphical user interface, and all operations are performed using a command line interface. It is possible to add a more elaborate user interface on top of this infrastructure, however, the focus of this package was on a programmatic approach to enable the convenient use of novel algorithmic methods, and their easy integration with other existing software for the statistical analysis of mass spectrometry data.

Rdisop is available through Bioconductor [50], an open-source project[11] for the collaborative development of extensible software for computational biology and bioinformatics. Although started as an initiative for the statistical analysis of microarray data, meanwhile, Bioconductor has evolved into one of the standard tools for analyzing of various types of high-throughput genomic and metabolomic data [50].

The biological and chemical background on the metabolites and isotope patterns have been given in Chapter 4, while the algorithms for the mass decomposition and simulation of the isotope patterns have been extensively discussed in Chapters 3 and 4. Thus, in the following, we focus on the implementation and usage of our software.

6.3.2 Implementation and Use

The core algorithms in Rdisop are implemented in C++. They have been integrated in the R statistical environment using a package Rcpp, available at Comprehensive R Archive Network (CRAN)[12]. Rcpp is a tool that facilitates the process of using C++ libraries from within the R system, by providing the user with the set of C++ classes that allows the simple usage of C++ implementations from R, without necessity to know technical details about R API internals.

Like any R package, Rdisop is command-line oriented. The functions are called by the user, possibly with arguments and options. Any session using Rdisop in R starts with the command

```
> library(Rdisop)
```

that makes the functions of Rdisop available in the R environment.

---

[10] `http://www.r-project.org/`

[11] `http://www.bioconductor.org/`

[12] `http://www.cran.r-project.org/`

Initialization of Elements and Molecules. The key object in Rdisop is a `Molecule` that represents a list containing its molecular formula, an isotope pattern, a score and other properties. There are principally two ways to initialize molecules. The explicit initialization can be performed by providing the molecular formula to `getMolecule()` function or using a predefined set of elements: For example, the six most common elements in living beings, CHNOPS, can be initialized using `initializeCHNOPS()` function. Most functions in Rdisop operate only a subset of the periodic system of elements (PSE). Alternatively, molecules can be created implicitly by executing the algorithm either for masses alone using `decomposeMass()`, or for both masses and intensities using `decomposeIsotopes()` function.

Function `getMolecule()` returns a list object containing the data for a single element or composite molecule. For example, an amino acid alanine can be initialized as follows:

```
> alanine <- getMolecule("CH3CHNH2COOH")
> alanine$formula

[1] "C3H7NO2"

> alanine$exactmass

[1] 89.04768
```

Note that the molecule's formula is written in a canonical form, and the monoisotopic mass includes decimals, while the nominal mass of alanine equals 89 Da. By default, the elements in use are CHNOPS that are most relevant in metabolomic research. To create a subset of different elements of PSE, one can use another initialization function, for example, as follows:

```
> newElements <- initializeCHNOPSMgKCaFe()
> chlorophyll <- getMolecule("C55H72MgN4O5H", z = 1,
+  elements = newElements)
> isotopes <- getIsotope(chlorophyll, seq(1,4))
> isotopes

            [,1]        [,2]        [,3]         [,4]
[1,] 893.5431133 894.5459715 895.5462516 896.54753129
[2,]   0.4117684   0.3176796   0.1793288   0.06834484
```

In the above listing, an extended set of elements has been created that includes other elements occurring in metabolites such as magnesium (Mg), kalium (K), calcium (Ca), and iron (Fe). Using this set, a composite molecule chlorophyll has been created that contains a metal ion Mg, and is positively, singly charged ($z = +1$). To examine the leading four isotopes of the chlorophyll, the `getIsotope()` function was used. For the visual inspection the isotope pattern can be plotted using the following command:

```
> plot(t(isotopes), type = "h", xlab = "m/z", ylab = "intensity").
```

Decomposing Masses. There are two functions that execute the identification algorithm: `decomposeMass()` and `decomposeIsotopes()`. Given a mass and an error, the `decomposeMass()` returns a list of molecules with monoisotopic masses that fall into a specified mass range:

```
> molecules <- decomposeMass(57.0214, ppm = 20)

$formula
[1] "C2H3NO"

$score
[1] 1

$exactmass
[1] 57.02146

$charge
[1] 0

$parity
[1] "e"

$valid
[1] "Valid"

$DBE
[1] 2

$isotopes
$isotopes[[1]]
           [,1]        [,2]         [,3]         [,4]         [,5]
[1,] 57.0214640 58.02401863 59.025741860 6.002822e+01 6.103017e+01
[2,]  0.9715877  0.02618779  0.002171668 5.238267e-05 4.294902e-07
         [,6]          [,7]  [,8]         [,9] [,10]
[1,]  6.203062e+0  16.303616e+01 6.404216e+01 6.504829e+01     66
[2,] 1.087790e-09   4.333328e-13 6.224049e-17 3.043903e-21      0
```

Here, a single molecule, the amino acid glycine exists with the monoisotopic mass that lies in the specified mass region. In addition to a molecular formula and isotopes, each candidate is supplied with the parity, the validity, and the *double-bonds equivalent* (DBE), which are simple and frequently used indicators to test the plausibility of a solution. The parity indicates the evenness of the number of electrons in the molecule. The validity is determined by the *nitrogen rule*, which states that the molecule with an odd number of nitrogens should have an odd nominal molecular mass. DBE is equivalent to the *rings-plus-double-bonds equivalent* (RDBE) or the *degree of unsaturation* (DU) [96], which is calculated as $1 + \frac{1}{2}\sum n_i(v_i - 2)$, where n_i is the number of atoms with valence v_i.

The candidate molecules that are marked as invalid, or those with the unusual DBE values, are not discarded by default. Instead, the decision regarding the relevant filtering strategy is left to the user. Doing so, we avoid removing rare, but legimitate compound classes that do not obey common chemical rules.

Regarding the nitrogen rule, it should be applied only with nominal masses. However, small non-nominal mass contributions from a large number of constituting elements add up in higher mass regions, leading to the monoisotopic mass which rounds to the next integer compared to the true nominal mass. For example, consider a molecule $C_{38}H_{64}N_2$ with the monoisotopic mass 548.50692 Da that rounds not to the true nominal mass 548 Da, and thus do not follow the nitrogen rule. Regarding the DBE value, it is calculated using the atom valences. However, as mentioned in Chapter 4, several elements can have different valence states in different molecules. For example, nitrogen and phosphor can have valence 3 or 5, whereas sulphur can have valence 2, 4, or 6. Thus, molecules with negative DBE values can not be excluded beforehand, because normal valence state may be exceeded. For example, a molecule $CH_2F_{10}S_2$ contains sulphur at a valence state of six with a DBE value of -4.

Taking into account these considerations, we want to refrain from removing any legitimate compound *a priori*. The user can, after all, arrange the appropriate filtering rules to report only the relevant portion of results.

Decomposing Isotope Patterns. On the modern mass spectrometers, such as FT-ICR or Orbitrap, a molecule's isotope pattern of outstanding accuracy can be obtained. This allows to use both masses and intensities in the identification process to improve the accuracy of the molecular formula prediction. Given the isotope pattern of a sample molecule and a mass measurement accuracy, `decomposeIsotopes()` returns a ranked list of molecule candidates sorted by a score, that represents a similarity measurement between the theoretical and measured isotope patterns (see Section 4.3 on page 44):

```
> masses <- c(196.100708, 197.102185, 198.105295)
> intensities <- c(0.8122, 0.1549, 0.0329)
> molecules <- decomposeIsotopes(masses, intensities, ppm = 3.0)
> cbind(molecules$formula,molecules$score,molecules$valid)

     [,1]          [,2]                   [,3]
[1,] "C13H12N2"    "0.99999998888609"     "Valid"
[2,] "C7H18NO3S"   "1.11087825603777e-08" "Invalid"
[3,] "C9H15N3P"    "5.1275127846229e-12"  "Invalid"
[4,] "C3H21N2O3PS" "1.97564553565218e-33" "Valid"
[5,] "C5H18N4P2"   "1.12078165228393e-47" "Valid"
[6,] "H26N3S4"     "2.59829944976493e-91" "Invalid"
[7,] "C2H141NO"    "5.39181020551907e-105" "Invalid"
[8,] "CH21N5P3"    "5.0512131881472e-200" "Invalid"
```

```
 [9,] "H20O11"      "0"                          "Valid"
```

The above listing contains 9 candidate molecules whose monoisotopic masses lie within 3.0 ppm from the measured mass of 196.100708 Da. Each candidate can be inspected on the validity with respect to the nitrogen rule. The true molecule $C_{13}H_{12}N_2$ is clearly identified as the top candidate, based on the score of its isotope pattern.

In the mass spectrometry experiment, the measured peak list typically corresponds to some adduct ion, such as $[M + H]+$ in LC/ESI-MS experiment. In order to obtain the molecular formula of the actual sample molecule, the adduct can be removed using `subMolecules()` function:

```
> query <- subMolecules("C5H10NO4", "H")
> query$formula

[1] "C5H9NO4"
```

Similarly, if during the ionisation an in-source fragmenation occurs, the lost fragment can be added to the sample molecule using `addMolecules()` function.

Related Bioconductor Packages. The data from a mass spectrometer is typically obtained in the form of a raw machine data. The current release of Bioconductor (version 2.4) contains two packages that handle the peak picking of the raw data: MassSpecWavelet [33] and XCMS [111]. As the latter contains a wrapper for MassSpecWavelet, we can suggest XCMS to be used to transform the raw data into the input peak lists for Rdisop. Another package, called CAMERA [120], which is at the time of this writing available only in a Bioconductor development version[13], can be used to extract the isotope peaks from the peak lists generated by XCMS.

After the candidate molecular formulas are generated, the results can be further queried in open-access compound databases, including PubChem[14] from the National Center for Biotechnology Information (NCBI), and Chemical Entities of Biological Interest (ChEBI)[15], to find out if any information about this compound exists. Note that a match or non-match does not indicate a correct or incorrect formula, but facilitates in the subsequent validation or structure elucidation steps.

Summary. We have presented Rdisop, a new R package for the identification of the molecular formula of metabolites solely from its accurate mass and isotope pattern as obtained from high resolution mass spectrometers. The efficient algorithm implementations in Rdisop are complemented with the front-end features to perform the subsequent statistical analysis. The distributions of Rdisop for Unix/Linux and Windows operating systems are available under the GNU General Public License (GPL). As other Bioconductor packages, Rdisop offers a high level of standardized documentation through the dedicated vignette and help pages[16].

---

[13] `http://bioconductor.org/packages/devel/bioc/html/CAMERA.html`

[14] `http://pubchem.ncbi.nlm.nih.gov/`

[15] `http://www.ebi.ac.uk/chebi/`

[16] `http://www.bioconductor.org/packages/2.4/bioc/html/Rdisop.html`

Rdisop fills the need for a simple and efficient tool for experiments in metabolomics on a small scale. For a large scale application, a relevant graphical user interface is an indispensable feature for the successful interpretation of the molecular formulas of unknown sample fragments. In the next chapter, we address this issue by presenting a standalone program, that complements the efficient computational methods with a powerful graphical user interface for the large scale analysis of MS data.

7 SIRIUS

According to Human Metabolome Project[1], there exist about 2 900 common metabolites that are detectable in the human body, not accounting for secondary metabolites. Yet, until recently less than 30 % have been identified. Similar is the situation for higher eukaryotes: Up to 20 000 metabolites are currently estimated to remain uncharacterized for any given organism [41]. This lack of information is mostly due to the high diversity and variability encountered in the metabolome. Consequently, one of the key challenges in metabolomics, analytical chemistry, and other related life sciences is to establish the knowledge of metabolite identity.

As mentioned in the previous chapter, for a small scale application, a multitude of various tools exists for the identification of small molecules. For a large scale analysis of metabolomic data, the existing software solutions are mainly concern with the processing and storage of the mass spectrometry (MS) data. One of the most recent examples in this category is MeltDB [87], a web-based program for the preprocessing, annotation and storage of datasets obtained from metabolomic experiments. Other recent solutions include, e.g., TagFinder [78] and mzMine [69]. Whereas the focus of these packages is on the preprocessing and annotation functionality, the compound identification is mainly performed by mass spectral database lookup. Clearly, these methods are confined to identifying metabolites and chemical compounds that have been included in some reference library. Unlike in evolved genomics and proteomics with a number of existing target databases, the state-of-art in metabolomics is yet under very developmental stage with the limited number of available databases [28]. Hence, *de novo* identification solutions are highly sought. Moreover, the visualization tools often suffer from poor or no support of computational methods, while the algorithmic tools are usually command line based and require good understanding of the software environment [86, 111].

We have developed a new java-based application SIRIUS (Sum formula Identification by Ranking Isotope patterns Using mass Spectrometry), which aims at user-oriented software framework for *de novo* identification of molecular formulas of metabolites using accurate mass spectrometry data. SIRIUS core comprises efficient algorithms for generating all elemental compositions for a given mass and error, calculating isotope patterns for all chemically relevant compositions, and matching and ranking the candidate molecules against the input spectrum. These computational methods in SIRIUS are combined with a powerful graphical user interface for a fully featured analysis of MS data. An extensive management system allows simplified data handling, and offers an easy way to integrate new algorithms and data structures in the application. Through a user-friendly interface, SIRIUS allows user (i) to import datasets in most known mass

---

[1] `http://www.metabolomics.ca/`

spectrometry file formats, (ii) automatic recognition of molecular ion adducts available in the input spectrum, (iii) handy visualization of identified molecular formulas and their isotope patterns, (iv) customizable export of identification results to common file formats. For the means of data post-processing, we provide basic functionality to search for molecular formulas identified by the algorithm in NCBI PubChem database[2].

The remainder of this chapter is organized as follows: we start by describing the system architecture, and give some technical details on the used libraries. Then, we present a basic application workflow and outline the cornerstones of the data and analysis preparation. We also give a brief overview of the existing mass spectrometry data formats, and the file formats that are supported by SIRIUS. Finally, we present the application functionalities for the analysis of algorithm results, including visualization, data export, and searching for molecular formulas in external databases.

7.1 System Architecture

Similar to a typical three-tier application, SIRIUS structure consists of three layers: the *presentation* layer, the *functional* or *business logic* layer, and the *data* layer. The presentation layer consists of the *graphical user interface* (GUI), and is responsible for the data visualization and processing the user requests to the next, functional layer. The functional logic layer includes the implementations of the algorithms and general utility components, which can be used at all layers of application. The third layer comprises a set of *domain objects* (DOs) that represent basic entities used in the problem domain, for which the application is created. Domain objects are typically designed to be made persistent in a database or flat files, and thus, have to be clearly distinguished from the other types of data, such as user interface widgets or algorithm data structures. Separating domain objects from the user interface and algorithms improves the application scalability and performance. In Figure 7.1, a schematic illustration of the SIRIUS architecture is depicted.

7.1.1 Domain Objects

Domain objects typically represent logical entities containing some important information in the problem domain space [48]. As our problem domain is the identification of sample molecules using mass spectrometry, the domain objects represent the corresponding entities of this domain such as, e.g., chemical elements, compounds, peak lists etc. The diagram of the structure of the domain objects is shown in Figure 7.2.

In general, domain objects should know how to copy and compare themselves to other domain objects of the same type. This is of particular importance in the large scale applications, as it keeps implementations error free and easily extendable. Since any domain object can contain references to other domain objects, the copying and comparison of the whole hierarchical structures can become tedious and error prone. To overcome

---

[2] `http://pubchem.ncbi.nlm.nih.gov/`

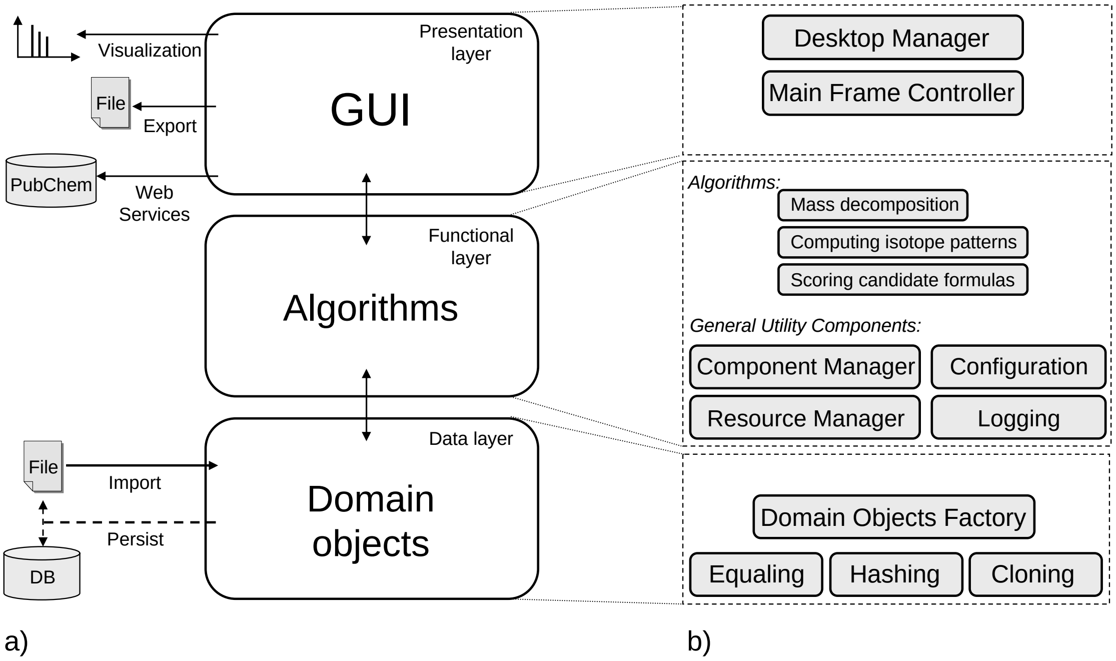


Figure 7.1: a) The three-tier architecture of the SIRIUS framework. b) Schematic representation of the main SIRIUS components.

this difficulty, any domain object in SIRIUS extends the basic `AbstractDomainObject` that takes the responsibility over the cloning and comparing functionality.

`AbstractDomainObject` has a default implementation of `clone()`, `hashCode()`, and `equals()` methods using Java Reflection[3]. To obtain this built-in functionality, the extending class has to only implement `getClonePrototype()` method and the cloning of the collections. If some DO needs a specific implementation, the default behavior of these methods can be overridden. Note that this design sets one important requirement on the DOs structure: Domain objects need to form a directed acyclic graph (DAG), otherwise the default implementations of `clone()`, `hashCode()`, and `equals()` will produce an infinite recursion and end up with `StackOverflowError` exception.

All DOs instances have to be created using `AbstractDomainObjectFactory`. This ensures that if the implementation of one DO needs to be changed, the other parts of code can remain untouched as long as the new functionality implements the same domain object interface. This behavior resembles the well known *Abstract Factory* design pattern [48].

Keeping domain objects as simple as possible enhances the portability, and simplifies their integration with the entities from other domains. Thus, domain objects in SIRIUS are realized as Plain Old Java Objects (POJOs). Furthermore, each domain object should have a DO *interface* that abstracts its implementation from other parts of the

---

[3] `http://java.sun.com/docs/books/tutorial/reflect/`

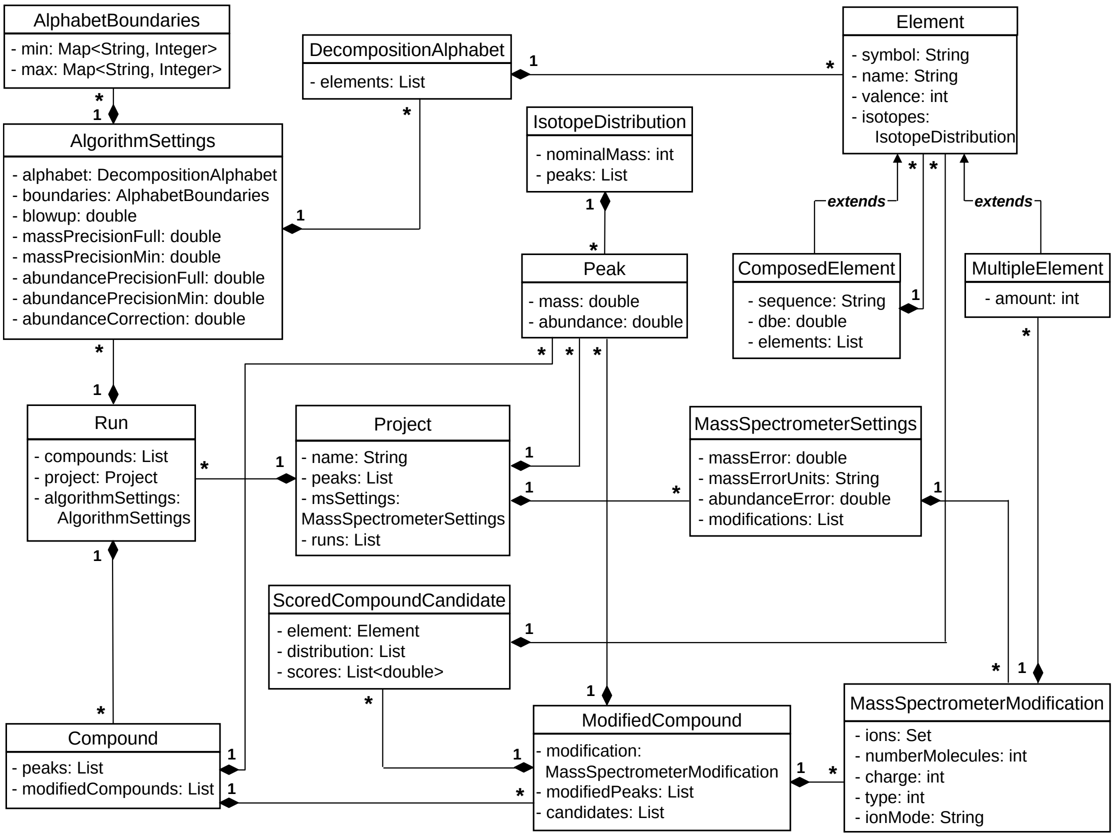


Figure 7.2: Overview of the structure of the domain objects in SIRIUS.

functional layer. The DO interfaces are used to make changes in the implementation easier and faster. Therefore, all algorithms and GUI components should handle only instances typed to these interfaces, not to the actual implementing classes.

7.1.2 Functional Logic Layer

The *functional* or *business logic* layer consists of the algorithm implementations and *general utility components* that can be used at all application layers, see Figure 7.1. Major algorithm implementations in SIRIUS include (i) generating of all elemental decompositions for a given mass and error, (ii) computation of the molecule's isotope pattern for a given molecular formula, and (iii) scoring the theoretical isotope pattern against a given measured peak list. Each implementation is clearly encapsulated from other parts allowing the execution of any individual algorithm separately. As algorithms have been described in detail in Chapter 4, here we omit further discussion on their implementations; more details can be found in the documentation to the *application programming interface* (API) on the project web page[4].

---

[4] `http://bio.informatik.uni-jena.de/trac/sirius`

SIRIUS components are the part of the central functional layer as they can be used by the top and bottom application layers as well. They include *Component Manager*, *Configuration*, *Resource Manager*, *Logging* and others, see Figure 7.1. Here, we give a brief introduction to the several basic components, while the description of the remaining units can be found in the associated API documentation.

Component Manager. *Component Manager* is the central component in the application infrastructure. It is responsible for the creation and deletion of other general utility components at the business logic layer and other layers as well. By keeping the references to all existing components, it allows to statically obtain such reference at any level of the application. This way, *Component Manager* acts as a "container" for all other components. For example, if user is about to create a new domain object, the presentation layer requests from the *Component Manager* the reference to the object factory of the current domain, and performs an instantiation.

The initialization of the components in *Component Manager* is performed in two steps: First, the basic components such as *Logging* and *Configuration*, are created. Then, the remaining units are initialized, which by the creation can employ the functionality of the basic components. For example, if any of the latter initializations fails, the error can be traced back using the *Logging* component.

Other Utility Components. *Configuration* component can be used to store various configuration values that are defined at application startup. For the initialization, the component uses the contents of the application configuration XML file that contains a list of configuration key-value pairs. After creation, clients can query the component for the configuration values giving a corresponding key as a parameter for the query. This way, any configuration parameter given at startup ,e.g., the current application version can be retrieved from this component.

Clear and verbose logging is vital both for the development and maintenance phase of any software. The logging responsibility in SIRIUS is handled by the special purpose *Logging* component, which is built upon a widely used Log4j Library 1.2.14[5]. Log4j offers an extensive functionality for advanced and verbose logging at all application levels. Similar to a typical logging framework, *Logging* component in SIRIUS divides the log output to five different levels: `DEBUG`, `INFO`, `WARNING`, `ERROR`, `FATAL`. Note that the debug level should be used sparingly, such that at least in production it is completely left away from the actual file output. To make the logging available at the presentation layer, a dedicated `Log Panel` exists that accumulates the log output and informs user with notifications.

The component *Resource Manager* is used to obtain references to any external binary resources such as images, sounds, videos etc. Each resource used in the application has its entry in the resource map in *Configuration.* Using a relevant key, clients can query the *Resource Manager* to provide a *Uniform Resource Locator* (URL) of the corresponding

---

[5] `http://logging.apache.org/log4j/`

resource. One of the basic examples of using the *Resource Manager* is to provide an icon file to be shown in the presentation layer.

Another general utility component is a *Biological Data Service*, which provides any biochemical data necessary for our analysis, such as the properties of the relevant chemical elements that includes, e. g., a distribution of natural isotopes, and valences. Upon the component creation, this information is initialized from the configuration XML file using the XStream Library 1.2.2[6], which allows a simplified (de-)serialization of XML files, and is used for this task throughout our application. To de-serialize any file into the application, a *Serialization Manager* is used, which unmarshalls the contents of the file available from the hard disk or class loader into the corresponding data structure. Similar, a serialization of any object back to the XML file is performed.

7.1.3 Presentation Layer

The SIRIUS presentation layer consists of the *graphical user interface* (GUI), which is based on the standard Java Swing Library[7]. SIRIUS employs a wide variety of other Swing-based technologies that can be reused to keep a unified design and simplify the integration of new functionalities. Here, we only outline several major aspects of the GUI infrastructure; the remaining parts can be found in the associated API documentation.

Application Desktop. Frames and panels on the application desktop are arranged and managed by the docking system of the VLDocking Framework 2.1.4[8]. Using docking, the application views can be arranged or *docked* on the desktop with respect to their responsibility, allowing for a simple navigation and customization of the application desktop. Docking of UI components at the SIRIUS application frame is handled by the *Desktop Manager* component. *Desktop Manager* contains two desktops: a *view* desktop, which corresponds to the full application frame, and a *data* desktop, which is a segment of the view desktop, and displays the output of the algorithms. Correspondingly, all UI components are divided in two categories: components that display the algorithm results are grouped together on the data desktop, while the remaining units are spread around occupying the remaining space, and can be toggled on or off depending on the user needs. In Figure 7.4, a typical view of the SIRIUS main frame is shown: views containing the algorithm output are tabbed together in the central pane, while other components are placed on the boundaries, including `Project Explorer` (left), `Properties View` (left bottom), and `Log Panel` (bottom). Such layout clearly separates the output of the computational methods from other information, allowing clear navigation and customization of the application frame. Adding a new component to the desktop structure is as easy as providing a component's unique identifier, a group to which the component belongs, and other supplementary information such as a title and icon.

---

[6] `http://xstream.codehaus.org/`

[7] `http://java.sun.com/docs/books/tutorial/uiswing/`

[8] `http://www.vlsolutions.com/en/index.php`

Dialogs and Panels. Dialogs and panels in SIRIUS are based on the JGoodies Forms 1.0.7[9], which provides a powerful and precise layout for the GUI components. To keep a standardized look-and-feel, SIRIUS supplies the base classes `DefaultDialog` and `DefaultPanel`, which can be extended by newly created GUI components. In addition, `UIUtils` class provides a set of useful utility functions for the consistent GUI design, including the creation of the default form builders, columns, rows, borders etc.

Wizards. For creation of new projects and analyses, SIRIUS employs Wizard dialogs from the SwingLabs Wizard Project (version 0.992)[10]. An open-source Wizard API offers a powerful framework for making the comprehensive wizards by coupling together several settings pages for entering the user input. Although wrapped together for the wizard purposes, the input components in SIRIUS are fully decoupled from the Wizard API itself, and can be reused in different from the wizard-based implementations. For example, handling of the algorithm specific parameters is performed on the `AlgorithmSettingsPanel`, which is used both in a wizard-based creation of each particular analysis and for modifying the default algorithm settings globally.

Application Startup and Shutdown. SIRIUS startup is realized using Swing Application Framework (SAF) 1.03[11], which offers developers a rich set of elements typical to any Swing-based application, such as lifecycle, session storage, threading and others. SAF has already been successfully integrated in large scale applications for the analysis of phylogenetic [53] data. SIRIUS integrates several SAF features, including the application lifecycle and session state functionality.

Each application has a well-defined life cycle, such as launching the application, starting the GUI, and shutting down. Each stage has a corresponding method that our application can extend or use the default SAF functionality. For example, SIRIUS overrides only two methods `startup()` and `shutdown()` from the SAF `Application` class to provide the specific behavior to the corresponding lifecycle stages, whereas the remaining work to successfully start and close the application is left to the underlying framework. Upon startup, `Application` reads the configuration file with application settings that include, e. g., an icon, title, screener, and launches the program. Upon shutdown, various runtime properties of the application can be stored. These properties can be used to store small amounts of client specific graphical configuration, including window size, internal frame locations, and other graphical properties. To save this data locally and restore by the next start, SIRIUS utilizes the session storage functionality of the SAF `ApplicationContext` class.

After starting up SAF default functionally, SIRIUS performs a few more preparation steps. First, a *Component Manager* is created, which performs the initialization of all data layer components, general utilities components, and *Desktop Manager*. Then, *Main Frame Controller* is initialized, which, in turn, setups the remaining UI widgets on the

---

[9] `http://www.jgoodies.com/freeware/forms/`

[10] `https://wizard.dev.java.net/`

[11] `https://appframework.dev.java.net/`

main frame including an icon and main menu. *Main Frame Controller* is responsible for handling all application events coming to the main view. This way, it functions as a *Controller* in a Model-View-Controller (MVC) design pattern [44], which is one of the cornerstones of any Swing-based application. Here, we omit further details about the *Main Frame Controller.* The complete description of this and other components involved in the application startup can be found in the associated API documentation.

Finally, the application desktop is shown, indicating the readiness of application to receive and handle user requests. For a “quick start” example, SIRIUS provides some demo data from the datasets described in Section 4.4. Once this data is uploaded, a few demo projects become available for submission to the identification algorithm. A detailed survey to the typical application workflow is the topic of the next section.

7.2 Application Workflow

A typical SIRIUS workflow includes the following steps: preparing the input data for the analysis, setting up and running the algorithm, and analyzing the algorithm results. In this section, we discuss each of these steps in more detail.

7.2.1 Preparing Input Data

A project is a root object in the hierarchy of the SIRIUS domain, and serves as a “portfolio” for the input and output data of the algorithm analysis. Originated from the laboratory experiment, the input for the computational analysis includes the processed MS data in the form of peak lists, and settings of the mass spectrometer on which this data has been obtained. Instrument settings include a mass, an abundance error, and a set of molecular ion adducts typical for this instrument.

In an initial step, the input data has to be added to the project. For an upload of the peak lists only, user can perform a dedicated `Import Data` task from the pop-up or file menu, which reads the MS data from file(s) and adds it to one of the available projects. Alternatively, the input peak lists and machine settings can be entered manually[12] in a new project or analysis wizard. The corresponding tasks are, again, available from the pop-up or file menu. After initialization, the input peak list and machine settings can be reused for multiple analyses within the same project.

To import mass spectrometry data, SIRIUS employs the ProteomeCommons.org IO Framew̃ork 6.21 [38], which allows reading most of the available MS data formats. In the remainder of this section, we give a short overview of the existing file formats, and provide recommendations upon conversion of the raw data to file formats supported by SIRIUS.

Supported Mass Spectrometry Data Formats. Following the variety of commercial and custom-build mass spectrometers, a multitude of different MS data formats exists.

---

[12] For the peak lists, the file(s) upload is also available.

All available file formats can be divided into three groups: binary data, plain text, and more recent XML-based format.

The raw data obtained from a mass spectrometer is stored in a vendor binary file format. Several well known binary formats include raw/Xcalibur (Thermo Electron), wiff/Analyst (ABI and MDS Sciex), raw/MassLynx (Waters), and baf (Bruker). Before computational analysis, MS spectra has to be extracted from raw data to ASCII or XML data formats. A number of various conversion tools are available for vendor specific file formats[13]. Alternatively, the raw data can be converted into a network Common Data Form (netCDF), a binary data format, which is widely used for saving experimental data including MS data[14].

Typical examples of the plain text format are mgf (MASCOT), dta (SEQUEST), and pkl (ProteinLynx, Micromass). While saving low resolution or low volume MS data in plain text can be a plausible solution, as far as high resolution or chromatography MS data is concerned, the usage of ASCII format becomes impractical due to the huge file size. Moreover, plain text files do not contain precursor scans, thus, lacking valuable information for comprehensive computational analysis.

More recently, an open XML file format mzXML [95] has been introduced for storing data generated by mass spectrometry runs, and has been quickly supplied with a number of tools to convert manufacturer formats, such as Mascot Distiller[15], ReAdW, mzBrucker, and others[16]. However, mzXML has not been designed to contain biological descriptive information. About the same time, the mzData format [90] has been developed with the primary concern of encapsulating the peak lists from raw data, and the durable archiving of the extracted data in public repositories. As the existence of two separate formats for essentially the same purpose has been causing considerable confusion and extra programming support, it has been decided to create a new format unifying mzXML and mzDATA. In June 2009, a joined mzML 1.1.0 format was released under the guidance of the HUPO Proteomics Standards Initiative (PSI)[17], which addresses the problem of standardizing mass spectrometry-related data formats and vocabulary.

As the importing functionality in our application basically extends the ProteomeCommons.org IO Framework, SIRIUS can read all mass spectrometry formats that are supported by this library[18]. A more detailed list of existing file formats and conversion utilities can be found on the web[19].

7.2.2 Analysis Preparation and Parameter Input

After importing the input data, the next step of the SIRIUS application workflow is to set up a new analysis run. To achieve this, user can start a new analysis wizard on the selected project. If no project with relevant data exists, a wizard for the creation of a

---

[13] `http://sashimi.sourceforge.org`

[14] `http://www.unidata.ucar.edu/software/netcdf/`

[15] `http://www.matrixscience.com/distiller.html`

[16] `http://tools.proteomecenter.org/software.php`

[17] `http://www.psidev.info`

[18] `http://www.proteomecommons.org/current/531/`

[19] `http://www.proteomecommons.org`,`http://tools.proteomecenter.org/software.php`

new project can be invoked, which also includes the analysis setup. Preparation of a new algorithm run includes the following steps: setting up the algorithm parameters, extracting isotope patterns from the input peak list, and validating the list of compounds for the subsequent computational analysis.

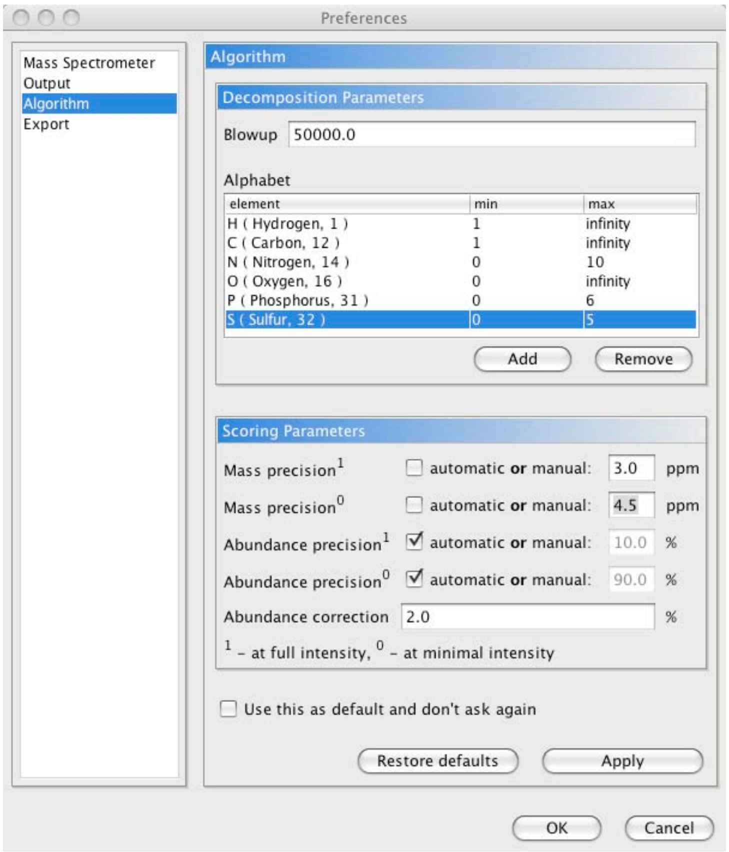
Algorithm Parameters. Algorithm parameters include the settings for the mass decomposition algorithm, such as a blowup factor and alphabet (see Section 3.2 on page 31), and for scoring of the isotope patterns (see Section 4.3 on page 44). Furthermore, the user can specify bounds on the minimal and maximal amount of elements in the candidate molecule. Often in *de novo* interpretations of MS data, a partial tag or other sample preparation information can suggest such constraints. Providing the lower and upper bounds on the number of elements restricts the search space to only relevant candidate molecules and can significantly speed up the search, see Section 6.1.2 on page 72.

SIRIUS provides the user with reasonable default values for the algorithm parameters. To customize the preferences to his own needs, user can open the `Preferences` dialog and modify the algorithm settings globally, see Figure 7.3. The applied changes will be saved in the user local workspace and automatically reloaded in any subsequent wizard.

Extracting and Validating Isotope Patterns. To extract isotope patterns, the input peak list is parsed and divided into signal groups related to different compounds. A peak list can also contain several signal groups belonging to the same compound, modified by different molecular ion adducts. Typical ion adducts, such as $[M+H]+$ and $[M+Na]+$, are provided as a part of the default MS instrument settings, and can be further customized. Identifying modifications is done by calculating mass differences between monoisotopic peak masses. In view of the small number of adducts, we apply a simple exhaustive search to find all matching mass differences. If there is no prior knowledge on the source of modification, the user can manually assign one or more adduct types to an isotope pattern. Moreover, if an automatic arrangement of individual peaks to isotope patterns has failed, each peak can be manually assigned to a proper signal group using a simple drag-and-drop.

After grouping isotope patterns to compounds, one or more compounds of interest can be selected for the subsequent identification analysis. Before executing the algorithm, the results on the approximate number of candidates for each compound can be previewed. To obtain these estimates, we use Equation (3.4) to compute the approximate number of decompositions for a given mass, error, and alphabet. If the approximate number of solutions exceeds a certain customizable threshold, the user receives a notification and can modify the input before submitting it to the algorithm.

After creation, the analysis is added to the corresponding project, and is shown as its child node in the `Project Explorer`, see the left pane in Figure 7.4. For the visual inspection of entered data, the contents of each node can be viewed in the `Properties` view, see the left bottom window in Figure 7.4.



Blowup 50000.0

Alphabet

element	min	max
H (Hydrogen, 1)	1	infinity
C (Carbon, 12)	1	infinity
N (Nitrogen, 14)	0	10
O (Oxygen, 16)	0	infinity
P (Phosphorus, 31)	0	6
S (Sulfur, 32)	0	5

Mass precision^1 ☐ automatic **or** manual: 3.0 ppm

Mass precision^0 ☐ automatic **or** manual: 4.5 ppm

Abundance precision^1 ☑ automatic **or** manual: 10.0 %

Abundance precision^0 ☑ automatic **or** manual: 90.0 %

Abundance correction 2.0 %

^{1} – at full intensity, ^{0} – at minimal intensity

☐ Use this as default and don't ask again

Figure 7.3: The SIRIUS preferences dialog.

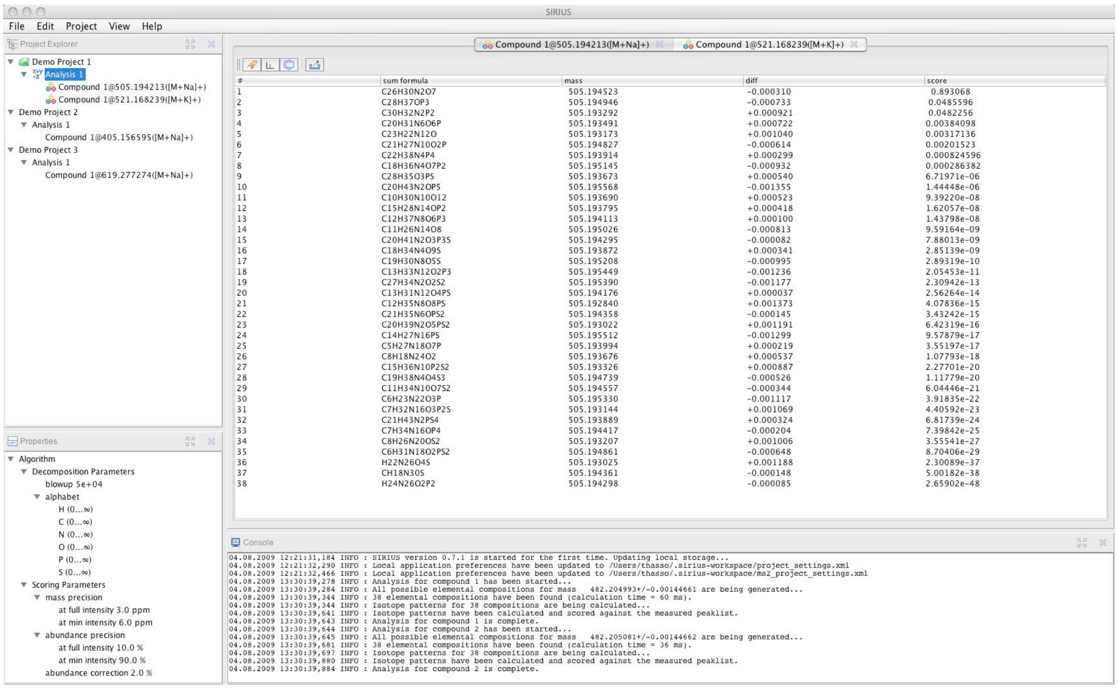
7.2.3 Analyzing Algorithm Results

The output of the algorithm is a list of candidate molecular formulas for each compound. Candidates are ranked with respect to the score that reflects the probability that this candidate is a true molecular formula for the analyzed compound, see Section 4.3. In the results view, molecular formulas are listed in the summary table, sorted in decreasing order of likelihoods. To view an entry in more detail, the user can select and compare several theoretical and measured isotope patterns visually.The visual comparison of isotope patterns is realized using an open-source JFreeChart Library 1.0.5[20], which offers an extensive set of functions to create professional quality charts. SIRIUS extends this rich functionality, e.g., for printing or zooming in the chart for detailed examination of the differences between isotope patterns.

Another possibility to analyze the results of the algorithm is to export them to the user workspace in the form of PDF reports, and open for further evaluation using the default system or user specified editor. Furthermore, the algorithm results can be exported

---

[20] `http://www.jfree.org/jfreechart/`


SIRIUS
File Edit Project View Help
Project Explorer
Compound 1@505.194213([M+Na]+)
Compound 1@521.168239([M+K]+)
Properties
Console

- Demo Project 1
  - Analysis 1
    - Compound 1@505.194213([M+Na]+)
    - Compound 1@521.168239([M+K]+)
- Demo Project 2
  - Analysis 1
    - Compound 1@405.156595([M+Na]+)
- Demo Project 3
  - Analysis 1
    - Compound 1@619.277274([M+Na]+)

#	sum formula	mass	diff	score
1	C26H30N2O7	505.194523	−0.000310	0.893068
2	C28H37OP3	505.194946	−0.000733	0.0485596
3	C30H32N2P2	505.193292	+0.000921	0.0482256
4	C20H31N6O6P	505.193491	+0.000722	0.00384098
5	C23H22N12O	505.193173	+0.001040	0.00317136
6	C21H27N10O2P	505.194827	−0.000614	0.00201523
7	C22H38N4P4	505.193914	+0.000299	0.000824596
8	C18H36N4O7P2	505.195145	−0.000932	0.000286382
9	C28H35O3PS	505.193673	+0.000540	6.71971e-06
10	C20H43N2OP5	505.195568	−0.001355	1.44448e-06
11	C10H30N10O12	505.193690	+0.000523	9.39220e-08
12	C15H28N14OP2	505.193795	+0.000418	1.62057e-08
13	C12H37N8O6P3	505.194113	+0.000100	1.43798e-08
14	C11H26N14O8	505.195026	−0.000813	9.59164e-09
15	C20H41N2O3P3S	505.194295	−0.000082	7.88013e-09
16	C18H34N4O9S	505.193872	+0.000341	2.85139e-09
17	C19H30N8O5S	505.195208	−0.000995	2.89319e-10
18	C13H33N12O2P3	505.195449	−0.001236	2.05453e-11
19	C27H34N2O2S2	505.195390	−0.001177	2.30942e-13
20	C13H31N12O4PS	505.194176	+0.000037	2.56264e-14
21	C12H35N8O8PS	505.192840	+0.001373	4.07836e-15
22	C21H35N6OPS2	505.194358	−0.000145	3.43242e-15
23	C20H39N2O5PS2	505.193022	+0.001191	6.42319e-16
24	C14H27N16PS	505.195512	−0.001299	9.57879e-17
25	C5H27N18O7P	505.193994	+0.000219	3.55197e-17
26	C8H18N24O2	505.193676	+0.000537	1.07793e-18
27	C15H36N10P2S2	505.193326	+0.000887	2.27701e-20
28	C19H38N4O4S3	505.194739	−0.000526	1.11779e-20
29	C11H34N10O7S2	505.194557	−0.000344	6.04446e-21
30	C6H23N22O3P	505.195330	−0.001117	3.91835e-22
31	C7H32N16O3P2S	505.193144	+0.001069	4.40592e-23
32	C21H43N2PS4	505.193889	+0.000324	6.81739e-24
33	C7H34N16OP4	505.194417	−0.000204	7.39842e-25
34	C8H26N20OS2	505.193207	+0.001006	3.55541e-27
35	C6H31N18O2PS2	505.194861	−0.000648	8.70406e-29
36	H22N26O4S	505.193025	+0.001188	2.30089e-37
37	CH18N30S	505.194361	−0.000148	5.00182e-38
38	H24N26O2P2	505.194298	−0.000085	2.65902e-48

- Algorithm
  - Decomposition Parameters
    - blowup 5e+04
    - alphabet
      - H (0...∞)
      - C (0...∞)
      - N (0...∞)
      - O (0...∞)
      - P (0...∞)
      - S (0...∞)
  - Scoring Parameters
    - mass precision
      - at full intensity 3.0 ppm
      - at min intensity 6.0 ppm
    - abundance precision
      - at full intensity 10.0 %
      - at min intensity 90.0 %
    - abundance correction 2.0 %

```
04.08.2009 12:21:31,184 INFO : SIRIUS version 0.7.1 is started for the first time. Updating local storage...
04.08.2009 12:21:32,290 INFO : Local application preferences have been updated to /Users/thasso/.sirius-workspace/project_settings.xml
04.08.2009 12:21:32,466 INFO : Local application preferences have been updated to /Users/thasso/.sirius-workspace/ms2_project_settings.xml
04.08.2009 13:30:39,278 INFO : Analysis for compound 1 has been started...
04.08.2009 13:30:39,284 INFO : All possible elemental compositions for mass   482.204993+/-0.00144661 are being generated...
04.08.2009 13:30:39,344 INFO : 38 elemental compositions have been found (calculation time = 60 ms).
04.08.2009 13:30:39,344 INFO : Isotope patterns for 38 compositions are being calculated...
04.08.2009 13:30:39,641 INFO : Isotope patterns have been calculated and scored against the measured peaklist.
04.08.2009 13:30:39,643 INFO : Analysis for compound 1 is complete.
04.08.2009 13:30:39,644 INFO : Analysis for compound 2 has been started...
04.08.2009 13:30:39,645 INFO : All possible elemental compositions for mass   482.205081+/-0.00144662 are being generated...
04.08.2009 13:30:39,681 INFO : 38 elemental compositions have been found (calculation time = 36 ms).
04.08.2009 13:30:39,697 INFO : Isotope patterns for 38 compositions are being calculated...
04.08.2009 13:30:39,880 INFO : Isotope patterns have been calculated and scored against the measured peaklist.
04.08.2009 13:30:39,884 INFO : Analysis for compound 2 is complete.
```

Figure 7.4: The SIRIUS main frame.

into machine–readable file formats such as plain text and XML documents, that can be further used as an input in a larger algorithmic pipeline.

Querying Post-processed Data. The candidate molecular formulas can be queried in an open-access NCBI PubChem database to obtain any information available for the compound with a given molecular formula. The retrieved information can facilitate in further verification or structure elucidation steps.

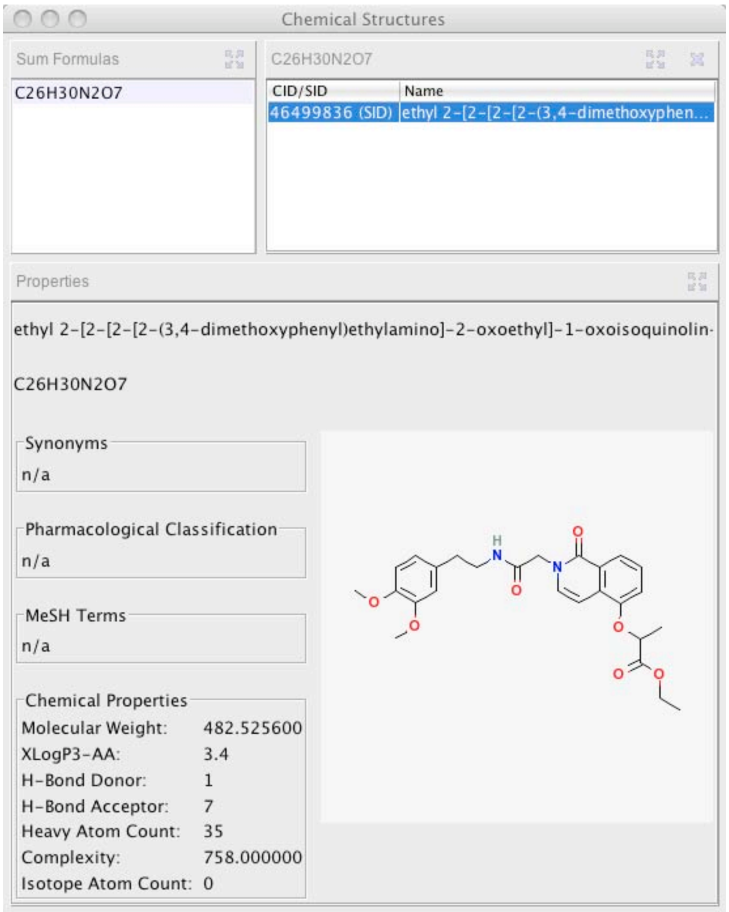
PubChem Compound and Substance databases belong to the NCBI *Entrez* cross-database search system[21], which provides a set of tools, Entrez Programming Utilities (eUtils), for the retrieval of *Entrez* data outside of the regular web query interface. The NCBI eUtils Web Service[22] provides developers an access to eUtils via SOAP protocol. For the short introduction to Web Services, see Section 6.1.2.

For any NCBI Entrez database, a simple search query for one or more molecular formulas can be created using a corresponding database client. SIRIUS employs a simple strategy to search for the compound information: First, the PubChem Compound database is queried, which contains a chemical structural information. If no match is found, then the PubChem Substance database is searched on the chemical description of the compound. Moreover, the latter can contain the information on the compound

---

[21] `http://www.ncbi.nlm.nih.gov/sites/gquery`

[22] `http://www.ncbi.nlm.nih.gov/entrez/query/static/esoap_help.html`



C26H30N2O7

CID/SID	Name
46499836 (SID)	ethyl 2-[2-[2-[2-(3,4-dimethoxyphen...

ethyl 2-[2-[2-[2-(3,4-dimethoxyphenyl)ethylamino]-2-oxoethyl]-1-oxoisoquinolin-

C26H30N2O7

Synonyms
n/a

Pharmacological Classification
n/a

MeSH Terms
n/a

Chemical Properties

Molecular Weight:	482.525600
XLogP3-AA:	3.4
H-Bond Donor:	1
H-Bond Acceptor:	7
Heavy Atom Count:	35
Complexity:	758.000000
Isotope Atom Count:	0

Figure 7.5: Query results to the NCBI PubChem Database.

biological activity. Upon retrieval, the results on putative compounds are shown in the dedicated view, see Figure 7.5. Displayed information include a molecule's name, a Compound ID (CID) or Substance ID (SID), a chemical structure, and other compound properties.

7.3 Summary

We have presented SIRIUS, a new java-based software framework for *de novo* identification of molecular formulas of metabolites using high-resolution mass spectra. SIRIUS employes recently developed, efficient computational methods, and combines them with a powerful graphical user interface for the comprehensive analysis of MS data. Well-defined infrastructure of our application provides a simple way to integrate new computation and visualization functionalities in the framework.

As the focus of our application lies on the computational analysis of MS data, SIRIUS currently supports the export of the analysis results as flat files only. However, after continuous usage of the application, the amount of analyzed data inevitably grows, and saving this information by means of flat files soon becomes impractical. Therefore, we are currently performing further developments to add a database support to our application for the local storage and reuse of the user specific project and analysis data.

Moreover, for execution of particularly memory consuming tasks and for non-interactive uses, we are planning to extend SIRIUS to run in the batch processing mode. For efficient memory usage, any input/output or computationally intensive tasks should be executed in a background thread. As our application constantly handles algorithmically intensive problems, we would like to enhance current SIRIUS implementations with the rich threading functionality of the Swing Application Framework for starting, stopping, and monitoring the progress of background tasks.

SIRIUS is freely available for download at the project website[23]. SIRIUS distributions for various operating systems include Unix/Linux, Windows and Mac OS. Alternatively, the user can deploy and execute the application over the network using Java Web Start[24]. To run SIRIUS, a Java Runtime Environment (JRE), version 1.5 or newer, is required. Regarding hardware requirements, we recommend at least 256 MB, or better 512 MB of free memory.

To facilitate in further collaborative development, we provide a SIRIUS Trac[25], an extensive bug and issue tracking system, which makes the application development process transparent for the end users, and provides means for an immediate feedback. Following the SIRIUS Trac, one can (i) examine the list of changes on SIRIUS over the time line, (ii) learn more about our future plans and milestones looking at the road map, (iii) report a problem or suggestion by creating a new ticket using the guest login, and (iv) view tickets on their resolution state. Finally, SIRIUS Trac includes a complete API documentation and a detailed user manual for the guidance to all nuances of the application usage.

---

[23] `http://bio.informatik.uni-jena.de/sirius/`

[24] `http://java.sun.com/products/javawebstart/`

[25] `http://bio.informatik.uni-jena.de/trac/sirius`

8 Conclusion

In this thesis, we have discussed several algorithms and applications for identification of sample molecules using high resolution mass spectrometry data.

After the completion of a multitude of various genomes, the bioinformatics research has progressed by focusing on the study of different functional levels of biological systems including transcriptome, proteome, and metabolome. These studies allow an understanding of living organisms at the molecular system level, and provide an insight into the complex functioning of biological systems.

With the major advances in MS instrumentation, an era of "precision" proteomics has been recently announced [81]. This implies a routine high-throughput acquisition of very accurate MS data, such as of several parts-per-million (ppm) mass measurement accuracy. For metabolomics, this accurate information can greatly facilitate in resolving its major challenge – the identification of metabolites, whose majority remains unknown to date. So far, this bottleneck has prevented metabolomics to unroll its full potential in life science. With regards to human health alone, great benefits of metabolomic investigations can be expected: Recent experimental studies have shown the potential of metabolomics to provide new biomarkers indicative of human disease [2]. However, in contrast to the emerging fields of genomics and proteomics, the progress in metabolomics regarding the number of existing computational methods, and applications for the analysis of MS data is far from satisfying. Due to the enormous complexity and high dynamics of the metabolome, the number of available metabolomics databases is limited [28]. Consequently, *de novo* identification solutions are highly sought.

We have presented an algorithm that employs accurate isotopic abundances for identifying the molecular formula of metabolites. Revealing the molecular formula is a crucial step in identifying a metabolite, as it greatly reduces the search space of possible molecular structures that can be further examined for automatic structure elucidation of molecules. We have proposed the notion of an *isotope pattern*, and presented methods for the efficient computation of isotope patterns, which is important for the analysis of larger molecules where the search space increases rapidly. We have evaluated our method on several datasets obtained from different MS techniques, and achieved results of excellent quality: For orthogonal time-of-flight mass spectrometry, we correctly identified molecular formulas for more than 90 % of the molecules with masses up to 1 000 Da; in other cases the search space was reduced to only a few candidates. Regarding the performance of our method, it is time and memory efficient and can be executed on a common desktop computer.

Next, we presented an algorithm to generate all solutions of a multi-dimensional equality constrained integer knapsack problem. This problem arises in the context of finding all amino acid sequences of a peptide with given molecular formula. The results both on simulated and experimental data demonstrate a superior performance of our *mixed matrix* approach which is about two orders of magnitude faster than the second-best approach.

Based on the implementations of our algorithms, we have developed several application tools that can be used for a small scale analysis of MS data. We have created DECOMP, a web-based application for decomposition of integer and real-valued masses over an arbitrary alphabet. DECOMP can be applied both for the interpretation of real-valued MS data and for solving instances of the Money Changing Problem. We have demonstrated the applicability of our software as a part of another algorithm, called CompNovo, for *de novo* peptide sequencing using tandem mass spectra. We also have created Rdisop, a R/Bioconductor package for analysis of small sample molecules using an accurate isotope pattern information.

Finally, we have developed the java-based software framework SIRIUS, which implements all the presented algorithms for identification of molecular formulas of metabolites, and combines them with an easy-to-use graphical user interface for the comprehensive analysis of MS data. A refined application structure provides a simple way to integrate new computation and visualization methods to the framework. A rich set of SIRIUS post-processing functionalities, including visualization of isotope patterns, customizable data export, and searching for molecular formulas in biological databases, is the essential part of our software allowing an efficient interpretation of the algorithm results.

Future Work. Although the results computed by our algorithms have achieved a remarkable quality, there are still some unsolved problems.

The current version of our algorithm uses information from only one molecular ion adduct per compound. Combining information from several signal groups belonging to the same compound (if they exist), could improve identification rates. Since measured data from different ion adducts of the same compound could be of different quality, our evaluation shows that a simple intersection of candidate lists makes results worse. Instead, a more refined approach is needed. We want to develop a probabilistic model which takes into account the quality of measurements when combining data from multiple adducts. This data can originate from multiply charged ions, so far ignored by our algorithm, but upon inclusion, can potentially improve the identification accuracy.

As a more theoretical aspect, one open problem is to improve the approximation of the number of decompositions over an alphabet with a considerable number of elements, such as the 20 standard amino acids. This can be achieved by deriving further coefficients of the polynomial in Equation (3.7) but is far from trivial (Takao Komatsu, personal communication).

Regarding our mixed matrix approach for the inference of amino acid composition from a given molecular formula, this work has just started and has further applications beyond the field of proteomics. We are currently conducting evaluations of our method to compare its performance to other related algorithms such as the one proposed in [93], that solve the general case of a system of linear Diophantine equations with negative coefficients. Improving existing solutions for the general linear Diophantine systems can have further implications, e.g, for decomposing complex chemical reactions in analytical chemistry.

Several practical aspects regarding our future work on the SIRIUS software should be noted. SIRIUS is a growing software project, and since its introduction has been subject to continuous improvement. In release 0.7, called SIRIUS starburst, a novel algorithm for *de novo* analysis of metabolites using tandem mass spectra [21] has been integrated, which employs the fragmentation information from several medium-quality mass spectra of the same compound, and searches for the most probable molecular formula that can cause the observed fragmentation pattern. In the future SIRIUS releases, we plan to integrate other related algorithms such as the one presented in [22], to establish a software platform for the comprehensive analysis of metabolites from MS data of various types such as single and tandem mass spectra, fragmentation trees, and others.

Another important aspect of our future work is to evaluate the performance of SIRIUS as an identification tool. We have started addressing this issue by including the functionality to search for compounds in the NCBI PubChem database in the current version of SIRIUS. In future releases, we plan to extend this feature towards fully automated searching for molecular formulas in various databases.

We hope that this work is a first step towards the development of efficient computational methods and applications for further discovery and comprehensive analysis of unknown sample molecules in all species including humans.

Bibliography

[1] K. Aardal and A. K. Lenstra. Hard equality constrained integer knapsacks. In *Proc. of Integer Programming and Combinatorial Optimization (IPCO 2002)*, volume 2337 of *Lect. Notes Comput. Sc.*, pages 350–366. Springer, 2002.

[2] B. L. Ackermann, J. E. Hale, and K. L. Duffin. The role of mass spectrometry in biomarker discovery and measurement. *Curr. Drug Metab.*, 7:525–539, 2006.

[3] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, 2003.

[4] G. Audi, A. Wapstra, and C. Thibault. The AME2003 atomic mass evaluation (ii): Tables, graphs, and references. *Nucl. Phys. A*, 729:129–336, 2003.

[5] V. Bafna and N. Edwards. SCOPE: A probabilistic model for scoring tandem mass spectra against a peptide database. *Bioinformatics*, 17:S13–S21, 2001.

[6] N. Bandeira, J. Ng, D. Meluzzi, R. G. Linington, P. Dorrestein, and P. A. Pevzner. De novo sequencing of nonribosomal peptides. In *Proc. of Research in Computational Molecular Biology (RECOMB 2008)*, volume 4955 of *Lect. Notes Comput. Sc.*, pages 181–195. Springer, 2008.

[7] C. Bartels. Fast algorithm for peptide sequencing by mass spectrometry. *Biomed. Environ. Mass Spectrom.*, 19:363–368, 1990.

[8] J. Baumbach. Coryneregnet 4.0 = a reference database for corynebacterial gene regulatory networks. *BMC Bioinformatics*, 8, 2007.

[9] M. Beck, I. M. Gessel, and T. Komatsu. The polynomial part of a restricted partition function related to the Frobenius problem. *Electron. J. Comb.*, 8(1):N7, 2001.

[10] M. W. Bern and D. Goldberg. EigenMS: De novo analysis of peptide tandem mass spectra by spectral graph partitioning. In *Proc. of Research in Computational Molecular Biology (RECOMB 2005)*, volume 3500 of *Lect. Notes Comput. Sc.*, pages 357–372. Springer, 2005.

[11] M. J. Bertrand, P. Thibault, M. J. Evans, and D. Zidarov. Determination of the empirical formula of peptides by fast atom bombardment mass spectrometry. *Biomed. Environ. Mass Spectrom.*, 14(6):249–256, 1987.

[12] A. Bertsch, A. Leinenbach, A. Pervukhin, M. Lubeck, R. Hartmer, C. Baessmann, Y. A. Elnakady, R. Müller, S. Böcker, C. G. Huber, and O. Kohlbacher. De novo peptide sequencing by tandem mass spectrometry using complementary collision-induced dissociation and electron transfer dissociation. *Electrophoresis*, Jul 2009. Accepted for publication.

[13] R. Bino, R. Hall, O. Fiehn, J. Kopka, K. Saito, J. Draper, B. Nikolau, P. Mendes, U. Roessner-Tunali, M. Beale, R. Trethewey, B. Lange, E. Wurtele, and L. Sumner. Potential of metabolomics as a functional genomics tool. *Trends Plant Sci.*, 9(9):418–425, 2004.

[14] S. Böcker. Sequencing from compomers: The puzzle. *Theory Comput. Syst.*, 39(3):455–471, 2006.

[15] S. Böcker, M. Letzel, Zs. Lipták, and A. Pervukhin. Decomposing metabolomic isotope patterns. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2006)*, volume 4175 of *Lect. Notes Comput. Sc.*, pages 12–23. Springer, 2006.

[16] S. Böcker, M. Letzel, Zs. Lipták, and A. Pervukhin. SIRIUS: Decomposing isotope patterns for metabolite identification. *Bioinformatics*, 25(2):218–224, 2009.

[17] S. Böcker, Z. Lipták, M. Martin, A. Pervukhin, and H. Sudek. DECOMP—from interpreting mass spectrometry peaks to solving the Money Changing Problem. *Bioinformatics*, 24(4):591–593, 2008.

[18] S. Böcker and Zs. Lipták. Efficient mass decomposition. In L. M. Liebrock, editor, *Proc. of ACM Symposium on Applied Computing (ACM SAC 2005)*, pages 151–157, Santa Fe, USA, 2005. ACM Press.

[19] S. Böcker and Zs. Lipták. A fast and simple algorithm for the Money Changing Problem. *Algorithmica*, 48(4):413–432, 2007.

[20] S. Böcker and A. Pervukhin. Inferring peptide composition from molecular formulas. In *Proc. of Computing and Combinatorics Conference (COCOON 2009)*, volume 5609 of *Lect. Notes Comput. Sc.*, pages 277–286. Springer, 2009.

[21] S. Böcker and F. Rasche. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24:I49–I55, 2008. Proc. of *European Conference on Computational Biology* (ECCB 2008).

[22] S. Böcker, F. Rasche, and T. Steijger. Annotating fragmentation patterns. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2009)*, volume 5724 of *Lect. Notes Comput. Sc.*, pages 13–24. Springer, 2009.

[23] D. C. Chamrad, G. Körting, K. Stühler, H. E. Meyer, J. Klose, and M. Blüggel. Evaluation of algorithms for protein identification from sequence databases using mass spectrometry data. *Proteomics*, 4:619–628, 2004.

[24] T. Chen, M.-Y. Kao, M. Tepel, J. Rush, and G. M. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *J. Comput. Biol.*, 8(3):325–337, 2001.

[25] K. R. Clauser, P. Baker, and A. L. Burlingame. Role of accurate mass measurement (+/- 10 ppm) in protein identification strategies employing MS or MS/MS and database searching. *Anal. Chem.*, 71(14):2871–2882, Jul 1999.

[26] V. Dančík, T. A. Addona, K. R. Clauser, J. E. Vath, and P. A. Pevzner. De novo peptide sequencing via tandem mass spectrometry: A graph-theoretical approach. *J. Comput. Biol.*, 6(3/4):327–342, 1999.

[27] R. Datta and M. Bern. Spectrum fusion: Using multiple mass spectra for de novo peptide sequencing. In *Proc. of Research in Computational Molecular Biology (RECOMB 2008)*, volume 4955 of *Lect. Notes Comput. Sc.*, pages 140–153. Springer, 2008.

[28] K. Dettmer, P. A. Aronov, and B. D. Hammock. Mass spectrometry-based metabolomics. *Mass Spectrom Rev.*, 26(1):51–78, 2007.

[29] E. W. Deutsch, H. Lam, and R. Aebersold. Data analysis and bioinformatics tools for tandem mass spectrometry in proteomics. *Physiological Genomics*, 33:18–25, March 2008.

[30] P. A. DiMaggio and C. A. Floudas. De novo peptide identification via tandem mass spectrometry and integer linear optimization. *Anal. Chem.*, 79(4):1433–1446, 2007.

[31] Q. Ding, L. Xiao, S. Xiong, Y. Jia, H. Que, Y. Guo, and S. Liu. Unmatched masses in peptide mass fingerprints caused by cross-contamination: An updated statistical result. *Proteomics*, 3:1313–1317, 2003.

[32] B. Domon and R. Aebersold. Mass spectrometry and protein analysis. *Science*, 312:212–217, February 2006.

[33] P. Du, W. A. Kibbe, and S. Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22:2059–2065, 2006.

[34] I. Eidhammer, K. Flikka, L. Martens, and S.-O. Mikalsen. *Computational Methods for Mass Spectrometry Proteomics.* Wiley, 2007.

[35] T. Ejaz, S. Rahmann, and J. Stoye. Online abelian pattern matching. Technical report, Bielefeld University, Technische Fakultät der Universität Bielefeld, Abteilung Informationstechnik, Postfach 10 01 31, 33501 Bielefeld, Germany, January 2008.

[36] J. E. Elias, F. D. Gibbons, O. D. King, F. P. Roth, and S. P. Gygi. Intensity-based protein identification by machine learning from a library of tandem mass spectra. *Nat. Biotechnol.*, 22(2):214–219, 2004.

[37] J. K. Eng, A. L. McCormack, and J. R. Yates III. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J. Am. Soc. Mass Spectr.*, 5:976–989, 1994.

[38] J. Falkner, J. Falkner, and P. Andrews. Proteomecommons.org io framework: reading and writing multiple proteomics data formats. *Bioinformatics*, 23:262–263, 2007.

[39] J. Fenn, M. Mann, C. Meng, S. Wong, and C. Whitehouse. Electrospray ionisation for mass spectrometry of large biomolecules. *Science*, 246:64–71, 1989.

[40] D. Fenyö and R. C. Beavis. A method for assessing the statistical significance of mass spectrometry-based protein identifications using general scoring schemes. *Anal. Chem.*, 75(4):768–774, 2003.

[41] A. R. Fernie, R. N. Trethewey, A. J. Krotzky, and L. Willmitzer. Metabolite profiling: from diagnostics to systems biology. *Nat. Rev. Mol. Cell Biol.*, 5(9):763–769, 2004.

[42] O. Fiehn. Metabolomics–the link between genotypes and phenotypes. *Plant Mol. Biol.*, 48(1-2):155–171, 2002.

[43] B. Fischer, V. Roth, F. Roos, J. Grossmann, S. Baginsky, P. Widmayer, W. Gruissem, and J. M. Buhmann. NovoHMM: a hidden Markov model for de novo peptide sequencing. *Anal. Chem.*, 77(22):7265–7273, 2005.

[44] M. Fowler. *Patterns of Enterprise Application Architecture.* Addison-Wesley, 2002.

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

[46] A. M. Frank, M. M. Savitski, M. N. Nielsen, R. A. Zubarev, and P. A. Pevzner. De novo peptide sequencing and identification with precision mass spectrometry. *J. Proteome Res.*, 6(1):114–123, September 2007.

[47] A. Fürst, J.-T. Clerc, and E. Pretsch. A computer program for the computation of the molecular formula. *Chemom. Intell. Lab. Syst.*, 5:329–334, 1989.

[48] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[49] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *J. Proteome Res.*, 3:958–964, 2004.

[50] R. C. Gentleman, V. J. Carey, D. M. B., B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith,

G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. BioConductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

[51] P. Gilmore and R. Gomory. Multi-stage cutting stock problems of two and more dimensions. *Oper. Res.*, 13(1):94–120, 1965.

[52] A. H. Grange, M. C. Zumwalt, and G. W. Sovocool. Determination of ion and neutral loss compositions and deconvolution of product ion mass spectra using an orthogonal acceleration time-of-flight mass spectrometer and an ion correlation program. *Rapid Commun. Mass Spectrom.*, 20(2):89–102, 2006.

[53] T. Griebel, M. Brinkmeyer, and S. Böcker. EPoS: a modular software framework for phylogenetic analysis. *Bioinformatics*, 24(20):2399–2400, 2008.

[54] J. Gross. *Mass Spectrometry: A textbook.* Springer, Berlin, 2004.

[55] W. Haas, B. K. Faherty, S. A. Gerber, J. E. Elias, S. A. Beausoleil, C. E. Bakalarski, X. Li, J. Ville, and S. P. Gygi. Optimization and use of peptide mass measurement accuracy in shotgun proteomics. *Molecular & Cellular Proteomics*, 5.7:1326–1337, 2006.

[56] M. Hardman and A. A. Makarov. Interfacing the orbitrap mass analyzer to an electrospray ion source. *Anal. Chem.*, 75(7):1699–1705, 2003.

[57] G. G. Harrigan and R. Goodacre, editors. *Metabolic Profiling: Its Role in Biomarker Discovery and Gene Function Analysis.* Springer, 2003.

[58] F. He, C. L. Hendrickson, and A. G. Marshall. Baseline mass resolution of peptide isobars: A record for molecular mass resolution. *Anal. Chem.*, 73(3):647–650, 2001.

[59] W. J. Henzel, T. M. Billeci, J. T. Stults, S. C. Wong, C. Grimley, and C. Watanabe. Identifying proteins from two-dimensional gels by molecular mass searching of peptide fragments in protein sequence databases. *Proc. Natl. Acad. Sci. U.S.A.*, 90(11):5011–5015, 1993.

[60] W. J. Henzel, C. Watanabe, and J. T. Stults. Protein identification: The origins of peptide mass fingerprints. *J. Am. Soc. Mass Spectr.*, 14(9):931–942, 2003.

[61] F. Hillenkamp, M. Karas, R. Beavis, and B. Chait. Matrix-assisted laser desorption/ionization mass spectrometry of biopolymers. *Anal. Chem.*, 63:1193A–1203A, 1991.

[62] D. H. Horn, R. A. Zubarev, and F. W. McLafferty. Automated reduction and interpretation of high resolution electrospray mass spectra of large molecules. *J. Am. Soc. Mass Spectr.*, 11:320–332, 2000.

[63] C. S. Hsu. Diophantine approach to isotopic abundance calculations. *Anal. Chem.*, 56(8):1356–1361, 1984.

[64] Y. Iijima, Y. Nakamura, Y. Ogata, K. Tanaka, N. Sakurai, K. Suda, T. Suzuki, H. Suzuki, K. Okazaki, M. Kitayama, S. Kanaya, K. Aoki, and D. Shibata. Metabolite annotations based on the integration of mass spectral information. *Plant J.*, 54(5):949–962, Jun 2008.

[65] P. Jones, R. G. Côté, L. Martens, A. F. Quinn, C. F. Taylor, W. Derache, H. Hermjakob, and R. Apweiler. PRIDE: a public repository of protein and peptide identifications for the proteomics community. *Nucleic Acids Res.*, 34(Database-Issue):659–663, 2006.

[66] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.*, 34:D354–D357, 2006.

[67] E. A. Kapp, F. Schütz, L. M. Connolly, J. A. Chakel, J. E. Meza, C. A. Miller, D. Fenyo, J. K. Eng, J. N. Adkins, G. S. Omenn, and R. J. Simpson. An evaluation, comparison, and accurate benchmarking of several publicly available ms/ms search algorithms: Sensitivity and specificity analysis. *Proteomics*, 5:3475–3490, 2005.

[68] M. Karas and F. Hillenkamp. Laser desorption ionization of proteins with molecular masses exceeding 10,000 Daltons. *Anal. Chem.*, 60:2299–2301, 1988.

[69] M. Katajamaa, J. Miettinen, and M. Oresic. Mzmine: toolbox for processing and visualization of mass spectrometry based molecular profile data. *Bioinformatics*, 22(5):634–636, 2006.

[70] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[71] S. Kim, N. Gupta, and P. A. Pevzner. Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *J. Proteome Res.*, 7(8):3354–3363, 2008.

[72] T. Kind and O. Fiehn. Metabolomic database annotations via query of elemental compositions: Mass accuracy is insufficient even at less than 1 ppm. *BMC Bioinformatics*, 7(1):234, Apr 2006.

[73] T. Kind and O. Fiehn. Seven golden rules for heuristic filtering of molecular formulas obtained by accurate mass spectrometry. *BMC Bioinformatics*, 8:105, 2007.

[74] T. Komatsu. On the number of solutions of the diophantine equation of frobenius - general case. *Mathematical Communications*, 8:195–206, 2003.

[75] H. Kubinyi. Calculation of isotope distributions in mass spectrometry: A trivial solution for a non-trivial problem. *Anal. Chim. Acta*, 247:107–119, 1991.

[76] E. Lange, C. Gröpl, K. Reinert, O. Kohlbacher, and A. Hildebrandt. High-accuracy peak picking of proteomics data using wavelet techniques. In *Proc. of Pacific Symposium on Biocomputing (PSB 2006)*, pages 243–254, 2006.

[77] Zs. Lipták. *Strings in Proteomics and Transcriptomics. Algorithmic and Combinatorial Questions in Mass Spectrometry and EST Clustering.* PhD thesis, Universität Bielefeld, Bielefeld, Germany, May 2005.

[78] A. Luedemann, K. Strassburg, A. Erban, and J. Kopka. TagFinder for the quantitative analysis of gas chromatography–mass spectrometry (GC-MS)-based metabolite profiling experiments. *Bioinformatics*, 24(5):732–737, Mar 2008.

[79] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report TR-178, Department of Electrical Engineering, Princeton University, March 1975.

[80] M. Mann, P. Hojrup, and P. Roepstorff. Use of mass spectrometric molecular weight information to identify proteins in sequence databases. *Biol. Mass Spectrom.*, 22(6):338–345, 1993.

[81] M. Mann and N. L. Kelleher. Precision proteomics: The case for high resolution and high mass accuracy. *PNAS*, 105(47):18132–18138, November 2008.

[82] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Chichester, 1990.

[83] L. McHugh and J. W. Arthur. Computational methods for protein identification from mass spectrometry data. *PLoS Comput. Biol.*, 4(2):e12, February 2008.

[84] J. S. Morris, K. R. Coombes, J. Koomen, K. A. Baggerly, and R. Kobayashi. Feature extraction and quantificatioon for mass spectrometry in biomedical applications using the mean spectrum. *Bioinformatics*, 21(9):1764–1775, 2005.

[85] A. I. Nesvizhskii, O. Vitek, and R. Aebersold. Analysis and validation of proteomic data generated by tandem mass spectrometry. *Nature Methods*, 4(10):787–797, September 2007.

[86] S. Neumann, A. Pervukhin, and S. Böcker. *Mass decomposition with the Rdisop package*, October 13 2007. Part of BioConductor 2.2.

[87] H. Neuweger, S. P. Albaum, M. Dondrup, M. Persicke, T. Watt, K. Niehaus, J. Stoye, and A. Goesmann. MeltDB: a software platform for the analysis and integration of metabolomics experiment data. *Bioinformatics*, 24(23):2726–2732, 2008.

[88] S. Ojanperä, A. Pelander, M. Pelzing, I. Krebs, E. Vuori, and I. Ojanperä. Isotopic pattern and accurate mass determination in urine drug screening by liquid chromatography/time-of-flight mass spectrometry. *Rapid Commun. Mass Spectrom.*, 20(7):1161–1167, 2006.

[89] J. V. Olsen, L. M. F. de Godoy, G. Li, B. Macek, P. Mortensen, R. Pesch, A. Makarov, O. Lange, S. Horning, and M. Mann. Parts per million mass accuracy on an orbitrap mass spectrometer via lock mass injection into a c-trap. *Molecular & Cellular Proteomics*, 4:2010–2021, 2006.

[90] S. Orchard, L. Montechi-Palazzi, E. W. Deutsch, P.-A. Binz, A. R. Jones, N. Paton, A. Pizarro, D. M. Creasy, J. Wojcik, and H. Hermjakob. Five years of progress in the standardization of proteomics data 4th annual spring workshop of the hupo-proteomics standards initiative. *Proteomics*, 7:3436–3440, 2007.

[91] P. M. Palagi, P. Hernandez, D. Walther, and R. D. Appel. Proteome informatics i: Bioinformatics tools for processing experimental data. *Proteomics*, 6:5435–5444, 2006.

[92] T. Palzkill. *Proteomics.* Kluver Academic Publishers, 2002.

[93] D. Papp and B. Vizvári. Effective solution of linear diophantine equation systems with an application in chemistry. *J. Math. Chem.*, 39(1):15–31, 2006.

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

[95] P. G. A. Pedrioli, J. K. Eng, R. Hubley, M. Vogelzang, E. W. Deutsch, B. Raught, B. Pratt, E. Nilsson, R. H. Angeletti, R. Apweiler, K. Cheung, C. E. Costello, H. Hermjakob, S. Huang, R. K. Julian, E. Kapp, M. E. McComb, S. G. Oliver, G. Omenn, N. W. Paton, R. Simpson, R. Smith, C. F. Taylor, W. Zhu, and R. Aebersold. A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.*, 22(11):1459–66, 2004.

[96] V. Pellegrin. Molecular formulas of organic compounds: the nitrogen rule and degree of unsaturation. *J. Chem. Educ.*, 60(8):626–633, 1983.

[97] D. N. Perkins, D. J. Pappin, D. M. Creasy, and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567, 1999.

[98] S. Pillai, V. Silventoinen, K. Kallio, M. Senger, S. Sobhany, J. Tate, S. Velankar, A. Golovin, K. Henrick, P. Rice, P. Stoehr, and R. Lopez. Soap-based services provided by the european bioinformatics institute. *Nucleic Acids Res.*, 33(Web Sever issue):W25–W28, 2005.

[99] S. C. Pomerantz, J. A. Kowalak, and J. A. McCloskey. Determination of oligonucleotide composition from mass spectromectrically measured molecular weight. *J. Am. Soc. Mass Spectrom.*, 4:204–209, 1993.

[100] A. L. Rockwood and S. L. Van Orden. Ultrahigh-speed calculation of isotope distributions. *Anal. Chem.*, 68:2027–2030, 1996.

[101] A. L. Rockwood, J. R. Van Orman, and D. V. Dearden. Isotopic compositions and accurate masses of single isotopic peaks. *J. Am. Soc. Mass Spectr.*, 15:12–21, 2004.

[102] S. Rogers, R. A. Scheltema, M. Girolami, and R. Breitling. Probabilistic assignment of formulas to mass peaks in metabolomics experiments. *Bioinformatics*, 25(4):512–518, 2009.

[103] M. S. Sabatine, E. Liu, D. A. Morrow, E. Heller, R. McCarroll, R. Wiegand, G. F. Berriz, F. P. Roth, and R. E. Gerszten. Metabolomic identification of novel biomarkers of myocardial ischemia. *Circulation*, 112:3868–3865, 2005.

[104] A. Salomaa. Counting (scattered) subwords. *B. Euro. Assoc. Theo. Comp. Sci.*, 81:165–179, 2003.

[105] M. M. Savitski, M. L. Nielsen, F. Kjeldsen, and R. A. Zubarev. Proteomics-grade de novo sequencing approach. *J. Proteome Res.*, 4:2348–2354, 2005.

[106] S. Schneiker, O. Perlova, O. Kaiser, K. Gerth, A. Alici, M. O. Altmeyer, D. Bartels, T. Bekel, S. Beyer, E. Bode, H. B. Bode, C. J. Bolten, J. V. Choudhuri, S. Doss, Y. A. Elnakady, B. Frank, L. Gaigalat, A. Goesmann, C. Groeger, F. Gross, L. Jelsbak, L. Jelsbak, J. Kalinowski, C. Kegler, T. Knauber, S. Konietzny, M. Kopp, L. Krause, D. Krug, B. Linke, T. Mahmud, R. Martinez-Arias, A. C. McHardy, M. Merai, F. Meyer, S. Mormann, J. Munöz-Dorado, J. Perez, S. Pradella, S. Rachid, G. Raddatz, F. Rosenau, C. Rückert, F. Sasse, M. Scharfe, S. C. Schuster, G. Suen, A. Treuner-Lange, G. J. Velicer, F.-J. Vorhölter, K. J. Weissman, R. D. Welch, S. C. Wenzel, D. E. Whitworth, S. Wilhelm, C. Wittmann, H. Blöcker, A. Pöhler, and R. Müller. Complete genome sequence of the myxobacterium sorangium cellulosum. *Nat. Biotechnol.*, 25(11):1281–1289, 2007.

[107] J. Senior. Partitions and their representative graphs. *Am. J. Math.*, 73(3):663–689, 1951.

[108] M. W. Senko, C. L. Hendrickson, and M. R. Emmet. External accumulation of ions for enhanced electrospray ionization fourier transform ion cyclotron resonance mass spectrometry. *JASMS*, 8:970–976, 1997.

[109] I. Shadforth, D. Crowther, and C. Bessant. Protein and peptide identification algorithms using ms for use in high-throughput, automated pipelines. *Proteomics*, 5:4082–4095, 2005.

[110] G. Siuzdak. *Mass Spectrometry for Biotechnology.* Academic Press, 1996.

[111] C. A. Smith, E. J. Want, G. O'Maille, R. Abagyan, and G. Siuzdak. XCMS: Processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Anal. Chem.*, 78(3):779–787, Feb 2006.

[112] R. D. Smith, G. A. Anderson, M. S. Lipton, L. Pasa-Tolic, Y. Shen, T. P. Conrads, T. D. Veenstra, and H. R. Udseth. An accurate mass tag strategy for quantitative and high-throughput proteome measurements. *Proteomics*, 2:513–523, 2002. Issue 5.

[113] A. P. Snyder. *Interpreting Protein Mass Spectra: A Comprehensive Resource.* Oxford University Press, 2000.

[114] M. D. Soffer. The molecular formula generalized in terms of cyclic elements of structure. *Science*, 127(3303):880, April 1958.

[115] B. Spengler. De novo sequencing, peptide composition analysis, and composition-based sequencing: A new strategy employing accurate mass determination by fourier transform ion cyclotron resonance mass spectrometry. *J. Am. Soc. Mass Spectrom.*, 15(5):703–714, 2004.

[116] M. Sturm, A. Bertsch, C. Gröpl, A. Hildebrandt, R. Hussong, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, A. Zerck, K. Reinert, and O. Kohlbacher. Openms - an open-source software framework for mass spectrometry. *BMC Bioinformatics*, 9:163, 2008.

[117] J. E. P. Syka, J. J. Coon, M. J. Schroeder, J. Shabanowitz, and D. F. Hunt. Peptide and protein sequence analysis by electron transfer dissociation mass spectrometry. *Proc. Natl. Acad. Sci. U.S.A.*, 101(26):9528–9533, 2004.

[118] S. Tanner, S. H. Payne, S. Dasari, Z. Shen, P. A. Wilmarth, L. L. David, W. F. Loomis, S. P. Briggs, and V. Bafna. Accurate annotation of peptide modifications through unrestrictive database search. *J. Proteome Res.*, 7:170–181, 2008.

[119] S. Tanner, H. Shu, A. Frank, L.-C. Wang, E. Zandi, M. Mumby, P. A. Pevzner, and V. Bafna. Inspect: Identification of posttranslationally modified peptides from tandem mass spectra. *Anal. Chem.*, 77:4626–4639, 2005.

[120] R. Tautenhahn, C. Böttcher, and S. Neumann. Annotation of lc/esi-ms mass signals. In *Proc. of Conference on Bioinformatics Research and Development (BIRD 2007)*, volume 4414 of *Lect. Notes Comput. Sc.* Springer, 2007.

[121] J. A. Taylor and R. S. Johnson. Sequence database searches via de novo peptide sequencing by tandem mass spectrometry. *Rapid Commun. Mass Spectrom.*, 11:1067–1075, 1997.

[122] J. A. Taylor and R. S. Johnson. Implementation and uses of automated de novo peptide sequencing by tandem mass spectrometry. *Anal. Chem.*, 73:2594–2604, 2001.

[123] J. van Lint and R. Wilson. *A Course in Combinatorics.* Cambridge University Press, 2001.

[124] S. M. Watkins and J. B. German. Toward the implementation of metabolomic assessments of human health and nutrition. *Curr. Opin. Biotechnol.*, 13(5):512–516, October 2002.

[125] J. T. Watson and O. D. Sparkman. *Introduction to Mass Spectrometry: Instrumentation, Applications, and Strategies for Data Interpretation.* Wiley, 2007.

[126] C. Whitehouse, R. Dreyer, M. Yamashita, and J. Fenn. Electrospray interface for liquid chromatographs and mass spectrometers. *Anal. Chem.*, 57:675–679, 1985.

[127] H. Wilf. *generatingfunctionology.* Academic Press, 1990.

[128] C. Yang, Z. He, and W. Yu. Comparison of public peak detection algorithms for maldi mass spectrometry data analysis. *BMC Bioinformatics*, 10(4), 2009.

[129] J. R. Yates III, J. K. Eng, A. L. McCormack, and D. Schieltz. Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Anal. Chem.*, 67(8):1426–1436, 1995.

[130] J. R. Yates III, S. Speicher, P. R. Griffin, and T. Hunkapillar. Peptide mass maps: A highly informative approach to protein identification. *Anal. Biochem.*, 214:397–408, 1993.

[131] J. A. Yergey. A general approach to calculating isotopic distributions for mass spectrometry. *Int. J. Mass Spectrom. Ion Phys.*, 52(2–3):337–349, 1983.

[132] J. Zhang, W. Gao, J. Cai, S. He, R. Zeng, and R. Chen. Predicting molecular formulas of fragment ions with isotope patterns in tandem mass spectra. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2(3):217–230, 2005.

[133] N. Zhang, R. Aebersold, and B. Schwikowski. ProbID: a probabilistic algorithm to identify peptides through sequence database searching using tandem mass spectral data. *Proteomics*, 2(10):1406–1412, Oct 2002.

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

[135] Z. Zhang. De novo peptide sequencing based on a divide-and-conquer algorithm and peptide tandem spectrum simulation. *Anal. Chem.*, 76:6374–6383, 2004.

[136] R. A. Zubarev, N. L. Kelleher, and F. W. McLafferty. Electron capture dissociation of multiply charged protein cations. a nonergodic process. *J Am Chem Soc*, 120:3265–3266, 1998.